Búsqueda de Mejoras en la Detección Automática de Estrellas Variables

Jeremías Rodríguez

Director: Dr. Pablo M. Granitto

Codirector: Dr. Juan B. Cabral

Universidad Nacional de Rosario
Facultad de Ciencias Exactas, Ingeniería y Agrimensura

Rosario, Argentina
2021

Resumen

La astronomía está atravesando una profunda transformación debido al desarrollo de modernos telescopios terrestres y satelitales, que han fomentado la realización de enormes relevamientos astronómicos. Ante la abrumadora cantidad y calidad de los datos generados, se vuelve imprescindible el uso de procedimientos automatizados. Consecuentemente, diversas técnicas de aprendizaje automatizado y minería de datos surgen como una elección natural a la hora de analizar y extraer información de modernos datasets astronómicos.

En este trabajo se hará uso de mediciones generadas por el relevamiento VVV del infrarrojo cercano (realizado en Parnal, Chile), que relevó aproximadamente 10^9 estrellas durante un período de 5 años. Se aplicarán diversas técnicas de aprendizaje automatizado con el objeto de identificar estrellas de tipo RR Lyrae, las cuales son extremadamente valiosas pues permiten estimar distancias a viejas poblaciones estelares. En concreto, se hará uso de clasificadores de tipo Random Forest y Support Vector Machine, haciendo énfasis en comprender por qué los primeros parecen tener significativamente mejor performance en este tipo de datasets astronómicos.

Contenido

1. Introducción y Estado del Arte

1.1. Introducción

La astronomía está atravesando una profunda transformación debido al desarrollo de modernos telescopios terrestres y satelitales, que han fomentado la realización de enormes relevamientos astronómicos [Benjamin et al., 2003] [Skrutskie et al., 2006] [Brown et al., 2018] [Soszyński et al., 2019]. Ante la abrumadora cantidad y calidad de los datos generados, se vuelve necesario el uso de procedimientos automatizados. Consecuentemente, diversas técnicas de aprendizaje automatizado y minería de datos surjen como una elección natural a la hora de analizar y extraer información de modernos datasets astronómicos. [Richards et al., 2011] [Ball and Brunner, 2010]

Vista Variables in the Via Lactea (VVV) es un relevamiento público del infrarrojo cercano que cubrió aproximadamente 520 deg^2 del disco y bulbo galáctico, utilizando el telescopio VISTA en Paranal, Chile [Minniti et al., 2010] (ver figura **1-1**). El VVV relevó aproximadamente 10^9 fuentes astronómicas en las bandas ZYJHK durante un período de 5 años, teniendo entre sus principales objetivos la realización de un mapa tridimensional del bulbo galáctico.

En este contexto, las estrellas variables de tipo RR Lyrae son una poderosa herramienta para obtener distancias a poblaciones de estrellas en la Vía Láctea, debido a que satisfacen una relación entre sus períodos y sus luminosidades absolutas que permite estimar distancias [Shapley, 1918] [Baade, 1946]. Sin embargo, a causa de la inmensa cantidad de información generada, la clasificación manual de estrellas se vuelve impracticable. Se requiere, por lo tanto, un método automatizado para identificar estrellas variables de tipo RR Lyrae de entre las $10^6 - 10^7$ estrellas variables que se espera encontrar en el área explorada [Cabral et al., 2020a].

Investigaciones recientes han mostrado que algoritmos de aprendizaje automatizado basados en ensambles de árboles tienen un excelente desempeño en este tipo de tareas, en tanto que otros métodos tradicionales como Support Vector Machines parecen no ser tan efectivos [Elorrieta López et al., 2016] [Cabral et al., 2020a]. El objetivo de esta tesina es intentar mejorar los resultados previos en el problema de detección de RR Lyrae en la VVV utilizando Support Vector Machines, así como comprender por qué métodos basados en ensambles de

Figura 1-1.: Vista aérea de la plataforma del "ESO Very Large Telescope (VLT)", en la cima del Cerro Parnal, en el desierto chileno de Atacama. Créditos: J.L. Dauvergne y G. Hüdepohl

árboles aparentan funcionar mejor en este tipo de datos.

Durante este trabajo se hizo uso de una amplia variedad de técnicas de aprendizaje automatizado, incluyendo diversos tipos de preprocesamiento, reducción de dimensionalidad y selección de atributos, visualización de los datos y corrección de imbalance de clases.

1.2. Conceptos astronómicos

1.2.1. Estrellas RR Lyrae

Una **estrella** es un objeto astronómico que consiste en una esfera luminosa de plasma, la cual mantiene su forma debido a su propia gravedad. Características como luminosidad, tamaño, evolución y duración de vida son definidas principalmente por su masa inicial.

Aquellas estrellas cuya luminosidad (magnitud aparente[1]) exhibe cambios periódicos o aleatorios se denominan **estrellas variables.** Las estrellas de tipo variable han jugado un rol

[1]Este valor indica la medida del brillo y cantidad de energía por segundo por metro cuadrado que se recibe de un objeto celeste por un observador en la Tierra. Como dicha cantidad recibida depende de la transmisión de la atmósfera en dichas bandas, las magnitudes aparentes se normalizan a un valor fuera de la atmósfera terrestre. La escala de magnitudes es una relación inversa logarítmica por la cual la estrella más brillante es la que tiene menor magnitud

crucial en la historia de la astronomía. El Catálogo General de Estrellas Variables [Samus et al., 2009] enumera más de 110 clases y subclases de estrellas variables.

Las estrellas variables pueden, en primera instancia, clasificarse en **variables intrínsecas** y **variables extrínsecas**, dependiendo del origen de la variabilidad observada. Si la causa de la variabilidad observada es inherente a la estrella, como es el caso de una estrella que periódicamente se expande y contrae, se le denomina intrínseca. Contrariamente, aquellas estrellas cuya variabilidad se debe a factores externos, como un compañero orbital que ocasionalmente la eclipsa, se denominan variables extrínsecas.

Probablemente el subgrupo más importante de las estrellas variables intrínsecas son las **estrellas pulsantes**, que varían en radio y luminosidad en el tiempo, expandiéndose y contrayéndose con períodos tan breves como minutos, o tan extensos como años, dependiendo del tamaño de la estrella. Para una revisión moderna de la fenomenología de estrellas pulsantes, puede remitirse a [Márcio Catelan, 2015].

Dentro de las estrellas pulsantes, se encuentran las subclases **RR Lyrae** y **Cepheids**, las cuales satisfacen una relación entre sus períodos y sus luminosidades absolutas que las convierte en invaluables herramientas para una de las tareas más complejas en astronomía: estimar distancias [Shapley, 1918] [Baade, 1946].

Las estrellas de tipo **RR Lyrae** (en adelante, RRL) tienen períodos de entre 0.2 y 1.2 días [Smith, 2004], y tienen una edad de aproximadamente 14 Gyr[2]. Son consideradas excelentes candelas estándar, especialmente importantes a la hora de explorar distancias y propiedades de viejas problaciones estelares. Basándose en relevamientos recientes, se estima que hay alrededor de 140000 estrellas RRL en nuestra galaxia [Soszyński et al., 2019] [Brown et al., 2018]

En base a sus curvas de luz, las RRL fueron originalmente separadas en subtipos a, b y c [Bailey, 1902], aunque posteriores trabajos unificaron los dos primeros subtipos en uno solo, RRab, dado que son fundamentalmente distintos del tercero, RRc [Schwarzchild, 1938]. Adicionales subtipos, extremadamente raros, fueron añadidos posteriormente.

1.2.2. El relevamiento VVV

"Vista Variables in the Via Lactea (VVV) ESO Public Survey" es un relevamiento fotométrico (terrestre) del bulbo galáctico y parte del disco interno de la Vía Láctea, que se llevó a cabo utilizando el telescopio VISTA en Parnal, Chile. Uno de los objetivos principales de la VVV

[2] 1 gigayear (Gyr) = 10^9 años

es adquirir un mayor entendimiento del origen, estructura y evolución de la Vía Láctea. Para una descripción detallada de la VVV puede consultarse [Minniti et al., 2010], en tanto que un artículo más reciente con énfasis en variabilidad puede consultarse en [Catelan et al., 2014]

El VVV, así como su sucesor el "VVV Survey eXtended", i.e. VVVX, persiguen el objetivo de producir un atlas profundo de una gran parte del Bulbo de la Vía Láctea, así como de una fracción del Disco Galáctico interno. Este mapa se produjo mediante un monitoreo sistemático de estas regiones, habiéndose completado 1929 horas de observación durante cinco años, comenzando en 2010.

Dentro del barrido total del relevamiento, es de interés astronómico la generación de catálogos de cualquier tipo de objeto, esperándose identificar no sólo estrellas variables, sino planetas, asteroides y otros fenómenos. En la figura **1-2** se ilustra el número de fenómenos estimado.

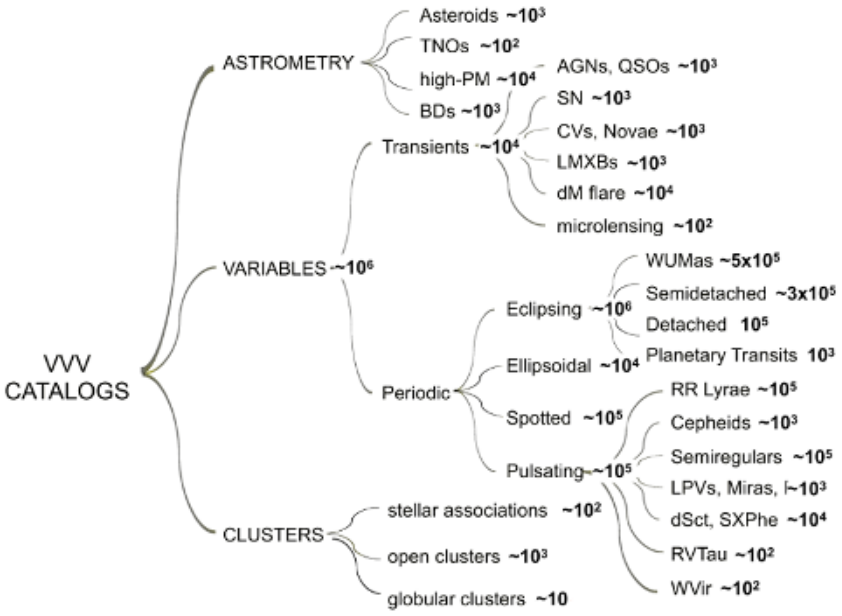


Figura 1-2.: Número esperado de fenómenos astrofísicos que espera detectar VVV en sus catálogos. Minniti et. al. 2010.

Los datos del VVV se presentan en una unidad llamada baldosa (tile, en inglés), una zona rectangular del cielo de $1{,}501 deg^2$ relevada a través del tiempo. Se requirieron 196 tiles para mapear el área del Bulbo, así como 152 tiles para el Disco. Cada tile se compone de varias imágenes en alta resolución para diferentes tipos de filtros (bandas anchas) de frecuencias lumínicas en el infrarrojo cercano (cinco en total). Asimismo, por cada imagen existe una base de datos con los valores de posición, magnitud y color de las fuentes de luz presentes en la imagen, llamada "catálogo fotométrico". En el caso del VVV, la totalidad de las baldosas

que constituyeron el relevamiento puede apreciarse en la figura **1-3**

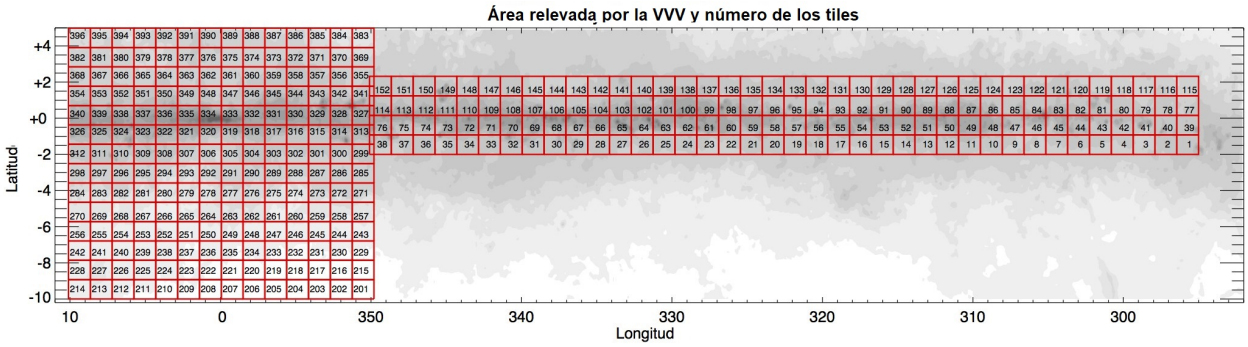


Figura 1-3.: Área del relevamiento VVV, en coordenadas galácticas. En escala de grises se puede apreciar la densidad estelar del área relevada. [Skrutskie et al., 2006]

La identificación de aquellas fuentes de luz que corresponden a estrellas RRL es de particular interés para VVV, dado que determinar distancias es vital para cumplir uno de sus objetivos primordiales: mapear el Bulbo Galáctico. Más aún, la existencia de trabajos previos [Gran et al., 2015] [Gran et al., 2016] que identifican algunos cientos de fuentes de luz como RRL dentro del área mapeada por la VVV, implica que se cuenta con un conjunto de datos que puede ser utilizado para entrenar y testear clasificadores de aprendizaje automatizado.

1.2.3. Extracción de atributos

En esta tesina se hace un uso extensivo de los datos producidos en [Cabral et al., 2020a]. Por cada tile en un subconjunto de tiles del relevamiento, se generó un dataset numérico, donde cada fila corresponde a una estrella, descripta por 62 atributos (del inglés, features) numéricas. En esta sección se explicará, a alto nivel, cómo se obtuvieron dichos atributos a partir de las mediciones de fotometría del infrarrojo cercano producidas por VVV.

Durante el relevamiento VVV se generaron enormes volúmenes de datos (1.4 TB/noche). El pipeline de VVV [Emerson et al., 2004] brinda, por cada imagen generada, una base de datos de archivos con los valores de posición, magnitud y color de las fuentes de luz presentes, llamada **catálogo fotométrico**. Aunque existen varios tipos de catálogos, sólo dos son utilizados en este trabajo:

- En primer lugar, el catálogo **Pawprint Stack** contiene lecturas de los sensores infrarrojos de la cámara VIR-CAM del telescopio VISTA, correspondientes a la observación de un tile en una fecha dada (también llamado época).

- En segundo lugar, el catálogo **Band Merge** se utiliza para tener una lista confiable de estrellas en un cierto tile, dado que ciertas fuentes de luz pueden no estar presentes entre distintos Pawprint Stacks debido a condiciones meteorológicas o problemas en los instrumentos de medición.

Para determinar la variación en magnitud de cada estrella presente en cada Band Merge, fue necesario identificar todas sus ocurrencias en diferentes Pawprint Stacks. De esta forma, se puede formar una serie temporal de observaciones de la estrella en cuestión, haciendo uso de técnicas de cross-matching [Gray et al., 2007]. El resultado son series temporales como la que se puede apreciar en la figura **1-4**. Nótese que el tiempo de muestreo puede ser muy irregular, por lo que se presupone que es aleatorio.

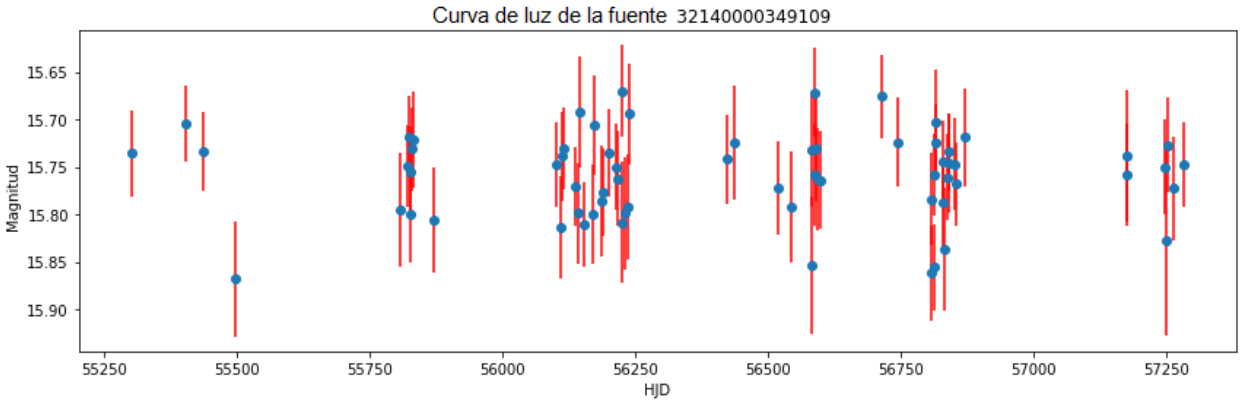


Figura 1-4.: Curva de luz de una de las estrellas relevadas por VVV. El eje x muestra el tiempo expresado en días heliocéntricos medios (HJD), en tanto que el eje y muestra la magnitud aparente observada. Cada punto azul corresponde a una observación en un cierto Pawprint Stack, mientras que las líneas rojas indican el error asociado a cada medición. Imagen tomada de la documentación de Carpyncho, `https://carpyncho-py.readthedocs.io/`

Habiéndose obtenido las series temporales de cada estrella, el siguiente paso fue estimar su período. Se utiliza el método de Fast Lomb-Scargle [Lomb, 1976] [Scargle, 1983] [VanderPlas, 2018] para recuperar cada período, el cual consiste en ajustar una función sinusoidal a los datos muestreados.

A partir de las curvas de luz y el período estimado, se extrajeron 49 atributos (o características) numéricas utilizando la herramienta *Feets* [Cabral et al., 2018]. Estos atributos se refieren al período, variación de magnitud y morfología de las curvas de luz. Una descripción de cada atributo puede consultarse en el apéndice A.

Una contribución original de [Cabral et al., 2020a] fue la inclusión de atributos que describen las temperaturas o colores intrínsecos de cada estrella. Dado que la presencia de polvo interestelar ocasiona un fenómeno denominado extinción, que provoca un enrojecimiento de los colores, fue necesario realizar una corrección utilizando mapas de extinción [McWilliam, 2011]. Como resultado final, se extrajeron 13 atributos numéricos describiendo el color de cada estrella. Nuevamente, la lista completa de atributos de color, junto con sus descripciones está disponible en el apéndice A.

1.2.4. Descripción de los datos

En esta sección se describirán en detalle los datasets generados en [Cabral et al., 2020a], que pueden ser accedidos por el público a través de la librería Python **Carpyncho** [Cabral et al., 2020b].

Cada dataset generado corresponde a un cierto tile de la VVV. Cada fila de un dado dataset se corresponde a una cierta estrella variable, descripta por 62 atributos numéricos tal y como se describió en la sección anterior. La siguiente tabla resume los datos liberados por Carpyncho:

Tile	**Épocas**	**Tamaño**	**RRL**	**Unknown**	**RRL / Tamaño**
b206	73	157825	47	157778	0.03 %
b214	74	149557	34	149523	0.02 %
b216	73	168996	43	168953	0.03 %
b220	73	209798	65	209733	0.03 %
b228	73	199853	28	199825	0.01 %
b234	73	293013	126	292887	0.04 %
b247	73	406386	192	406194	0.05 %
b248	74	417839	218	417621	0.05 %
b261	74	555693	252	555441	0.05 %
b262	74	573873	314	573559	0.06 %
b263	94	568110	317	567793	0.06 %
b264	94	595234	307	594927	0.05 %
b277	73	718567	429	718138	0.06 %
b278	74	742153	436	741717	0.06 %
b360	74	939110	669	938441	0.07 %
b396	73	486639	15	486624	0.00 %
Total	1216	7182646	3492	7179154	0.05 %

Como se puede observar, algunas de las estrellas variables de cada dataset están identificadas como RRL. Estas etiquetas provienen de realizar cross-matching con catálogos de estrellas

variables de otros relevamientos con zonas de observación superpuestas a la VVV: *OGLE-III* [Udalski, 2004], *OGLE-IV* [Udalski et al., 2015] y VizieR [Ochsenbein, F. et al., 2000].

En todos los experimentos de clasificación de este trabajo, se consideró como clase positiva a aquellas estrellas etiquetadas como RRL, siendo el resto la clase negativa. Es importante mencionar que nuestras clases positivas y negativas no son completamente certeras, dado que no han sido validadas con una inspección manual de cada fuente. Es posible que ciertas estrellas etiquetadas como RRL en realidad no lo sean y viceversa. Esto es especialmente válido en tiles como *b396*, para las cuales hay pocas RRL previamente clasificadas en relevamientos anteriores.

Para la mayoría de los experimentos en este trabajo, se utilizó un subconjunto de tiles (*b234*, *b261*, *b278* y *b360*) correspondiente a áreas del Bulbo que resultan de particular interés:

- Hay una buena densidad de RRL, pues las zonas se superponen bien con relevamientos anteriores. Por lo tanto, es probable que haya menos RRLs etiquetadas como no-RRL.
- *b278* es de particular interés pues contiene la llamada ventana de Baade, una zona con poco polvo intergaláctico en la línea visual de la tierra al núcleo galáctico [Baade, 1946]. Esta zona ha sido históricamente utilizada para estudiar RRL.

1.3. Aprendizaje automatizado

Esta sección introducirá brevemente algunos conceptos básicos de aprendizaje automatizado, y está fuertemente basada en los capítulos introductorios de [Mitchell, 1997] y [Hastie et al., 2001].

1.3.1. Definición y tipos de aprendizaje

El campo del **aprendizaje automatizado** consiste en el estudio de algoritmos que mejoran automáticamente su rendimiento gracias a la experiencia. Formalmente, Tom Mitchell en su libro *"Machine Learning"* [Mitchell, 1997] define aprendizaje automatizado como sigue:

> Se dice que un programa de computadora aprende de la experiencia E respecto a una tarea T y una medida de desempeño P, si el desempeño medido con P en una tarea T , mejora con la experiencia E.

Dada una muestra de datos, los algoritmos de aprendizaje automatizado construyen un modelo con el objeto de realizar predicciones o tomar decisiones *sin haber sido expresamente programados para hacerlo*. Tales algoritmos pueden ser aplicados a un amplio rango de tareas, tales como aprender a conducir un vehículo autónomo [Pomerleau, 1989], aprender a jugar

backgammon a nivel profesional [Tesauro, 1995] o clasificar estructuras astronómicas [Weir et al., 1995].

Tradicionalmente, las distintas tareas a llevar a cabo por algoritmos de aprendizaje automatizado se dividen en tres paradigmas [Russell and Norvig, 2009]:

- **Aprendizaje supervisado**: Consiste en aprender una función $f : I \mapsto O$, basándose en un conjunto de ejemplos $T \subset I \times O$. En otras palabras, el algoritmo de aprendizaje automatizado intenta inferir una función a partir de un conjunto de entradas de ejemplo, etiquetadas con su valor de salida. Si el conjunto O es discreto, se trata de una tarea de **clasificación**; en tanto que si es continuo, se trata de una tarea de **regresión**.
- **Aprendizaje no supervisado**: En aprendizaje no supervisado, sólo hay entradas pero no salidas, por lo que el objetivo es aprender patrones y relaciones en conjuntos de datos no etiquetados.
- **Aprendizaje por refuerzo (Reinforcement learning)**: Consiste en aprender a tomar decisiones en un ambiente dinámico, con el objeto de maximizar una cierta noción de recompensa acumulada.

Sin importar el tipo de supervisión, todo aprendizaje automático tiene una fase de entrenamiento y una fase de predicción. La **fase de entrenamiento** se basa en suministrar datos a los modelos para ajustar sus parámetros internos. Asimismo, hay varios modelos que poseen hiperparámetros que suelen ser configurados previamente a la fase de entrenamiento. Luego, en la **fase de predicción**, los modelos pueden usarse para predecir alguna información acerca de datos desconocidos.

En este trabajo nos enfocaremos en técnicas de aprendizaje automatizado supervisado, dado que se desea abordar la tarea de clasificar estrellas variables en RRL o no-RRL.

1.3.2. Medidas de performance

Como se mencionó anteriormente, el aprendizaje supervisado consiste en aproximar una función $f : I \mapsto O$, donde usualmente $I = \mathbb{R}^n$ y $O \subseteq \mathbb{R}$. Durante una primera fase de entrenamiento, un conjunto de m ejemplos $x_i \in I,\ i = 1, \ldots, m$ junto con sus salidas asociadas $y_i \in O$ es utilizado por un cierto algoritmo de aprendizaje automatizado supervisado para producir una estimación de f, a la que llamaremos $\hat{f}$.

Para evaluar el desempeño de tal algoritmo de aprendizaje automatizado, se necesita alguna forma de medir cuán bien $\hat{f}$ aproxima a f. Dado un conjunto de pares etiquetado $S = \{(a_i, b_i) \mid a_i \in I,\ b_i \in O,\ i = 1, \ldots, k\}$, las siguientes métricas son de interés:

- En problemas de regresión, la métrica más comúnmente utilizada es el *error cuadrático medio*:

$$MSE = \sum_{i=1}^{k} (b_i - \hat{f}(a_i))^2$$

- En problemas de clasificación, podemos calcular la proporción de errores realizados:

$$Error\ Rate = \frac{1}{k} \sum_{i=1}^{k} I(b_i, \hat{f}(a_i))$$

  Donde:

$$I(y_i, \hat{y}_i) = \begin{cases} 0 & \text{si } y_i = \hat{y}_i \\ 1 & \text{si } y_i \neq \hat{y}_i \end{cases}$$

Si S es exactamente el conjunto de entrenamiento, entonces ambas métricas se llaman *training MSE* y *training Error Rate* respectivamente; y nos indican cuán bien $\hat{f}$ aproxima a f para los elementos de entrenamiento. Sin embargo, en general, estamos interesados en evaluar el desempeño del clasificador en elementos nuevos que no pertenecen al conjunto de entrenamiento.

Usualmente, se reserva una porción de los datos disponibles para ser utilizada como **conjunto de test**. Estos datos no son utilizados en la fase de entrenamiento, sino que son utilizados para calcular métricas como *test Error Rate* una vez terminada la fase de entrenamiento. Métricas calculadas sobre un conjunto de test proveen una noción de cuán bien el clasificador funcionará en datos no antes vistos.

Una alternativa para medir cuán bien $\hat{f}$ aproxima a f sin necesidad de tener que separar los datos etiquetados disponibles en un conjunto de entrenamiento y uno de testeo es utilizar **cross validation** [Berrar, 2018]. Dado un dataset de elementos del dominio etiquetados D, cross validation consiste en:

1. Permutar los elementos de D de forma aleatoria.
2. Particionar D en k grupos
3. Para cada partición p:
   - Entrenar el algoritmo de aprendizaje automatizado utilizando todas las particiones excepto p
   - Utilizar el modelo entrenado para realizar predicciones sobre los elementos de p, calculando la métrica de performance deseada (por ejemplo, *Error rate*).
   - Guardar la métrica obtenida, y descartar el modelo entrenado.
4. Combinar las k métricas obtenidas, por ejemplo calculando su promedio. El valor obtenido representa la performance del algoritmo de aprendizaje automatizado sobre la totalidad de los datos.

1.3.3. Medidas de performance en clasificación binaria

En este trabajo estamos particularmente interesados en problemas de **clasificación binaria**, es decir, problemas de aprendizaje automatizado supervisado en los cuales el conjunto de etiquetas contiene solo dos elementos, o clases, a menudo llamados clase positiva y negativa. En el contexto de clasificación binaria, dado un modelo entrenado, cada elemento se puede clasificar en una de las siguientes cuatro categorías:

	Clase predicha: Negativa	**Clase predicha: Positiva**
Clase real: Negativa	Verdadero negativo (TN)	Falso positivo (FP)
Clase real: Positiva	Falso negativo (FN)	Verdadero positivo (TP)

Los falsos positivos son aquellos elementos de clase negativa, clasificados incorrectamente por el modelo con etiqueta positiva. Similarmente, los falsos negativos son aquellos elementos con etiqueta positiva, clasificados como negativos. Los verdaderos positivos (negativos), son aquellos elementos positivos (negativos), que fueron clasificados correctamente.

Dado un conjunto de testeo, la performance de un clasificador binario puede visualizarse al calcular una **matriz de confusión**. La matriz de confusión mostrará la suma de elementos de cada una de las cuatro categorías explicadas. En particular, podemos reescribir la métrica Error Rate como:

$$ErrorRate = \frac{FP+FN}{FP+FN+TP+TN}$$

Equivalentemente, es usual analizar la **tasa de aciertos**:

$$Acc = \frac{TP+TN}{FP+FN+TP+TN}$$

Otras métricas de interés son:

- **Precisión**: La proporción de instancias clasificadas como positivas, que realmente lo son:

$$Precision = \frac{TP}{TP+FP}$$

- **Recall**: La proporción de instancias positivas que son detectadas por el clasificador.

$$Recall = \frac{TP}{TP+FN}$$

1.3.4. El dilema sesgo-varianza

En esta sección, se discutirá brevemente el dilema sesgo-varianza en aprendizaje automatizado [Hastie et al., 2001]. Supongamos que la variable que estamos intentando predecir es Y, basándonos en un conjunto de atributos X. Asumimos que hay una relación subyacente entre ambos:

$$Y = f(X) + \epsilon$$

Donde ϵ se distribuye de forma normal, con media cero y varianza σ^2, y representa ruido en los datos. Utilizando algún algoritmo de aprendizaje automatizado, y a partir de un dataset de entrenamiento D, se obtiene un modelo $\bar{f}$ que intenta aproximar f. El error cuadrático de predicción para un input x es[3]:

$$Err(x) = E[(Y - \bar{f}(x))^2]$$

Lo cual puede reescribirse como:

$$Err(x) = (E[\bar{f}(x)] - f(x))^2 + E[(\bar{f}(x) - E[\bar{f}(x)])^2] + \sigma^2$$

Donde podemos darle un nombre a cada término de esa suma:

$$Err(x) = Sesgo^2 + Varianza + ErrorIrreducible$$

El **error irreducible**, que es la varianza de la función objetivo, está fuera de nuestro control. No puede ser reducido creando buenos modelos, y es una medida de la cantidad de ruido en nuestros datos. Los otros dos términos, en cambio, están bajo nuestro control.

El **sesgo** es la diferencia entre la predicción promedio de nuestro modelo, y el valor correcto que intentamos predecir. Este error puede ser pensado como el error causado por asunciones simplistas del modelo, como por ejemplo asumir que la distribución de los datos es lineal. Un modelo con alto sesgo presta muy poca atención a los datos de entrenamiento.

La **varianza** es la variabilidad de la predicción de un modelo, es decir, cuánto varía $\bar{f}(x)$ alrededor de su media. Modelos con alta varianza prestan mucha atención a los datos de entrenamiento, pero no generalizan tan bien en datos no vistos anteriormente.

De forma general, cuando la complejidad de nuestro modelo crece, la varianza tiende a aumentar en tanto que el sesgo disminuye. Lo opuesto sucede cuando la complejidad del modelo disminuye.

[3]Nótese que las esperanzas se calculan sobre las diferentes elecciones de dataset de entrenamiento D.

La figura **1-5** muestra el comportamiento típico de la varianza y el sesgo en función de la complejidad del modelo. El error de entrenamiento tiende a decrecer cuando incrementamos la complejidad del modelo, es decir, cuando ajustamos los datos con mayor precisión. Sin embargo, un exceso de ajuste (**overfitting**) conduce a un modelo que se adapta demasiado a los datos de entrenamientos, y no generaliza bien en test. En este caso, las predicciones tendrán gran varianza. En contraste, si el modelo no es muy complejo, no se ajusta lo suficiente a los datos de entrenamiento (**underfitting**) y tendrá un alto sesgo, resultando nuevamente en alto error de generalización.

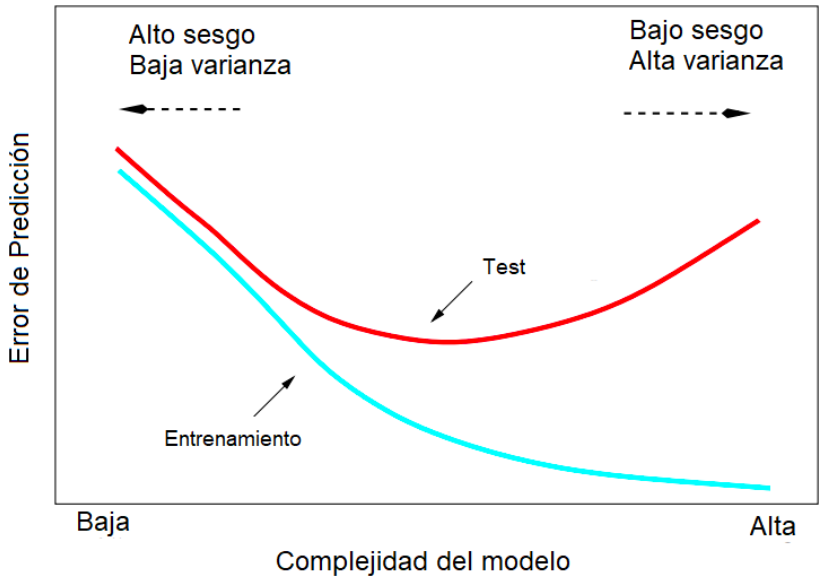


Figura 1-5.: Error de test y de entrenamiento en función de la complejidad del modelo. Créditos: [Hastie et al., 2001]

Como se puede ver en la figura **1-5**, el error de entrenamiento será muy bajo en modelos complejos. Sin embargo, sería inadecuado utilizar el error de entrenamiento como una medida de cuán bien el modelo funciona, pues puede no funcionar tan bien en un conjunto de testeo no antes visto. Por este motivo es que se deben utilizar métricas calculadas sobre los datos de testeo para realizar afirmaciones sobre la performance de un modelo.

Utilizar métricas obtenidas a través de cross-validation también es correcto, pues debe notarse que en el procedimiento descripto en la sección 1.3.2 las métricas de cada partición siempre se calculan sobre datos que no han sido utilizados para entrenar los modelos.

1.3.5. Problemas desbalanceados

A la hora de estudiar problemas de clasificación, es importante tener en cuenta la proporción de datos de cada clase. Un **problema de clasificación desbalanceado** es aquel en el cual la cantidad de elementos de cada clase no es proporcional. Ciertas aplicaciones como diagnóstico médico o detección de transacciones fraudulentas con tarjeta de crédito presentan datasets con un número muy pequeño de instancias positivas[4], que son sin embargo cruciales de clasificar correctamente [Akbani et al., 2004].

Si se utilizan datasets imbalanceados para entrenar y testear, es inadecuado utilizar métricas de desempeño basadas en clasificar la *mayor cantidad* de elementos de forma correcta, como *Acc*. La hipótesis mas sencilla, clasificar todas las instancias como pertenecientes a la clase mayoritaria, a menudo maximiza tales métricas.

Un primer paso a la hora de trabajar con datos desbalanceados es, entonces, utilizar métricas que no son sensitivas al imbalance de clases. En clasificación binaria precision y recall son métricas adecuadas.

A menudo, muchos métodos de aprendizaje automatizado para clasificación binaria producen un puntaje[5] en $[0, 1]$ para cada instancia a clasificar. Puntajes cercanos a 0 indican que el clasificador tiene mayor certeza de que una cierta entrada pertenece a la clase negativa, en tanto que valores cercanos a 1 indican pertenencia a la clase positiva.

En tal escenario, resulta necesaria la elección de un umbral $t \in [0, 1]$, de forma tal que sólo instancias cuyos puntajes son mayores a t sean clasificadas como positivas, siendo las demás negativas. En consecuencia, resulta importante determinar un t adecuado para el dominio de aplicación en estudio, dado que distintos valores de t tendrán un impacto directo en la matriz de confusión producida.

Valores altos de t tenderán a producir clasificadores con mayor precisión y menor recall, en tanto que valores pequeños de t tenderán a clasificadores con mayor recall y menor precisión. Una forma gráfica de analizar esta correspondencia es a través de la curva de precision-recall del clasificador, como la de la figura **1-6**, en la cuál se grafica la precisión y el recall obtenidos para un rango de valores de t.

Si se está abordando un problema desbalanceado, una forma determinar si un clasificador tiene mejor performance que otro es comparar sus curvas de precision-recall en el dataset de

[4]En problemas de clasificación binaria, usualmente se considera clase positiva a la clase minoritaria, en tanto que la clase mayoritaria es la negativa

[5]Se utiliza el término puntaje (del inglés *score*), dado que no se utilizará un enfoque probabilístico.

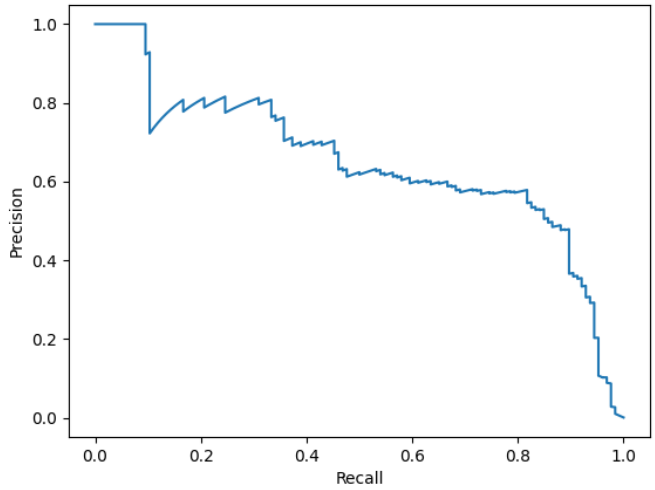


Figura 1-6.: Ejemplo de una curva de precision-recall. Cada punto de la curva corresponde a la precisión y el recall obtenidos para un cierto valor de t.

test. Dado que las curvas de precision-recall tienden a ser bastante ruidosas en áreas de bajo recall, en este trabajo se utilizó en repetidas ocasiones el área bajo la curva de precision-recall restringida al dominio [0.35,1] para obtener una medida de performance numérica, a la que llamaremos R-AUPRC (del inglés Robust Area Under the Precision Recall Curve, área debajo de la curva de precision-recall robusta).

R-AUPRC resulta útil cuando se desea comparar decenas o cientos de curvas a la vez, siendo imposible su inspección visual y prefiriéndose aquel modelo que maximiza el R-AUPRC. Otra métrica alternativa es utilizar la precisión del clasificador a un cierto valor fijo de recall.

Además de utilizar métricas adecuadas para evaluar el desempeño, otras técnicas para trabajar con datos desbalanceados son:

- Corregir manualmente el imbalance de clases en los datos. Una posibilidad es eliminar datos de la clase mayoritaria aleatoriamente (**undersampling**) [Japkowicz, 2000] hasta alcanzar el balance deseado. Una segunda posibilidad es generar nuevas instancias de la clase minoritaria (**oversampling**). [He and Garcia, 2009]
- Forzar a que el clasificador preste más atención a la clase minoritaria [Akbani et al., 2004]. Por ejemplo, como se verá en secciones posteriores, para ciertos métodos de aprendizaje automatizado como Support Vector Machines es posible asociar un peso (*class_weight*) a cada clase, de tal forma que el clasificador será menos permisivo a la hora de clasificar incorrectamente clases con alto peso.

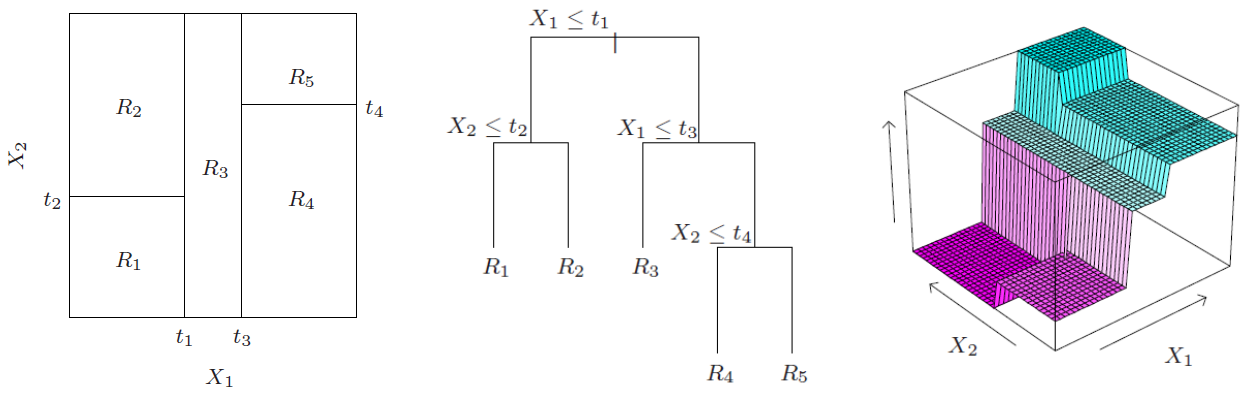
1.3.6. Árboles de decisión

Esta sección está enteramente basada en [Hastie et al., 2001], y presentará uno de los métodos de aprendizaje automatizado supervisado para clasificación más ampliamente utilizados: Árboles de Decisión.

Los métodos basados en árboles particionan el dominio de los atributos en un conjunto de rectángulos, y luego ajustan un modelo sencillo a cada uno. Son conceptualmente simples, pero poderosos.

Consideremos un problema de regresión donde se intenta predecir una variable continua Y, basándose en dos atributos X_1 y X_2, ambos tomando valores en el intervalo $[0, 1]$.

La figura **1-7**(a) muestra una partición del espacio de atributos. Se utilizó una partición simple, restringiéndose a particiones recursivas binarias. En primer lugar se dividió el espacio en dos regiones, y se modeló la respuesta de la media de Y en cada región. Se escogió el atributo y el punto de división (split) para encontrar el mejor ajuste. Recursivamente, cada una de las dos regiones obtenidas se divide en otras dos regiones, hasta alcanzar algún criterio de terminación.



(a) (b) (c)

Figura 1-7.: Ejemplo de un árbol de decisión sencillo. Fuente: [Hastie et al., 2001]

En el ejemplo de la figura **1-7**(a), primero se dividió el espacio de atributos en $X_1 = t_1$. Luego, la región $X_1 \leq t_1$ se dividió en $X_2 = t_2$ y la región $X_1 > t_1$ se dividió en $X_1 = t_3$. Finalmente, la región $X_1 > t_3$ se dividió en $X_2 = t_4$. El resultado de este proceso es una partición en cinco regiones $R_1, R_2, \ldots, R_5$. El modelo de regresión producirá predicciones constantes $c_1, \ldots, c_5$ de Y para una nueva instancia de acuerdo a la partición en que sus atributos se encuentren:

$$\hat{f}(X) = \sum_{m=1}^{5} (c_m I\{(X_1, X_2) \in R_m\})$$

Este mismo modelo puede ser representado por el árbol de decisión binario de la figura **1-7**(b). El dataset completo se encuentra en la raíz, en la parte superior del árbol. Instancias que satisfacen la condición en cada nodo son asignadas a la rama izquierda, en tanto que las otras son asignadas a la rama derecha. Los nodos terminales del árbol (también llamados hojas) corresponden a las regiones $R_1, \ldots, R_5$. La superficie de decisión obtenida para $c_1 = -5, c_2 = -7, c_3 = 0, c_4 = 2, c_5 = 4$ puede ser observada en la figura **1-7**(c).

Una de las ventajas clave de los árboles de decisión binarios es su interpretabilidad, incluso cuando hay más de dos atributos.

Cassification And Regression Tree (CART)

Veamos a continuación cómo se construye un árbol de decisión utilizando el algoritmo CART. Supongamos que el problema consta de p atributos de entrada y una variable de salida, y se cuenta con N observaciones (x_i, y_i) para $i = 1, \ldots, n$, donde $x_i = (x_{i1}, \ldots, x_{ip})$. El algoritmo necesita automáticamente decidir qué atributos y qué valores de tales atributos utilizará en cada nodo (split), así como la topología (forma) del árbol.

Supongamos que tenemos una partición del dominio en M regiones $R_1, \ldots, R_M$, y modelamos la respuesta con una constante c_m en cada región:

$$f(X) = \sum_{m=1}^{M} c_m I\{(X_1, X_2) \in R_m\}$$

Si el problema es de regresión, adoptando como criterio de minimización la suma de los errores cuadráticos $\sum(y_i - f(x_i))^2$, es sencillo ver que el mejor $\hat{c}_m$ es el promedio de y_i en la región R_m:

$$\hat{c}_m = avg(y_i \mid x_i \in R_m)$$

En problemas de clasificación, puede usarse la etiqueta más frecuente (moda) de y_i en la región R_m.

Para encontrar la mejor partición binaria se utiliza un algoritmo greedy, buscando el atributo j y el valor de división s que resuelve:

$$\min_{j,s}[\min_{c_1} \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2]$$

Donde $R_1(j, s) = \{X|X_j \leq s\}$ y $R_2(j, s) = \{X|X_j > s\}$.

En problemas de clasificación, se utilizan otras métricas para decidir cuál es el mejor valor de corte (split), intentando incentivar la homogeneidad de la variable objetivo en los subconjuntos producidos. Dos métricas muy populares son *Gini Impurity* e *Information Gain* [Raileanu and Stoffel, 2004].

Para cada atributo, determinar el mejor punto de división s puede hacerse muy rápidamente y por lo tanto, escaneando todos los atributos, se puede hallar el mejor par (j, s).

Habiendo encontrado la mejor división, se particionan los datos en las dos regiones resultantes y se repite recursivamente el proceso de división en cada una de las dos regiones. Una cuestión a considerar es cuándo detener este proceso, pues árboles muy grandes pueden sobreajustar los datos, en tanto que árboles muy pequeños pueden no capturar la estructura del problema. La estrategia preferida es construir un árbol grande, deteniendo el proceso de división cuando una cantidad mínima de de nodos es alcanzada. Posteriormente, este árbol es *podado*, reduciendo su tamaño.

Otras consideraciones sobre árboles de decisión

- El algoritmo previamente descripto, CART, es uno de los métodos más populares para construir árboles de decisión. Otras opciones ampliamente utilizadas incluyen ID3 [Quinlan, 1986], C4.5 [Quinlan, 1993] y C5.0 [Kuhn and Johnson, 2013].
- Los árboles de decisión son un ejemplo de un modelo que tiene bajo sesgo, pues no hacen casi ninguna asunción acerca de la función objetivo. Sin embargo, los árboles de decisión complicados (profundos) son modelos con alta varianza, pues son altamente susceptibles a variaciones en los datos de entrenamiento y tienden a sobreajustar. Técnicas de podado (prunning) pueden ser utilizadas para reducir la complejidad de los árboles y evitar sobreajustar los datos de entrenamiento.

1.3.7. Random Forests

En esta sección se introducirá uno de los dos métodos de clasificación utilizados en este trabajo: Random Forests [Breiman, 2001], y está basada enteramente en [Hastie et al., 2001].

Métodos bootstrap

Bootstrap es una técnica general para analizar estadísticas sobre una población. Supongamos que tenemos un modelo de aprendizaje automatizado, entrenado con un cierto conjunto de

datos $Z = (z_1, \ldots, z_N)$ donde $z_i = (x_i, y_i)$. La idea básica es muestrear datasets aleatoriamente con reemplazo a partir de Z, cada uno con el mismo tamaño que Z.

Este procedimiento se repite B veces, produciendo B datasets bootstrap. Cada dataset bootstrap construido se utilizará como conjunto de entrenamiento para entrenar un nuevo modelo de aprendizaje automatizado. Finalmente, se examina el comportamiento de los distintos modelos entrenados sobre las B réplicas, tal y como se muestra en la figura **1-8**.

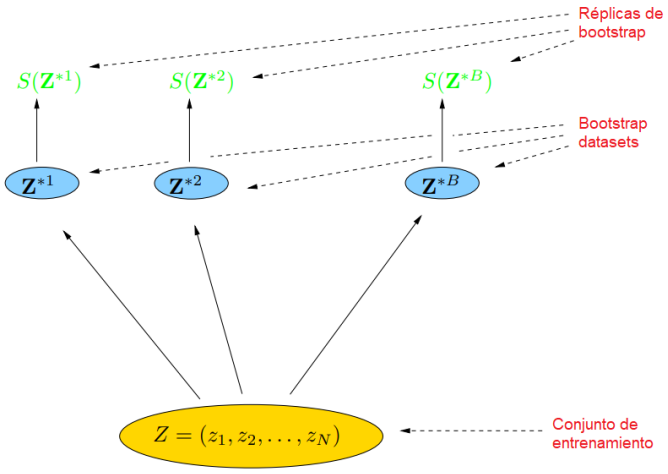


Figura 1-8.: Semántica del proceso bootstrap. Se desea estimar la precisión estadística de una métrica $S(Z)$ computada sobre un dataset Z. B datasets de entrenamiento Z^{*b}, cada uno con tamaño N, son muestreados con reemplazo a partir del dataset original Z. La métrica de interés $S(Z)$ se computa sobre cada dataset bootstrap, y los valores $S(Z^{*1}), \ldots, S(Z^{*N})$ pueden usarse para estimar la precisión estadística de $S(Z)$. Fuente: [Hastie et al., 2001]

En la figura, $S(Z)$ es algún tipo de métrica computada a partir de los datos Z. Una posibilidad sería que $S(Z)$ fuese el error de predicción sobre Z. Sin embargo, esto sería metodológicamente incorrecto pues muchos de los elementos del conjunto de entrenamiento estarían actuando como elementos del conjunto de testeo. Esta superposición puede hacer que los resultados sean incorrectamente optimistas.

Emulando la idea aplicada en Cross Validation (1.3.2), una mejor estimación sería únicamente considerar predicciones obtenidas con bootstraps que no contienen esa observación. Las métricas obtenida nos permiten estimar el error en test de un cierto clasificador.

Bagging

Se presentó boostrap como una forma de estimar el error en test de un modelo. En esta subsección, se mostrará cómo utilizar bootstrap para mejorar las predicciones en sí mismas.

Consideremos en primera instancia un problema de regresion. Supongamos que se cuenta con un conjunto de entrenamiento $Z = (z_1, \ldots, z_N)$ donde $z_i = (x_i, y_i)$, y se calculan B boostrap datasets a partir de él.

Bootstrap aggregation o **bagging** [Breiman, 1996] consiste en entrenar un modelo con cada bootstrap dataset Z^{*b}, cada uno de los cuales produce una predicción $\hat{f}^{*b}(x)$ para un dado input x. Posteriormente, se utiliza el promedio de todas las predicciones como predicción final, por lo tanto reduciendo la varianza del resultado:

$$\hat{f}_{bag} = \frac{1}{B} \sum_{b=1}^{B} \hat{f}^{*b}(x)$$

Por otro lado, en un problema de clasificación, puede utilizarse la moda de las predicciones asociadas a cada bootstrap como predicción final.

Bagging tiene la capacidad de reducir dramáticamente la varianza de predictores inestables como árboles sin introducir sesgo, lo cual conduce a mejores resultados. Cada árbol entrenado con un bootstrap dataset diferente tenderá a involucrar diferentes atributos, y a tener una topología diferente, tal y como se puede ver en la figura **1-9**.

Random Forest

La idea esencial de bagging es promediar muchos modelos ruidosos que tienen bajo sesgo, reduciendo la varianza. Los árboles de decisión son candidatos ideales para bagging, dado que pueden capturar interacciones de estructuras complejas en los datos, y tienen un sesgo relativamente bajo si se les permite tener profundidad suficiente.

- Dado que cada árbol generado en bagging está idénticamente distribuido, la esperanza del promedio de B árboles es igual a la esperanza de cualquiera de ellos. Esto implica que el sesgo de bagging con árboles de decisión es el mismo que el de árboles de decisión individuales.
- Dadas B variables aleatorias idénticamente distribuidas, no necesariamente independientes y con correlación a pares positiva ρ, la varianza del promedio es:

  $$\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$$

  Para B suficientemente grande, el segundo término desaparece, y sigue que el tamaño de la correlación a pares de los árboles calculados sobre cada bootstrap limita los beneficios del promedio.

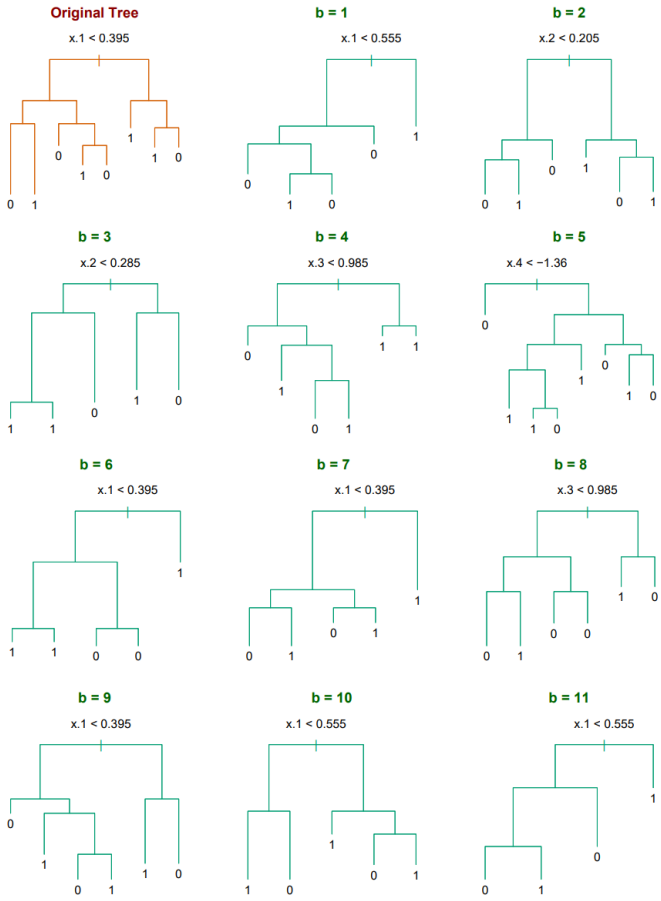


Figura 1-9.: En esta imagen puede verse un árbol de decisión, en color naranja, construido a partir de un cierto dataset. Los restantes 11 árboles fueron construidos a partir de distintos bootstraps del dataset en cuestión. Nótese que la topología de los árboles y los atributos utilizadas en cada nodo son notablemente distintas para cada b. Fuente: [Hastie et al., 2001]

Random forest [Breiman, 2001], en adelante RF, es una modificación sustancial de bagging que intenta mejorar la reducción de varianza obtenida por bagging reduciendo la correlación entre los árboles. Un RF formado por B árboles de decisión, se construye de la siguiente forma:

1. Para b=1 hasta B:
   a) Muestrear un bootstrap Z^{*b} de tamaño $|Z|$ a partir de los datos de entrenamiento Z.
   b) Entrenar un árbol de decisión T_b utilizando Z^{*b} como dataset de entrenamiento, repitiendo los siguientes pasos para cada nodo terminal del árbol, hasta que una

cantidad mínima de nodos n_{min} se haya alcanzado:

1) Seleccionar m atributos aleatoriamente de entre los p atributos.
2) Elegir el mejor atributo y el mejor punto de corte (split) de entre los m atributos seleccionados.
3) Dividir el nodo en dos nodos hijos.

2. Retornar el ensamble de arboles $\{T_b\}_1^B$

Para realizar una predicción de un nuevo punto, utilizamos el promedio de la predicción de todos los árboles (regresión) o la moda de las predicciones (clasificación).

La reducción de correlación entre los árboles sucede en la fase de entrenamiento, al seleccionar $m \leq p$ de los atributos aleatoriamente como candidatos para cada nodo. Intuitivamente, valores pequeños de m reducirán la correlación entre cada par de árboles del ensamble, y por lo tanto, reducirán la varianza en promedio.

Una característica importante de RF es el potencial uso de elementos OOB (out-of-bag). Para cada observación $z_i \in Z$, es posible construir una predicción RF promediando sólo las predicciones de aquellos árboles correspondientes a bootstraps Z^{*b} en los cuales z_i no fue muestreado. Esto implica que podemos calcular una estimación de cuán bien el predictor funcionará en instancias no antes vistas, con un razonamiento similar al utilizado en Cross Validation.

RF es un método muy flexible y poderoso, y es cotidianamente utilizado en aplicaciones industriales. Una de sus principales ventajas es su facilidad de uso, dado que no se requiere una optimización de hiperparámetros muy compleja y brinda generalmente buenos resultados en datasets arbitrarios.

1.3.8. Support Vector Machines

En esta sección se presentará el segundo método de aprendizaje automatizado supervisado para clasificación binaria que se utilizará en este trabajo: Support Vector Machines (SVM) [Cortes and Vapnik, 1995] [Boser et al., 1996]

Una tarea de clasificación binaria puede ser vista como la tarea de separar las clases en el espacio de atributos. En la figura **1-10** podemos ver un ejemplo en el cual se decide utilizar un hiperplano para realizar dicha separación. Como se puede ver hay infinitos separadores lineales para este ejemplo.

Dado un conjunto de entrenamiento $\{(x_i, y_i),\ i = 1, \ldots, N,\ x_i \in \mathbb{R}^p, y_i \in \{1, -1\}\}$, un hiperplano se define como:

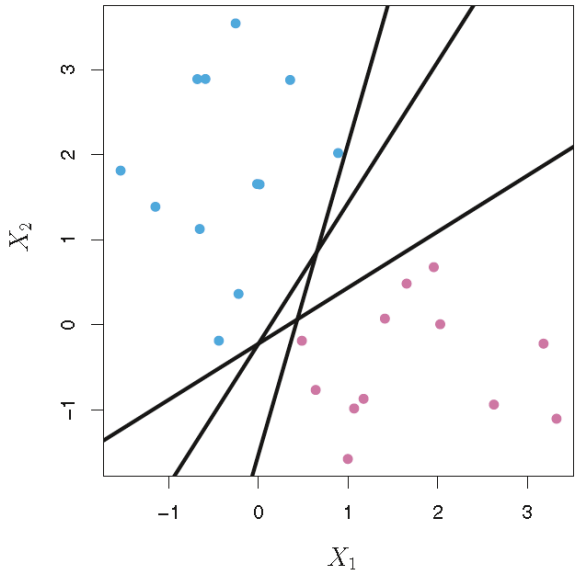


Figura 1-10.: En esta imagen podemos ver un ejemplo de clasificación binaria utilizando un separador lineal. Se ilustra la existencia de múltiples separadores lineales que podrían ser utilizados. Créditos: Drew Wilimitis, `towardsdatascience.com`.

$$\{x : f(x) = x^T\beta + \beta_0 = 0\}$$

Donde β es un vector unitario. Una regla de clasificación inducida por $f(x)$ es:

$$G(x) = sign[x^T\beta + \beta_0]$$

La función $f(x_0)$ es la distancia de un punto x_0 al hiperplano $x^T\beta + \beta_0 = 0$. Si las clases son separables, es posible encontrar β y β_0 tal que $y_i f(x_i) > 0\ \forall i$. Las SVM buscan elegir el hiperplano que separe ambas clases maximizando el margen entre ambas clases, donde el margen se define como la mínima distancia entre un punto de training y el hiperplano separador. Esta situación se ilustra en la figura **1-11**.

El siguiente problema de optimización captura este concepto [Hastie et al., 2001]:

$$\begin{aligned} &\min_{\beta,\beta_0} \|\beta\| \\ \text{sujeto a} \quad &y_i(x_i^T\beta + \beta_0) \geq 1,\ i = 1, \ldots, N \end{aligned}$$

Un primer problema a considerar es que las clases pueden estar superpuestas en el espacio de atributos. Una forma de lidiar con esta superposición consiste en maximizar el margen, pero permitir que algunos puntos estén en el lado equivocado, como se ilustra en la figura **1-12**. Con este propósito, se definen variables slack $\xi = (\xi_1, \ldots, \xi_N)$, y se modifica el problema de optimización como sigue:

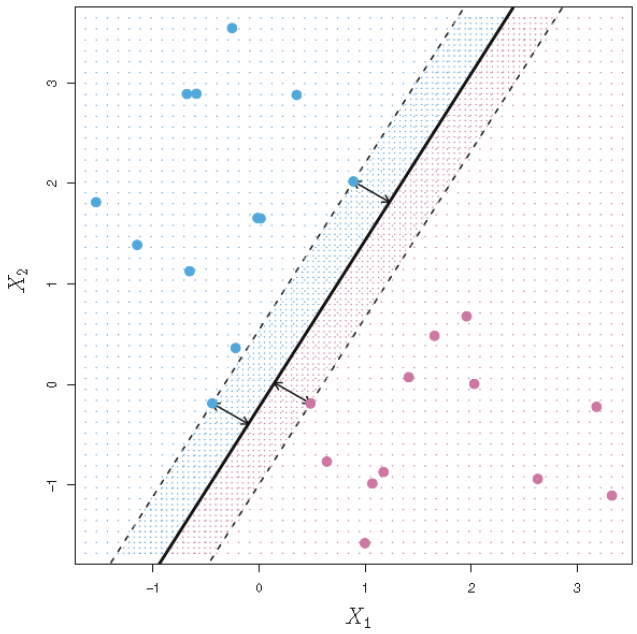


Figura 1-11.: En esta imagen se ilustra el hiperplano escogido por SVM. Aquellos puntos que están a la menor distancia del hiperplano separador se denominan vectores soporte. SVM escoge el hiperplano que maximiza la separación entre ambas clases. Créditos: Drew Wilimitis, `towardsdatascience.com`

$$\begin{aligned} &\min_{\beta,\beta_0} \|\beta\| \\ \text{sujeto a} \quad &y_i(x_i^T\beta + \beta_0) \geq 1 - \xi_i, \quad i = 1, \ldots, N \\ &\xi_i \geq 0 \quad i = 1, \ldots, N \\ &\sum_{i=0}^{N} \xi_i \leq constant. \end{aligned}$$

Computacionalmente es conveniente re-expresar este problema como sigue:

$$\begin{aligned} &\min_{\beta,\beta_0} \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{N} \xi_i \\ \text{sujeto a} \quad &y_i(x_i^T\beta + \beta_0) \geq 1 - \xi_i, \quad i = 1, \ldots, N \\ &\xi_i \geq 0 \quad i = 1, \ldots, N \end{aligned}$$

Donde el parámetro de costo (o regularización) C reemplaza la restricción $\sum_i \xi_i \leq constant$. Los detalles matemáticos de la resolución de este problema de optimización utilizando multiplicadores de Lagrange pueden ser consultados en [Hastie et al., 2001].

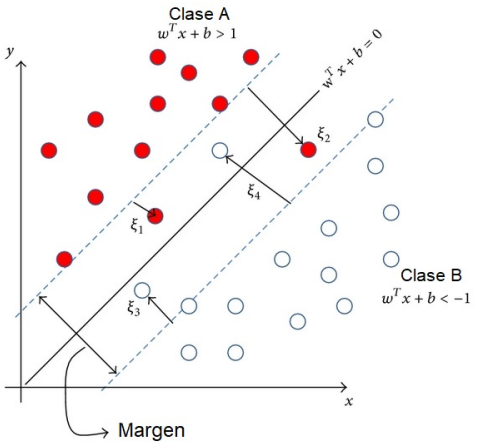


Figura 1-12.: Utilización de variables slack que permiten a algunos puntos de entrenamiento estar dentro del margen, e incluso del lado equivocado del hiperplano. El parámetro C permite regular cuán permisivo es el modelo a la hora de permitir errores de clasificación. Imagen tomada de *Hybrid Model Based on Genetic Algorithms and SVM Applied to Variable Selection within Fruit Juice Classification*, Carlos Fernandez-Lozano et al.

El truco del kernel

El algoritmo descripto tiene una importante limitación: sólo es capaz de encontrar fronteras lineales en el espacio de atributos. Esto puede solucionarse mediante el uso de una función ϕ que mapea los datos a algún espacio de mayor dimensionalidad donde, potencialmente, las clases serán separables por una frontera lineal. Esta situación es ilustrada en la figura **1-13**

$$\begin{aligned} &\min_{\beta,\beta_0} \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{N}\xi_i \\ \text{sujeto a} \quad &y_i(\phi(x_i)^T\beta + \beta_0) \geq 1 - \xi_i, i = 1, \ldots, N \\ &\xi_i \geq 0 \ \forall i \end{aligned}$$

La solución a este nuevo problema de optimización toma la forma:

$$\hat{f}(x) = \sum_{i=1}^{N} \hat{\alpha}_i y_i K(x, x_i) + \hat{\beta}_0$$

Donde $K(a, b) = \langle\phi(a), \phi(b)\rangle$. Nuevamente, el lector puede consultar los detalles matemáticos en [Hastie et al., 2001]. Un punto clave es que la solución sólo involucra productos internos en el espacio de mayor dimensionalidad, por lo que nunca es necesario calcular el mapeo explícito de alguna instancia x a $\phi(x)$. Todo lo que se requiere es definir una **función kernel** K que

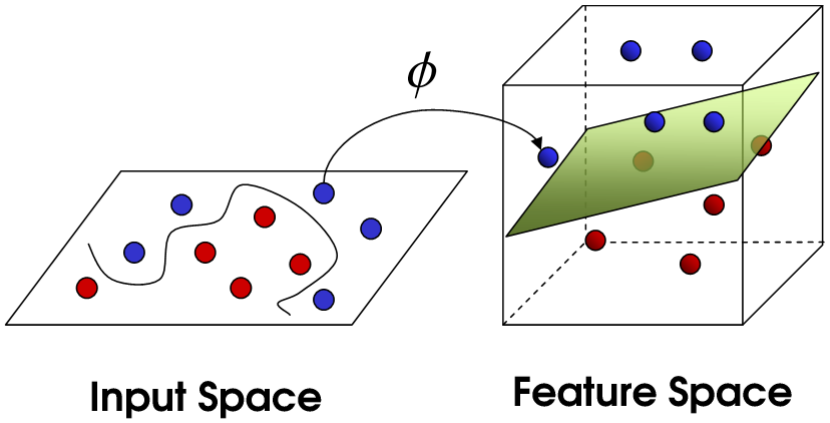


Figura 1-13.: Utilización de una función kernel para mapear puntos que no son linealmente separables a un espacio de mayor dimensionalidad, donde pueden ser separados por un hiperplano. Créditos: Drew Wilimitis, `towardsdatascience.com`

calcule el producto interno en el espacio de mayor dimensionalidad. Las tres elecciones más populares de kernel son:

- **Lineal**: $K(x_i, x_j) = x_i^T x_j$
- **Polinomial**: $K(x_i, x_j) = (\gamma x_i^T x_j + r)^d,\ \gamma > 0$
- **Radial Basis Function** (RBF)[6]: $K(x_i, x_j) = exp(-\gamma \left\| x_i - x_j \right\|^2),\ \gamma > 0.$

Donde γ y d son parámetros del kernel.

Función de pérdida y regularización

El problema de optimización presentado en la sección anterior:

$$\min_{\beta,\beta_0} \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{N}\xi_i$$
$$\text{sujeto a} \quad y_i(\phi(x_i)^T\beta + \beta_0) \geq 1 - \xi_i, i = 1, \ldots, N$$
$$\xi_i \geq 0\ \forall i$$

Puede reformularse como sigue [Hastie et al., 2001] [Lopez-Martinez, 2017]:

[6]El kernel RBF, también llamado Gaussiano, puede reescribirse como $K(x_i, x_j) = \sum_{n=0}^{+\infty} \frac{(x_i.x_j)^n}{\sigma^n.n!}$. Es decir, el kernel RBF es una combinación de todos los kernels polinomiales de grado $n \geq 0$. Dado que n puede ser arbitrariamente grande, decimos que el kernel Gaussiano trabaja con vectores en infinitas dimensiones.

$$\min_{\beta,\beta_0} \sum_{i=0}^{n} [1 - y_i(\beta_0 + x_i^T \beta)]_+ \frac{\lambda}{2} \|\beta\|^2$$

Donde el primer término de la suma es el término de pérdida (loss) y el segundo término es el término de regularización. λ es un parámetro de regularización, relacionado a $\frac{1}{C}$ en la formulación inicial; y la función $(1-a)_+ = \text{máx}(0, 1-a)$ es llamada función de pérdida de bisagra (del inglés, hinge loss function).

El término de perdida mide cuán bien el modelo aproximará las etiquetas del conjunto de entrenamiento. Es posible utilizar una función de pérdida distinta, por ejemplo el cuadrado de la función de pérdida de bisagra. Otras funciones de pérdida pueden apreciarse en la figura **1-14**. En este trabajo se usará en todos los experimentos el cuadrado de la función de bisagra como penalización.

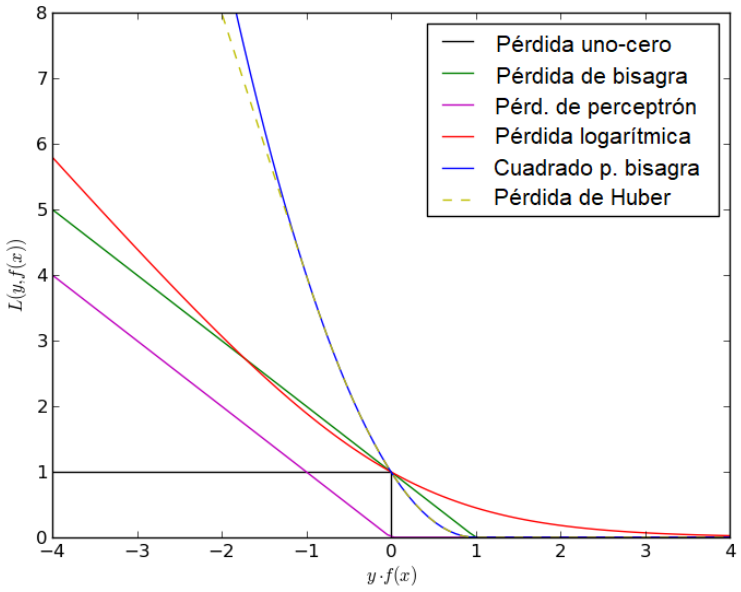


Figura 1-14.: Gráfico de distintas funciones de pérdida. Créditos: Cagdas Ozgenc

Por otro lado, el término de regularización impone algún tipo de penalidad sobre la complejidad del modelo. Es posible utilizar distintos tipos de norma sobre los coeficientes β en el término de regularización (e.g. L_1, L_2). Tanto L_1 como L_2 buscan mantener los coeficientes β cercanos a cero, aunque presentan ciertas diferencias [Lopez-Martinez, 2017]:

- L_1 tiene la capacidad de llevar algunos de los coeficientes β_i al valor 0 para λ suficientemente grande, promoviendo soluciones β dispersas y realizando selección de atributos (ver sección 5.1.1) de forma implícita. Una desventaja es que L_1 tiende a descartar

atributos correlacionados, incluso cuando todos son relevantes para el problema a tratar. En otras palabras, L_1 falla en identificar atributos que son relevantes de forma conjunta.

- Contrariamente, L_2 no puede producir coeficientes β dispersos, y por lo tanto no puede realizar selección de atributos de forma automática. Esto implica que atributos correlacionados que no deberían ser ignorados por el modelo no serán ignorados.

En este trabajo se utilizará la penalidad $L2$ en todos los experimentos.

Finalmente, cabe mencionar que SVM es uno de los métodos de predicción con mayor sustento teórico, dado que se basa en aprendizaje estadístico o teoría VC (Vapnik-Chervonenkis). [Vapnik and Chervonenkis, 1971] [Vapnik and Chervonenkis, 1974].

1.4. Antecedentes relevantes a esta tesina

Como se adelantó anteriormente, en esta tesina se estudiará la aplicación de algoritmos de aprendizaje automatizado al problema de clasificar estrellas variables de tipo RRL en los datasets publicados por Carpyncho, correspondientes al relevamiento VVV.

Previamente, se ha explorado el desempeño de clasificadores automatizados de estrellas variables en varios estudios [Debosscher et al., 2007], [Dubath et al., 2011], [Armstrong et al., 2015], [Paegert et al., 2014]. Los dos antecedentes más relevantes de esta tesina, [Elorrieta López et al., 2016] y [Cabral et al., 2020a], proponen procedimientos automatizados para clasificar RRL de subtipo ab en la VVV, basándose en sus curvas de luz.

En [Elorrieta López et al., 2016] se concluye que AdaBoost [Freund and Schapire, 1997], un método basado en ensambles de árboles, obtiene consistentemente el mejor desempeño de entre una amplia selección de algoritmos de aprendizaje automatizado incluyendo SVM y redes neuronales profundas. El desempeño es estimado utilizando cross-validation, y a través de la comparación entre dos datasets que fueron clasificados por expertos humanos. Nótese que varias de las métricas utilizadas en este trabajo, como AUC o F_1, no son adecuadas para trabajar con datasets altamente desbalanceados.

Posteriormente, en [Cabral et al., 2020a], se refinaron las métricas de desempeño y se extiende el conjunto de atributos utilizado para describir estrellas variables, incorporando atributos de pseudocolor. En uno de los experimentos, se realizó una comparación entre el desempeño de distintos métodos de clasificación, incluyendo Random Forests y SVM. El mayor desempeño se obtuvo, nuevamente, utilizando ensambles de árboles.

1.5. Comparación entre Random Forest y Support Vector Machines

Como ya se ha mencionado, Random Forest y SVM son dos potentes métodos de aprendizaje automatizado, que pueden ser aplicados tanto a problemas de regresión como de clasificación. En esta sección, se hará hincapié en ciertas características que distinguen ambos métodos, buscando comparar y contrastar la forma en que ambos operan. El objetivo es enumerar razones que puedan explicar por qué Random Forest parece funcionar mejor que SVM en estudios previos de clasificación de RRL.

En primer lugar, se debe notar que ambos métodos proceden de formas radicalmente distintas. En la fase de training, los árboles de decisión que conforman un random forest observan cada atributo uno a la vez, analizando si es beneficioso realizar divisiones para ciertos valores [Mitchell, 1997]. Por otro lado, las SVM consideran cada elemento del dataset como un punto en el espacio, utilizando los valores de todos los atributos en simultáneo para calcular un hiperplano separador [Cortes and Vapnik, 1995].

Una primera consecuencia es que RF es inmune a **diferentes escalas** en los atributos [Hastie et al., 2001]. Los datos no necesitan estar normalizados ni centrados para que RF funcione, mientras que la naturaleza geométrica de SVM implica que los distintos atributos necesariamente han de estar expresados en la misma escala [Hsu et al., 2003]. Más aún, las rutinas de optimización que usualmente implementan la fase de training en SVM pueden fallar en converger, o tardar significativamente más tiempo, si los datos no se encuentran escalados a [0,1] o [-1,1] [Hsu et al., 2003]. Por lo tanto, resulta vital realizar un preprocesamiento adecuado a los datos a la hora de aplicar SVM.

Una segunda consecuencia es que RF puede **ignorar atributos que no son muy informativos** fácilmente, pues los árboles de decisión tienen un mecanismo simple para darse cuenta de que no sirven (ningún split será un buen candidato) [Hastie et al., 2001]. De acuerdo a [Hastie et al., 2001], los árboles de decisión "realizan una selección de atributos internamente como una parte integral en su forma de proceder". Ignorar atributos no informativos puede ser más complejo para SVM, pues la rutina de optimización tendría que asignar un valor nulo a los coeficientes correspondientes en w. La elección del tipo de penalidad (ver sección 1.3.8) es importante en este sentido. Un cierto atributo puede ser poco informativo por distintos motivos: puede haber sido incluido en el dataset pero no contener información relevante para predecir la variable target, puede ser extremadamente ruidoso, etc. Las técnicas de selección de atributos (Feature Selection) que se aplicarán en capítulos posteriores son una forma adecuada de deshacerse de atributos no informativos, y podrían reducir el impacto de este problema en SVM.

Resulta interesante explorar la potencial presencia de **atributos altamente correlacionados**. RF puede verse negativamente afectado por este fenómeno, pues la información contenida en dos atributos altamente correlacionados tendrá mayor probabilidad de ser utilizada para realizar una división en un nodo. Por otro lado, el impacto negativo puede verse disminuido por la forma en que los árboles de decisión son construidos. El algoritmo comprenderá que, si ya se realizó un split utilizando un cierto atributo, realizar el mismo split utilizando un atributo altamente correlacionado probablemente no ayudará a separar las clases objetivo. Resulta interesante explorar si las SVM se ven perjudicadas por la presencia de atributos correlacionados, y analizar el impacto de eliminar tales redundancias [Kuperman et al., 2016]. La elección del tipo de penalidad (ver sección 1.3.8) es un factor a considerar en este sentido

Otro punto a considerar es cómo manejar el **overfitting** (sobreajuste) durante la fase de training. RF requiere la estimación de **hiperparámetros** para manejar apropiadamente el sobreajuste [Louppe, 2015], tal como el número máximo de atributos a considerar en cada nodo. Por otro lado, SVM requiere una cuidadosa estimación del hiperparámetro de regularización C, así como la elección de algún tipo de penalidad (1.3.8). Más aún, si se utiliza un kernel con parámetros adicionales como RBF, se requiere es imar cuidadosamente otros hiperparámetros [Hsu et al., 2003]. Contrariamente, Random Forest es considerado un método off-the-shelf [Wyner et al., 2015], pues suele producir buenos resultados sin una optimización de hiperparámetros tan meticulosa. Por lo tanto, resulta de vital importancia una cuidadosa optimización de hiperparámetros de regularización y de kernel en SVM.

El marcado **imbalance de clases** presente en los datos, con aproximadamente 2000 elementos de clase negativa (no-RRL) por cada elemento de clase positiva (RRL), puede afectar negativamente el desempeño de RF y SVM. En [Cabral et al., 2020a], se determinó que corregir el imbalance de clases de estos datos mediante undersampling conduce a una pérdida de desempeño en Random Forest, siendo conveniente entrenar los modelos utilizando los datos sin balancear. Por otro lado, es posible que esto no sea cierto para SVM, dado que el imbalance de clases puede ocasionar pérdida de performance pues el hiperplano soporte elegido estará más alejado de la clase positiva, y habrá mayor cantidad de vectores soporte de la clase negativa [Akbani et al., 2004]. Resulta por lo tanto interesante explorar diversas técnicas de undersampling y oversampling a la hora de aplicar SVM.

La existencia de **atributos ruidosos y outliers** en los datos es de esperarse en datasets astronómicos, que pueden incluir errores de medición y observación. RF no se ve afectado por outliers, pues los árboles de decisión funcionan creando cubetas (bins) que no se ven influenciadas por valores extremos [Hastie et al., 2001]. Atributos que son altamente ruidosos no serán informativos a la hora de dividir la clase objetivo, y podrán ser ignorados por los árboles de decisión. Es interesante explorar distintos preprocesamientos tendientes

a limpiar o suavizar los datos a la hora de aplicar SVM, pues puede ayudar a mitigar el efecto de outliers y ruido en los datos. Nótese que la función de pérdida utilizada por SVM (ver sección 1.3.8) es menos sensible a outliers que otro tipo de funciones de pérdida, y la inclusión de regularización tratará de minimizar el valor de los coeficientes de SV, reduciendo el sobreajuste a outliers en los datos.

La siguiente tabla resume los puntos discutidos en esta sección. Durante el resto de este trabajo, se explorará cada uno de ellos.

Aspecto	**Capítulo Relevante**
Estimación de hiperparámetros óptimos Overfitting	Capítulos 2, 3.
Diferentes escalas en los datos	Capítulo 4: Preprocesamientos
Atributos ruidosos Existencia de outliers	Capítulo 4: Preprocesamientos
Atributos no informativas	Capítulo 5: Feature selection y feature extraction
Atributos altamente correlacionados	Capitulo 6: Inspeccionando los datos
Imbalance de clases	Capítulo 7: Imbalance de clases

Finalmente, en los capítulos 8 y 9 se ponderarán distintas teorías en base a los resultados obtenidos, y se extraerán conclusiones.

2. Clasificación de estrellas RRL utilizando Random Forest

El objetivo de esta sección es determinar la performance de clasificadores Random Forest aplicados a la tarea de clasificar estrellas variables de tipo RRL. Estos resultados serán utilizados como punto de referencia en secciones posteriores, donde se trabajará con clasificadores basados en Support Vector Machines sobre los mismos datos. La implementacion de Random Forest utilizada es la provista por Scikit-Learn [Buitinck et al., 2013] [Pedregosa et al., 2011].

2.1. Optimización de hiperparámetros

Se realizó un experimento utilizando una metodología muy similar a la descripta en la sección "Model Selection"de [Cabral et al., 2020a]. En primer lugar, se optimizaron los hiperparámetros de Random Forest utilizando 10-fold Cross Validation Grid Search sobre el tile b278. Los hiperaparámetros a optimizar fueron:

- $n\_estimators$: El número de árboles en cada ensamble. Se exploró un amplio rango de valores entre 10 y 750 árboles.
- max_AUPRCs: El número de atributos a considerar cuando se busca el mejor split. Las opciones exploradas fueron la raíz cuadrada del número de atributos ("sqrt"), y el logaritmo en base dos del número de atributos ("log2").
- *criterion*: La función utilizada para medir la calidad de cada split. Las opciones consideradas fueron Gini Impurity ("gini") e Information Gain ("entropy")

La medida de performance utilizada fue el área bajo la curva de precision-recall. Los resultados completos pueden observarse en la figura **2-1**. Las siguientes conclusiones se desprenden de este experimento:

- Utilizar entropy como criterion es consistentemente más beneficioso que utilizar gini.
- A partir de aproximadamente 200 árboles, la ganancia de agregar más árboles al ensamble es muy pequeña. A partir de aproximadamente 400 árboles, la ganancia de agregar más árboles parece ser nula.

- Los hiperparámetros óptimos resultaron ser: *criterion*="entropy", *n_estimators*=400 y *max_features*="sqrt"

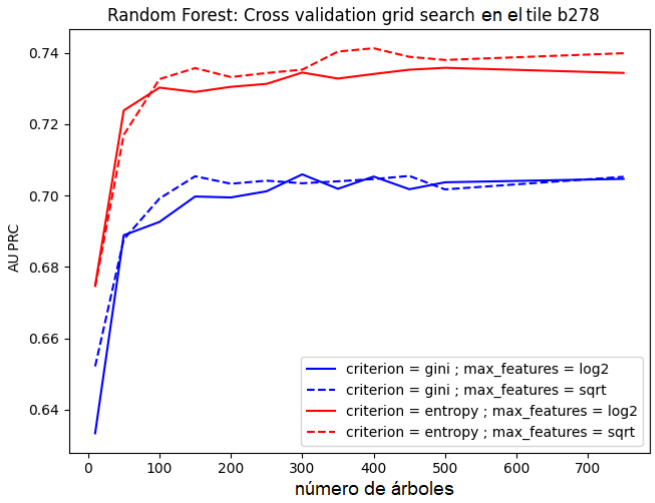


Figura 2-1.: Resultados de la optimización de hiperparámetros de Random Forest en el tile b278

2.2. Análisis de performance de Random Forest

Para estimar la performance de Random Forest utilizando los parámetros óptimos que se obtuvieron en la sección anterior, se utilizará un subconjunto fijo de tiles: { b234, b261 , b278, b360 }. Este subconjunto de tiles fue escogido como un compromiso entre la densidad y cobertura de estrellas de tipo RRL [Cabral et al., 2020a]. Para cada par de tiles t_1 y t_2, se entrenará un clasificador Random Forest utilizando t_1 como dataset de training y t_2 como dataset de test, obteniendo curvas de precision-recall.

Las curvas resultantes pueden observarse en la figura **3-3**, en color azul. Los valores obtenidos son consistentes con aquellos reportados en trabajos previos [Cabral et al., 2020a]. Pueden observarse diversos comportamientos, dependiendo de los datos de training y testing seleccionados. En la mayoría de los casos, parece ser posible fijar un recall de 0.5 y obtener una precisión igual o mayor a 0.5. Si se desea una mayor precisión, por ejemplo 0.9; parece ser posible fijar un recall de 0.2 para obtenerla. Esto significa que, utilizando random forests, sería posible detectar automáticamente y con alta precisión al menos el 20 % de las RLL mapeadas por el VVV.

3. Clasificación de estrellas RRL utilizando Support Vector Machines

En este capítulo se repitieron los experimentos descriptos en el capítulo anterior, utilizando SVM en vez de RF. Esto permitirá realizar una primera comparación entre el desempeño de ambos métodos.

En trabajos previos se ha reportado que la performance de SVM es inferior a la de métodos basados en ensembles de árboles a la hora de clasificar estrellas variables de tipo RRL [Cabral et al., 2020a] [Elorrieta López et al., 2016]. El objetivo de este capítulo es reproducir estos resultados, obteniendo una base de referencia de performance (baseline) para SVM que intentaremos mejorar y entender en capítulos posteriores.

Nuevamente se utilizaron implementaciones de SVM provistas por SciKit Learn [Buitinck et al., 2013] [Pedregosa et al., 2011]. Para todos los experimentos de esta sección, se aplicó un escalado estándar a los tiles de training y testing antes de utilizarlos (restando la media en training de cada atributo y dividiendo por la varianza en training).

3.1. Optimización de hiperparámetros para SVM Lineal

En primera instancia, se utilizó SVM con kernel lineal. Se utilizó la implementacion **LinearSVC** de SciKit Learn, la cuál está implementada en terminos de liblinear [Fan et al., 2008]. Esta implementación se caracteriza por escalar casi linealmente a datasets con millones de elementos, lo cual nos permite entrenar con tiles completas.

Para optimizar el parámetro de regularización C, se realizó 10-fold Cross Validation Grid Search sobre el tile b278. Se exploró un amplio rango de valores para C, utilizando una escala logarítmica. La medida de performance elegida fue, nuevamente, el area bajo la curva de precision-recall. Adicionalmente se estudió la precisión a valores fijos de recall. En la figura **3-1** pueden apreciarse los resultados completos de este experimento.

El valor óptimo encontrado fue C=0.1. Valores pequeños de C conducen a modelos muy pobres, lo cuál indica que modelos donde el margen elegido por SVM es muy grande (permitiendo demasiadas clasificaciones incorrectas) no generan buenos resultados. Por otro lado,

incrementar el valor de C más allá de 0.1 tampoco produce mejoras en performance, generando modelos que son innecesariamente más complejos.

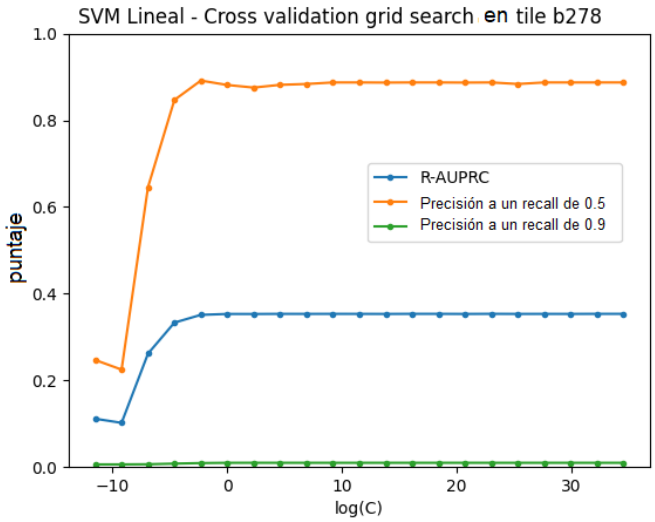


Figura 3-1.: Resultados de la optimización de hiperparámetros de SVM-Lineal en el tile b278

3.2. Aproximando SVM Kernel

En la sección 1.3.8 se introdujo el truco del kernel, que consiste en mapear puntos desde un espacio de menor dimensionalidad a un espacio de mayor dimensionalidad. Un punto clave es que, en realidad, únicamente se computan los productos internos en el espacio de mayor dimensionalidad, y nunca es necesario mapear los datos realmente a este espacio.

Sin embargo, una de las características de los algoritmos que utilizan este tipo de truco, como SVM kernel, es que su complejidad temporal escala con el número de instancias de entrenamiento ($\mathcal{O}(n^3)$), en tanto que la complejidad espacial será cercana a $\mathcal{O}(n^2)$. Esto vuelve prohibitivo utilizar SVM kernel en los tiles de Carpyncho, que tienen cientos de miles de instancias cada uno.

En resumidas cuentas, SVM kernel funciona muy bien en datos que no son linealmente separables, pero no escala bien a datasets con muchas instancias. Por otro lado, como se mencionó en la sección anterior, existen implementaciones de SVM Lineal que escalan de forma lineal a la cantidad de instancias de entrenamiento.

Una tendencia reciente es combinar ambos, dejando de lado la idea fundamental del truco del kernel. La idea base es mapear los datos literalmente a un espacio de mayor dimensionalidad y luego aplicar un clasificador lineal, el cuál generará fronteras de decisión no lineales en el espacio original.

Dado que algunos kernels, como RBF, mapean los datos a un espacio de infinitas dimensiones; se busca mapear los datos a un espacio con una cantidad finita razonablemente grande de dimensiones. Afortunadamente, se sabe que únicamente un subespacio finito de este espacio con infinitas dimensiones es necesario para resolver el problema de SVM: el espacio inducido por las imágenes de los datos de entrenamiento.

Sin embargo, mapear a este espacio inducido sería al menos tan costoso como SVM kernel exacto. Una solución aproximada es utilizar solo un subconjunto de $m < n$ puntos de entrenamiento para generar un mapeo a $\mathbb{R}^m$. Este algoritmo es llamado Método de Nystroem, y permite obtener un mapeo aproximado, pudiéndose controlar la complejidad asociada ajustando el parámetro m.

Los detalles del procedimiento recien descripto pueden consultarse en las siguientes publicaciones: [Williams and Seeger, 2001], [Yang et al., 2012], [Rahimi and Recht, 2008].

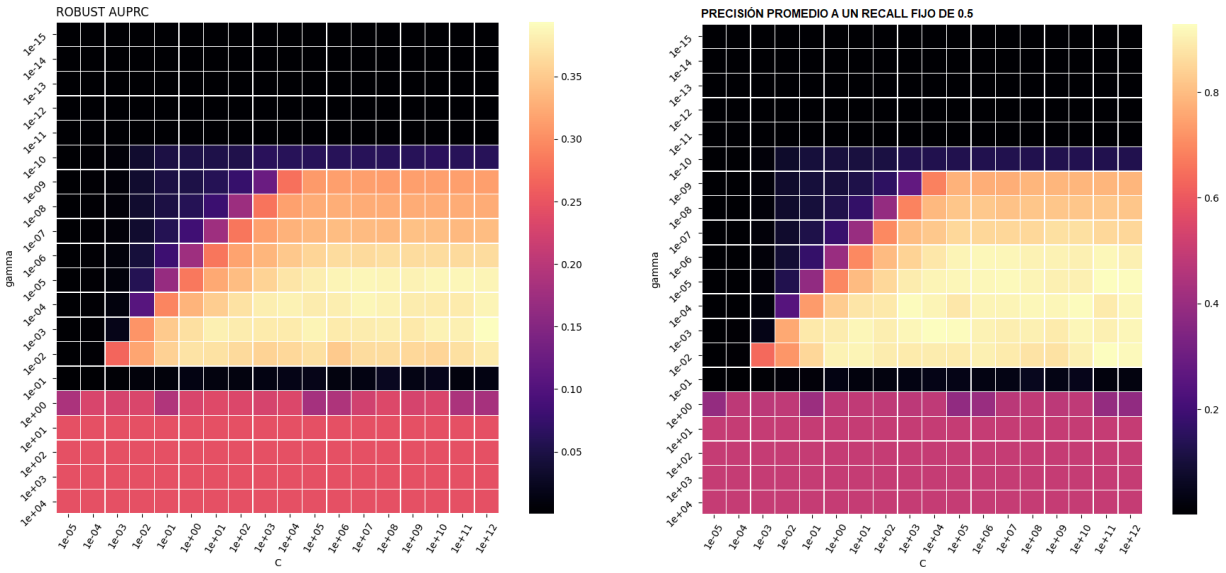
3.3. Optimización de hiperparámetros para SVM RBF

En este segundo experimento, se utilizó SVM con un kernel no lineal: Radial Basis Function. La optimización de hiperparámetros se vuelve significativamente más costosa, dado que se deben optimizar los parámetros C y γ en simultáneo. Como ya se mencionó, debido al inmenso tamaño de nuestros tiles (aproximadamente 400000 filas cada uno), resultó imposible utilizar implementaciones puras de kernel SVM.

Se decidió utilizar un aproximador del kernel RBF, Nystroem con $m = 300$, seguido de la eficiente implementación de LinearSVC descripta en la sección anterior. Esto permite aproximar los resultados de kernel SVM, reduciendo dramáticamente la complejidad computacional requerida.

Durante el experimento, se realizó 10-fold Grid Search Cross Validation sobre el tile b278. Se exploró un amplio rango logarítmico de valores para los hiperparámetros α y C. Los resultados pueden observarse en **3-2**.

Podemos observar que SVM-RBF es muy sensible a la elección de hiperparámetros, a diferencia de RF que no se ve muy afectado por la elección del número de árboles. Los hiperparámetros óptimos fueron $C = 10^5$ y $\gamma = 10^{-4}$.



(a) Area bajo la curva de Precision-Recall (b) Precision a un recall fijo de 0.5

Figura 3-2.: Resultados de la optimización de hiperparámetros para SVM-RBF en el tile b278

3.4. Análisis de performance de SVM

Habiendo estimado los hiperparámetros óptimos de SVM con kernel lineal y RBF, se procedió a entrenar y testear clasificadores usando cada par de tiles en { b234, b261 , b278, b360 }. Las curvas de precision-recall de ambos modelos, así como las de RF, pueden observarse en la figura **3-3**.

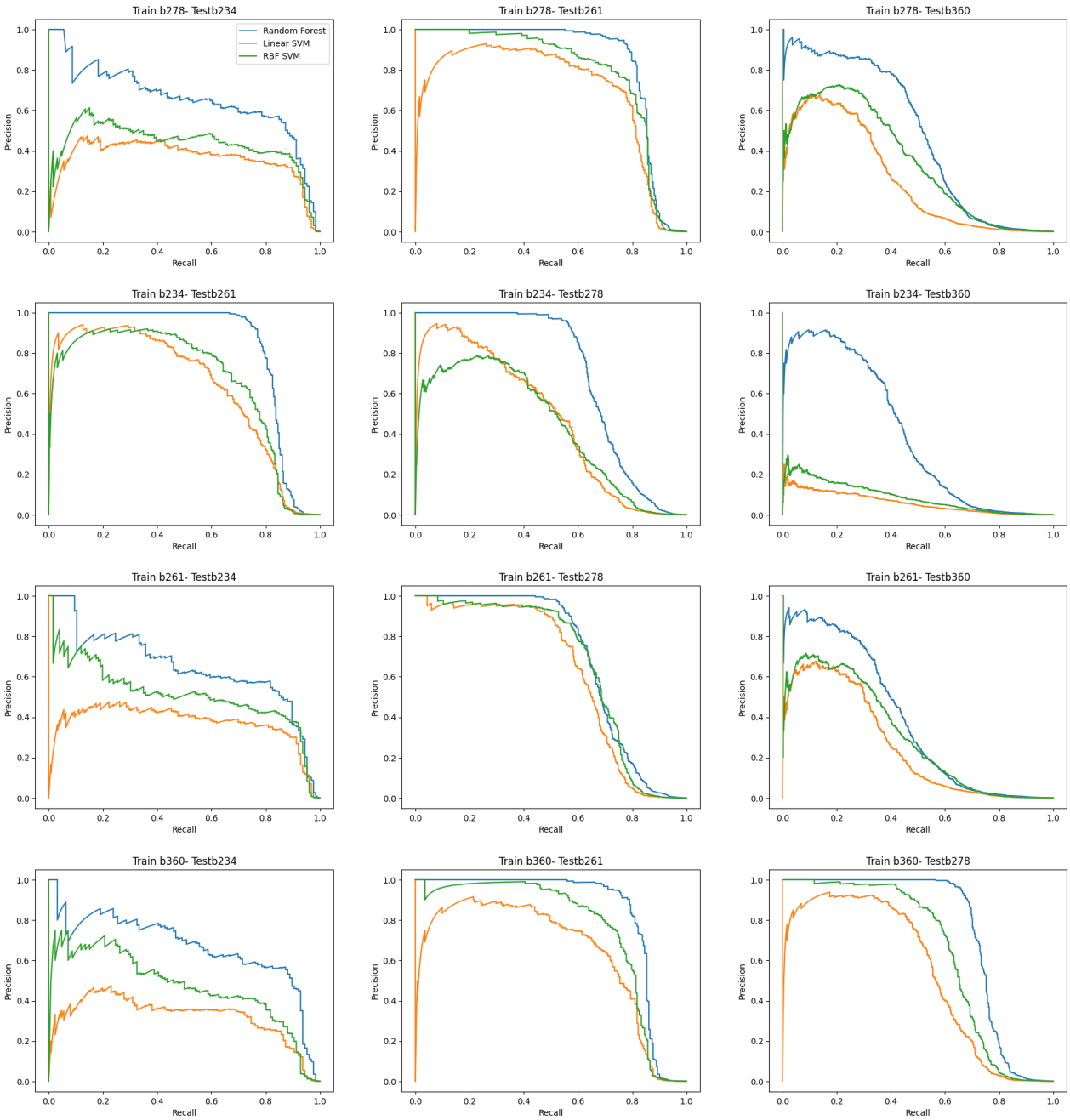


Figura 3-3.: Curvas de precision-recall obtenidas utilizando Random Forest, Support Vector Machine con kernel lineal y Support Vector Machine con kernel RBF

Algunas observaciones:

- En las 12 combinaciones testeadas la performance de random forest es superior a la de SVM-RBF, la cuál es a su vez superior a SVM Lineal. Esto se condice con los resultados

reportados en [Cabral et al., 2020a]. El hecho de haber utilizado las tiles completas para entrenar y testear nos permite apreciar con exactitud cuán mejores son las curvas generadas por Random Forest, lo cual no era evidente en trabajos previos.

- El hecho de que SVM Lineal tenga la menor performance parece indicar que la superficie de decisión no es lineal.

En los capítulos siguientes se aplicarán distintas técnicas con el objeto de mejorar las curvas obtenidas por SVM. Adicionalmente, se intentará comprender por qué RF parece funcionar mucho mejor que SVM en este tipo de datos.

4. Preprocesamientos

En este capítulo se describirán distintos tipos de preprocesamientos que pueden ser aplicados a los datos. Posteriormente, se procederá a realizar experimentos utilizando cada uno de estos preprocesamientos en combinación con SVM, evaluando si hay alguna mejora en los resultados que fueron expuestos en el capítulo anterior. Se utilizarán las implementaciones provistas por el módulo *preprocessing* de SKLearn [Buitinck et al., 2013], cuyos detalles pueden consultarse en la documentación de SKLearn[1].

4.1. Estandarización

En primer lugar, se consideraron distintas técnicas básicas que consisten en modificar la media y escalar la varianza de los datos. [Han et al., 2012]

- **MaxAbsScaler**: Escalar cada atributo por separado al rango $[-1, 1]$. Para cada columna c, se dividen todos los elementos de esa columna por su máximo valor absoluto:

$$c := \frac{c}{max(abs(c))}$$

- **MinMaxScaler**: Escalar cada atributo por separado al rango $[0, 1]$. Para cada columna c, se aplica la siguiente transformación [Han et al., 2012]:

$$c := \frac{c - min(c)}{max(c) - min(c)}$$

- **StandardScaler**: Estandarizar cada atributo, restándole su media y escalando a varianza unitaria.

$$c := \frac{c - mean(c)}{stdev(c)}$$

- **RobustScaler**: Estandarizar cada atributo usando métricas robustas a outliers, restando la mediana y escalando los datos de acuerdo al rango intercuartil.

4.2. Transformaciones no lineales y discretización

En esta sección se enumeran distintas técnicas que modifican la distribución de probabilidad subyacente de cada atributo:

[1] `https://scikit-learn.org/stable/modules/preprocessing.html`

- **PowerTransformer**: Aplica una transformación paramétrica, monotónica que transforma cada atributo para que tenga una distribución de probabilidad más Gaussiana. La transformación utilizada es Yeo-Johnson [Yeo and Johnson, 2000]. Un escalado estándar (StandardScaler) es aplicado previamente a los datos.

- **QuantileTransformer**: Transformar cada atributo por separado, obteniendo valores entre 0 y 1. Una primera opción es mapear los datos a una distribución uniforme, una segunda opción es mapear los datos a una distribución normal [Krzysztofowicz, 1997]. Un parámetro de esta transformación es el número de cuantiles computado para realizar la transformación.

- **Binning**: Discretizar cada atributo por separado, distribuyendo sus valores en k cubetas (bins) [Han et al., 2012]. Distintos criterios pueden seguirse para decidir cómo se distribuyen los datos en cada cubeta:
  - **Uniforme**: Usar cubetas de tamaño constante. Es decir, dividir el rango de valores del atributo en k intervalos del mismo tamaño; y reemplazar cada valor por el índice de su intervalo.
  - **Quantile**: Colocar la misma cantidad de elementos en cada cubeta.
  - **Kmeans**: Definir cada cubeta basado en un algoritmo de clústering k-means.

Es importante remarcar que para todos los preprocesamientos discutidos hasta el momento, se estimarán los parámetros de la transformación usando únicamente los datos de entrenamiento. Por ejemplo, los límites de los bins son calculados sobre los datos de entrenamiento y estos mismos límites se utilizarán para discretizar los datos de test. Análogamente, a la hora de realizar un escalado estándar por ejemplo, tanto a los datos de entrenamiento como a los de test se les resta la media del dataset de entrenamiento y se les divide por la varianza del detaset de entrenamiento.

4.3. Preprocesamientos en SVM Lineal

Se aplicaron los preprocesamientos descriptos en las secciones anteriores a los datos, antes de utilizar SVM Lineal. En los siguientes experimentos, se considera como base de referencia la performance reportada en la última sección del capítulo anterior, obtenida aplicando únicamente StandardScaler antes de entrenar y testear.

- Los resultados de aplicar estandarización pueden ser observados en **4-1**. De este experimento se desprende que StandardScaler es la mejor opción, dado que es la única opción que se encuentra consistentemente entre los preprocesamientos con mejor performance.

- Los resultados de aplicar QuantileTransformer se pueden observar en **4-2**. Se exploraron ambas variantes: mapear la distribución de cada atributo a una distribución

normal, y a una distribución uniforme. Para cada una de estas configuraciones, se analizó el R-AUPRC en función del número de cuantiles utilizado en el transformador. Podemos observar que todas las combinaciones de parámetros exploradas efectivamente incrementan la performance respecto a la base de referencia; aunque el número de cuantiles óptimo es dependiente de los datos.

- Los resultados de aplicar binning, seguido de un escalado estándar, se pueden observar en **4-3**. Se exploraron las tres estrategias anteriormente mencionadas (Uniform, Quantile y Kmeans), estudiando la evolución del R-AUPRC en función del número de cubetas. Se puede apreciar que la estrategia Quantile es consistentemente la mejor opción. Nótese que la estrategia KMeans es computacionalmente impracticable para grandes cantidades de cubetas.

- Las curvas de precision-recall de los preprocesamientos que tuvieron mejor desempeño en los experimentos anteriores, junto con la correspondiente a PowerTransformer (que no requiere parámetros), son comparadas en la figura **4-4**. De esta comparación final se desprende que el preprocesamiento óptimo para SVM Lineal es binning, seguido de un escalado estándar. Al final de este capítulo se especula sobre los motivos por los cuales binning produce tales mejoras.

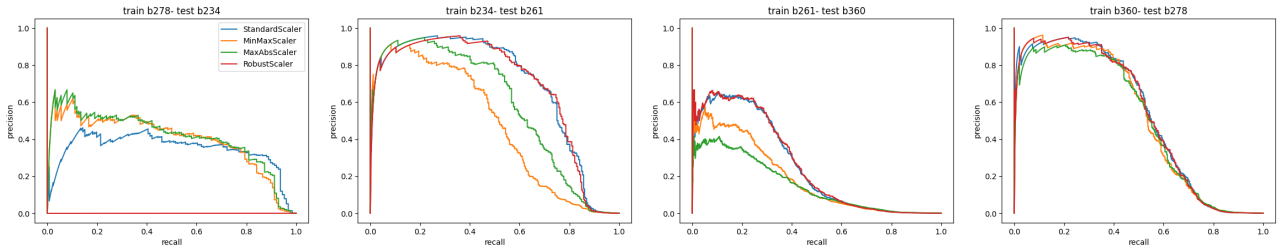


Figura 4-1.: Estandarización en SVM Lineal.

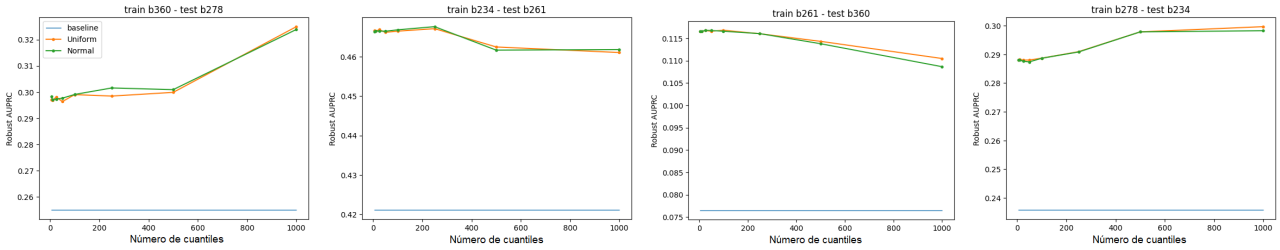


Figura 4-2.: QuantileTransformer en SVM Lineal

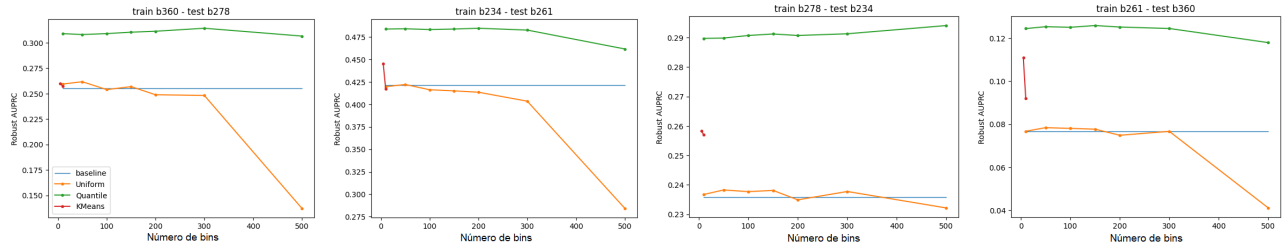


Figura 4-3.: Binning en SVM Lineal

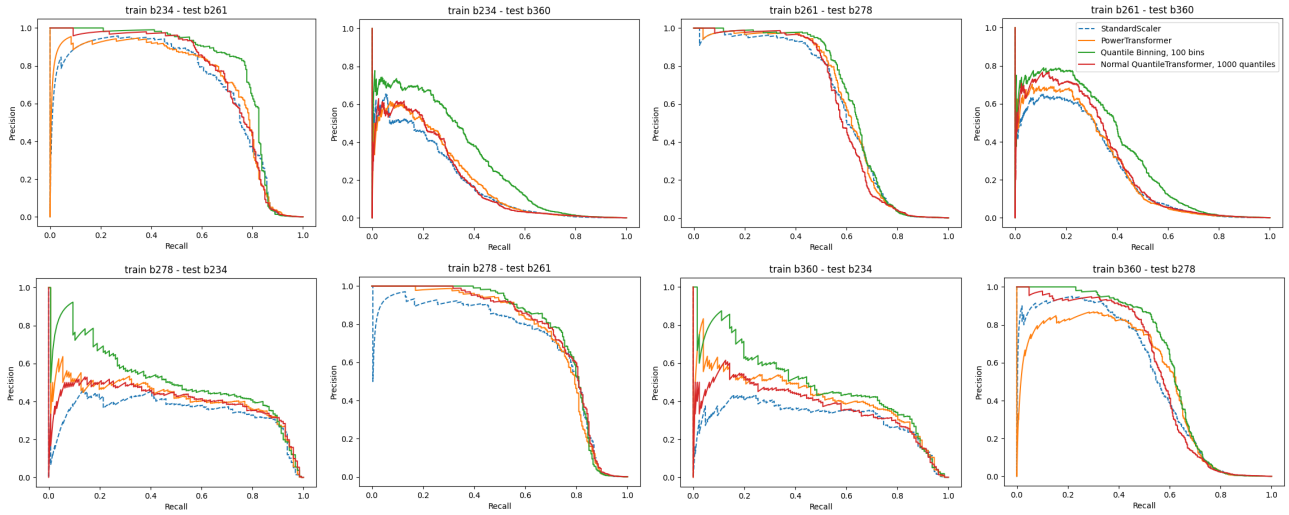


Figura 4-4.: Comparación de los mejores preprocesamientos, SVM Lineal

4.4. Preprocesamientos en SVM RBF

Se procedió a realizar los mismos experimentos utilizando SVM con kernel RBF:

- Los resultados de aplicar estandarización pueden ser observados en **4-5**. Nuevamente, StandardScaler parece ser la mejor opción, estando consistentemente entre los que mejor funcionan. El escalado robusto, si bien produce mejores resultados en ciertos tiles, ocasiona una pérdida significativa de performance en otros.

- Los resultados de aplicar QuantileTransformer se pueden observar en **4-6**. A diferencia de en SVM-Lineal, la performance oscila considerablemente dependiendo del número de cuantiles, y es difícil encontrar una asignación de parámetros para este método que sea óptima para todos los tiles. Más aún, aplicar este preprocesamiento implica una pérdida de performance respecto a la base de referencia en algunas de las combinaciones testeadas, por lo cuál se descarta su aplicación en el resto de este trabajo.

- Los resultados de aplicar binning, seguido de un escalado estándar, se pueden observar en **4-7**. Los resultados son consistentes a los observados en SVM Lineal: La estrategia

Quantile es consistentemente superior a las otras estrategias, y dependiendo del número de bins escogido mejora la performance respecto a la base de referencia.

- Las curvas de precision-recall de los preprocesamientos que tuvieron mejor desempeño en los experimentos anteriores, junto con la correspondiente a PowerTransformer (que no requiere parámetros), son comparadas en la figura **4-8**. De esta comparación final se desprende que el preprocesamiento óptimo para SVM RBF es Binning, aunque la mejora es mucho más modesta que la obtenida en SVM Lineal.

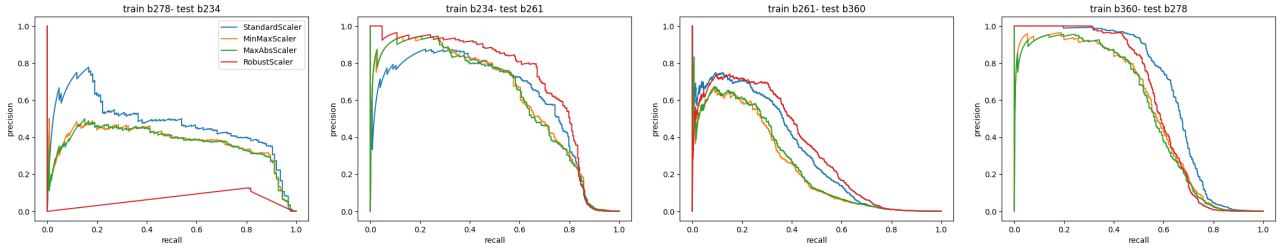


Figura 4-5.: Estandarización en SVM RBF

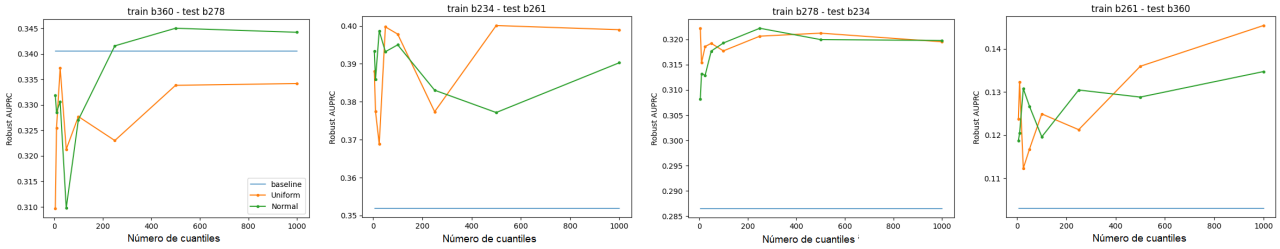


Figura 4-6.: QuantileTransformer en SVM RBF

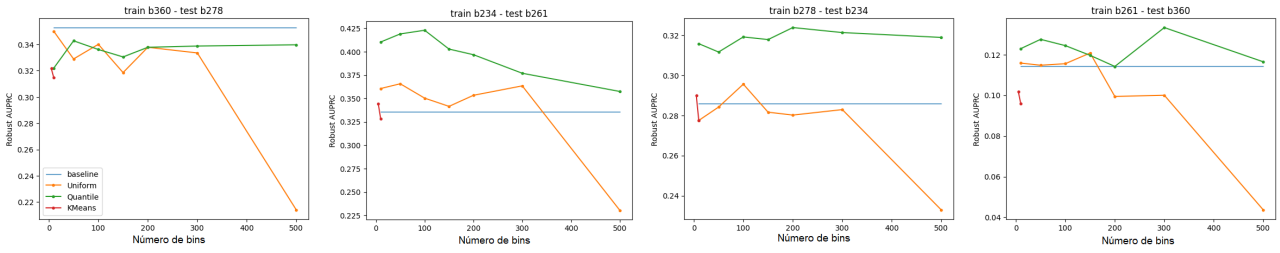


Figura 4-7.: Binning en SVM RBF

4.5. Análisis de resultados

Se exploró una amplia variedad de preprocesamientos a ser aplicados a los datos. Aquellas técnicas que transforman la distribución de probabilidad de cada atributo a una normal o

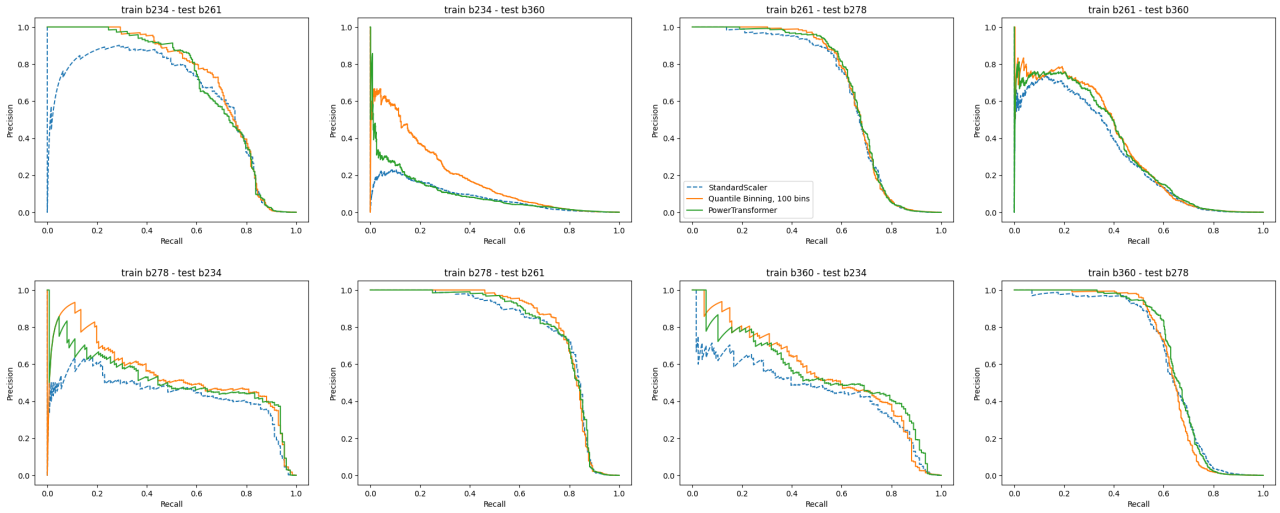


Figura 4-8.: Comparación de los mejores preprocesamientos, SVM RBF

uniforme fueron las que mostraron un mayor beneficio, siendo Binning el preprocesamiento escogido para el resto de este trabajo.

Hay distintos motivos por los cuales binning quantile puede estar mejorando consistentemente la performance de SVM:

- Errores de medición: Se reduce el efecto de errores menores de observación. Es decir, se aplica un efecto de suavizado local (local smoothing) sobre los datos. Esto es razonable para datasets astronómicos, donde los instrumentos de medición que generan los datos siempre tendrán un error inherente.
- Manejo de outliers: Valores extremos son asignados a las cubetas de los extremos, evitando que tengan un efecto significativo sobre los parámetros del modelo.
- Binning tiene una cierta similitud a la forma en que los árboles de decisión funcionan: dividir un atributo continuo en subintervalos, eligiendo ciertos puntos de corte; y tomar decisiones dependiendo de en qué subintervalo un cierto valor está. Es posible que para nuestros datos, ciertos atributos tengan un efecto particular sobre la clase objetivo a un threshold específico; y binning ayude a SVM a *entender* esto con mayor facilidad, concentrándose en interacciones de alto orden entre los atributos.
- La distribución de probabilidad subyacente de cada atributo será convertida a una distribución uniforme.

Claramente, la potencial pérdida de información asociada a discretizar los datos no perjudica a nuestro modelo. Nótese que se aplica un escalado estándar luego de Binning, dado que Binning en su forma pura mapearía cada valor a $[0, NBINS)$; y SVM funciona mejor con

valores pequeños alrededor de cero.

4.6. Reoptimización de hiperparámetros

Luego de definir el preprocesamiento óptimo de cada método (Quantile 100-Binning + StandardScaler), se procedió a reoptimizar los hiperparámetros tal y como se describió en el capítulo 3, esta vez incluyendo los preprocesamientos. Los nuevos parámetros óptimos fueron:

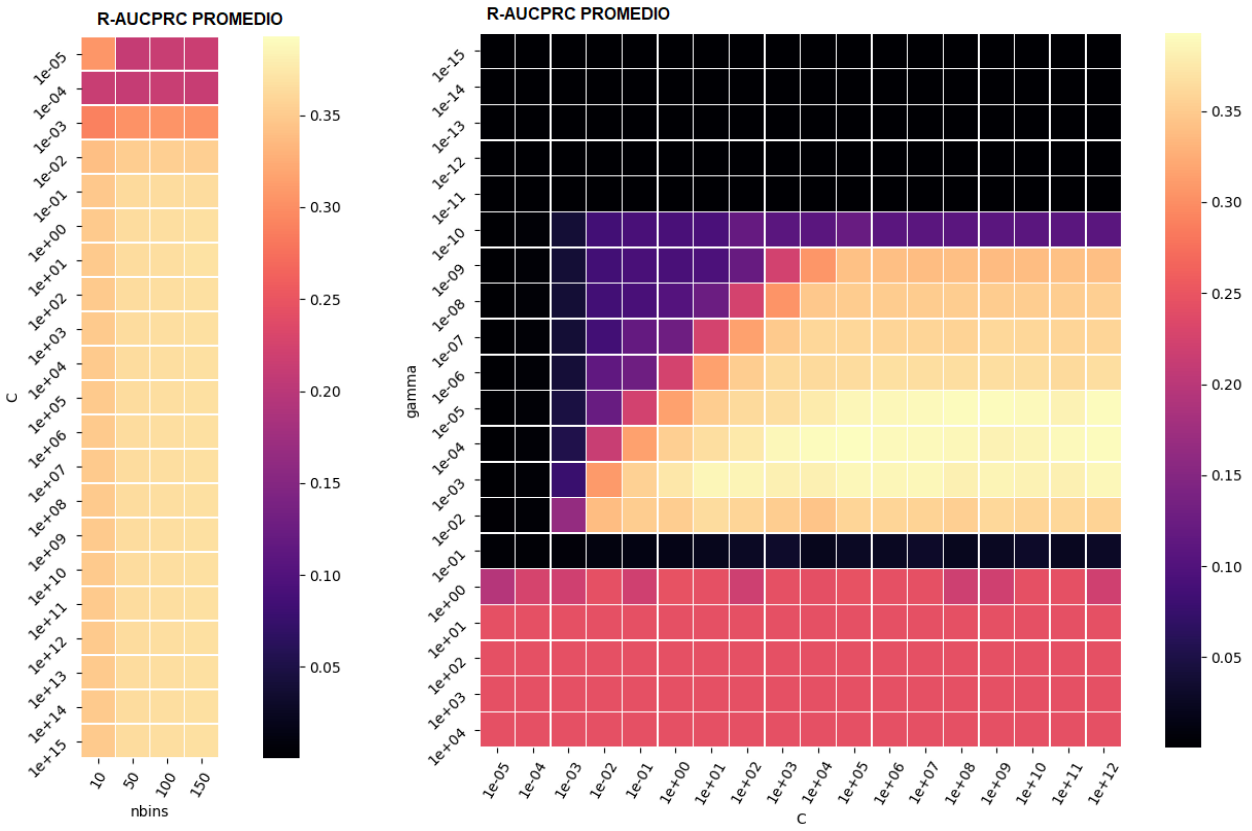
- **SVM-Lineal:** $C = 10$ y 150 bins.
- **SVM-RBF:** $C = 10^4$, $\gamma = 10^{-4}$, 100 bins.

Notemos que el parámetro C ahora es dos órdenes de magnitud mayor en SVM-Lineal, y un orden de magnitud menor en SVM-RBF. El R-AUPRC en cross-validation obtenido para cada clasificador puede observarse en la figura **4-9**.

Utilizando estos nuevos parámetros y preprocesamientos, se volvieron a calcular las curvas de precision-recall para cada par de tiles, que se se pueden ver en la figura **4-10**.

4.7. Conclusiones

Las tablas **4-1** y **4-2** nos permiten medir numéricamente el incremento en R-AUPRC obtenido en las curvas al utilizar preprocesamientos, respecto a las presentadas en capítulos anteriores. Tanto SVM lineal como SVM RBF se ven beneficiados al utilizar binning, siendo SVM lineal el método que muestra un mayor incremento en performance.



(a) SVM Lineal (b) SVM RBF, 100 bins

Figura 4-9.: Resultados de la optimización de hiperparámetros para SVM-Lineal y RBF en el tile b278, incluyendo preprocesamientos.

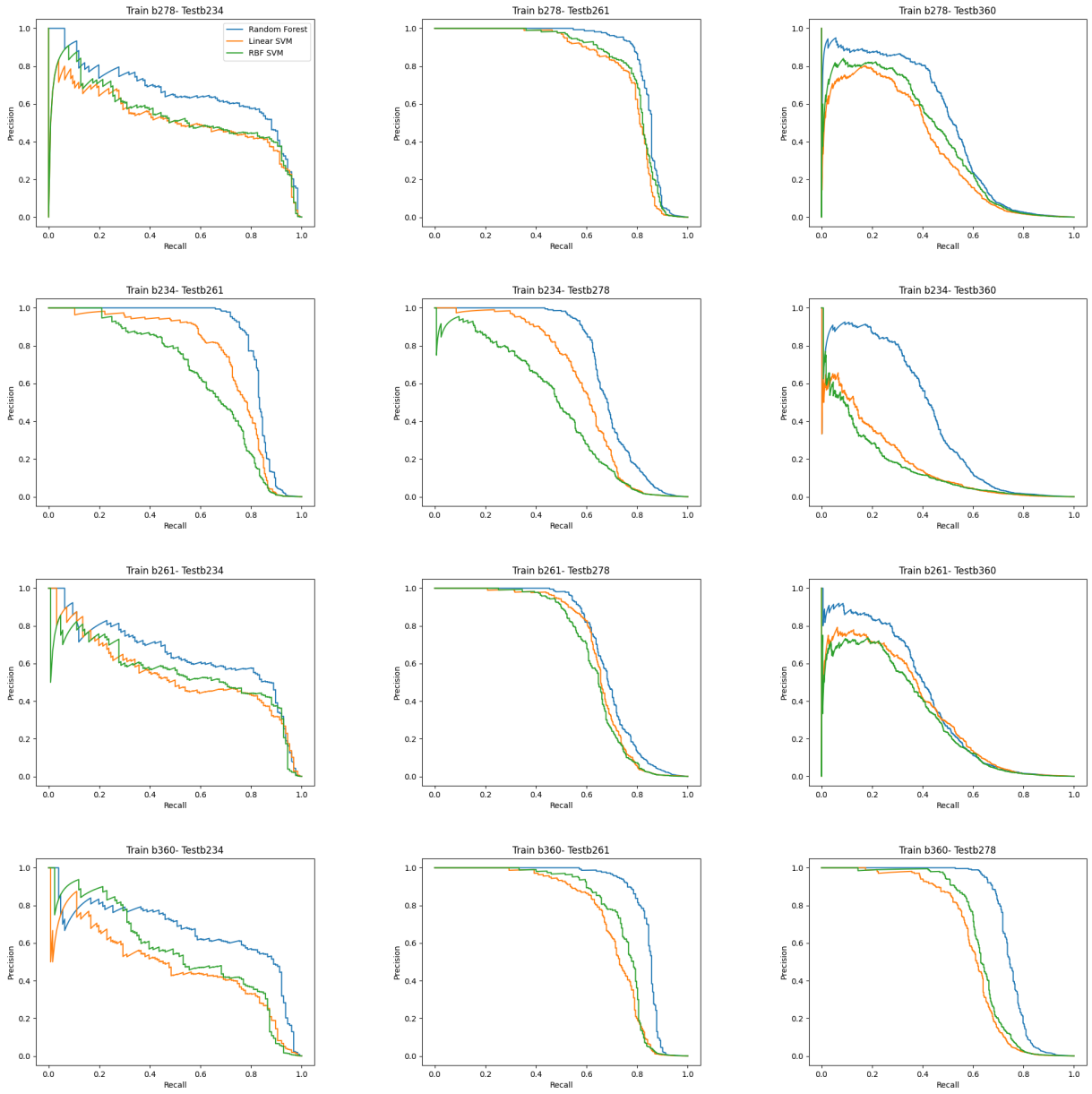


Figura 4-10.: Curvas de precision-recall obtenidas utilizando Random Forest, Support Vector Machine con kernel linal y Support Vector Machine con kernel RBF, incluyendo hiperparámetros optimizados con preprocesamiento incluido.

Train tile	Test tile	RF	SVM-L	Preproc+SVM-L	Gain
b278	b234	0.40	0.25	0.31	0.06
b278	b261	0.54	0.44	0.48	0.04
b278	b360	0.21	0.08	0.14	0.06
b234	b278	0.39	0.21	0.29	0.09
b234	b261	0.53	0.37	0.44	0.07
b234	b360	0.14	0.02	0.04	0.02
b261	b278	0.39	0.33	0.36	0.02
b261	b234	0.39	0.25	0.30	0.06
b261	b360	0.13	0.07	0.12	0.05
b360	b278	0.45	0.27	0.30	0.04
b360	b234	0.42	0.20	0.26	0.06
b360	b261	0.54	0.39	0.41	0.02
avg		.38	.24	.29	.05

Tabla 4-1.: En esta tabla podemos ver el incremento en R-AUPRC que se obtiene al utilizar preprocesamientos en SVM lineal.

Train tile	Test tile	RF	SVM-RBF	Preproc+SVM-RBF	Gain
b278	b234	0.40	0.28	0.32	0.04
b278	b261	0.54	0.49	0.50	0.01
b278	b360	0.21	0.14	0.16	0.02
b234	b278	0.39	0.22	0.30	0.09
b234	b261	0.53	0.41	0.45	0.04
b234	b360	0.14	0.03	0.07	0.03
b261	b278	0.39	0.37	0.37	0.00
b261	b234	0.39	0.30	0.33	0.03
b261	b360	0.13	0.11	0.12	0.02
b360	b278	0.45	0.34	0.34	0.00
b360	b234	0.42	0.27	0.30	0.03
b360	b261	0.54	0.46	0.45	-0.01
avg		.38	.29	.31	.03

Tabla 4-2.: En esta tabla podemos ver el incremento en R-AUPRC que se obtiene al utilizar preprocesamientos en SVM-RBF. Si bien el aumento es menor al observado en SVM Lineal, se puede observar una mejoría consistente.

5. Selección y extracción de atributos

En esta sección se aplicarán diversas técnicas para reducir la cantidad de atributos necesaria para describir cada estrella. Esta reducción se puede alcanzar de dos formas [Jovic et al., 2015]:

- **Selección de Atributos** (Feature Selection): Utilizar un subconjunto de atributos del dataset original, sin transformarlos, por lo tanto manteniendo su interpretación original. Ejemplo: Filtros univariable. (Ver sección 5.1.2).
- **Extracción de Atributos** (Feature Extraction): Transformar los atributos originales en un nuevo conjunto de atributos, reduciendo la cantidad de atributos e intentando conservar la información relevante. Ejemplo: PCA (Ver sección 5.2.2).

5.1. Selección de atributos

5.1.1. Introducción a selección de atributos

En la mayoría de los problemas de clasificación con datos provenientes del mundo real, los atributos relevantes son a menudo desconocidos a priori. Usualmente, muchos atributos candidatos son introducidos con el objeto de representar el dominio lo mejor posible. Desafortunadamente, muchos de estos atributos candidatos son parcial o completamente irrelevantes o redundantes para el concepto objetivo [Dash and Liu, 1997].

Selección de atributos, como ya se mencionó, consiste en elegir sólo un subconjunto de atributos de un cierto dataset, descartando los demás de acuerdo a un cierto criterio [Liu et al., 2010]. Realizar selección de atributos como un paso previo a aplicar un algoritmo de aprendizaje automatizado puede proveer varias ventajas [Guyon and Elisseeff, 2003]:

- **Desempeño**: En ciertos datasets, muchos atributos no son informativas para el problema en cuestión. Al eliminar atributos irrelevantes, ruidosos o redundantes, se reduce el riesgo de sobreajuste, mejorando el desempeño del clasificador en test. Más aún, algunos algoritmos simplemente trabajan mucho mejor con menos variables.
- **Eficiencia**: Reducción en complejidad computacional y espacial.

- **Entendimiento**: El modelo resulta más sencillo y fácil de interpretar. Adicionalmente, muchas técnicas utilizadas para realizar selección de atributos permiten en el proceso descubrir información sobre el problema, por ejemplo cuáles son las variables más importantes (Ver capítulo 6).

En este trabajo se aplicaron métodos de selección de atributos de **filtrado** [Jovic et al., 2015], que asignan un puntaje o ranking a atributos individuales o a grupos de atributos. Los **filtros univariable** asignan un puntaje a cada atributo por separado, mientras que los **filtros multivariable** evalúan subconjuntos de atributos a la vez.

5.1.2. Filtros univariable

En este trabajo se experimentó únicamente con filtros univariable, los cuales funcionan asignando puntajes a cada atributo intentando medir cuán correlacionado dicho atributo está con la variable a predecir. Posteriormente, se evaluó la performance de nuestro clasificador conservando únicamente los k atributos con mayor puntaje para entrenar y testear. Los filtros utilizados fueron:

- **f_classif**: Computa el ANOVA (Analysis of Variance) F-value [Han et al., 2012]. Este test estadístico determina si las medias de dos muestras de datos provienen de la misma distribución o no. Este test es capaz de medir correlaciones lineares. En la sección 6.1.1 se estudiará ANOVA en mayor detalle.

- **mutual_info**: Calcula la información mútua [Han et al., 2012] entre dos variables aleatorias, la cuál es una medida de la reducción en incertidumbre para una variable aleatoria suponiendo que se conoce el valor de la otra. Intuitivamente, mide cuánta información una variable aleatoria contiene sobre otra. La implementación provista por sklearn hace uso de modelos no paramétricos basados en estimar la entropía de distancias de KNN (K vecinos más cercanos), tal y como se describe en [Kraskov et al., 2004]. Este test es capaz de medir correlaciones no necesariamente lineares.

Nótese que en este trabajo no se han aplicado filtros multivariable. Es perfectamente posible que un cierto atributo sea inútil en sí mismo pero provea un aumento significativo en performance al ser utilizado con otros [Guyon and Elisseeff, 2003]. Sin embargo, los filtros multivariable son computacionalmente mucho más costosos y muestran tendencia a sobreajustar, por lo que se decidió no aplicarlos.

5.1.3. Experimentos en SVM-Lineal

En la figura **5-1** se encuentran los resultados de aplicar filtros univariable como paso previo a SVM-Lineal. Las curvas muestran el R-AUPRC en test obtenido utilizando únicamente los k atributos con mayor puntaje para entrenar y testear. Podemos concluir que:

- El aumento en R-AUPRC es muy modesto o nulo. Los filtros utilizados no tienen un impacto positivo significativo en el desempeño de nuestro clasificador.
- Hay aproximadamente 20 atributos que no contribuyen a mejorar el desempeño en clasificación usando SVM-L, pues eliminarlos no empeora las curvas de precision-recall significativamente. Se puede obtener una reducción importante en tiempo de entrenamiento y memoria eliminándolos. En la figura **5-2** se puede apreciar, al ver todas las curvas juntas, que el R-AUPRC deja de mejorar para $k \geq 40$ (i.e., las curvas tienden a aplanarse).
- Los ranking de atributos provistos por $f\_classif$ y $mutual\_info$ producen curvas consistentes. Ninguno de los dos filtros parece ser notablemente superior al otro. Nótese, sin embargo, que si se decide seleccionar un número muy pequeño de atributos ($k \leq 5$), $mutual\_info$ produce mejores resultados. Esto es irrelevante para nuestra aplicación, dado que los valores de R-AUPRC para k tan pequeños son muy inferiores a la base de referencia.

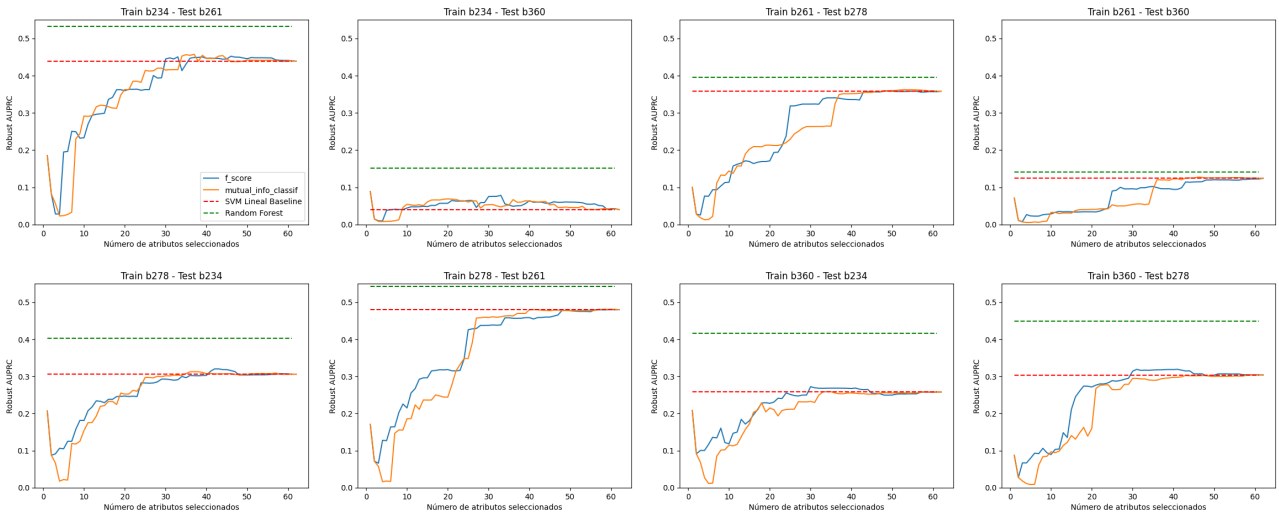
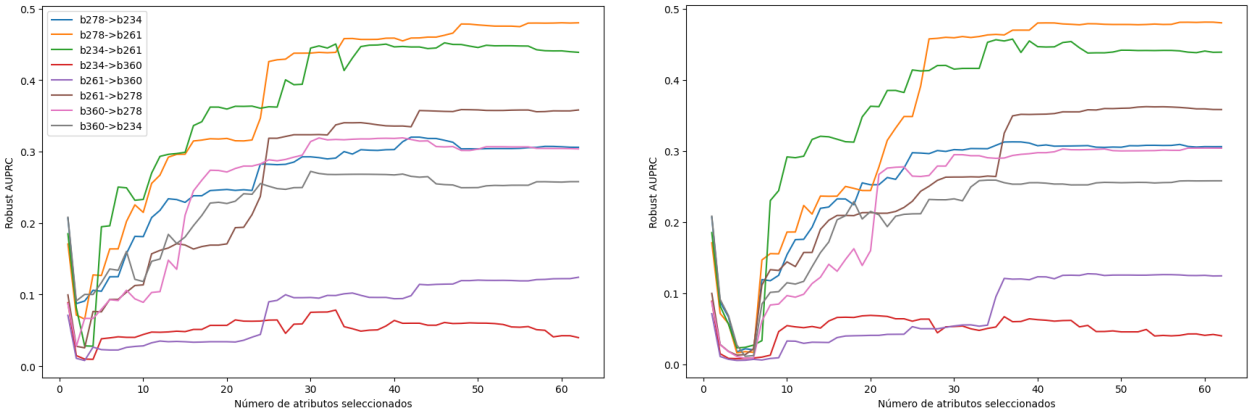


Figura 5-1.: Aplicación de filtros univariable en SVM-Lineal. Para cada par de tiles, se calcula el R-AUPRC de la curva de precision-recall en test obtenida entrenando con los k atributos de mayor puntaje. La línea roja punteada es la performance de SVM sin usar selección de atributos.

Para elegir un valor óptimo de k de forma analítica, se decidió calcular cuánto mejora (o empeora) el R-AUPRC para cada valor de k respecto a no utilizar selección de atributos. Dado que este análisis se realizó sobre 8 pares de tiles, en la figura **5-3** se muestra el promedio de la diferencia entre el R-AUPRC utilizando k atributos y la base de referencia. Adicionalmente, en color naranja, se muestra la diferencia obtenida por aquel par de tiles



(a) Filtro: $f\_classif$ (b) Filtro: $mutual\_info$

Figura 5-2.: Aplicación de filtros univariable en SVM-Lineal en distintos pares de tiles. Nótese que las curvas tienden a saturarse al utilizar un número de atributos mucho menor al total de atributos (62), lo cuál indica que a partir de un cierto punto hay muy poco beneficio en agregar más atributos.

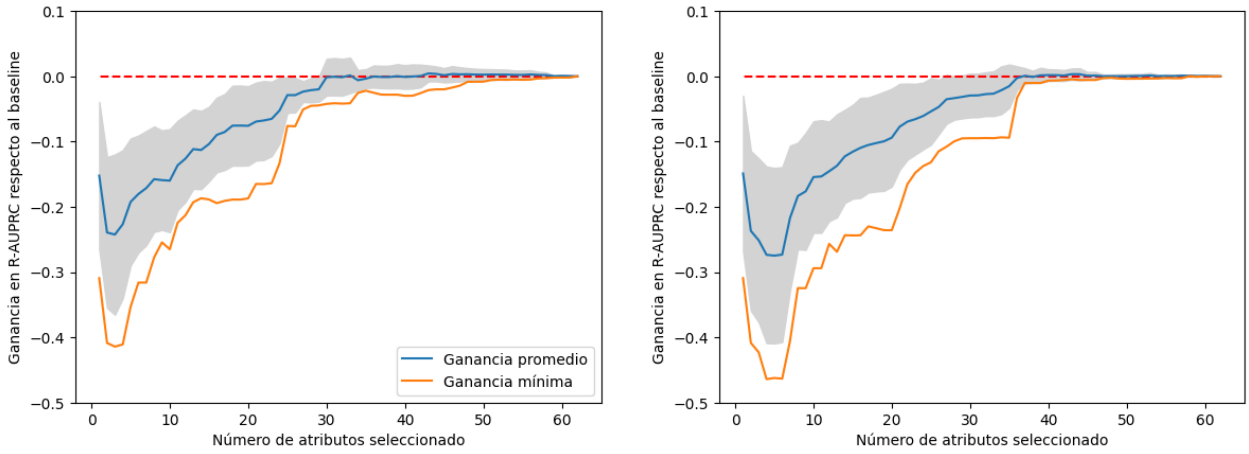
que menos se beneficia para cada valor de k.

Esta gráfica nos permite entender qué valores de k son más convenientes, o al menos no incurren en grandes pérdidas de R-AUPRC. Las conclusiones finales son:

- Ningún valor del hiperparámetro k permite, en promedio, mejorar el R-AUPRC en todos los tiles. Por otro lado, es perfectamente posible igualar la performance de la base de referencia eliminando varios atributos.
- Utilizando $f\_classif$, seleccionar 30 atributos no disminuye el R-AUPRC en promedio. Sin embargo, algunos tiles se ven altamente perjudicados, pues el mínimo toma valores negativos no despreciables. Seleccionar aproximadamente 50 atributos permitiría reducir esta variabilidad, obteniendo resultados más estables entre distintos pares de tiles.
- Utilizando $mutual\_info$ podemos permitirnos seleccionar una cantidad menor de atributos ($k \geq 40$), garantizando que ningún par de tiles tendrá una pérdida significativa de R-AUPRC.

5.1.4. Experimentos en SVM-RBF

En la figura **5-4** se encuentran los resultados de aplicar filtros univariable como paso previo a SVM-RBF. Las curvas muestran el impacto que tiene utilizar únicamente los k atributos con mayor puntaje en el R-AUPRC. Podemos concluir que:



(a) Filtro: $f\_classif$ (b) Filtro: $mutual\_info$

Figura 5-3.: Estas gráficas permiten apreciar la diferencia en R-AUPRC respecto a la base de referencia, obtenida al realizar seleccionar k atributos en SVM Lineal. Dado que esta diferencia se computó sobre 8 pares de tiles, para cada k se muestra la media, la desviación estándar y el mínimo valor de entre todos los pares testeados.

- Algunas curvas de SVM-RBF muestran una mejoría respecto a la base de referencia si se eliminan algunos atributos.
- Se puede reducir el tamaño del dataset aproximadamente a la mitad, seleccionando aproximadamente 30-35 atributos, mejorando o al menos no empeorando todas las curvas.
- Nótese que las curvas de SVM-Lineal comienzan a aplanarse a partir de $k = 40$, en tanto que en las curvas de SVM-RBF (**5-5**) tienden a aplanarse a partir de $k = 30$. Esto sugiere que SVM-RBF genera superficies de decisión no lineales; y que hay relaciones no lineales involucradas en nuestros datos que SVM lineal no es capaz de capturar.

Nuevamente, resulta complicado determinar el número óptimo de atributos a seleccionar a simple vista. Se procedió a resumir la información de las 8 curvas en la figura **5-6**, que analiza la ganancia promedio en R-AUPRC para cada k. Las conclusiones finales son:

- Seleccionar $35 \leq k \leq 55$ atributos de acuerdo al ranking de $mutual\_info$ permite mejorar el R-AUPRC respecto a la base de referencia en *todos* los pares testeados. En particular, $k = 45$ maximiza la mínima diferencia y el promedio.
- Seleccionar atributos de acuerdo al criterio $f\_classif$ es menos beneficioso, pues existen pocos valores de k para los cuales el R-AUPRC no empeora en ningún par de tiles testeado; y la mejora es un poco más modesta que la observada utilizando $mutual\_info$.

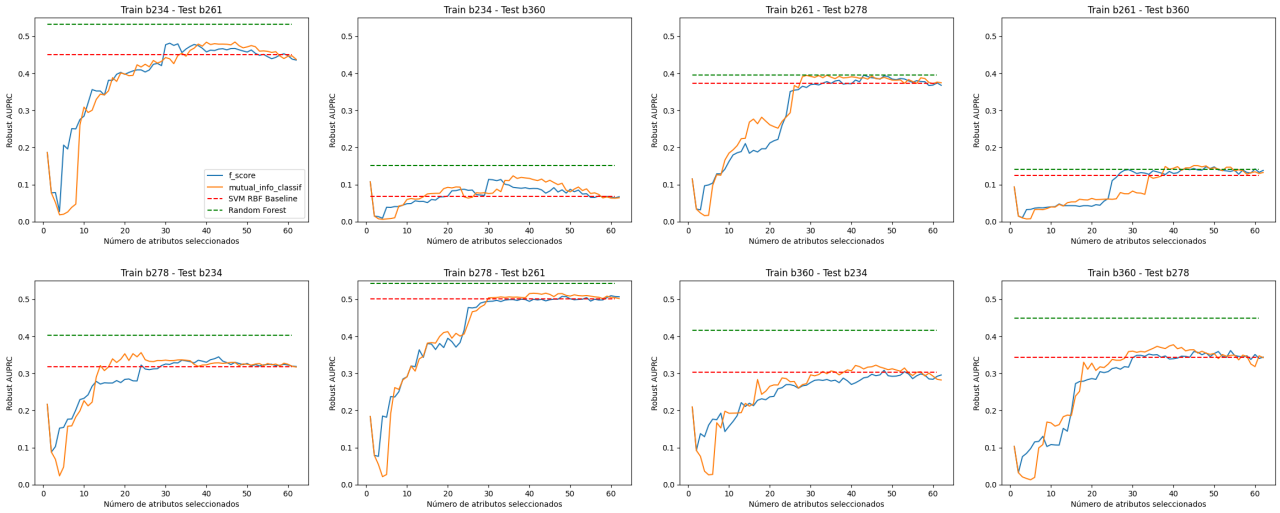
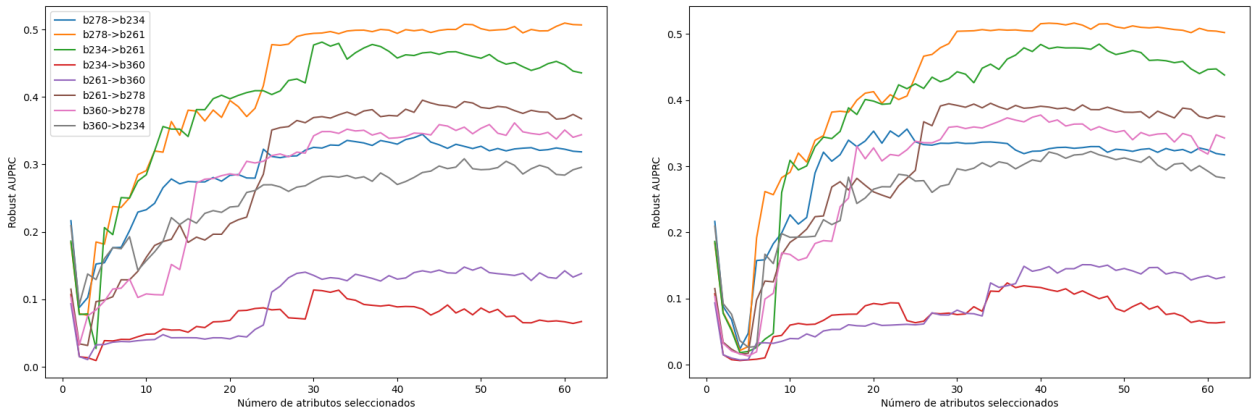


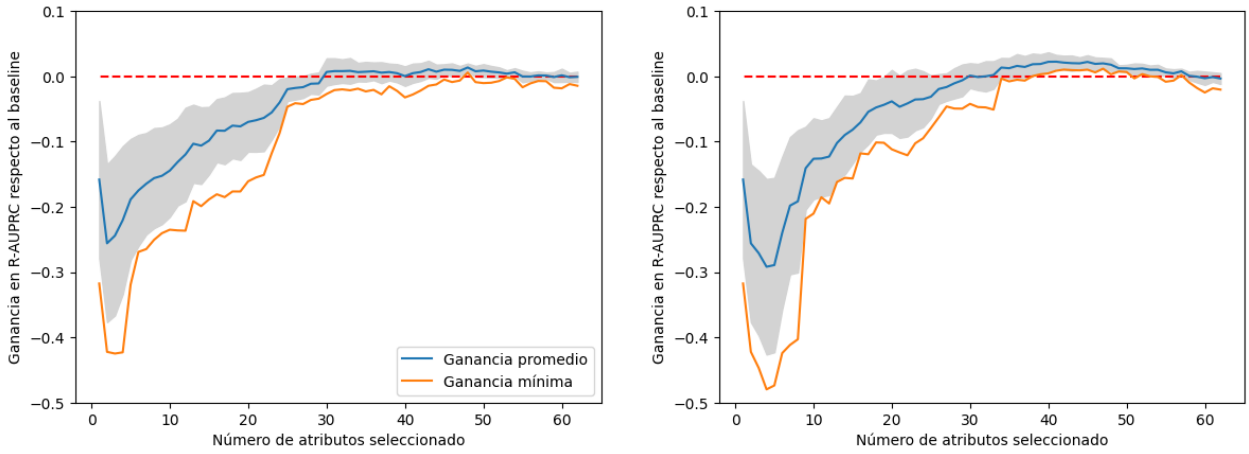
Figura 5-4.: Aplicación de filtros univariable en SVM-RBF. Para cada par de tiles, se calcula el R-AUPRC de la curva de precision-recall en test obtenida entrenando con los k atributos de mayor puntaje. La línea roja punteada es la performance de SVM sin usar selección de atributos.



(a) Filtro: $f\_classif$ (b) Filtro: $mutual\_info$

Figura 5-5.: Aplicación de filtros univariable en SVM-RBF en distintos pares de tiles. Nótese que las curvas tienden a saturarse al utilizar un número de atributos mucho menor al total de atributos (62), lo cuál indica que a partir de un cierto punto hay muy poco beneficio en agregar más atributos.

Esto puede deberse a que $f\_classif$ es incapaz de detectar correlaciones no lineales, y puede estar causando el descarte prematuro de atributos que resultan útiles a SVM-RBF para detectar RRLs.



(a) Filtro: $f\_classif$ (b) Filtro: $mutual\_info$

Figura 5-6.: Estas gráficas permiten apreciar la diferencia en R-AUPRC respecto a la base de referencia, obtenida al realizar seleccionar k atributos en SVM-RBF. Dado que esta diferencia se computó sobre 8 pares de tiles, para cada k se muestra la media, la desviación estándar y el mínimo valor de entre todos los pares testeados.

5.2. Extracción de atributos

5.2.1. Introducción a extracción de atributos

Consideremos un cierto fenómeno gobernado por L variables independientes. En la práctica, este fenómeno probablemente aparentará tener muchos más grados de libertad, debido a la influencia de factores como ruido, imperfección en el sistema de mediciones, adición de atributos irrelevantes, etcétera. Se define la **dimensión intrínseca** de un fenómeno como el *número de variables independientes que explica satisfactoriamente ese fenómeno* [Carreira-perpinan, 1997].

Dado un dataset X con d atributos numéricos, extracción de atributos consiste en hallar un mapeo $\mathbb{R}^d \mapsto \mathbb{R}^p$ con $p \leq d$; donde idealmente p será la dimensionalidad intrínseca de X. Este mapeo es utilizado para encontrar una representación más compacta de X sin perder información [Ghojogh et al., 2019]. Al igual que en selección de atributos, se espera obtener un beneficio en términos de desempeño y eficiencia. Adicionalmente, los datos pueden ser más fáciles de visualizar o separar en el nuevo espacio.

Los métodos de extracción de atributos pueden clasificarse en dos tipos [Ghojogh et al., 2019]:

- **Supervisados**: Tienen en cuenta las etiquetas (clases) de cada instancia a proyectar.

- **No supervisados**: No tienen en cuenta la separación entre clases, y se concentran principalmente en la variación y distribución de los datos.

5.2.2. Principal Component Analysis

En esta sección se describirá un método de extracción de atributos no supervizado muy popular: Principal Component Analysis (PCA) [Han et al., 2012].

A grandes rasgos, PCA consiste en entender cuáles son las direcciones más importantes de nuestros datos (i.e., las direcciones donde hay mayor variabilidad). Luego se transforman los datos, alineándolos a estas direcciones importantes. Finalmente, las direcciones menos importantes pueden ser eliminadas. Este proceso se ilustra en la figura **5-7**

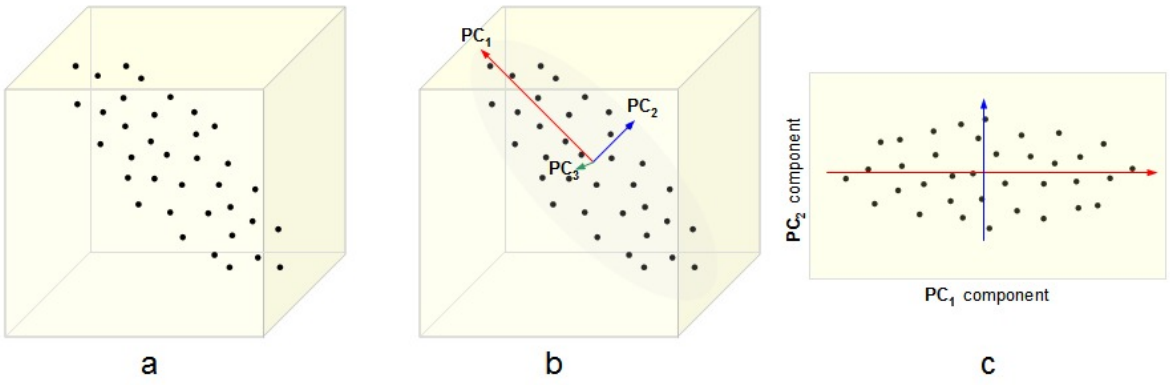


Figura 5-7.: Ejemplo de PCA. En la figura (a) se muestra un dataset en tres dimensiones. En la figura (b) se identifican las tres direcciones (perpendiculares entre sí) con mayor variabilidad. En la figura (c) se alinean los datos con las direcciones de mayor variabilidad, y se elimina la componente con menor variabilidad. Como consecuencia, la dimensionalidad de los datos se redujo en 1. Fuente: `cnx.org`

Supongamos que se desea reducir la cantidad de atributos de un dataset X descripto por p atributos. Se definen las **componentes principales** de X como la secuencia de p vectores unitarios, donde el i-ésimo vector es la dirección que mejor ajusta los datos en tanto que es ortogonal a los primeros $i-1$ vectores. En este contexto, la dirección o línea que mejor ajusta los datos es definida como aquella que minimiza la distancia cuadrada promedio de todos los puntos a la línea.

Estas direcciones constituyen una base ortonormal en la cual las distintas dimensiones individuales están linealmente decorrelacionadas. Equivalentemente, la i-ésima componente principal puede ser definida como la dirección ortogonal a las primeras $i-1$ componentes

principales que maximiza la varianza de los datos proyectados. En la figura **5-8** se puede ver un ejemplo ilustrativo.

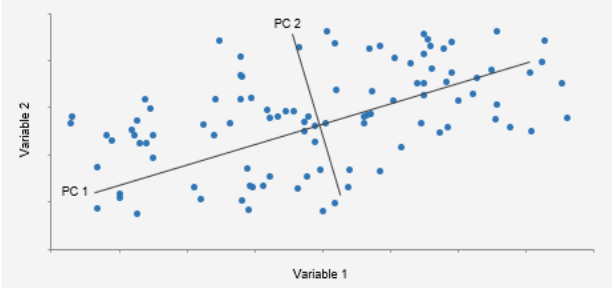


Figura 5-8.: En esta imagen se ilustran las dos componentes principales de un conjunto de datos bidimensional. La primer componente principal es aquella que maximiza la varianza de los datos proyectados, o, equivalentemente, la dirección que define la recta que mejor ajusta los datos. Nótese que las etiquetas de los datos no son utilizadas. Créditos: `statistixl.com`

PCA es el proceso de computar las componentes principales y usarlas para realizar un cambio de base en los datos, usualmente utilizando sólo las primeras $k < p$ componentes principales e ignorando las restantes.

En la figura **5-9** podemos ver el impacto de aplicar PCA como paso previo a SVM Lineal, en tanto que en la **5-10** se pueden apreciar los resultados del experimento equivalente en SVM RBF. Es importante remarcar que se calcula la transformación únicamente sobre los datos de entrenamiento, y esta misma transformación es aplicada a los datos de test.

Podemos ver que el uso de PCA tiene un efecto muy dispar en distintos pares de tiles testeados. Para ciertas combinaciones, utilizar PCA conduce a mejoras muy significativas, llegando a superar incluso la performance obtenida por Random Forests. Sin embargo, para otros pares de tiles el R-AUPRC disminuye drásticamente, lo cuál nos lleva a descartar este preprocesamiento para experimentos futuros.

5.2.3. Clustering de atributos

En esta sección se aplicó un último método de extracción de atributos no supervisado, llamado **clustering de atributos**. En primer lugar, se introducirán brevemente algunos conceptos de aprendizaje automatizado no supervisado para poder describir esta técnica.

Dado un dataset no etiquetado X con n elementos, la tarea de **clustering** [Jain and Dubes, 1988] consiste en dividir los n objetos en grupos, de tal forma que todos los objetos en un

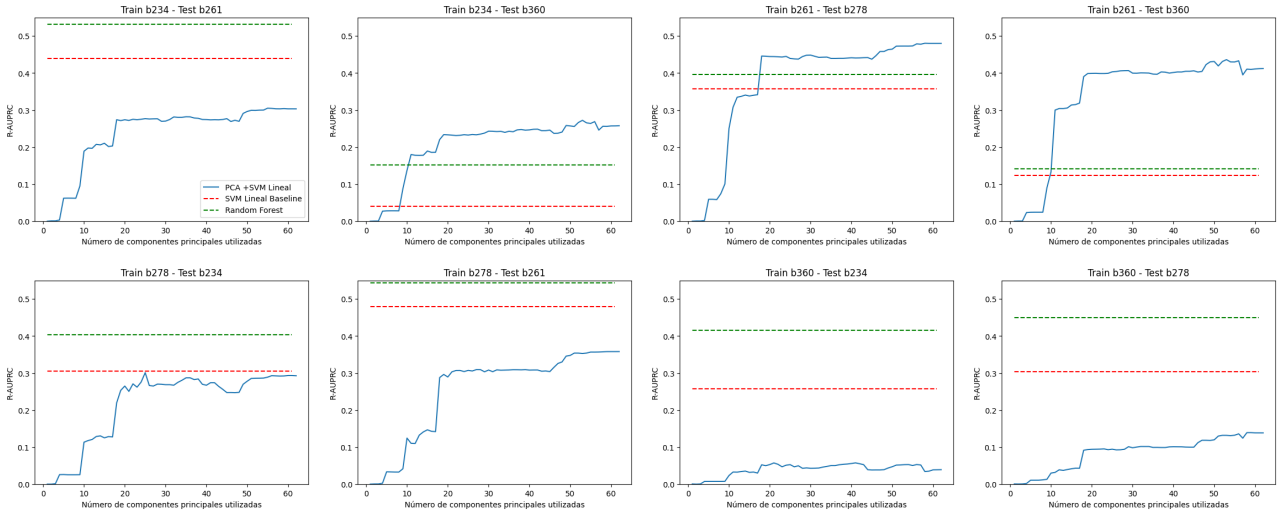


Figura 5-9.: Aplicación de PCA a SVM Lineal. Para cada par de tiles, se muestra el R-AUPRC obtenido al entrenar y testear usando las primeras k componentes principales. La línea punteada roja representa la base de referencia de SVM, obtenido sin utilizar extracción de atributos. La línea verde punteada representa la performance de RF.

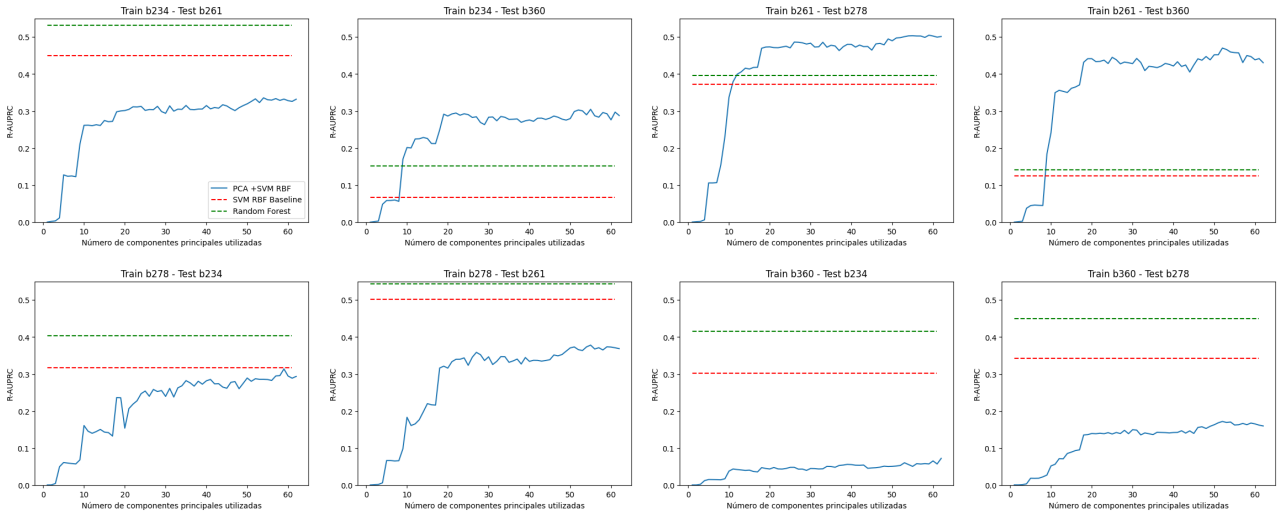


Figura 5-10.: Aplicación de PCA a SVM RBF. Para cada par de tiles, se muestra el R-AUPRC obtenido al entrenar y testear usando las primeras k componentes principales. La línea punteada roja representa el R-AUPRC de SVM-RBF obtenido sin utilizar extracción de atributos. La línea verde punteada representa la performance de RF.

mismo grupo (*cluster*) son más similares (en algún sentido) a aquellos que están en otros grupos (*clusters*).

Aquellas técnicas que construyen un anidado o jerarquía de clusters pertenecen al grupo de técnicas de **clustering jerárquico** [Jain and Dubes, 1988] [Rokach and Maimon, 2005]. Esto se ilustra en la figura **5-11**.

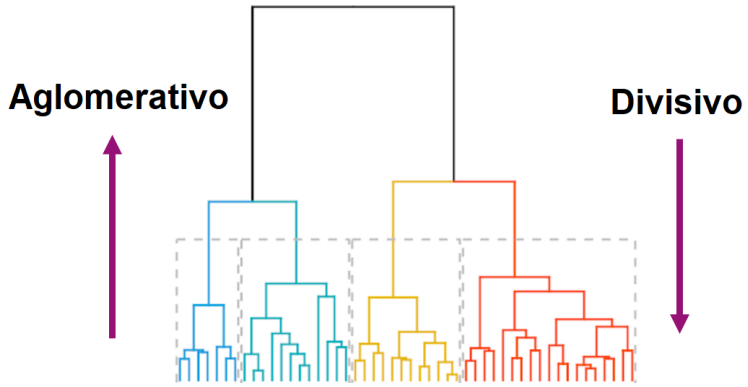


Figura 5-11.: Ejemplo de clustering jerárquico. Los elementos de un cierto dataset se agrupan en clusters que recursivamente pueden pertenecer a otros clusters. Créditos: Towards Data Science.

Los algoritmos de clustering jerárquico generalmente se dividen en dos grandes grupos [Rokach and Maimon, 2005] [Manning et al., 2008] :

- **Aglomerativos** (Bottom-up): Cada elemento del dataset comienza en su propio cluster, y pares de clusters son combinados recursivamente.
- **Divisivos** (Top-down): Todos los elementos del dataset comienzan en un mismo cluster, y divisiones son realizadas recursivamente.

En esta sección nos concentraremos en algoritmos de clustering jerárquico aglomerativos, que iterativamente combinan el par de clusters que minimiza una cierta métrica de disimilitud [Rokach and Maimon, 2005]. En este experimento se utilizaron las siguientes estrategias para definir los clusters:

- **Single linkage**: La distancia euclídea mínima entre pares de puntos, uno en cada cluster [Manning et al., 2008]. Esta métrica busca conectividad, no grupos compactos.
- **Complete linkage**: La distancia euclídea máxima entre pares de puntos, uno en cada cluster [Manning et al., 2008]. Intuitivamente, esta métrica busca agrupar conjuntos *completamente vecinos*, formando clusters compactos.

- **Average linkage**: Promedio de las distancias euclídea entre todas las observaciones de pares de clusters [Manning et al., 2008]. Esta métrica busca grupos conectados y compactos.
- **Ward**: Minimiza la suma de las diferencias euclídeas cuadradas dentro de todos los clusters [Ward, 1963]. Es un enfoque que busca minimizar la varianza.

En la figura **5-12** se pueden observar los clusters generados por las cuatro estrategias explicadas en tres datasets ilustrativos.

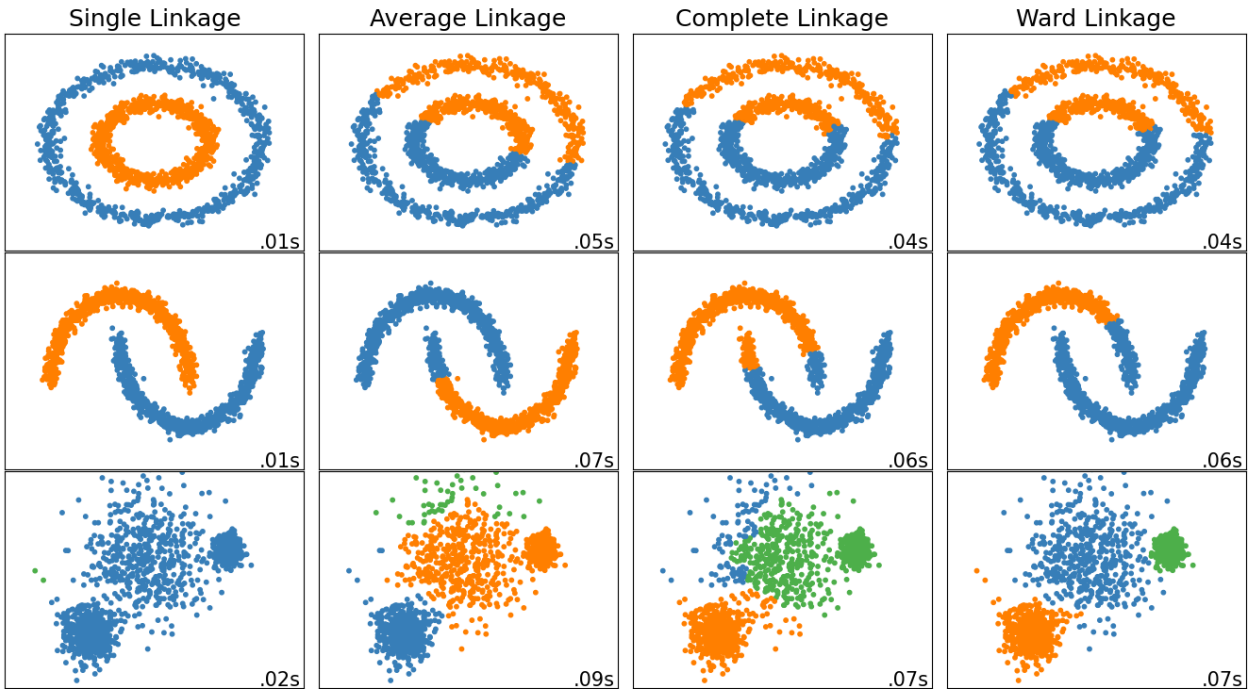


Figura 5-12.: Efecto de distintas estrategias en clustering jerárquico. Imagen tomada de la documentación oficial de Sklearn.

En esta sección se aplicó clustering para combinar atributos, en lugar de elementos de un dataset. En clustering aglomerativo estándar, el algoritmo recibe un dataset con n elementos descriptos por m atributos representado por una matriz $M \in \mathbf{R}^{n \times m}$. En **clustering de atributos** (feature agglomeration, en inglés) [Guyon and Elisseeff, 2003], el algoritmo de clustering aglomerativo simplemente recibe como argumento la transpuesta del dataset, M^T, e intentará combinar atributos que se comportan de forma similar, reemplazándolos por el centroide del cluster[1].

[1]Usualmente se utiliza el promedio, aunque se podría una función arbitraria (llamada *pooling function*) para definir cómo unificar clusters

Los resultados de estos experimentos pueden verse en las figuras **5-13** y **5-14** para SVM lineal y SVM RBF respectivamente. Las curvas generadas son notablemente similares a las obtenidas utilizando PCA.

La estrategia utilizada para hallar clústers no tiene un gran impacto en las curvas generadas. Al igual que en PCA, utilizar clustering de atributos produce aumentos muy interesantes de R-AUPRC en ciertos pares de tiles y desmejorías notables en otros; es por esto que se decidió no utilizar esta técnica en secciones posteriores.

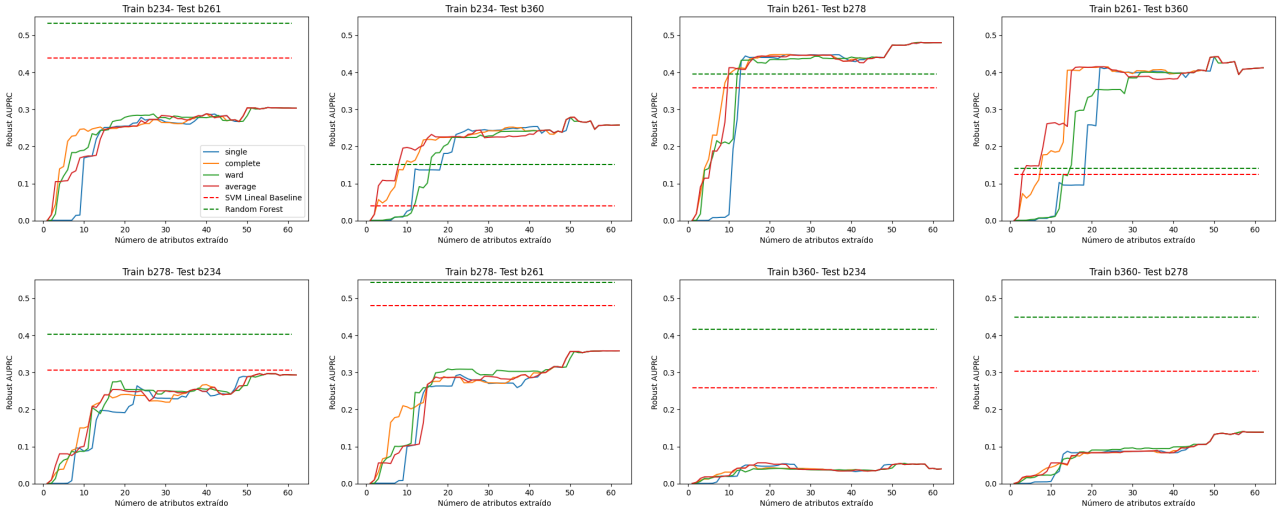


Figura 5-13.: Aplicación de clustering de atributos a SVM Lineal

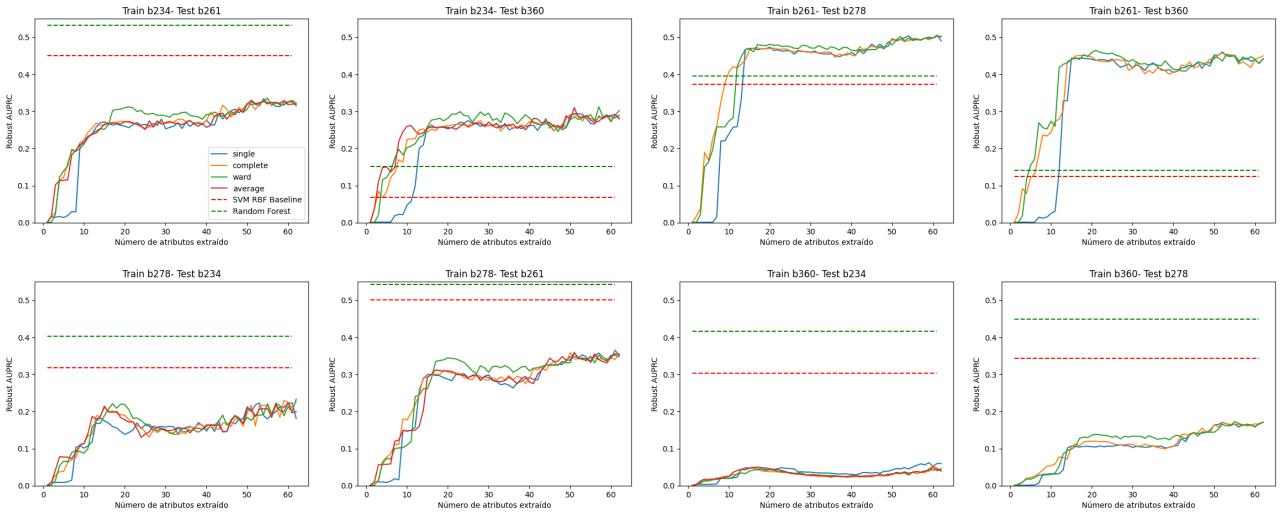


Figura 5-14.: Aplicación de clustering de atributos a SVM RBF

5.3. Conclusiones

En este capítulo se han explorado distintas técnicas para reducir la dimensionalidad de los datos. Ninguna de las técnicas estudiadas tuvo un impacto pronunciado en las curvas de precision-recall, por lo que la aplicación de estos métodos tiene sentido principalmente para reducir la complejidad temporal y espacial de nuestros clasificadores.

Dado que las técnicas de extracción de atributos probaron generar resultados bastante inestables, se decidió preferir el uso de selección de atributos (filtros univariable) en el resto de este trabajo. En base a los valores graficados en las figuras **5-3** y **5-6**, se decidió optar por aquellos hiperparámetros que maximizan la ganancia promedio en R-AUPRC, sujeto a no empeorar significativamente la performance en ningún par de tiles.

- Seleccionar los 48 atributos más importantes de acuerdo a $mutual\_info$ en SVM-Lineal.
- Seleccionar los 45 atributos más importantes de acuerdo a $mutual\_info$ en SVM-RBF.

En las tablas **5-1** y **5-2** podemos ver el impacto final de esta elección en R-AUPRC en SVM lineal y RBF respectivamente, en comparación a los clasificadores óptimos del capítulo anterior. La mejora en R-AUPRC es nula en SVM Lineal y modesta en SVM-RBF, y viene acompañada de una reducción en complejidad temporal y espacial. Esto último es especialmente notable en SVM-RBF, donde los tiempos de entrenamiento se redujeron significativamente.

Train tile	Test tile	RF	SVM-L sin FS	SVM-L con FS	Gain
b278	b234	0.40	0.31	0.31	0.00
b278	b261	0.54	0.48	0.48	-0.00
b278	b360	0.21	0.14	0.13	-0.00
b234	b278	0.40	0.29	0.30	0.01
b234	b261	0.53	0.44	0.44	0.00
b234	b360	0.15	0.04	0.04	0.00
b261	b278	0.40	0.36	0.36	-0.00
b261	b234	0.39	0.30	0.30	-0.00
b261	b360	0.14	0.12	0.12	0.00
b360	b278	0.45	0.30	0.30	-0.00
b360	b234	0.42	0.26	0.25	-0.00
b360	b261	0.54	0.41	0.41	-0.01
avg		0.38	0.29	0.29	0

Tabla 5-1.: En esta tabla podemos ver el incremento nulo en R-AUPRC que se obtiene al utilizar selección de atributos en SVM-Lineal.

Train tile	Test tile	RF	SVM-RBF sin FS	SVM-RBF con FS	Gain
b278	b234	0.40	0.32	0.33	0.01
b278	b261	0.54	0.50	0.52	0.02
b278	b360	0.21	0.16	0.18	0.02
b234	b278	0.40	0.30	0.34	0.04
b234	b261	0.53	0.45	0.47	0.02
b234	b360	0.15	0.07	0.09	0.02
b261	b278	0.40	0.37	0.39	0.02
b261	b234	0.39	0.33	0.33	0.00
b261	b360	0.14	0.12	0.15	0.02
b360	b278	0.45	0.34	0.36	0.02
b360	b234	0.42	0.30	0.32	0.02
b360	b261	0.54	0.45	0.45	0.00
avg		0.38	0.31	0.33	.02

Tabla 5-2.: En esta tabla podemos ver el incremento en R-AUPRC que se obtiene al utilizar selección de atributos en SVM-RBF.

6. Inspeccionando los datos

En este capítulo se aplicarán distintas técnicas con el objeto de entender en mayor detalle los datos astronómicos con que estamos trabajando, así como descubrir diferentes características del problema de clasificación de RRL. Tener un mayor entendimiento de los datos que se está manipulando puede ayudar a entender por qué RF funciona mejor que SVM.

6.1. Importancia de atributos

En esta sección se intentó determinar qué atributos son más relevantes a la hora de clasificar RRLs. Para ello se utilizarán:

- Tests estadísticos univariable, como los utilizados en la sección 5.1.2
- Ranking de importancia asignados por los clasificadores RF y SVM ya entrenados.

Contemplaremos todos los atributos en esta sección, es decir, se omitirá el paso de selección de atributos.

6.1.1. Importancia de atributos basada en tests estadísticos

Se aplicaron nuevamente tests estadísticos a cada uno de los 62 atributos de los datasets, esta vez con la intención de entender qué atributos son más importantes para predecir RRLs.

Sea x_i la variable aleatoria que rige el $i-$ésimo atributo de nuestro dataset ($0 \leq i < 62$), discretizado utilizando 100 bins quantile. Por lo tanto, x_i es una variable aleatoria discreta para $0 \leq i < 62$. Sea y la variable aleatoria discreta a predecir, que indica si una dada estrella es RRL o no.

Los dos tests estadísticos utilizados analizarán cada atributo x_i, planteando el siguiente test de hipótesis [Meyer, 1970]:

H_0: La variable x_i es independiente de y
H_1: La variable x_i no es independiente de y

Si la hipótesis nula es cierta, entonces x_i es irrelevante por sí misma para predecir y.

Test χ^2

El test Pearson χ^2 mide la dependencia entre dos variables estocásticas [Cochran, 1952]. Para explicar intuitivamente cómo funciona, consideremos el clásico dataset Titanic, en el cuál se intenta predecir si un pasajero sobreviviría al hundimiento en base a características como su género, edad y cantidad de familiares en el barco.

Supongamos que deseamos determinar si el atributo género es relevante o no para este problema:

H_0: El atributo género es independiente de la supervivencia de un pasajero.
H_1: El atributo género no es independiente de la supervivencia de un pasajero.

Con una simple visualización de los datos, como la mostrada en la figura **6-1**, podemos apreciar que la proporción de hombres y mujeres no es la misma para sobrevivientes que para no sobrevivientes. Esto nos indica, intuitivamente, que el género de una persona es ciertamente informativo para predecir la supervivencia de un pasajero. En la figura **6-2** (a) podemos ver estas frecuencias en una matriz de confusión.

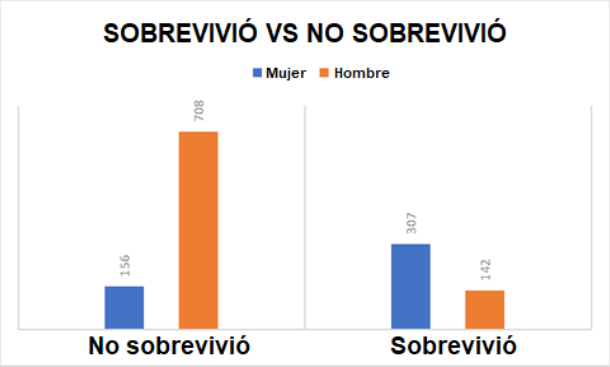


Figura 6-1.: Supervivencia de pasajeros del Titanic discriminada por género. Créditos: Dr. Saptarsi Goswami

	No Sobrev	**Sobrev**	total
Mujer	156	307	463
Hombre	708	142	850
total	864	449	1313

(a) Frecuencias observadas

	No Sobrev	**Sobrev**	total
Mujer	304,67	158.33	463
Hombre	559.33	290.67	850
total	864	449	1313

(b) Frecuencias esperadas

Figura 6-2.: Frecuencias en el dataset Titanic. Hay $n = 1313$ personas en este dataset.

Por otro lado, asumiendo que la hipótesis nula (independencia) es cierta, la siguiente igualdad vale:

$$P(genero \wedge sobrevivio) = P(genero) * P(sobrevivio)$$

Bajo esta asunción, se puede calcular la tabla de **frecuencias esperadas** de la figura **6-2** (b). Por ejemplo, para la primera celda, correspondiente a cuántas mujeres no sobrevivieron:

$$\begin{aligned} E_1 &= n * P(genero = Mujer \wedge sobrevivio = NoSobrevivio) \\ &= n * P(genero = Mujer) * P(sobrevivio = NoSobrevivio) \\ &= 1313 * \frac{463}{1313} * \frac{864}{1313} \\ &= 304{,}67 \end{aligned}$$

Si la hipótesis nula fuese cierta, las frecuencias de las dos tablas de la figura **6-2** deberían ser iguales celda a celda. χ^2 mide la disparidad entre ambas, y se define como sigue:

$$\chi^2 = \sum_i \frac{(O_i - E_i)^2}{E_i}$$

Donde O_i y E_i son la frecuencia observada y esperada para la i-ésima entrada de la tabla de confusión de las variables a testear. En la siguiente tabla podemos ver cómo se calcula el valor $\chi^2 = 327{,}7$ para nuestro ejemplo:

Género	**Sobrevivió**	O_i	E_i	$\frac{(O-E)^2}{E}$
Mujer	Sí	307	158.33	139.59
Mujer	No	156	304.67	72.54
Hombre	Sí	142	290.77	76.04
Hombre	No	708	559.33	39.51
χ^2 value				327.70

El valor χ^2 puede ser interpretado para decidir si rechazar o aceptar la hipótesis nula, donde un mayor valor de χ^2 indica evidencia en contra de la hipótesis nula. La interpretación se realiza calculando el p-value correspondiente al valor de χ^2 obtenido[1], que en conjunto con un nivel de significación α deseado puede interpretarse como:

- p-value $\leq \alpha$: Rechazar H_0, las variables son dependientes.
- p-value $> \alpha$: No rechazar H_0, las variables son independientes.

Para nuestro ejemplo, la diferencia entre las tablas es tan grande que el p-value asociado es ~ 0, por lo que la hipótesis nula es rechazada. El género de una persona es importante a la hora de predecir si sobrevivió o no. Aplicaremos este test estadístico a los atributos de los datasets de Carpyncho en la sección 6.1.1.

[1]La correspondencia entre χ^2 y p-values suele estar tabulada, pues es bastante compleja de calcular.

ANOVA F test

ANOVA (ANalysis Of VAriance) es un método estadístico utilizado para chequear si la media de dos o más grupos de datos son significativamente diferentes [Moore and McCabe, 1989] [Han et al., 2012]. Las hipótesis son:

H_0: Las medias de todos los grupos son iguales.
H_1: Al menos la media de uno de los grupos es distinta.

El resultado del test es una medida estadística, F score, que es inversamente proporcional al p-value asociado al test estadístico.

ANOVA puede ser utilizado para decidir si un cierto atributo de un dataset es relevante o no para predecir una variable objetivo. La figura **6-3** provee un ejemplo ilustrativo: se presenta un dataset con dos atributos x_1 (horizontal) y x_2 (vertical), en tanto que la variable a predecir y está demarcada con colores (rojo y azul). Se desea asignar un puntaje a cada atributo indicando cuán bien cada uno discrimina entre rojos y azules.

Al proyectar los elementos de cada color por separado en cada eje, podemos apreciar que x_1 es mucho mejor separando las clases que x_2, por dos motivos:

- Las medias de las distribuciones de probabilidad proyectadas están más separadas.
- Las distribuciones de probabilidad proyectadas son más compactas.

Estas dos cualidades son exactamente lo que el test ANOVA mide, numéricamente, para cada atributo. Como resultado, obtendremos un F score para cada atributo. El F score de x_1 será mayor que el F score asignado a x_2. Finalmente, el F-score de cada variable podrá ser convertido a un p-value para decidir si la hipótesis nula del test ANOVA puede ser rechazada o no. Si la hipótesis nula es rechazada, se puede concluir que el atributo en cuestión es relevante a la hora de predecir y.

Experimentos con tests estadísticos

Habiendo introducido brevemente los dos tests estadísticos que se utilizaron, se procederá a reportar los resultados de aplicar ambos tests a cada uno de los 62 atributos utilizados por Carpyncho para describir estrellas RRL. En la figura **6-4** podemos ver los p-values obtenidos por ambos tests para cada atributo del dataset *b278*. Gráficas similares para *b234*, *b360* y *b261* se encuentran en el anexo B.1.

Recordemos que altos p-values indican mayor evidencia a favor de la hipótesis nula. Es decir, altos p-values indican independencia entre el atributo en cuestión y la variable objetivo, por

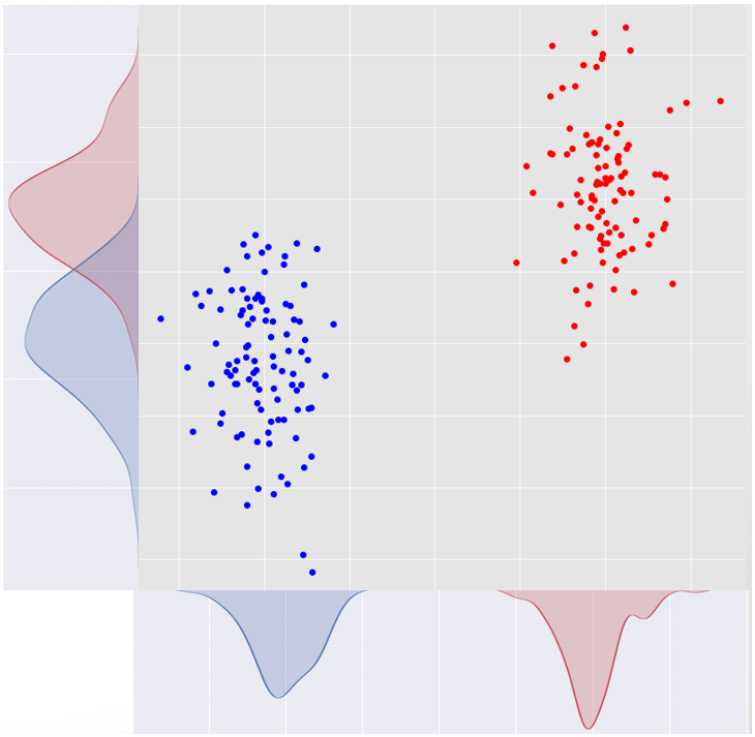


Figura 6-3.: Ejemplo de ANOVA F-value test. Créditos: Kasra Manshaei

lo tanto tales atributos son considerados irrelevantes para detectar RRLs por si mismos. Tomando un nivel de significación $\alpha = 0{,}05$, los siguientes atributos son consistentemente clasificados como irrelevantes de acuerdo a ANOVA:

- Freq1_harmonics_rel_phase_1, Freq1_harmonics_rel_phase_2, Freq1_harmonics_rel_phase_3
- Freq2_harmonics_rel_phase_1, Freq2_harmonics_rel_phase_2, Freq2_harmonics_rel_phase_3
- Freq3_harmonics_rel_phase_1, Freq3_harmonics_rel_phase_2, Freq3_harmonics_rel_phase_3
- Amplitude
- LinearTrend
- PairSlopeTrend

Utilizando el mismo nivel de significación, χ^2 parece ser un poco más conservador a la hora de declarar atributos como irrelevantes. Más aún, las conclusiones extraídas por χ^2 no son completamente consistentes entre distintos tiles. Los siguientes atributos son declarados como irrelevantes en la mayoría de los tiles:

- Freq1_harmonics_rel_phase_1, Freq1_harmonics_rel_phase_2

- Freq2_harmonics_amplitude_phase_2
- Freq2_harmonics_rel_phase_2, Freq2_harmonics_rel_phase_3
- Freq3_harmonics_rel_phase_1, Freq3_harmonics_rel_phase_2
- PairSlopeTrend

Podemos ver que ambos tests coinciden en que varios de los atributos que describen las componentes de Fourier parecen ser irrelevantes a la hora de diferenciar RRLs. Estos atributos están entre los primeros en ser eliminados en los experimentos de selección de atributos (ver sección 5.1.2) que se realizaron en el capítulo anterior, y se ha verificado que eliminarlos no reduce performance, de hecho la mejora en SVM-RBF.

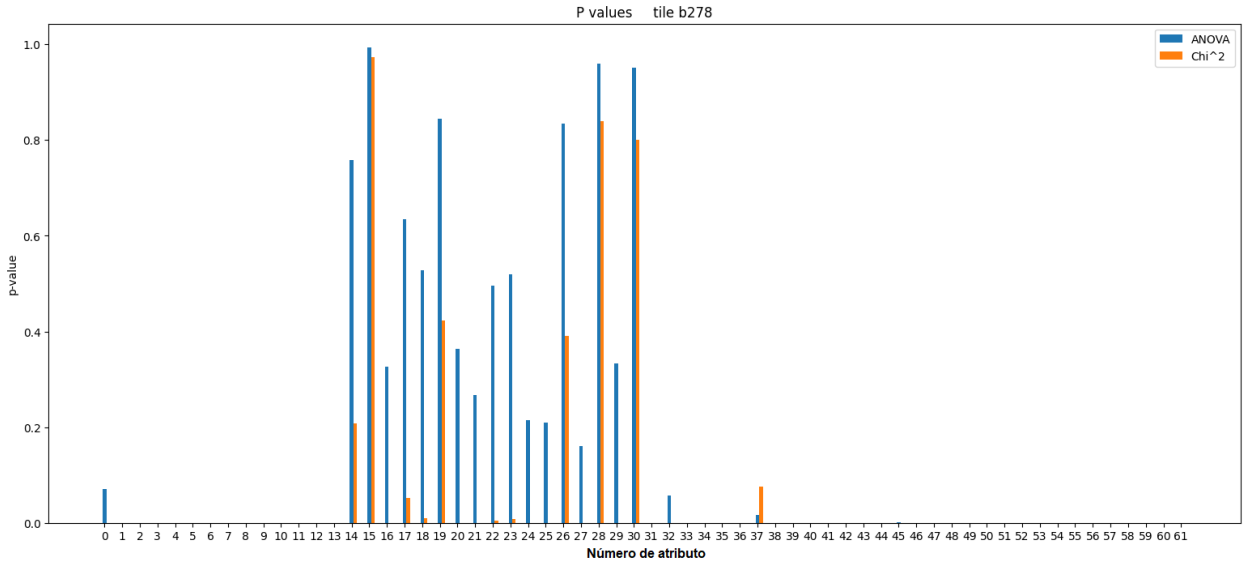


Figura 6-4.: p-values resultantes de los tests estadísticos ANOVA y χ^2 aplicados a cada atributo del tile b278. En el anexo A.3 puede consultarse la lista completa de atributos.

Por otro lado, si se desea estudiar qué atributos son más importantes, podemos inspeccionar cuáles de ellos maximizan las métricas χ^2, F-score e información mutua (ver sección 5.1.2). Nótese que, a diferencia de los p-values, estas métricas no se encuentran en una misma escala; y los valores absolutos obtenidos no son directamente comparables entre sí.

En la figura **6-5** se presentan las métricas para cada atributo en b278, unificadas en una misma gráfica. Cada métrica fue normalizada de forma tal que la sumatoria de los puntajes en todos los atributos es 1. Gráficas similares para *b234*, *b360* y *b261* se encuentran en el

anexo B.2.

En la práctica, la forma de utilizar estos puntajes muchas veces consiste en seleccionar los k atributos que maximizan la métrica utilizada. En este contexto, los valores absolutos de los puntajes no son relevantes, y resulta más informativo considerar la posición en el ranking que cada atributo recibe. Esto puede visualizarse en la figura **6-6**, en tanto que las gráficas correspondients a *b234*, *b360* y *b261* se encuentran en el anexo B.3.

- En primer lugar, veamos que mutual information presenta poca variación en sus puntajes, a excepción de cuatro atributos que son consistentemente destacados: Autocolor_length, Beyond1Std, PairSlopeTrend y Period_fit.
- El F-score destaca consistentemente los atributos Autocolor_length y Period_fit. También reciben puntajes considerables los atributos FluxPercentile y las primeras componentes de fourier, así como los índices Psi_CS y Psi_eta y algunos atributos de color.
- χ^2 destaca Psi_CS, Psi_eta, así como algunos atributos de color (51 52 57 58), FluxPercentile, las primeras componentes de Fourier, y Period_fit.

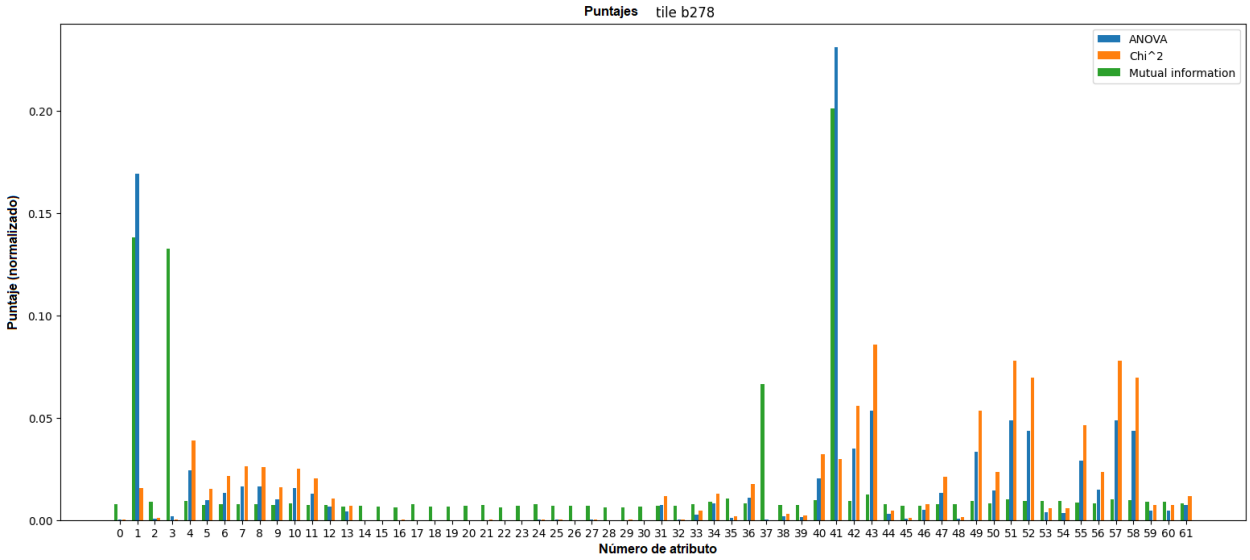


Figura 6-5.: Métricas χ^2, ANOVA F-score e información mutua calculadas para cada atributo del tile b278. En el anexo A.3 puede consultarse la lista completa de atributos.

6.1.2. Importancia de atributos basada en clasificadores

Los clasificadores RF pueden proveer un puntaje (importancia) a cada atributo del dataset de entrenamiento, llamado importancia Gini (o reducción media en impuridad, MDI) [Has-

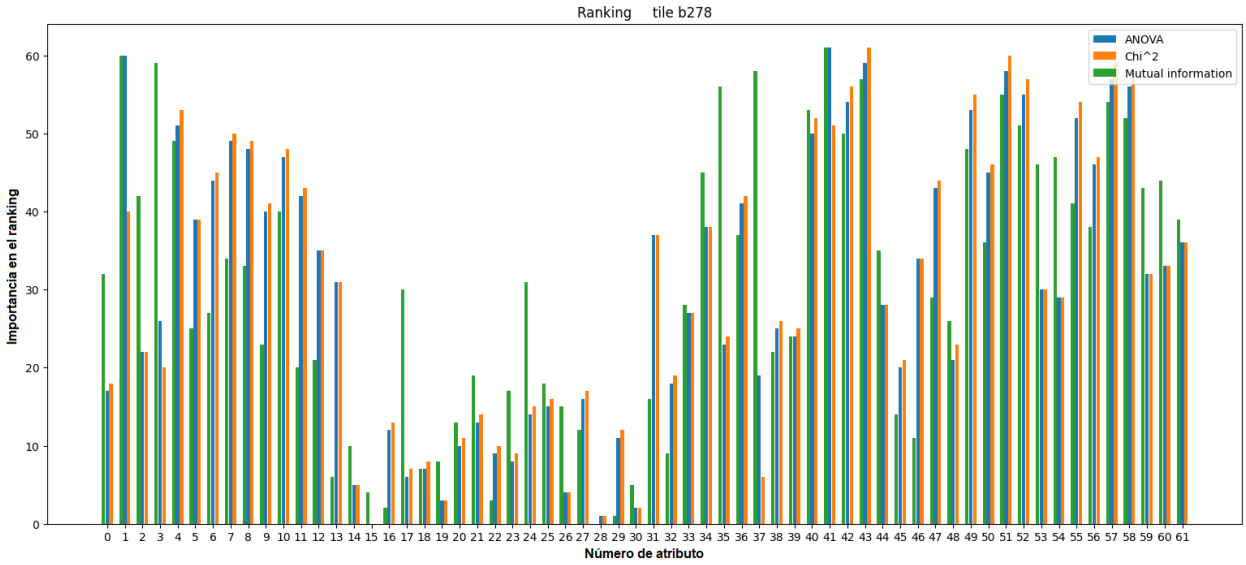


Figura 6-6.: Ranking de atributos obtenido por las métricas χ^2, ANOVA F-score e información mutua calculadas para cada atributo del tile b278. En el anexo A.3 puede consultarse la lista completa de atributos.

tie et al., 2001]. Para cada atributo, este valor se calcula como la cantidad de veces que ese atributo se utiliza como criterio de división en un nodo (en todos los árboles), ponderado por la cantidad de instancias que ese nodo divide. Nótese que se utilizan instancias 'out of the bag' para evitar realizar sobreajuste.

Por otro lado, luego de entrenar SVM con kernel lineal, podemos inspeccionar el valor absoluto del vector de pesos w para comprender cuánta importancia SVM le asigna a cada atributo [Rakotomamonjy, 2003]. Esto no es posible para SVM RBF pues los datos son transformados a otro espacio, por lo tanto el plano separador existe en otro espacio y los coeficientes no tienen una relación trivial con los atributos del espacio original.

En la figura **6-7** podemos ver la importancia (normalizada) que SVM-Lineal y RF asignan a cada atributo en el tile b278. Las gráficas para los tiles restantes se encuentran en el anexo B.4.

Los rankings de atributos generados por Random Forest son muy consistentes entre distintos tiles. Se puede observar que:

- El atributo más importante es **PeriodFit** seguido por **Psi_eta**. Nótese que tanto ANOVA como mutual information consistentemente colocaban a PeriodFit como el atributo más importante y a Psi_eta entre los 5 más importantes. Por otro lado, el ranking de χ^2 posiciona a Psi_eta con la máxima importancia pero en algunos tiles

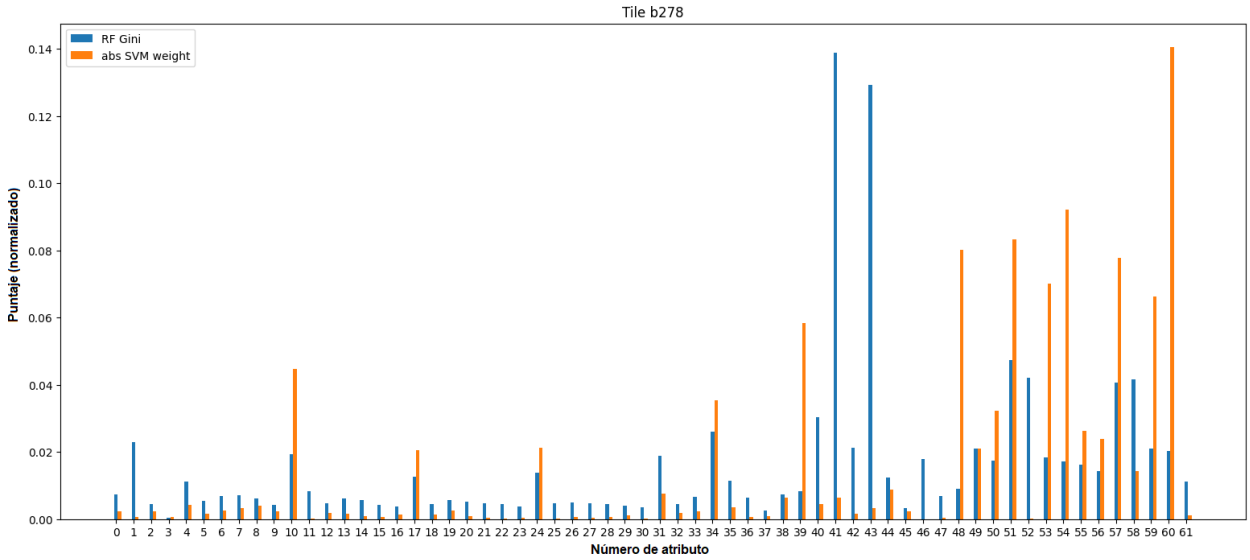


Figura 6-7.: Importancia asignada por RF y SVM-Lineal a cada atributo del tile b278. En el anexo A.3 puede consultarse la lista completa de atributos.

fallaba en detectar la verdadera importancia de PeriodFit.

- Con aproximadamente la mitad de importancia que los dos atributos recién mencionados, RF también considera importantes a Psi_CS, c89_jh_color y c89_jk_color, n_09_jh_color y n_09_jk_color, y Autocor_length. Con distintos grados de precisión, los tests estadísticos efectivamente marcan a estos atributos como relevantes para detectar RRLs.

Por otro lado, luego de analizar a qué atributos SVM-Lineal está ponderando, podemos observar que se está prestando alta atención a los atributos de color (48 a 60). Se puede también observar que SVM ignora atributos clave como PeriodFit y Psi_eta que, como ya hemos validado a través de tests estadísticos e importancia gini, son altamente informativos para detectar RRLs.

6.1.3. Importancia de atributos basada en permutaciones

Una tercer forma de estimar la importancia de cada atributo a la hora de identificar estrellas de tipo RRL es conocida como *importancia de atributos basada en permutaciones* (del inglés, permutation feature importance), y fue introducida en [Breiman, 2001], y posteriormente extendida en [Fisher et al., 2018]. Esta técnica de inspección de modelos puede ser utilizada con cualquier clasificador de aprendizaje automatizado, y es especialmente útil para estimar la importancia de cada atributo al utilizar modelos no lineales u opacos (como SVM RBF).

El concepto es muy sencillo: Medimos la importancia de cada atributo calculando el incremento en el error de predicción luego de permutar (aleatoriamente) todos los valores de dicho atributo [Molnar, 2019]. Esta permutación destruye la relación entre el atributo y la variable a predecir.

Un atributo se considera importante si permutar sus valores incrementa el error del modelo, pues en ese caso el modelo estaba utilizando dicho atributo para la predicción. Por otro lado, un atributo es poco importante si al permutar sus valores el error del modelo no incrementa [Molnar, 2019].

En la figura **6-8** podemos ver la importancia basada en permutaciones de cada atributo para modelos Random Forest, SVM lineal y SVM RBF en b278. Las gráficas correspondientes a b234, b261 y b360 se encuentran en el anexo B.5

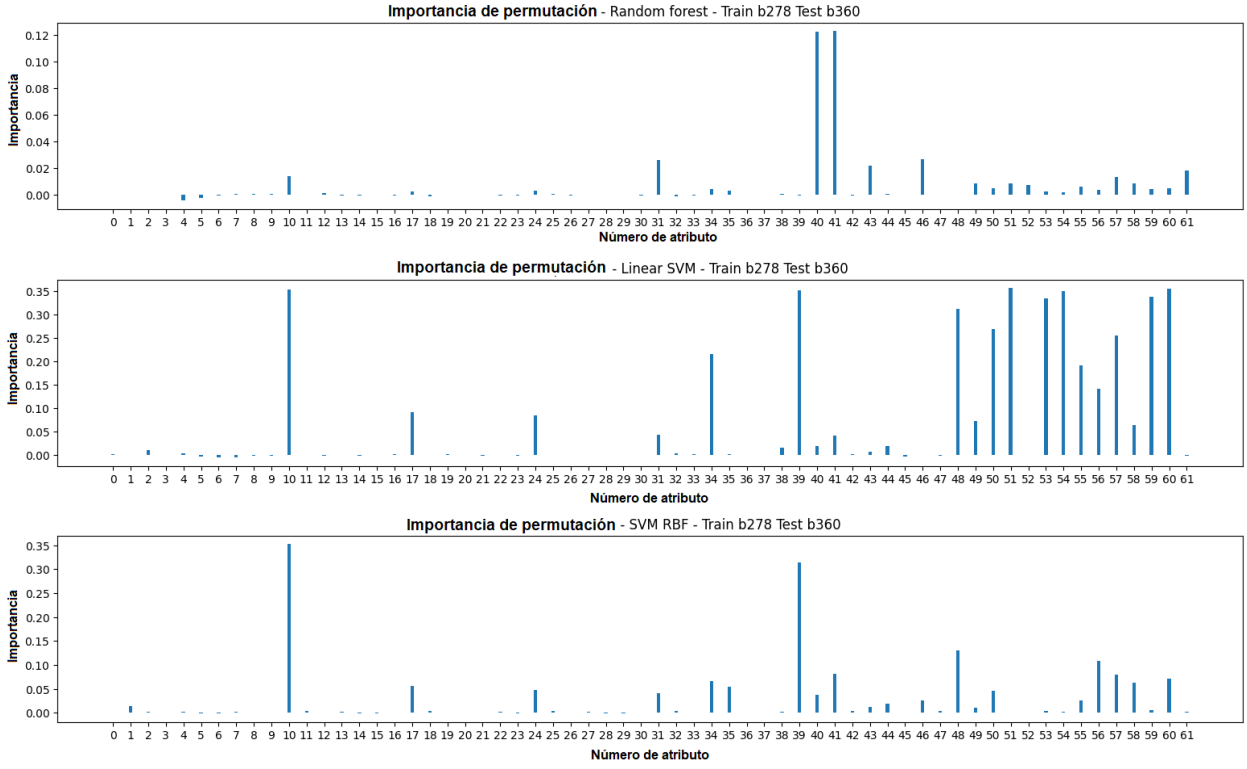


Figura 6-8.: Importancia de atributos basada en permutaciones para cada atributo del tile b278. La importancia de un atributo se define como la reducción en performance al permutar sus valores. En el anexo A.3 puede consultarse la lista completa de atributos.

Podemos observar que:

- La estimación de importancia de RF destaca los atributos 41, 40, 46, 31, 43, 61 y 10, y no se condice completamente con la obtenida utilizando la importancia Gini.

- La estimación de importancia de SVM Lineal se condice con la obtenida en la sección anterior utilizando el vector de coeficientes w. Nuevamente podemos ver que SVM Lineal considera muy importantes a los atributos de color (48-60), en tanto que considera irrelevantes a mayoría de los atributos que RF considera importantes.
- La estimación de importancia de SVM RBF tiene una cierta similitud a la de SVM Lineal, con la notable diferencia de que los atributos de color dejan de tener importancia mayor. Este puede ser el motivo pro el que SVM RBF funciona mejor que SVM Lineal, dado que la importancia basada en RF y en tests estadísticos indica que los atributos de color son de importancia menor.
- Es interesante resaltar que algunos atributos clave para SVM (10 y 39) son muy poco informativos para RF. Recíprocamente, atributos clave para RF (40 y 41) no son particularmente importantes para SVM. Esto parece indicar que ambos clasificadores están haciendo uso de atributos completamente distintos para producir su decisión.

6.2. Análisis de correlaciones

6.2.1. Correlación entre atributos

En esta sección nos concentraremos en entender las distintas correlaciones presentes entre los atributos de nuestros datasets, sin tener en cuenta la variable a predecir. Para ello, se computó un coeficiente de correlación en $[-1, 1]$ para cada par de atributos, que mide cuán fuertemente correlacionado cada par está. Un valor de ± 1 indica un grado de asociación perfecto entre los atributos, en tanto que valores cercanos a 0 indican una relación más débil. El signo indica si la relación es directa o inversa.

A la hora de realizar tales cálculos, es muy importante utilizar una métrica de correlación adecuada [Khamis, 2008]. Por ejemplo, la métrica más comúnmente utilizada, el coeficiente de correlación de Pearson [Pagano, 2010], sería inadecuada para nuestros atributos, dado que requiere que ambos atributos estén distribuidos de forma normal y sean continuos.

Dado que nuestros atributos (luego de binning) son de tipo discretos ordinales (con a lo sumo 100 valores ordenados), resulta apropiado el uso de ténicas que midan la semejanza de ordenamientos (o rankings) de los datos [Khamis, 2008] [Pagano, 2010], tales como el **Coeficiente de correlación de rango de Kendall** (τ, [Puth et al., 2015]) y el **Coeficiente de correlación de rango de Spearman** (ρ, [Puth et al., 2015])

Tanto ρ como τ son capaces de medir correlaciones monotónicas arbitrarias, a diferencia del coeficiente de Pearson que es únicamente capaz de medir correlaciones lineares [Pagano, 2010]. En la figura **6-9** podemos ver la matriz de correlación utilizando el coeficiente de

Spearman, en tanto que en la figura **6-10** podemos ver la matriz de correlación de acuerdo al coeficiente de Kendall. Las matrices de correlación de los tiles restantes se encuentran en el anexo B.6.

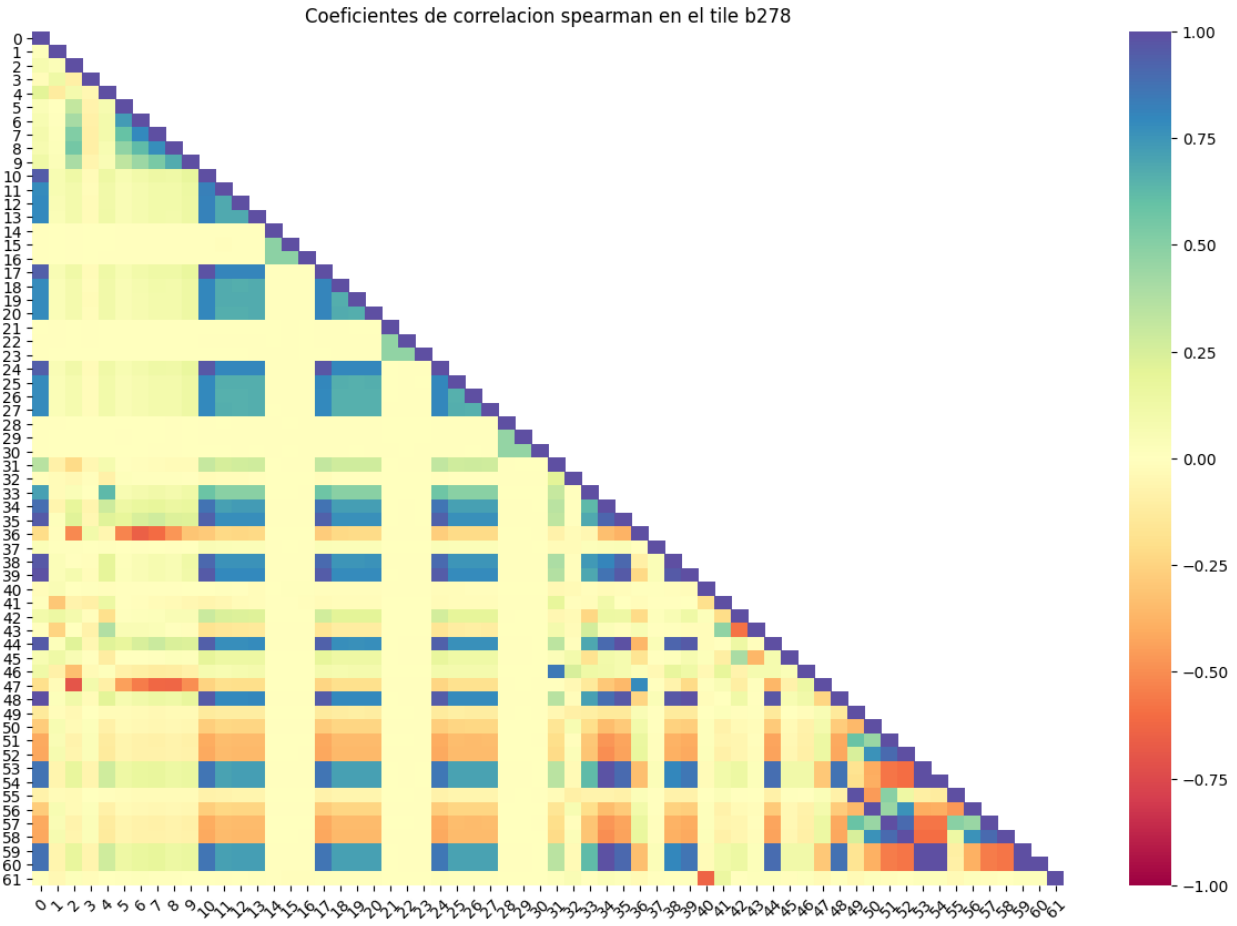


Figura 6-9.: Correlaciones lineales (métrica de Spearman) en el tile b278. En el anexo A.3 puede consultarse la lista completa de atributos.

Podemos observar que los coeficientes de correlación son bastante consistentes entre distintos tiles para ambos métodos. Ambos métodos, τ y ρ, identifican los mismos grupos de atributos como correlacionados; aunque los valores absolutos de correlación obtenidos por τ son ligeramente menores. Este fenómeno es esperable, dado que τ tiende a producir valores absolutos más pequeños que ρ.

Los bloques de atributos correlacionados más prominentes son:

- Los **atributos de color** (49 a 60) están, naturalmente, altamente correlacionados entre sí. Podemos notarlo porque la esquina inferior derecha de la matriz muestra altos valores entre la mayoría de los pares; así como con los atributos **Mean** y **Amplitude**.

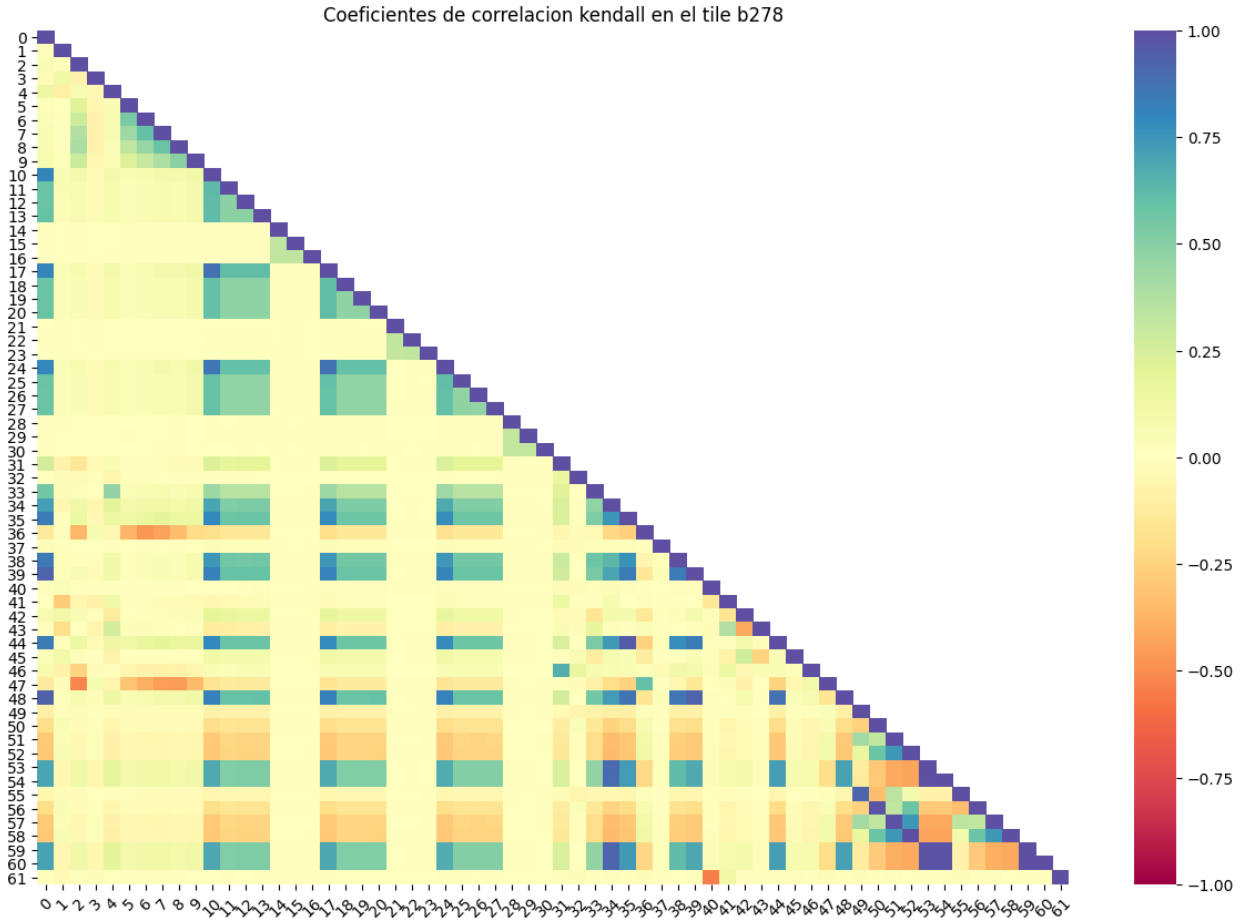


Figura 6-10.: Correlaciones lineales (métrica de Kendall) en el tile b278. En el anexo A.3 puede consultarse la lista completa de atributos.

- Los atributos **Amplitude**, **MedianAbsDev**, **PercentAmplitude**, **PercentDifferenceFluxPercentile**, **Q31** y **Std** tienden a estar altamente correlacionados entre sí. Esto es esperable, dado que todos estos atributos son estadísticas descriptivas calculadas a partir de las magnitudes observadas.
- **Beyond1Std**, **SmallKurtosis** y **MedianBRP**
- Naturalmente, los atributos **FluxPercentileRatioMid_*x*** para $x = 20, 35, 50, 65, 80$ exponen una leve correlación. Esta correlación no es tan fuerte, lo cuál indica que estos atributos contienen información ligeramente distinta.
- Los atributos que describen las amplitudes de las primeras tres componentes de Fourier de los tres primeros períodos candidatos (atributos 10-13, 17-20 y 24-27) están todos correlacionados entre sí. Esto indica que los primeros tres períodos candidatos de cada RRL son similares en amplitud.

Numéricamente, algunas de las correlaciones son extremadamente fuertes, alcanzando valores absolutos mayores a 0.99, lo cuál indica que ciertos pares de atributos contienen información completamente redundante.

6.2.2. Eliminación de atributos correlacionados

En esta sección se analizó si eliminar atributos altamente correlacionados trae algún beneficio interesante en las curvas de precision-recall obtenidas utilizando SVM. Para ello, se realizó el siguiente experimento dado un tile de entrenamiento y un tile de testing:

1. Calcular la matriz de correlaciones de Spearman en el dataset de entrenamiento.
2. Identificar el par de atributos tal que el valor absoluto de su coeficiente de correlación es máximo.
3. Seleccionar un atributo del par, y eliminarlo del dataset de entrenamiento y de test
4. Entrenar SVM utilizando el dataset de training resultante, y calcular el R-AUPRC en test.
5. Repetir (1) hasta que no queden más atributos.

En las figuras **6-11** (SVM-L) y **6-12** (SVM-RBF) podemos ver el efecto R-AUPRC obtenido cada vez que se elimina el atributo más correlacionado restante, así como la base de referencia obtenido en la sección 4.7.

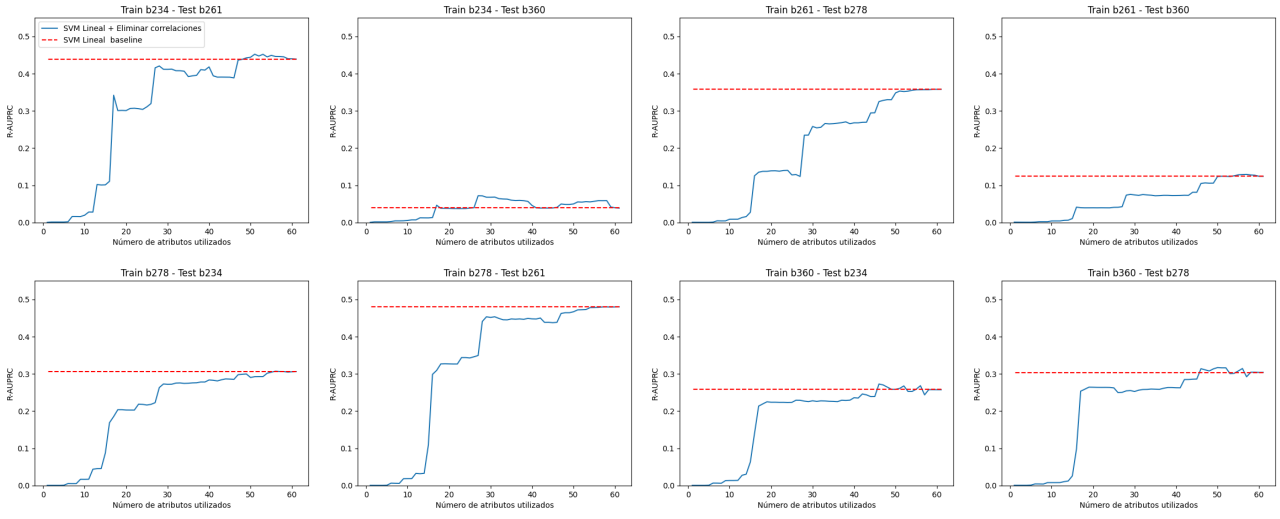


Figura 6-11.: Efecto de eliminar atributos altamente correlacionados en SVM Lineal

Para resumir la información de las figuras **6-11** y **6-12**, se calculó la distancia promedio y mínima a la base de referencia de todos los tiles. Estos cálculos están graficados en la figura **6-13**, y nos permiten concluir que:

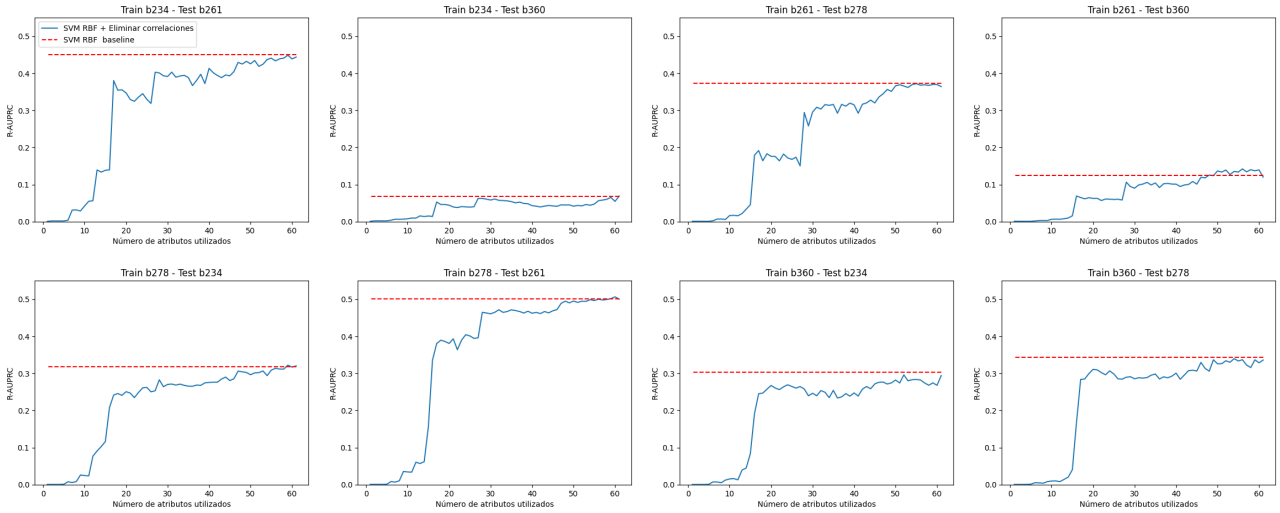


Figura 6-12.: Efecto de eliminar atributos altamente correlacionados en SVM RBF

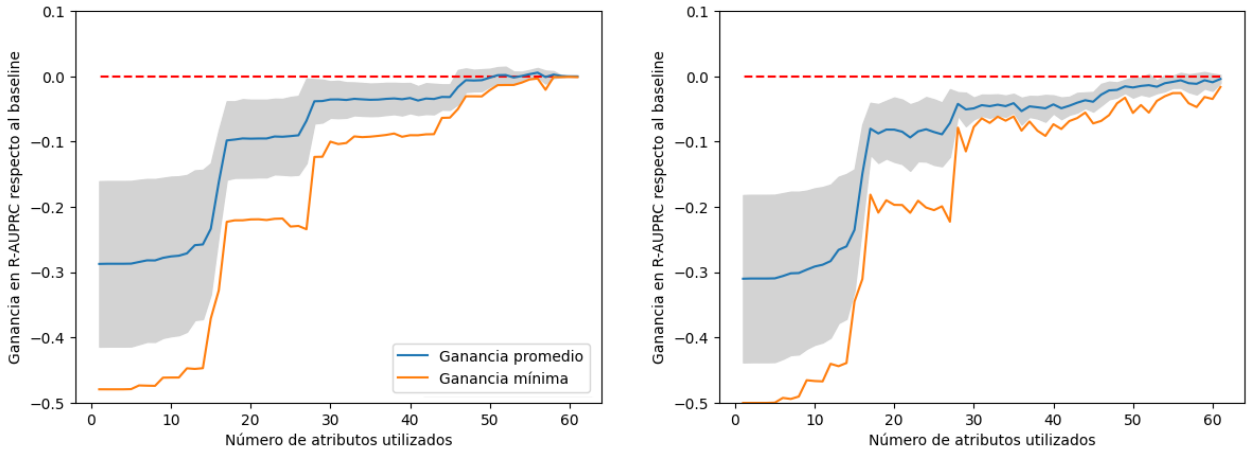
- En SVM Lineal, eliminar atributos altamente correlacionados no nos permite mejorar (en promedio) las curvas de precision-recall respecto a la base de referencia. Esto es válido incluso si nos remitimos a eliminar únicamente aquellos que muestran una correlación muy fuerte.
- Si bien no hay mejora en las curvas de precision-recall, en SVM Lineal podemos eliminar los primeros $\sim$ 17 atributos más correlacionados sin provocar pérdida de R-AUPRC en promedio. Esto indica que los atributos altamente correlacionados no contienen información vital para SVM-Lineal.
- A diferencia de SVM-Lineal, la performance de SVM-RBF se ve perjudicada al eliminar incluso los atributos con mayor coeficiente de correlación.

Como conclusión de esta sección, podemos decir que la existencia de atributos altamente correlacionados no tiene un impacto negativo en la performance de los clasificadores SVM. De hecho, prescindir de los atributos más correlacionados no mejora en absoluto la performance de los clasificadores estudiados. Las correlaciones entre atributos no son el motivo por el cuál SVM funciona peor que RF.

6.3. Distribuciones de probabilidad subyacentes

6.3.1. Histogramas de frecuencia

En esta última sección simplemente se intentará comprender la distribución de probabilidad subyacente a cada uno de los atributos que estamos utilizando para describir RRLs. En la figura **6-14** podemos ver histogramas de frecuencia de los datos sin preprocesamiento alguno,



(a) SVM-Lineal (b) SVM-RBF

Figura 6-13.: Estas gráficas permiten apreciar la diferencia en R-AUPRC respecto a la base de referencia, obtenida al realizar utilizar los k atributos menos correlacionados. Dado que esta diferencia se computó sobre 8 pares de tiles, para cada k se muestra la media, la desviación estándar y el mínimo valor de entre todos los pares testeados.

en el tile b278. Dado el inmenso tamaño de cada tile (aprox. 500000 elementos cada uno), podemos asumir que estos histogramas son una buena aproximación de la distribución de probabilidad subyacente. En el anexo B.7 se muestran las distribuciones correspondientes a los tiles b234, b261 y b360.

Se puede observar que varios de los atributos no están distribuidos de forma normal, de hecho algunos atributos tienen distribuciones algo peculiares. El preprocesamiento elegido en los capítulos anteriores, binning quantile, se encargará de transformar estas distribuciones originales en distribuciones uniformes.

Un punto clave a resaltar es que las distribuciones de probabilidad que rigen algunos atributos son *significativamente* diferentes entre distintos tiles. En la figura **6-15** se muestran algunos ejemplos prominentes.

6.3.2. Análisis de distribuciones

Se ha observado que las distribuciones de probabilidad subyacentes a cada tile son diferentes entre sí, dado que la enorme cantidad de instancias de cada tile nos garantiza que las diferencias no son por chance. Esto podría representar un gran problema para SVM por dos motivos:

- La performance de SVM es altamente dependiente de la elección de hiperparámetros C

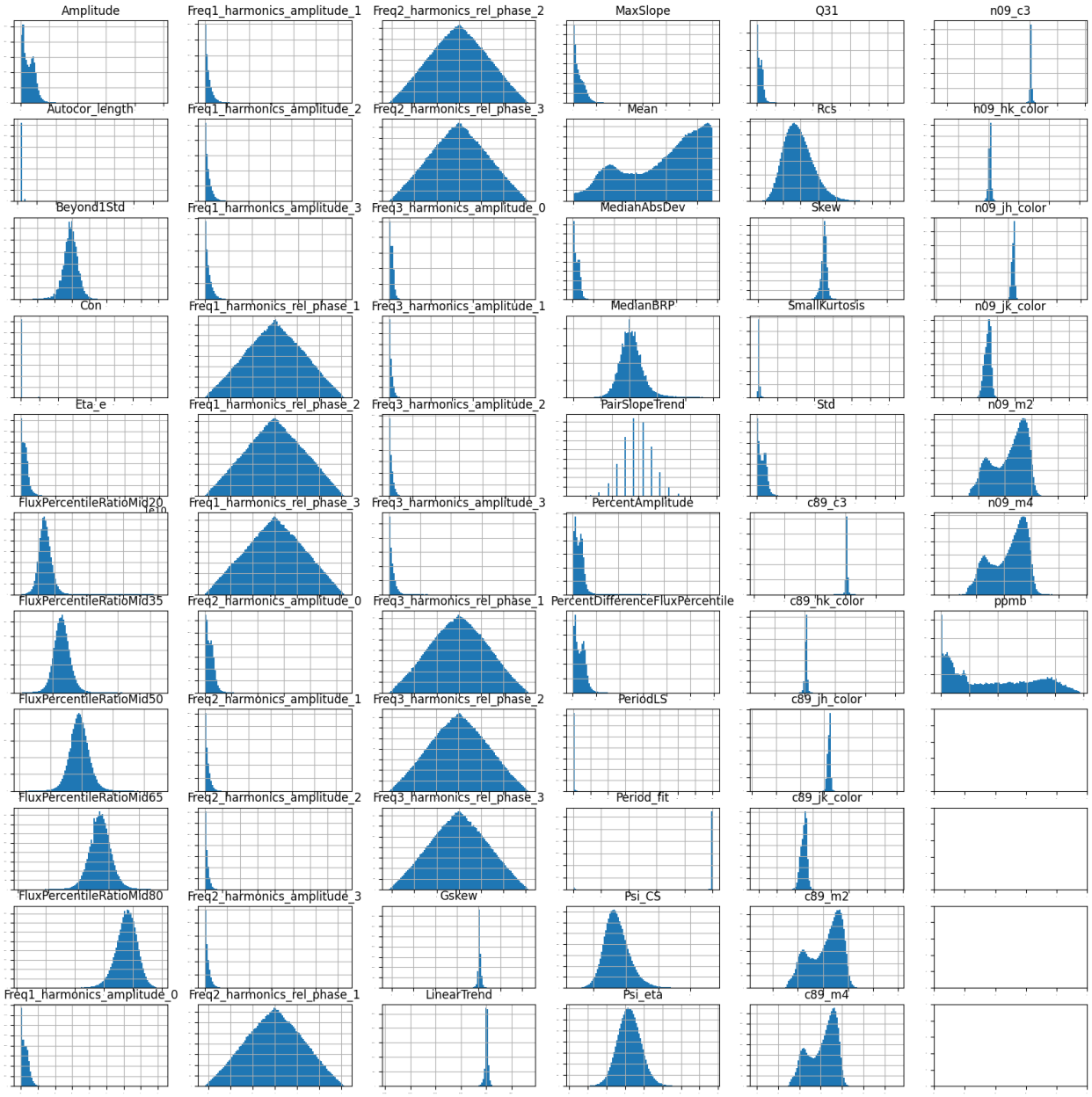


Figura 6-14.: Histogramas aproximando la distribución de probabilidad de cada atributo del tile b278.

y γ. Dado que hemos seleccionado valores para estos hiperparémtros realizando cross validation en b278, existe la posibilidad de que tales valores no generalicen muy bien al entrenar y testear con otros tiles, cuya distribución de probabilidad puede ser distinta. Lo mismo aplica a otros hiperparámetros que hemos hallado en capítulos anteriores: número de bins, atributos a eliminar en selección de atributos, etcétera.

- El hiperplano separador hallado al entrenar SVM con un cierto tile puede no generalizar

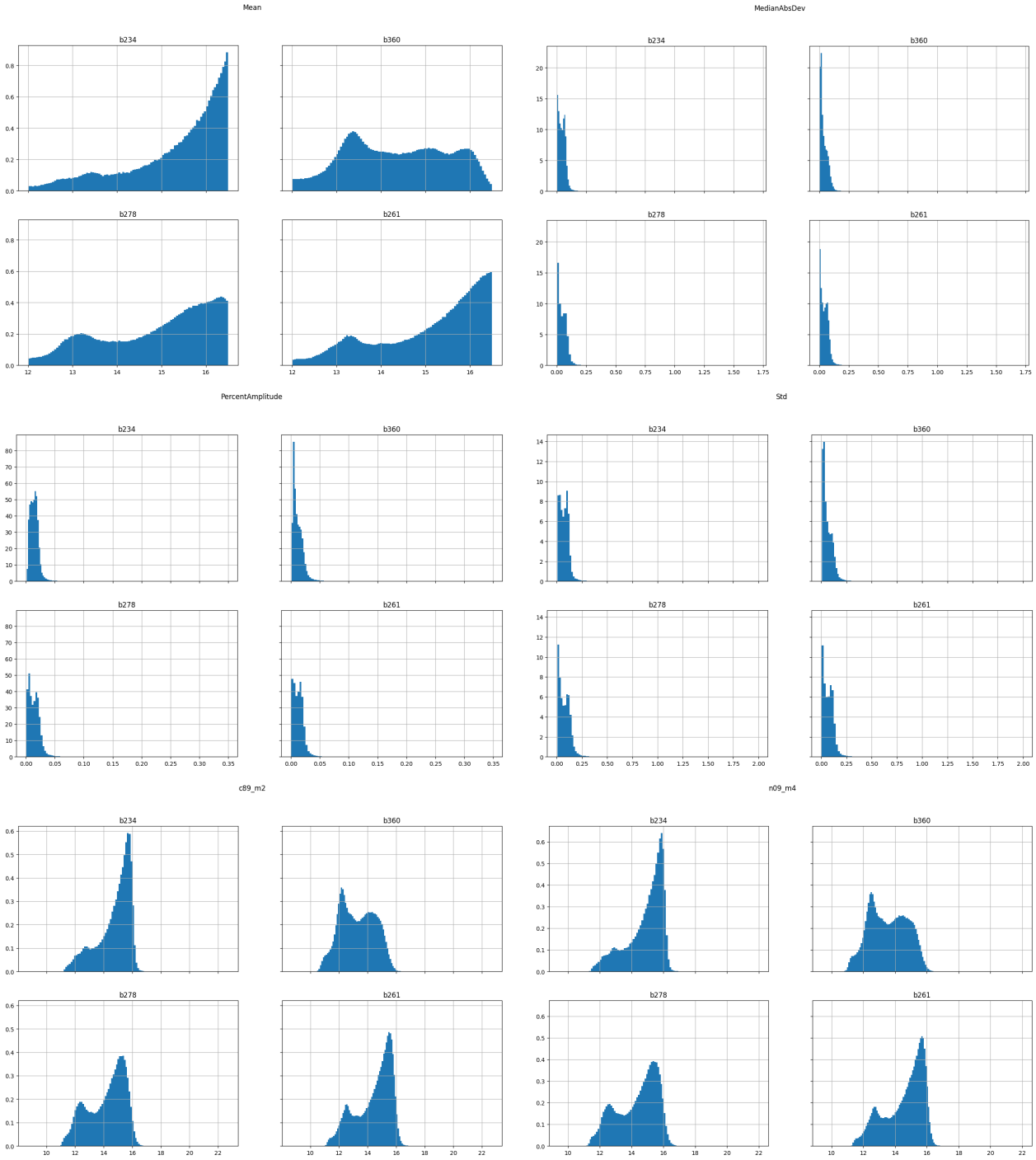


Figura 6-15.: En esta figura se grafican las distribuciones de probabilidad de algunos atributos que exhiben una notable diferencia entre distintos tiles.

muy bien al testear con otro tile cuya distribución de probabilidad es distinta.

Ambas ideas se exploran en detalle en el capítulo 8.

6.4. Conclusiones

En este capítulo hemos descubierto, mediante el uso de tests estadísticos, cuáles son los atributos de nuestro dataset que parecen no ser relevantes a la hora de distinguir RRLs. También hemos descubierto, mediante el uso de los mismos tests estadísticos y el índice Gini de RF que ciertos atributos, como PeriodFit y Psi_CS, contienen información muy valiosa para detectar RRLs. SVM lineal falla en reconocer la importancia de tales atributos.

Se determinó que ciertos atributos de nuestros datasets presentan altas correlaciones monotónicas. Sin embargo, la inclusión de estos atributos no tiene un efecto perjudicial en la performance de SVM.

Finalmente, se concluyó que las distribuciones de probabilidad subyacentes a los distintos tiles son inherentemente distintas. Esto podría traer mayores problemas para SVM que para RF, tales como dificultad de escoger hiperparámetros que generalicen bien entre distintos tiles. Esta última hipótesis se estudiará en mayor profundidad en el capítulo 8.

7. Imbalance de Clases

Como se detalló en la sección 1.2.4, los datasets exportados por Carpyncho presentan un gran desbalance en la cantidad de estrellas etiquetadas como clase positiva (RRL) y aquellas etiquetadas con clase mayoritaria o negativa (no-RLL). En la sección 1.3.5 se describió, a alto nivel, qué tipo de complicaciones pueden surgir al trabajar con datasets de esta naturaleza.

La mayoría de los clasificadores simplemente funcionan peor en datasets desbalanceados porque están diseñádos para generalizar hallando la hipótesis más simple que se ajuste a los datos de entrenamiento, basándose en la navaja de Ockham. En datos imbalanceados, la hipótesis más simple a menudo es aquella que clasifica (casi) todas las instancias como negativas [Akbani et al., 2004].

En particular, las SVM se ven perjudicadas por el imbalance de clases [Akbani et al., 2004]. Las instancias positivas (clase minoritaria) tienden a encontrarse más lejos de la frontera ideal [Wu and Chang, 2003]. Esto puede ilustrarse considerando un experimento en el cuál se generan n números aleatorios entre 1 y 100 a partir de una distribución uniforme. Las chances de obtener un número más cercano a 100 aumentan a mayores valores de n. Como consecuencia, la frontera aprendida por SVM tiende a ser más cercana a las clases positivas, tal y como se ilustra en la figura **7-1**.

En este capítulo se experimentará con distintas técnicas que buscan atenuar el impacto del imbalance de clases, con el objeto de intentar mejorar la performance de nuestros clasificadores.

7.1. Undersampling

Las técnicas de undersampling buscan corregir manualmente el imbalance de clases en los datos, eliminando instancias de la clase mayoritaria [Japkowicz, 2000] [He and Garcia, 2009]. La forma más sencilla de undersampling es **undersampling aleatorio**, en donde los elementos a descartar son elegidos aleatoriamente. Algunas consecuencias de utilizar undersampling son:

- Puede perderse información valiosa al descartar instancias de la clase mayoritaria.

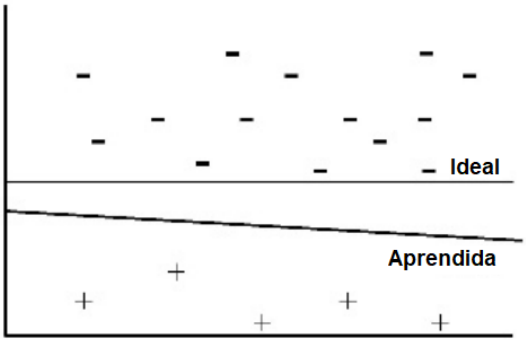


Figura 7-1.: Ejemplo de un dataset imbalanceado, donde las instancias positivas se encuentran más alejadas de la frontera ideal que las instancias negativas, simplemente por el hecho de ser menos. Como consecuencia, SVM aprende una frontera (línea inclinada) que está demasiado cerca de las instancias positivas. Créditos: [Akbani et al., 2004]

- La complejidad temporal y espacial para entrenar clasificadores se ve reducida, pues habrá menos datos que procesar.
- En undersampling aleatorio, la performance del clasificador en test puede variar mucho dependiendo del sampleo utilizado.

En [Cabral et al., 2020a] se experimentó utilizando undersampling aleatorio en clasificadores RF. Se concluyó que es conveniente el uso de tiles enteros, manteniendo el imbalance original entre las clases, dado que los clasificadores entrenados en tiles balanceados tenían menor performance en test. En esta sección se realizará un experimento similar utilizando SVM.

Se evaluó el R-AUPRC en test obtenido por clasificadores SVM entrenando con tiles artificialmente balancedas usando undersampling aleatorio, explorando distintas proporciones entre el número de no-RRLs y el número de RRLs. Nótese que cada sampleo fue obtenido del precedente, por ejemplo el sampleo con rate $1:1$ fue obtenido del sampleo con rate $1:10$ y así sucesivamente.

Este experimento se repitió tres veces, dado que los resultados obtenidos (especialmente para los datasets más balanceados) son sensibles al sampleo aleatorio utilizado. Los resultados para SVM lineal y SVM RBF se muestran en las figura **7-2** y **7-3** respectivamente. Es importante remarcar que en todos los experimentos de este capítulo la corrección de imbalance se realiza únicamente en el tile de entrenamiento, en tanto que se usan tiles completos para testear.

Utilizando las curvas de color negro, se calculó la ganancia promedio (entre todos los tiles)

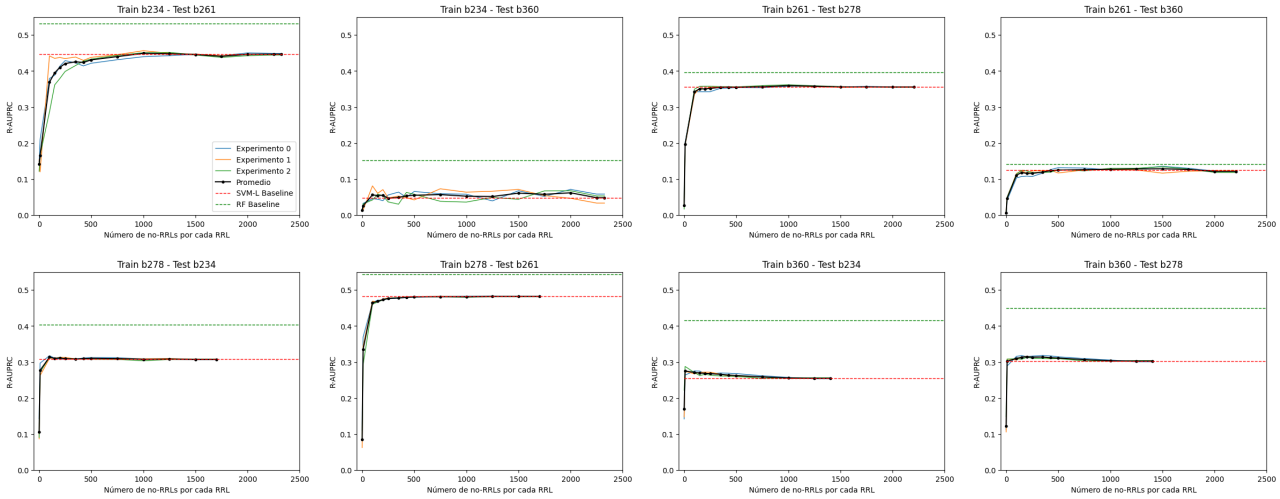


Figura 7-2.: Impacto de undersampling en el R-AUPRC en test de clasificadores SVM Lineal, en función de la proporción de no-RRLs por RRL. La línea punteada roja indica el R-AUPRC obtenido en la sección 5.3

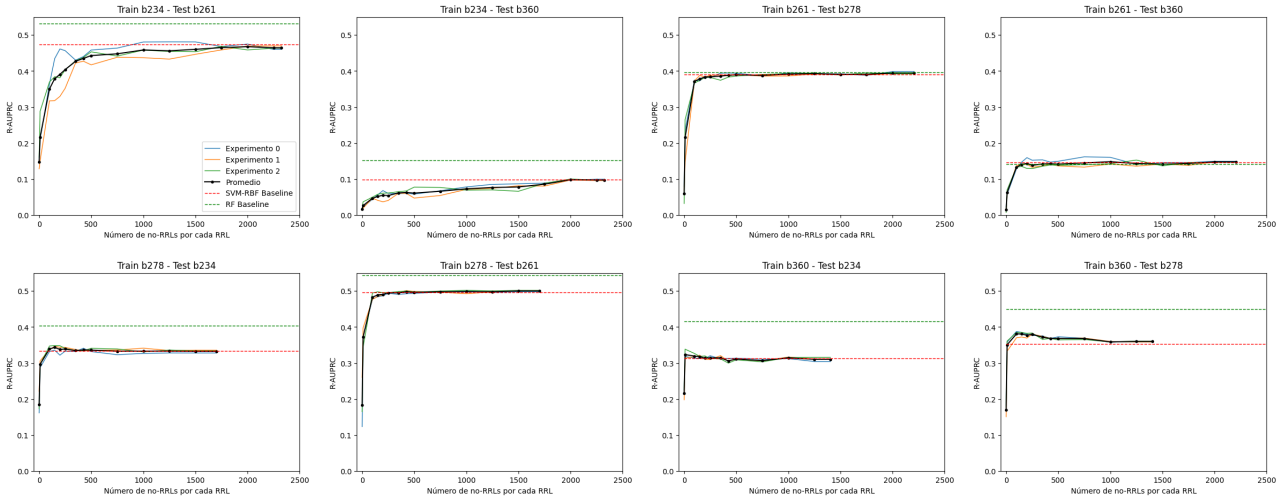


Figura 7-3.: Impacto de undersampling en el R-AUPRC en test de clasificadores SVM RBF, en función de la proporción de no-RRLs por RRL. La línea punteada roja indica el R-AUPRC obtenido en la sección 5.3

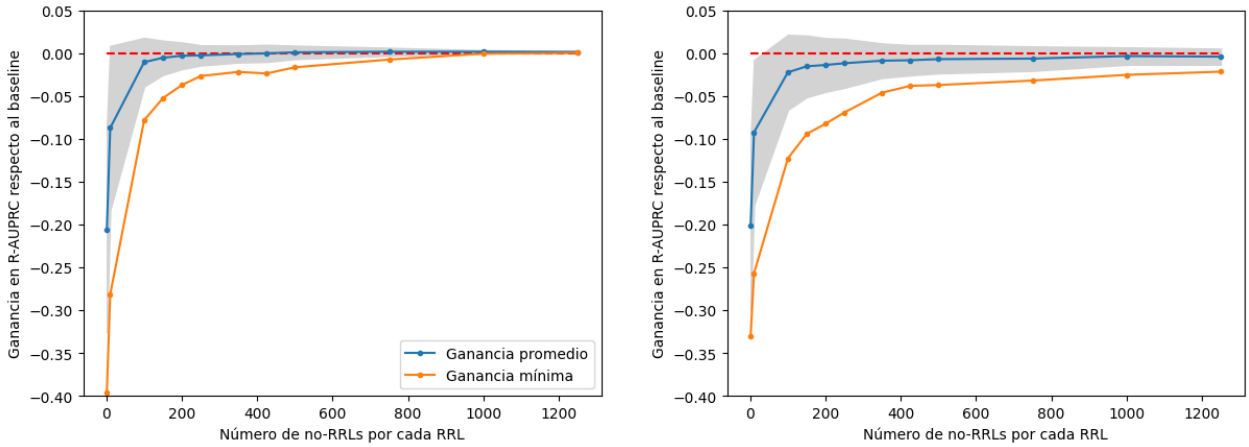
en R-AUPRC respecto a la base de referencia, para cada proporción RRL-noRRL estudiada. Los resultados de estos experimentos se encuentran en la figura **7-4**, de los cuáles podemos concluir:

- Balancear las clases en exceso es altamente perjudicial, pues reduce el R-AUPRC drásticamente para todos los tiles (ver las curvas en proporciones 1:1 a 1:250).
- En SVM Lineal, corregir el imbalance de forma moderada, dejando al menos 500 no-

RRL por RRL, permite igualar la base de referencia. Esto implica que se puede obtener al menos la misma performance en test utilizando tan solo un cuarto datos para entrenar. Una consecuencia inmediata es una drástica reducción en la complejidad temporal y espacial necesaria para entrenar clasificadores SVM Lineal.

- En SVM-RBF, undersampling incurre en una pérdida de R-AUPRC promedio para todas las proporciones testeadas.

Undersampling resulta, por lo tanto, no útil para mejorar el R-AUPRC de nuestros clasificadores. Únicamente resulta beneficioso en SVM Lineal, para obtener una reducción importante en tiempos de entrenamiento y consumo de memoria.



(a) SVM Lineal (b) SVM RBF

Figura 7-4.: Ganancia promedio en R-AUPRC respecto a la base de referencia al utilizar undersampling, en función de la proporción entre las clases. Nótese que el dominio de estos gráficos se extiende sólo hasta 1200, pues es la proporción original de algunos de los tiles estudiados. La mayoría de los tiles, sin embargo, presenta un imbalance original de 1:2000.

7.2. Oversampling

Una segunda forma de corregir manualmente el imbalance de las clases consiste en generar nuevas instancias de la clase minoritaria [He and Garcia, 2009]. En este trabajo se usaron tres métodos para lograr esto, ilustrados en la figura **7-5**:

- **Oversampling aleatorio**: Seleccionar elementos de la clase minoritaria (con reemplazo) con probabilidad uniforme, añadiéndolos nuevamente al dataset de entrenamiento [Menardi and Torelli, 2012]. Como consecuencia, habrá instancias repetidas.

- **ADASYN** (Adaptative Synthetic sampling method): Generar nuevas instancias utilizando interpolación, intentando crear puntos cerca de aquellas instancias que son incorrectamente clasificadas utilizando kNN. [He et al., 2008]

- **SMOTE** (Synthetic Minority Oversampling Technique): Generar nuevas instancias utilizando interpolación, sin hacer ninguna distinción entre instancias que son fáciles o difíciles de clasificar. [Chawla et al., 2002]

Adicionalmente, se experimentó con una técnica híbrida que combina undersampling y oversampling:

- **SMOTEENN**: Consiste en generar nuevas instancias utilizando SMOTE, seguido de una eliminación de instancias ruidosas utilizando el método de undersampling ENN (Edited Nearest Neighbors) [Batista et al., 2003]. Podemos ver un ejemplo de SMOTEENN en la figura **7-6**.

Se utilizaron las implementaciones provistas por **imbalanced-learn** (`imbalanced-learn.org/`), una librería que extiende scikit-learn con técnicas para abordar imbalance de clases. De forma similar a la sección anterior, se analizó el impacto en R-AUPRC en función a la proporción de RRLs y no-RRLs para cada método estudiado. Los resultados de este experimento para SVM Lineal y SVM RBF se encuentran en las figuras **7-7** y **7-8** respectivamente.

En primer lugar, podemos descartar el método SMOTEENN como preprocesamiento de nuestros tiles, debido a su pobre performance en todos los casos evaluados. Para los tres métodos restantes, se calcularon las ganancias promedio respecto a la base de referencia, las cuales se encuentran graficadas en la figura **7-9** . Podemos concluír que:

- En SVM Lineal, únicamente oversampling aleatorio es capaz de mejorar levemente el R-AUPRC respecto a la base de referencia en promedio. No hay beneficio en aplicar ADASYN o SMOTE.

- Similarmente, en SVM RBF, ADASYN y SMOTE empeoran el R-AUPRC respecto a la base de referencia en promedio. Oversampling aleatorio, en el mejor caso, logra igualar o superar el R-AUPRC promedio pero con una pérdida importante en el peor caso.

- Notar que, al igual que en undersampling, corregir el imbalance en exceso (500 o menos no-RRl por RRL) incurre en una reducción notable en R-AUPRC.

- Es importante notar que al hacer oversampling, una corrección moderada del imbalance (con 500 o más no-RRL por RRL) implica un aumento importante en el tamaño de los datasets de entrenamiento. Esto conlleva un mayor consumo de memoria y tiempo de entrenamiento.

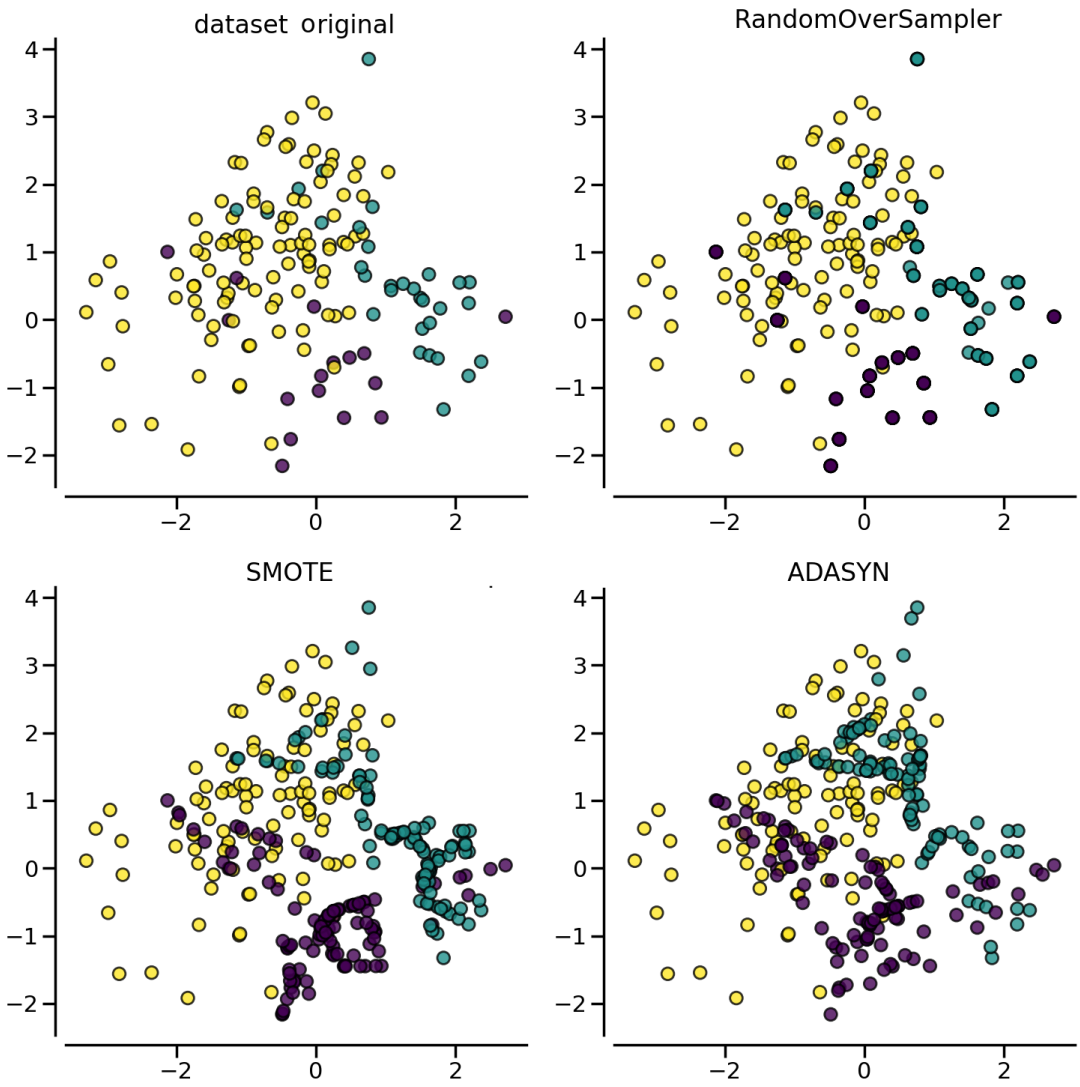
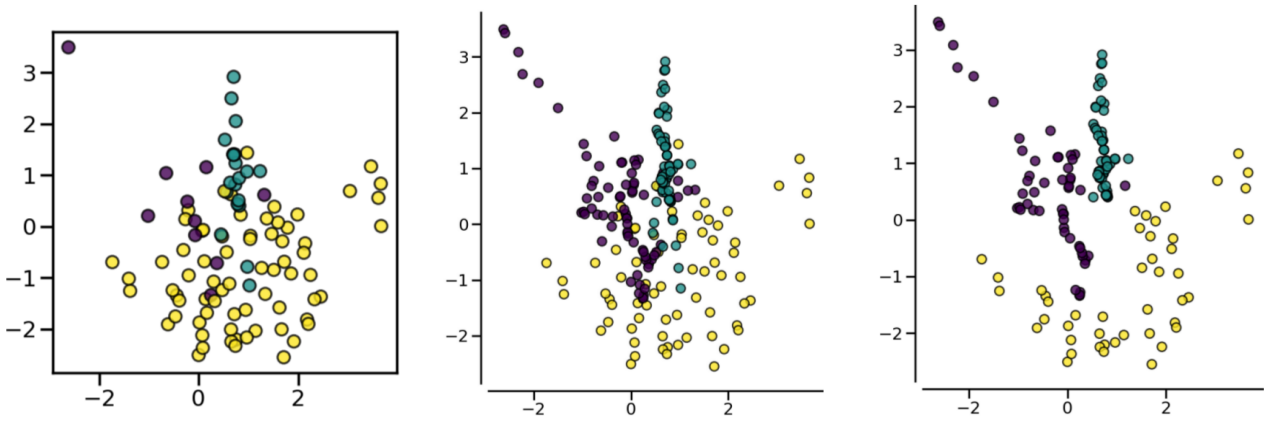


Figura 7-5.: Comparación de métodos de oversampling utilizados en este trabajo. Créditos: Documentación de *imbalanced-learn*.

En resumen, los resultados son similares a los vistos para undersampling: Oversampling no es beneficioso en SVM RBF, y es ligeramente beneficioso para SVM-Lineal. Sin embargo, esta mejoría en R-AUPRC conlleva un mayor costo de procesamiento y memoria.

7.3. Class Weight

Un tercer enfoque orientado a resolver el efecto del imbalance de clases es sesgar el clasificador de forma tal que preste más atención a la clase positiva [Akbani et al., 2004]. Esto puede hacerse, por ejemplo, incrementando la penalidad asociada a clasificar incorrectamente las



(a) Datos originales (b) Oversampling SMOTE (c) SMOTEENN

Figura 7-6.: Aplicación de SMOTEENN sobre el dataset imbalanceado (a). En primer lugar, se aplica oversampling SMOTE (b). Finalmente, se eliminan algunas de las nuevas instancias utilizando ENN (c). Créditos: Documentación de *imbalanced-learn*.

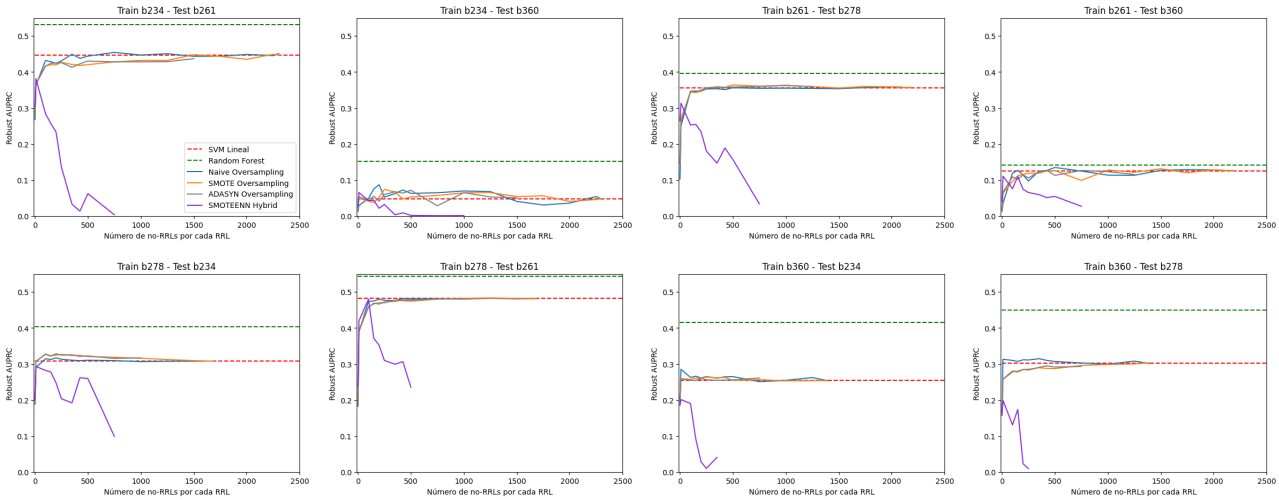


Figura 7-7.: Impacto de oversampling en el R-AUPRC en test de clasificadores SVM Lineal, en función de la proporción de no-RRLs por RRL. La línea punteada roja indica el R-AUPRC obtenido en la sección 5.3

instancias de clase RRL.

La implementación de SVM provista por sklearn permite asignar un peso α_i a cada clase i, de forma tal que se penalicen los errores de cada clase con una severidad diferente, ajustando el parámetro de regularización C como $\alpha_i * C$ para cada clase i . En problemas de clasificación binaria, llamaremos α_+ y α_- a los pesos de la clase minoritaria (positiva) y mayoritaria

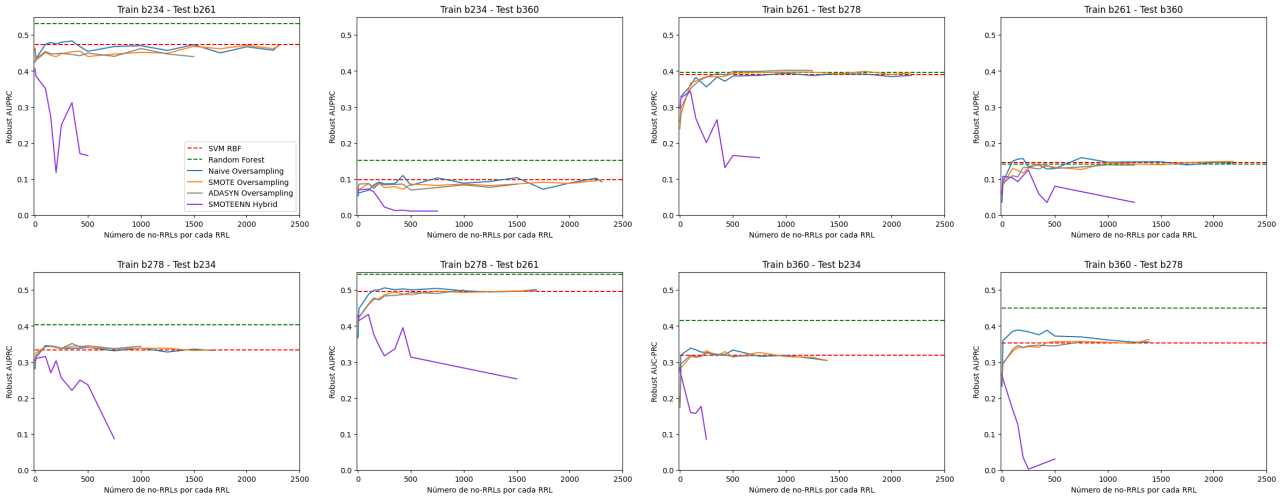


Figura 7-8.: Impacto de oversampling en el R-AUPRC en test de clasificadores SVM RBF, en función de la proporción de no-RRLs por RRL. La línea punteada roja indica el R-AUPRC obtenido en la sección 5.3

(negativa), respectivamente.

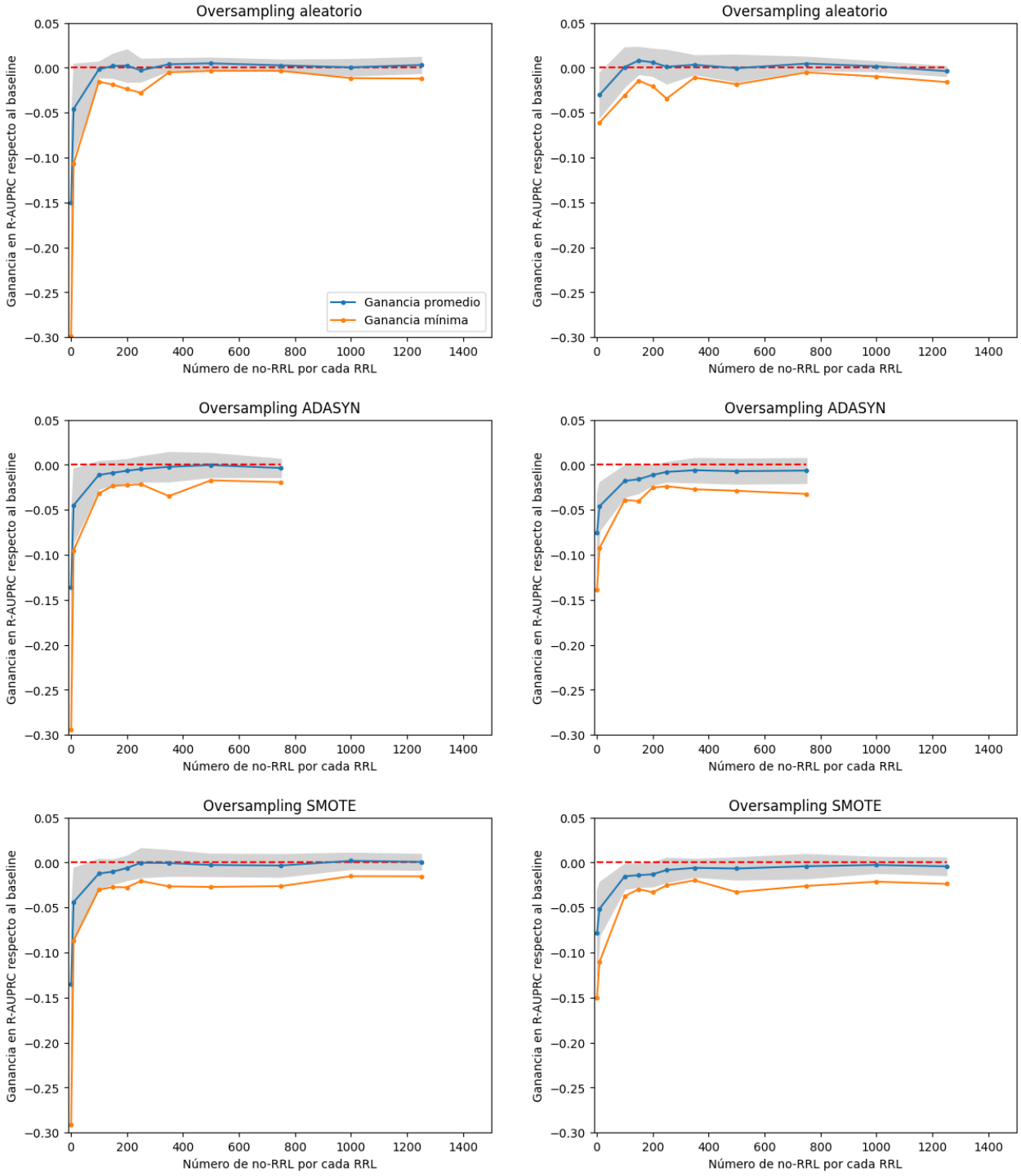
Se evaluó el impacto en R-AUPRC en test obtenido al utilizar SVM con distintos valores de α_+, fijando $\alpha_- = 1$. Los resultados para SVM Lineal y SVM RBF se encuentran en las figuras **7-10** y **7-11** respectivamente.

Si bien el uso de *class_weight* tiene un impacto positivo en algunos pares de tiles; tiene un impacto igualmente perjudicial en otros pares. La figura **7-12** resume esta información, permitiéndonos concluir que:

- *class_weight* permite obtener una mejora promedio leve en SVM-Lineal para $1 < \alpha_+ < 20$, y una mejora promedio casi nula en SVM-RBF para $1 < \alpha_+ < 10$.
- La mejoría promedio viene acompañada de reducción marcada en R-AUPRC para algunos pares de tiles. Esto indica que el uso de *class_weight* produce resultados un tanto inestables.

7.4. Conclusiones

Con el objeto de mitigar el impacto del imbalance de clases, en este capítulo se hizo uso de técnicas de oversampling y undersampling. También se intento sesgar los clasificadores



(a) SVM Lineal (b) SVM RBF

Figura 7-9.: Ganancia promedio en R-AUPRC respecto a la base de referencia al utilizar oversampling, en función de la proporción entre las clases

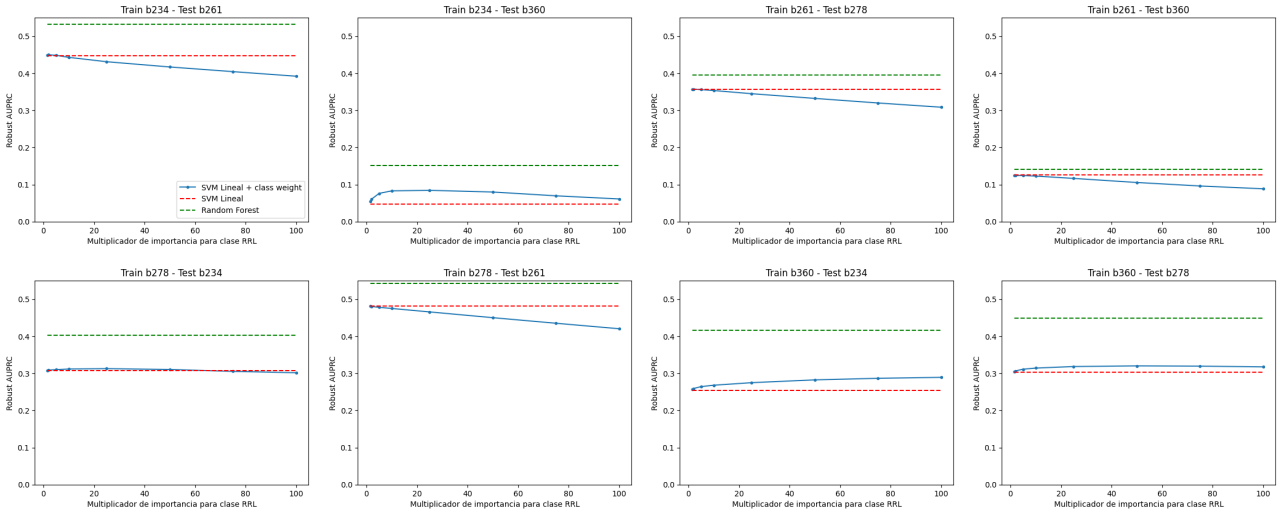


Figura 7-10.: Impacto de class weight en el R-AUPRC en test de clasificadores SVM Lineal, en función del multiplicador a la penalidad de la clase positiva. La línea punteada roja indica el R-AUPRC obtenido en la sección 5.3

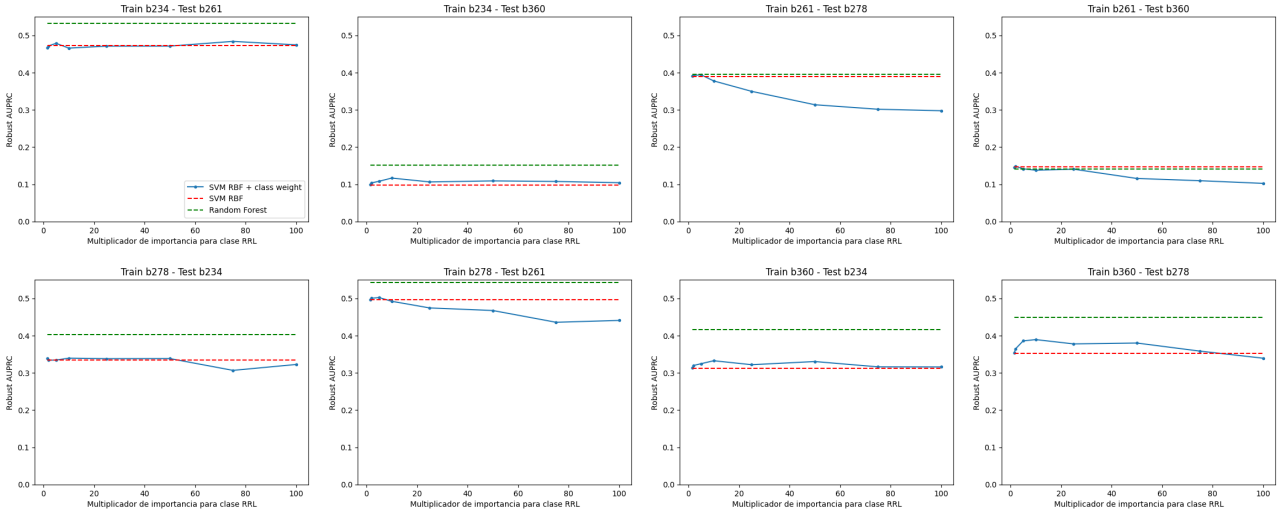
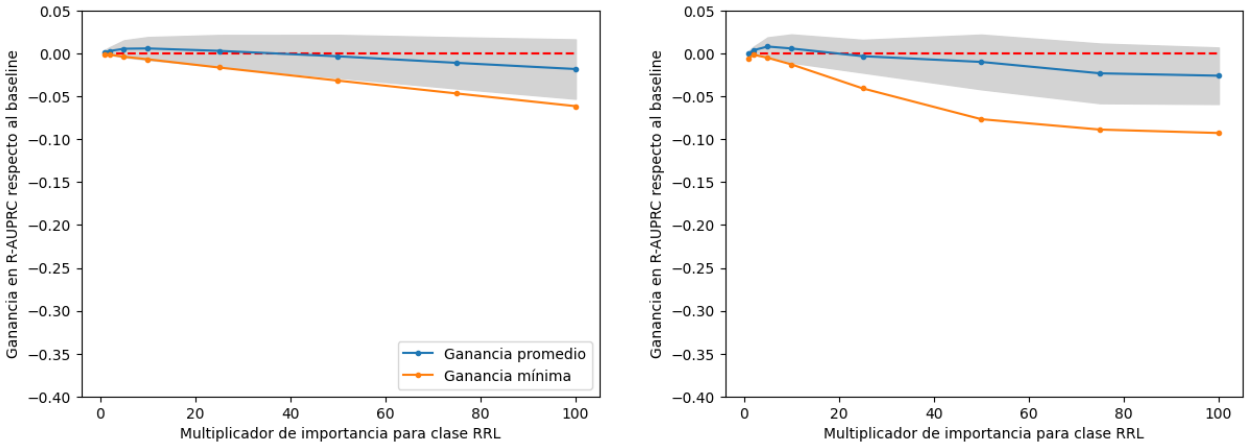


Figura 7-11.: Impacto de class weight en el R-AUPRC en test de clasificadores SVM RBF, en función del multiplicador a la penalidad de la clase positiva. La línea punteada roja indica el R-AUPRC obtenido en la sección 5.3

SVM para que presten más atención a la clase minoritaria. Ninguna de las técnicas estudiadas logró mejorar significativamente la base de referencia obtenida en capítulos anteriores.

Continuando con la política establecida en capítulos anteriores, se preferirá aquél preprocesamiento que maximice la ganancia promedio en R-AUPRC sujeto a no empeorar significativamente el R-AUPRC en ningún par de tiles estudiados. Esta última condición busca



(a) SVM Lineal (b) SVM RBF

Figura 7-12.: Ganancia promedio en R-AUPRC respecto a la base de referencia al utilizar class weight, en función del multiplicador a la penalidad de la clase positiva.

prevenir métodos con resultados inestables, que son altamente dependientes del par de tiles elegido.

Los hiperparámetros que maximizan esta métrica para SVM Lineal se muestran en la tabla **7-1**. Por otro lado, en SVM RBF, si bien algunas asignaciones de hiperparámetros para cada método logran mejorar el R-AUPRC promedio muy ligeramente, también ocasionan una pérdida no despreciable en R-AUPRC para ciertos pares (i.e. Ganancia mínima con valores negativos). En otras palabras, algunos pares de tiles tienen una involución muy marcada en R-AUPRC a cambio de una ganancia mínima promedio en todos los tiles. Los resultados para SVM RBF se encuentran en la tabla **7-2**.

	Hiperparámetro óptimo	**Avg gain**	**Min gain**
Undersampling aletorio	Proporción: 1:1000	0.004	0.001
Oversampling aleatorio	Proporción: 1:500	0.007	-0.001
Class weight	$\alpha_+ = 2$	0.005	0

Tabla 7-1.: Hiperparámetros óptimos para los mejores métodos estudiados en esta sección, SVM-L. El criterio de optimalidad es: maximizar la ganancia promedio respecto a la base de referencia, sujeto a no empeorar la performance en ningún par de tiles significativamente.

Se decidió elegir oversampling aleatorio con proporción 1:500 como preprocesamiento para SVM Lineal en el resto de este trabajo, aunque la mejoría es casi nula. En la tabla **7-3** se muestra la mejora obtenida para una selección de tiles. Para SVM-RBF, todos los métodos estudiados empeoran significativamente el R-AUPRC en alguno de los tiles estudiados, a

	Hiperparámetro óptimo	Avg gain	Min gain
Undersampling aletorio	∄	-	-
Oversampling aleatorio	Proporción: 1:750	0.003	-0.013
Class weight	$\alpha_+ = 5$	0.006	-0.015

Tabla 7-2.: Hiperparámetros óptimos para los mejores métodos estudiados en esta sección, SVM-RBF. Dado que no existen hiperparámetros que no empeoren la performance en ningún par de tiles significativamente, se muestran los hiperparámetros que maximizan la ganancia mínima.

cambio de una ganancia casi nula en promedio; por lo que se decidió no realizar corrección de imbalance alguna.

Como conclusión de este capítulo, podemos decir que el imbalance de clases parece no estar teniendo un impacto tan perjudicial en SVM, pues las técnicas de corrección de imbalance no permiten mejorar resultados significativamente.

Train tile	Test tile	RF	SVM-L sin overs.	SVM-L con overs.	Gain
b278	b234	0.40	0.307	0.312	0.005
b278	b261	0.54	0.479	0.48	0.001
b278	b360	0.21	0.134	0.135	0.001
b234	b278	0.40	0.305	0.313	0.009
b234	b261	0.53	0.439	0.45	0.011
b234	b360	0.15	0.044	0.057	0.014
b261	b278	0.40	0.358	0.355	-0.003
b261	b234	0.39	0.3	0.301	0.001
b261	b360	0.14	0.125	0.135	0.01
b360	b278	0.45	0.3	0.307	0.007
b360	b234	0.42	0.253	0.26	0.007
b360	b261	0.54	0.406	0.414	0.007
AVG		0.38	0.287	0.293	0.005

Tabla 7-3.: En esta tabla podemos ver el incremento en R-AUPRC que se obtiene al utilizar oversampling 1:500 en SVM-Lineal.

8. Variabilidad entre los datos de entrenamiento y testeo

A pesar de todos las técnicas aplicadas en los capítulos anteriores, no ha sido posible para SVM igualar la performance de RF en varios de los tiles analizados. Dado que es esperable que SVM lineal sea incapaz de alcanzar la performance de RF por su limitación de generar superficies de decisión no lineales, en este capítulo nos concentraremos en entender qué está limitando a SVM-RBF.

Una hipótesis que fue sugerida en la sección 6.3.2 es que el problema se encuentra en el hecho de que las distribuciones de probabilidad subyacentes a cada tile parecen ser significativamente distintas. En las distintas secciones de este capítulo se analizará esta hipótesis en detalle.

8.1. Impacto de la variabilidad de datos en la elección de hiperparámetros

Es importante resaltar que la performance de SVM-RBF es extremadamente sensible a la elección de hiperparámetros, como se pudo observar en la figura **3-2**. Pequeñas variaciones de C y γ pueden ocasionar saltos significativos en el R-AUPRC obtenido. Por otro lado, una de las virtudes más reconocidas de RF es el hecho de que no requiere una optimización muy cuidadosa de hiperparámetros, siendo un método off-the-shelf [Wyner et al., 2015].

Es posible que los hirparámetros C y γ que se estimaron en capítulos anteriores realizando cross-validation sobre un único tile ($b278$) no produzcan buenos resultados al utilizarse para entrenar y testear SVM utilizando otros pares de tiles. Para aclarar esto, se condujo un experimento que responderá a la siguiente pregunta:

> Dado un par de tiles arbitrario t_1 y t_2. ¿Existe una combinación de hiperparámetros C y γ tal que SVM-RBF entrenado con t_1 tenga igual o mejor R-AUPRC en test que RF, testeando en t_2?

Se calculó el R-AUPRC en test para cada combinación de hiperparámetros de una grilla de valores de γ y C, entrenando y testeando SVM-RBF en cada par de tiles. Los resultados se

encuentran en las figuras **8-1** y **8-2**. A partir de estos datos, podemos observar lo siguiente:

- En primer lugar, observando las curvas de la figura **8-2**, podemos ver que la performance de RF ya no es marcadamente superior a la de SVM, debido a las mejoras que se introdujeron en los capítulos anteriores. Sin embargo, RF continúa teniendo mejor performance en varios de los pares testeados.

- Una segunda observación que se desprende de las curvas de la figura **8-2** es que, en general, los hiperparámetros que se seleccionaron para SVM-RBF en capítulos anteriores ($C = 10^4$ y $\gamma = 10^{-4}$) son casi tan buenos como los hiperparámetros óptimos en cada par de tiles. Es decir, la elección de hiperparámetros que se venía utilizando generaliza de forma apropiada.

- Podemos ver que la asignación de hiperparámetros que mejor funciona en un cierto par de tiles, a menudo funciona muy mal en otro. Por ejemplo, escoger $10^{-4} \leq \gamma \leq 10^{-2}$ y $10^5 \leq C \leq 10^{11}$ produce resultados óptimos en $b278 \mapsto b261$, en tanto que produce consistentemente los peores resultados cada vez que se entrena con $b234$. Esto sugiere que es complicado (o imposible) hallar una asignación de hiperparámetros que sea óptima para todo par de tiles. Dicho esto, se vuelve a remarcar que la asignación de parámetros que se venía utilizando ($C = 10^4$ y $\gamma = 10^{-4}$), si bien no es óptima en todos los tiles, es una de las pocas que se encuentra consistentemente entre las que mejor funcionan en distintos tiles.

- Prestando atención a las celdas tachadas en la figura **8-1** y a las curvas de la figura **8-2**, se observa que SVM-RBF (incluso sobreajustando los hiperparámetros) no es capaz de empatar la performance de RF en cuatro de las combinaciones testeadas: $b234 \mapsto b360$, junto con todos los tiles entrenados en $b360$. El hecho de que el tile $b360$ esté involucrado en todos estos pares puede estar relacionado al hecho de que $b360$ se encuentra espacialmente a mayor distancia de $b234$, $b261$ y $b278$ (ver figura **1-3**). Esto podría ocasionar que la distribución subyacente de $b360$ sea más diferente a la de los otras tres tiles, siendo más difícil que clasificadores entrenados en otros tiles generalicen bien.

Como conclusión de este experimento, podemos descartar que la asignación de hiperparámetros escogida sea el problema por el cual SVM-RBF funciona peor que RF. Existen elecciones de hiperparámetros (como $C = 10^4$ y $\gamma = 10^{-4}$) que funcionan razonablemente bien en todos los pares testeados.

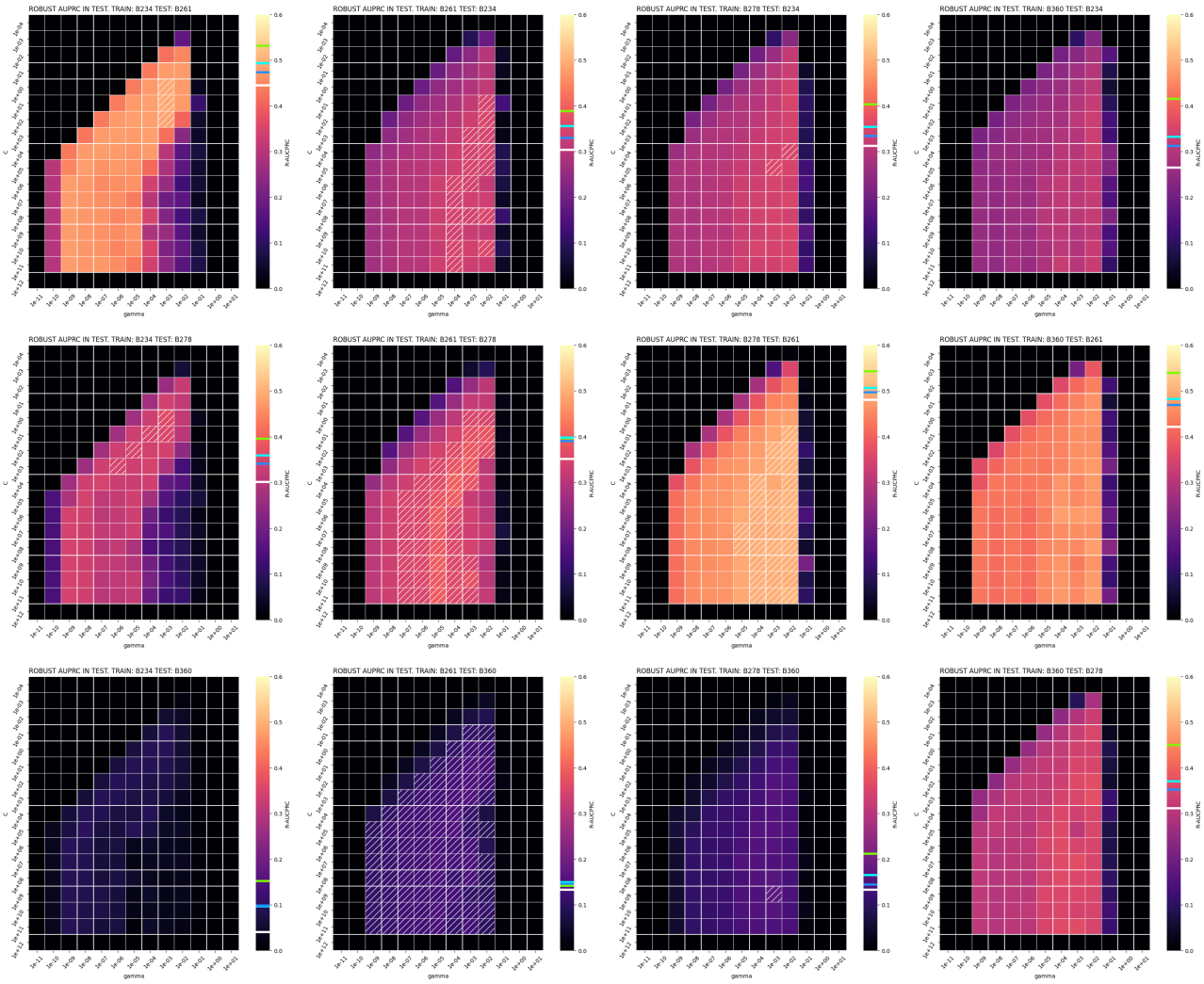


Figura 8-1.: Cada grilla muestra el R-AUPRC en test obtenido utilizando SVM-RBF con distintos hiperparámetros en distintos pares de tiles. En la barra de color se marca, con líneas horizontales, el R-AUPRC de Random Forest (verde), el valor óptimo de SVM-RBF encontrado en esta grilla (celeste), junto con los mejores resultados obtenidos en el capítulo anterior para SVM-RBF (azul) y SVM-Lineal (blanco). Adicionalmente, se tacharon aquellas celdas individuales cuyo R-AUPRC está a menos de 0.5 del de RF.

8.2. Impacto de la variabilidad de datos en los clasificadores

Si bien hemos comprobado que la diferencia en las distribuciones de cada tile no imposibilita hallar hiperparámetros razonables, la disparidad entre datos de entrenamiento y test puede aún estar perjudicando a SVM. En esta sección nos concentraremos en analizar esta

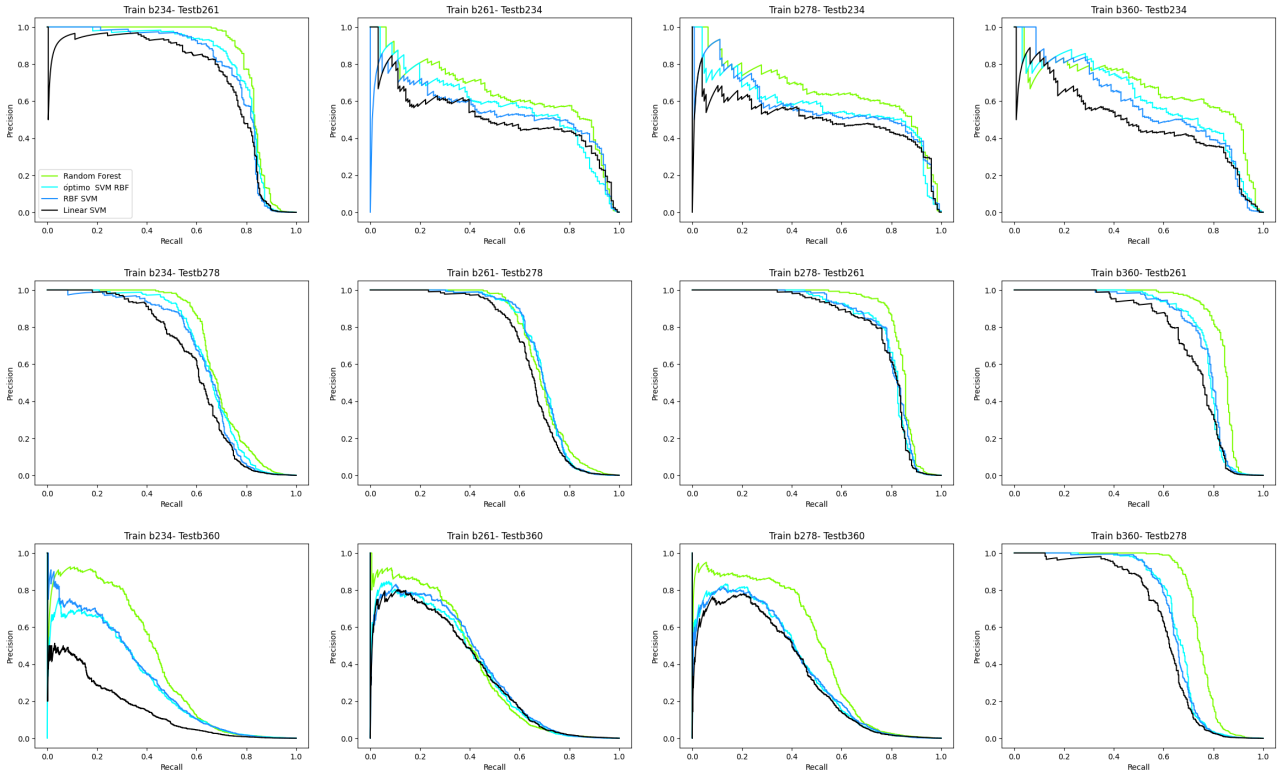


Figura 8-2.: Para cada una de las grillas calculadas en la figura **8-1**, se grafican las curvas de precision-recall correspondientes a RF y SVM-RBF con el par de parámetros óptimo de la grilla, junto con las mejores curvas de SVM-RBF y SVM-L obtenidas en capítulos anteriores.

disparidad en profundidad.

8.2.1. Introducción a dataset shift

A menudo, los científicos de datos se encuentran con que un cierto modelo funciona muy bien en los datos de entrenamiento, pero falla en obtener la misma performance en test, incluso luego de haberse asegurado de evitar sobreajustar, escogiendo el mejor modelo basado en cross-validation. En esta situación, hay algún tipo de patrón inherente a los datos de testeo que no está siendo capturado.

Una suposición crítica en el diseño de la amplia mayoría de algoritmos de aprendizaje automatizado supervisados es que los elementos del dataset de entrenamiento son muestreados independientemente de la misma distribución de probabilidad subyacente que los elementos sobre los cuales el modelo hará predicciones [Sugiyama and Kawanabe, 2012] [Kouw and Loog, 2019] [Zadrozny, 2004] [GeetaDharani. et al., 2019] [Yu et al., 2015]. Sin embargo, en una gran variedad de escenarios prácticos esta simple presunción suele incumplirse [Sugi-

yama and Kawanabe, 2012] [GeetaDharani. et al., 2019] [Quinonero-Candela et al., 2009]. Distintos trabajos muestran que el incumplimiento de esta premisa puede afectar seriamente la performance de la mayoría de los algoritmos de clasificación clásicos, tales como SVM [Zadrozny, 2004], árboles de decisión [Zadrozny, 2004], regresión lineal [Chen et al., 2016], y redes neuronales [Yu et al., 2015].

Para ejemplificar esta problemática, consideremos una situación donde se intenta modelar el comportamiento de compra de clientes. Supongamos que los datos de entrenamiento y de testeo lucen como en la figura **8-3**. El modelo será entrenado en clientes que tienen, en promedio, menor edad que los clientes con los que se testeará. Este modelo nunca habrá visto patrones de edad como los que habrá en los datos de testeo. Si la edad es un atributo importante para el problema en cuestión, la performance en test se verá afectada.

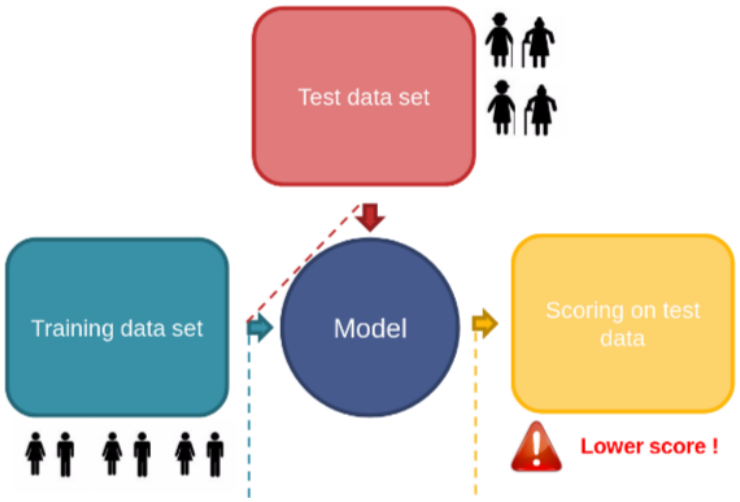


Figura 8-3.: En este ejemplo, un modelo se entrena utilizando una población joven pero se testea en una población que tiene mayor edad en promedio. Como consecuencia, el modelo no generaliza bien y se obtiene baja performance en test. Fuente: `analyticsvidhya.com`

Esta problemática ha recibido mucha atención en los últimos años, aunque a menudo pasa desapercibida, e incluso carece de un nombre estándar en la literatura, siendo referida con distintos nombres [Moreno-Torres et al., 2012] tales como: dataset shift [Quinonero-Candela et al., 2009], concept shift o concept drift [Widmer, 1998] [Widmer and Kubat, 1994] , changing environments [Alaiz and Japkowicz, 2008], etc. Formalmente, definiremos **dataset shift** (variabilidad de datasets) como aquella situación en la cual la distribución de probabilidad de los datos de entrenamiento es distinta a la distribución de probabilidad de los datos de test [Quinonero-Candela et al., 2009] [Sugiyama and Kawanabe, 2012].

$$P_{train}(x, y) \neq P_{test}(x, y)$$

La presencia de dataset shift es común en muchas aplicaciones prácticas y puede ser causada por distintos motivos, tales como:

- Ambientes no estacionarios: Datos obtenidos en distintas locaciones, en distintos instantes de tiempo, etcétera [Quinonero-Candela et al., 2009] [Raza et al., 2014] [Moreno-Torres et al., 2012]. Por ejemplo, un dataset recolectado en una cierta estación del año o en una cierta región geográfica es utilizado para realizar predicciones sobre datos recolectados en una estación del año o región geográfica notablemente distinta.
- Sesgo introducido por el diseño experimental mediante el cual se obtuvieron los datos [Moreno-Torres et al., 2012]. Por ejemplo, en ciencias sociales, habrá subconjuntos de una población general (e.g. estudiantes en la universidad que conduce un cierto experimento) que serán más fáciles de encuestar que otros. Tales subconjuntos de la población pueden estar sobre-representados, en tanto que otros subconjuntos (e.g. convictos) pueden ser completamente excluidos.
- Incapacidad de reproducir las condiciones de testeo al momento de entrenar.

Algunos ejemplos de dataset shift, extraídos de [Quinonero-Candela et al., 2009], son:

- El país Albodora hizo un estudio que muestra que la introducción de una cierta medida ayudó a disminuir el consumo de alcohol en menores de edad. Los políticos de Bodalecia están impresionados y quieren reutilizar las mismas medidas. ¿Deberían?
- Un algoritmo para detectar intrusiones en una red fue desarrollado utilizando aprendizaje automatizado en datos de hace cuatro años. ¿Funcionará hoy en día tan bien como cuando fue lanzado?

8.2.2. Tipos de dataset shift

Es interesante discriminar entre distintos tipos de dataset shift. La variedad de dataset shift más ampliamente estudiada es **covariate shift**[1] (también llamado covariate drift), en la cual la variabilidad entre los datos de entrenamiento y de testeo se encuentra únicamente en los atributos [Sugiyama and Kawanabe, 2012] [Quinonero-Candela et al., 2009] [Moreno-Torres et al., 2012] [Kouw and Loog, 2019]. Esto se ejemplifica en la figura **8-5**a.

Formalmente:

$$P_{train}(y|x) = P_{test}(y|x) \ \wedge \ P_{train}(x) \neq P_{test}(x)$$

Algunos ejemplos reales de covariate shift incluyen:

[1] En este contexto, la palabra *covariate* (en inglés) se utiliza como sinónimo de atributo o feature.

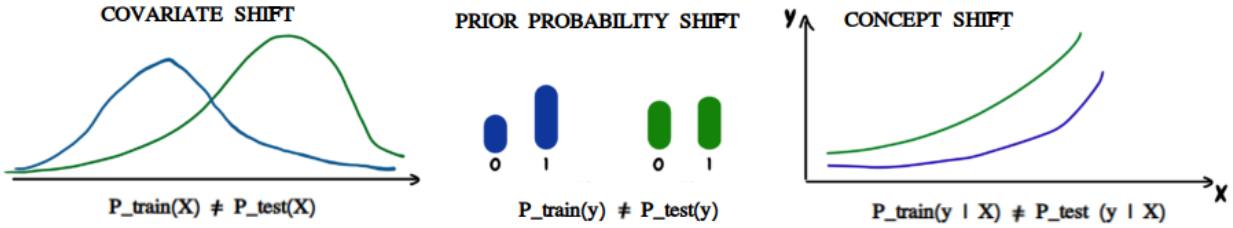


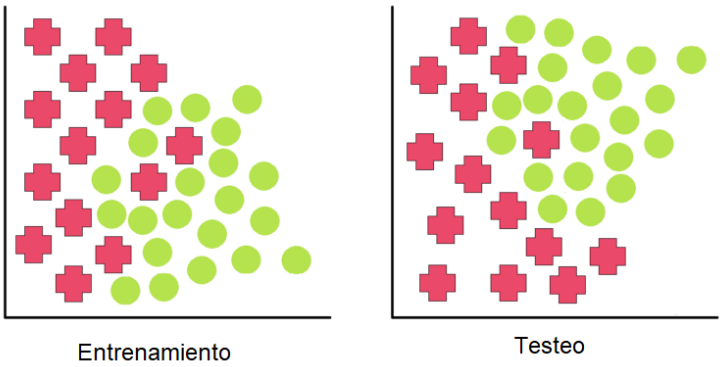
Figura 8-4.: Ilustración de tres tipos distintos de dataset shift. Créditos: Pawel Cislo `pawelcislo.com`

- Algoritmos de reconocimiento de rostros que son entrenados predominantemente con rostros jóvenes, pero los datos de testeo tienen una proporción mucho mayor de rostros de gente mayor.
- Predecir la expectativa de vida pero tener muy pocas instancias en los datos de entrenamiento de individuos que fuman, y muchos más en los datos de testeo.
- Clasificar imágenes como gatos o perros, pero omitir ciertas especies en los datos de entrenamiento que son comunes en los datos de testeo.

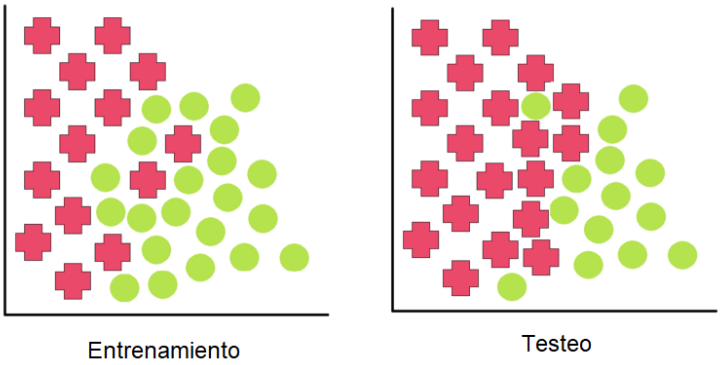
En particular, covariate shift puede causar muchos problemas al utilizar cross-validation. Cross-validation estimará la performance en test utilizando únicamente los datos de entrenamiento. Esta estimación no será realmente fiel a la performance en los datos de test. [López et al., 2014] [Sugiyama et al., 2007]

Otros tipos de dataset shift se ilustran en la figura **8-4** e incluyen:

- **Prior probability shift**: Mientras que covariate shift se concentra en cambios en los atributos x, prior probability shift sucede cuando únicamente la distribución de la variable objetivo y cambia [Quinonero-Candela et al., 2009] [Moreno-Torres et al., 2012]. Esto se ejemplifica en la figura **8-5**b.
- **Concept shift**: Sucede cuando el cambio no se encuentra puntualmente en las distribuciones de probabilidad, si no que la relación entre los atributos x y la variable objetivo y cambia [Moreno-Torres et al., 2012] [Kouw and Loog, 2019]. Concept drift puede suceder en situaciones donde los datos dependen altamente del tiempo. Por ejemplo, un modelo de aprendizaje automatizado entrenado para predecir el número de vuelos diarios en un cierto aeropuerto (ver figura **8-6**). Debido a variables que no son tenidas en cuenta por el modelo, como el contexto económico y social, la variable objetivo cambia con el tiempo. Un algoritmo entrenado con datos de 1949 a 1950 será incapaz de realizar predicciones en acertadas sobre 1960.



(a) Covariate shift



(b) Prior shift.

Figura 8-5.: Ejemplos de covariate shift (a) y prior shift (b). Notar que, en covariate shift, la probabilidad a priori de las clases no cambia entre train y test. Similarmente, en prior shift, la distribución de los atributos se mantiene constante. Créditos: `towardsdatascience.com`

En este capítulo nos concentraremos principalmente en analizar la presencia y el impacto de covariate shift en los tiles de Carpyncho, dado que experimentos llevados a cabo en capítulos anteriores han evidenciado la presencia de esta problemática en nuestros datos (ver figura **6-15**).

8.2.3. Efecto de covariate shift en SVM

En la figura **8-7** se ilustra el efecto perjudicial que covariate shift puede tener en SVM. En la subfigura de la izquierda, vemos el hiperplano separador óptimo que se calcula en base a los datos de entrenamiento (azul). Sin embargo, el dataset de test (rojo) exhibe una distribución de probabilidad ligeramente distinta a la distribución de entrenamiento. El hiperplano separador hallado durante la fase de entrenamiento no es capaz de separar correctamente las clases gobernadas por esta nueva distribución de probabilidad.

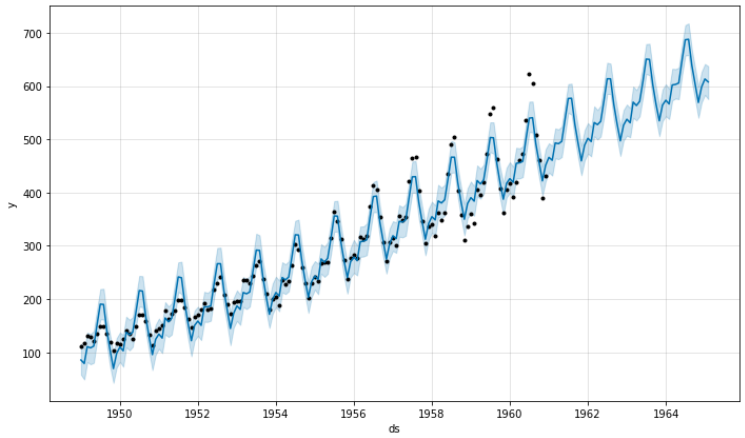


Figura 8-6.: Cantidad de vuelos de un aeropuerto en función del tiempo. Fuente: `gsarantitis.wordpress.com`

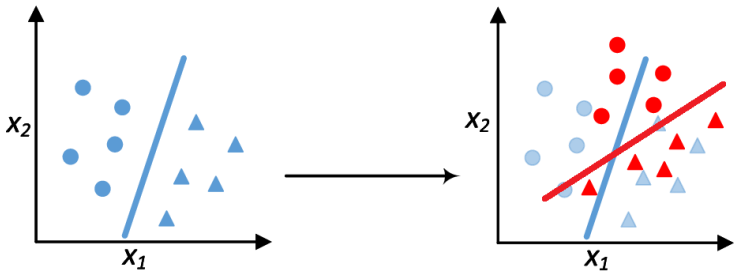


Figura 8-7.: En esta figura se ilustra el efecto negativo de covariate drift sobre SVM.

Distintos trabajos han estudiado el efecto que covariate drift tiene sobre clasificadores clásicos como SVM como árboles de decision [Sugiyama and Kawanabe, 2012]. En particular, los experimentos llevados a cabo en [Zadrozny, 2004] indican que los árboles de decisión toleran bastante bien la presencia de covariate drift:

> Sorpresivamente, C4.5 tiene muy buena performance en presencia de covariate drift. Esto podría ser explicado por el hecho de que, a pesar de que la elección de las divisiones en cada nodo (splits) está sesgada, las clases escogidas en las hojas no lo están.

En este mismo estudio, SVM muestra una mayor degradación en performance ante la presencia de covariate drift. Esto podría explicar por qué RF[2] parece verse menos afectado que SVM por la presencia de covariate drift en los tiles de Carpyncho.

[2]Si bien la implementación de RF utilizada en este trabajo utiliza el algoritmo CART (ver sección 1.3.6) en vez de C4.5, ambas implementaciones son muy similares. En la sección de trabajo futuro (9.2) se propone extender la investigación realizada por [Zadrozny, 2004].

Dataset shift es especialmente perjudicial en problemas de clasificación imbalanceados, dado que la clase positiva es muy sensible a errores de clasificación individuales debido la poca densidad de ejemplos [López et al., 2014].

8.2.4. Identificando la presencia de covariate shift

Teóricamente, a falta de mayores asunciones, cambios en la distribución de los atributos pueden causar una degradación arbitrariamente severa en la performance de clasificadores. Sin embargo, en la práctica, las distribuciones de train y test cambian constantemente, y a menudo estos cambios son benignos pues la diferencia es muy pequeña. Existe una amplia variedad de técnicas propuestas para determinar hasta qué punto los datos presentan covariate shift [Rabanser et al., 2019] [GeetaDharani. et al., 2019], incluyendo:

- **Distancia estadística**: Consiste en detectar si las distribuciones son distintas utilizando métricas como PSI (Population Stability Index), Kolmogorov-Smirnov statistic, Kullback-Lebler divergence, intersección de histogramas, etcétera. Esta última es ilustrada en la figura **8-8**. Estas técnicas, sin embargo, se vuelven más complejas de utilizar cuando los datos tienen alta dimensionalidad.

- **Detección de novedades** (Novelty detection): Dada una nueva instancia de test, utilizar métodos de detección de novedades (o detección de anomalías) para identificar si se condice o no a la distribución de entrenamiento [Cejnek and Bukovsky, 2018].

- **Distancia discriminatoria** (Discriminative distance): La intuición detrás de estos métodos es que, si hay covariate drift, entonces debería ser posible entrenar un clasificador que aprenda a separar los datos de entrenamiento y de testeo con alta performance. El error de este clasificador puede utilizarse como una medida de la distancia entre las dos distribuciones [Bickel et al., 2007] [GeetaDharani. et al., 2019]. Una ventaja de utilizar distancia discriminatoria es que puede utilizarse en dominios arbitrarios, incluyendo datos dispersos y con alta dimensionalidad.

Debido a la alta dimensionalidad de los tiles de Carpyncho, se decidió utilizar distancia discriminatoria para medir el nivel de covariate drift presente. Se procedió como sigue:

1. Dados dos tiles t_1 y t_2, sea n el mínimo entre la cantidad de filas de t_1 y la cantidad de filas de t_2.

2. Seleccionar aleatoriamente n elementos de t_1 y n elementos de t_2. Descartar las viejas etiquetas a predecir (RRL o noRRL). Se combinan todos los datos en un nuevo dataset t. Etiquetar cada fila de t con clase 1 si proviene del t_1 y 0 si proviene de t_2.

3. Dividir t en un conjunto de entrenamiento (75 % de los datos) y un conjunto de test (25 % de los datos).

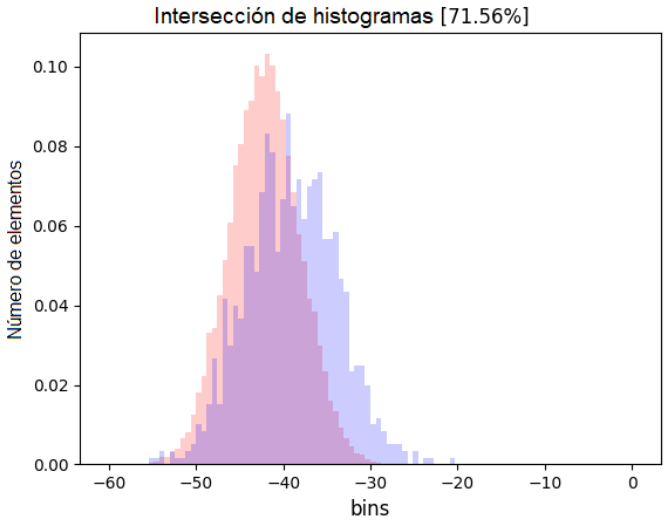


Figura 8-8.: Ilustración del cálculo de la intersección de histogramas de frecuencia correspondentes a los datos de test (rosa) y training (violeta). Hay aproximadamente 72 % de intersección entre las distribuciones, lo cual indica un nivel razonable de covariate shift entre las distribuciones. Fuente: `towardsdatascience.com`

4. Entrenar un clasificador RF y un clasificador de regresión logística (LR en adelante, [Hastie et al., 2001]) utilizando el dataset de entrenamiento.

5. Utilizar ambos modelos para calcular accuracy y el área debajo de la curva ROC[3] en test. Estas métricas son apropiadas por que t es un dataset completamente balanceado.

Algunos detalles del experimento: f

- RF se utiliza con tan solo 50 árboles. Los atributos no se escalan ni se discretizan. Previamente a entrenar el clasificador, se realiza selección de 45 atributos tal y como se explicó en la sección 5.3.

- Regresión logística se realizó con C=1. Se discretizaron los atributos utilizando binning y luego se escalan utilizando un escalado estándar. Previamente a entrenar el clasificador, se realiza selección de 45 atributos tal y como se explicó en la sección 5.3.

La figura **8-9** muestra los resultados obtenidos. En trabajos previos, se ha considerado que AUC-ROC mayor a 0.8 indica la presencia no despreciable de covariate shift [GeetaDharani. et al., 2019]. Como podemos ver, ambos clasificadores son sorprendentemente efectivos a la hora de distinguir la pertenencia a distintos tiles:

- Tanto LR como RF son capaces de distinguir con gran exactitud el tile de origen, LR en particular permite separar de forma casi perfecta algunos pares.

[3]La curva ROC se define en el glosario C

- Vemos que el tile $b360$ presenta el mayor covariate shift al comparársele con los demás tiles, especialmente con $b234$. Esto podría explicar por qué, en la figura **8-2**, las curvas donde se testea con $b360$ son las peores de todo el experimento (tanto para SVM como RF). Sin embargo, resulta curioso notar que al utilizar $b360$ para entrenar, la performance es notablemente buena a pesar de que el covariate shift se mantiene constante. La variación en la distribución subyacente, aparentemente, resulta más benigna en este último caso y los modelos obtenidos generalizan bien.

- El par $b261$ y $b278$ presenta la menor cantidad de covariate shift. Esto explicaría por qué las curvas $b261 \mapsto b278$ y $b278 \mapsto b261$ en la figura **8-2** son prácticamente idénticas.

- La severidad del covariate shift detectado parece ser proporcional a la distancia espacial de los tiles (ver figura **1-3**). De ser cierto, sería beneficioso entrenar y testear con tiles que se encuentran espacialmente cerca, siempre que sea posible.

- Utilizando el umbral de 0.8 AUC empleado en los trabajos previamente citados, ambos clasificadores coinciden en que todos los pares de tiles testeados se ven severamente afectados por covariate shift.

- Nótese que se muestra la matriz completa, en vez de mostrar sólo una matriz triangular. Esto es porque se distingue entre el tile de train y entrenamiento. El tile de train define cuáles serán los 45 atributos que se mantendrán en el experimento durante el paso de selección de atributos, previo a entrenar los clasificadores. A pesar de esta diferencia, vemos que el covariate shift medido con $t1 \mapsto t2$ es prácticamente el mismo que el medido con $t2 \mapsto t1$.

- Obviamente, cuando el tile de entrenamiento y de testeo son el mismo resulta imposible distinguirlos, y los clasificadores tienen la misma performance que un clasificador aleatorio (accuracy del 50 %, área debajo de la curva ROC 0.5).

8.2.5. Reduciendo el impacto covariate shift

Una vez que se ha identificado que los datos se ven afectados por covariate shift, distintos trabajos han propuesto multitud de técnicas para mitigar el efecto que tiene en performance. Algunas de las ideas más predominantes son:

- **Importance reweighting**: Consiste en aumentar la importancia de aquellos elementos del conjunto de entrenamiento que son más similares a las instancias en test [Kouw and Loog, 2019] [Sugiyama and Kawanabe, 2012] [GeetaDharani. et al., 2019]. Esencialmente, se intenta sesgar al clasificador para que preste más atención a aquellos datos de entrenamiento que se parecen más a los datos de test. Esto se ilustra en la figura **8-10**.

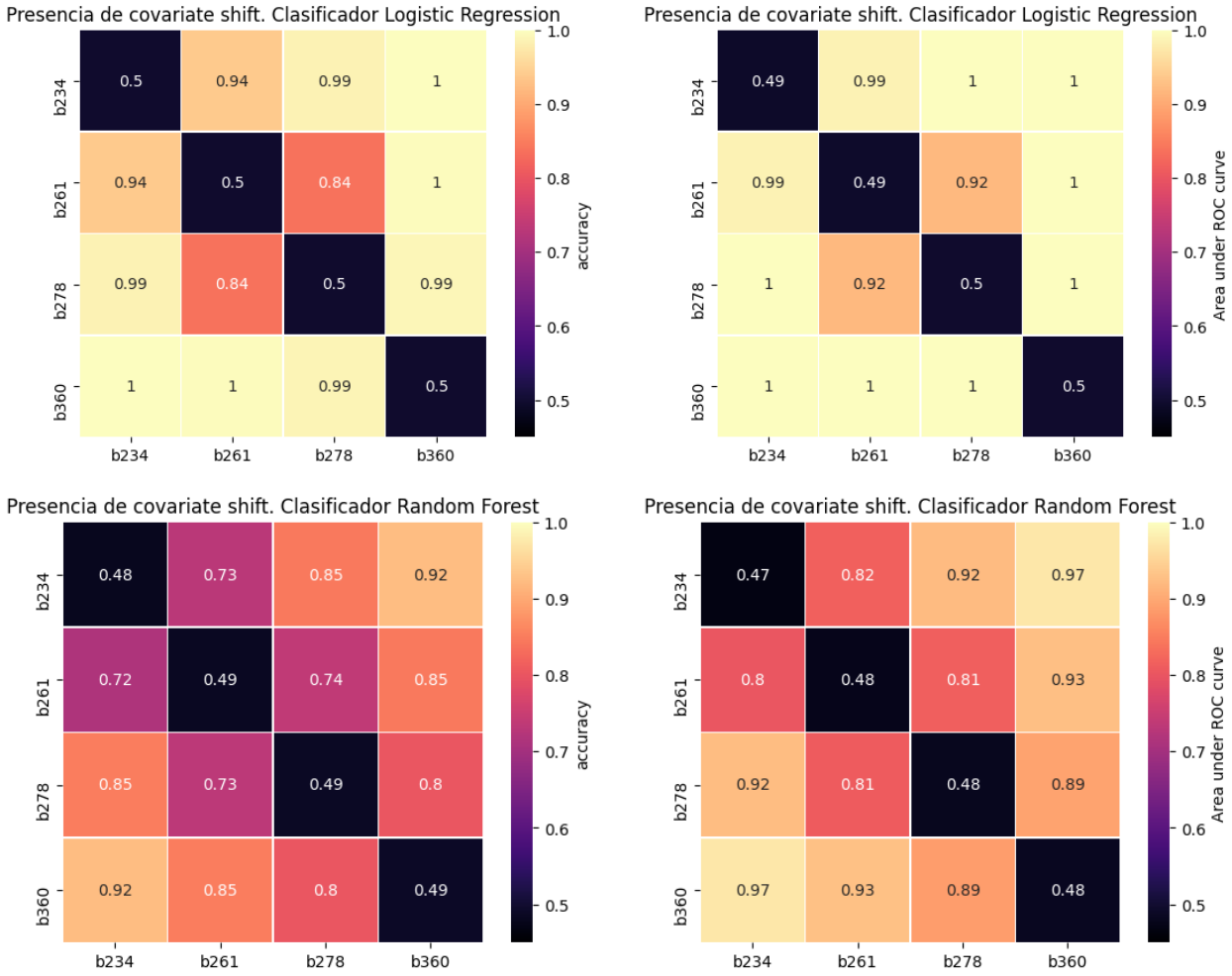


Figura 8-9.: Cálculo de distancia discriminatoria sobre un subconjunto de tiles de la VVV. Se intenta medir el grado de separación entre las distribuciones de probabilidad de distintos tiles, entrenando clasificadores RF y LR que buscan diferenciar ambas distribuciones.

- **Eliminar atributos que causan covariate shift**: Eliminar aquellos atributos que más contribuyen a la diferencia entre datasets. Una forma de hacer esto es, nuevamente, utilizando la idea de distancia discriminatoria desarrollada en la sección anterior:
  - Luego de entrenar un clasificador que distinga entre los datos de entrenamiento y de test, puede estimarse qué atributos son más importantes para el modelo (utilizando importancia gini en RF, por ejemplo). Aquellos atributos que no son fundamentales para el problema de clasificación original, y que contribuyen notablemente a diferenciar el tile de train del de test, pueden ser eliminados. Esto se ilustra en la figura **8-11**.
  - Una segunda opción es, para cada atributo, entrenar un clasificador que utilice

únicamente ese atributo para distinguir entre puntos de test y train. Aquellos atributos cuyo clasificador alcance un ROC-AUC en test mayor a un cierto umbral, por ejemplo 0.8 ([GeetaDharani. et al., 2019]), son candidatos a ser eliminados.

Variaciones de los métodos recién explicados pueden consultarse en [Raza et al., 2016] [Liu and Ziebart, 2017] [Sugiyama et al., 2007] [Raza et al., 2014] [Kouw and Loog, 2019]. La mayoría de los trabajos muestran claramente que hay algún beneficio en *hacer algo diferente* ante la presencia de covariate shift. Sin embargo, esto debe ser evaluado con cuidado pues la mayoría de los métodos requiere el uso de los datos de testing, si bien no etiquetados, durante la fase de entrenamiento. Esto puede considerarse filtrado de información, o ser simplemente imposible.

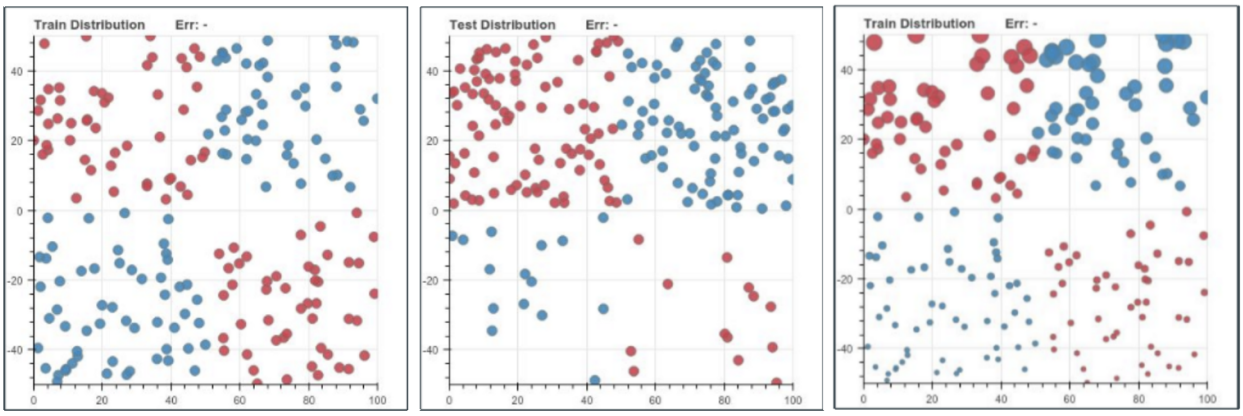


Figura 8-10.: Importance reweighting consiste en modificar el peso de cada instancia de entrenamiento, en base a la probabilidad relativa de los datos de entrenamiento y los datos de testeo. Fuente: `towardsdatascience.com`

Es factible aplicar los dos métodos recién descriptos a nuestro problema de clasificación de RRLs. En primer lugar, tanto RF como SVM permiten asignar un cierto peso o importancia a cada elemento de los datos de entrenamiento, lo cual posibilita el uso de importance reweighting. Hay una amplia variedad de técnicas propuestas para calcular los pesos adecuados, pero se ha decidido no aplicarlas en este trabajo para mantener la longitud del mismo acotada. Sería, sin embargo, muy interesante explorar los potenciales beneficios que se podría obtener.

Se optó por intentar eliminar atributos que causan covariate shift, estudiando puntualmente el par de tiles $b360 - b234$, pues muestra la mayor divergencia. En primer lugar, se analizó la importancia de cada atributo a la hora de diferenciar entre $b360$ y $b234$. Esta importancia se obtiene a partir de los clasificadores entrenados en la sección anterior; en el caso de RF es la importancia gini y en el caso de LR se obtienede forma transparente a partir de los coeficientes asignados a cada atributo [Molnar, 2019].

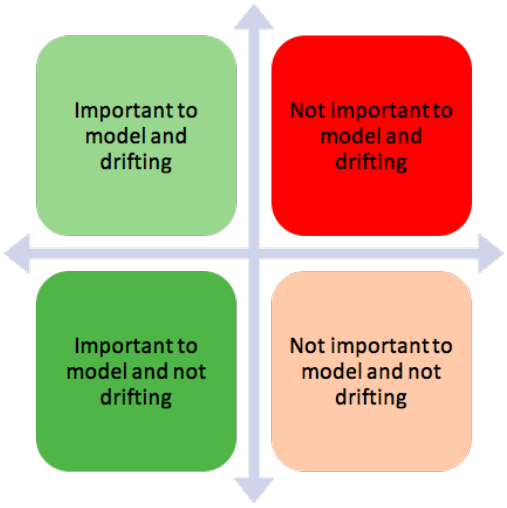


Figura 8-11.: Ante la presencia de covariate drift en un problema de clasificación, podemos clasificar los atributos en cuatro grupos. Por un lado cada atributo contribuye en cierto grado a la presencia de covariate drift, y por otro lado cada atributo tiene una cierta importancia para el problema de clasificación original. Se prefiere eliminar aquellos atributos que contribuyen en mayor medida a la presencia de covariate drift, y que no son vitales para modelar el problema original. Fuente: `kdnuggets.com`

En la figura **8-12** se muestra, para cada atributo, la importancia obtenida por RF y LR (normalizada) a la hora de separar ambos tiles. Notar que se aplicó selección de atributos previamente, por lo tanto sólo 45 atributos fueron evaluados. Los atributos están ordenados en el eje x de acuerdo a su importancia para clasificar RRLs en el tile b360, de acuerdo a mutual information (ver sección 6.1.1). Algunas observaciones:

- Tanto RF como LR señalan mayormente a atributos de color entre los más informativos para distinguir entre b360 y b234. Es decir, los atributos de color contribuyen mucho a la presencia de covariate shift. Quizás sea necesario realizar algún tipo de preprocesamiento extra de tal forma que los datos de color sean más homogéneos entre distintos tiles.

- Los 6 atributos más importantes para distinguir RRLs y no-RRLs (de acuerdo a mutual information) parecen no contribuir significativamente a la presencia de covariate drift.

Habiéndose obtenido los puntajes de importancia para cada atributo, se procedió a evaluar cuánto disminuye la presencia de covariate drift al eliminar (uno a uno) los atributos que más contribuyen. Sin embargo, se omite eliminar aquellos atributos que se encuentran entre los 5 atributos que más contribuyen a diferenciar RRL de no-RRL de acuerdo a mutual information, para evitar perder información crítica. Los resultados se presentan en la figura **8-13**. Algunas observaciones:

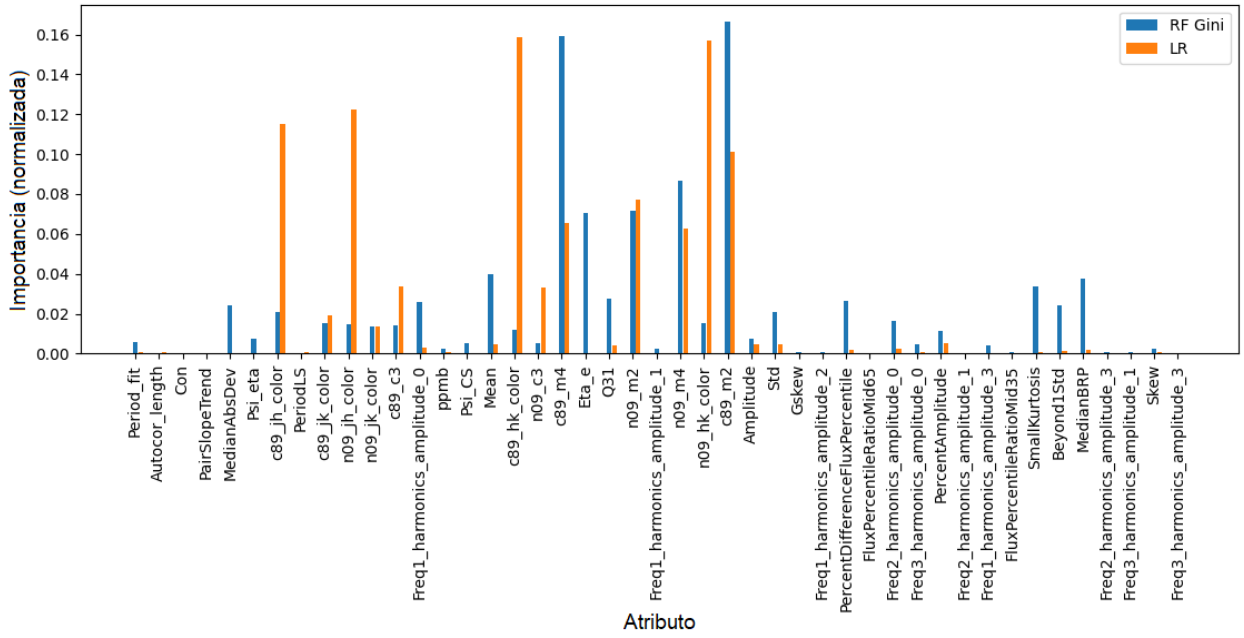


Figura 8-12.: Importancia de cada atributo a la hora de distinguir entre elementos de b360 y b234 provista por clasificadores RF y LR. Los puntajes de importancia provistos por cada método se muestran normalizados (de tal forma que la suma de todos los puntajes sea 1), pero no son directamente comparables entre distintos métodos.

- Podemos ver que la capacidad de distinguir entre b360 y b234 efectivamente se degrada al eliminar atributos que causan covariate drift. En particular, RF pierde performance mucho más rápido; en tanto que LR recién comienza a perder performance a partir de 10 atributos eliminados.
- Si deseamos disminuir la distancia discriminatoria a menos de 0.8 AUC, se necesita eliminar aproximadamente 20 atributos. Esto tiene un costo asociado, pues la pérdida de información puede degradar la performance en el problema de clasificación original.

El hecho de que la presencia de covariate drift no disminuya lo suficientemente rápido al eliminar los primeros atributos puede deberse a los siguientes motivos:

- El drift podría estar distribuido entre todos los atributos, por lo que los clasificadores pueden utilizar los atributos aún no eliminados para dividir entre b234 y b360 con alta efectividad.
- Algún atributo que tiene alta capacidad de diferenciar entre b234 y b360 no fue eliminado por que mutual information lo considera importante para separar RRL y no-RRL.

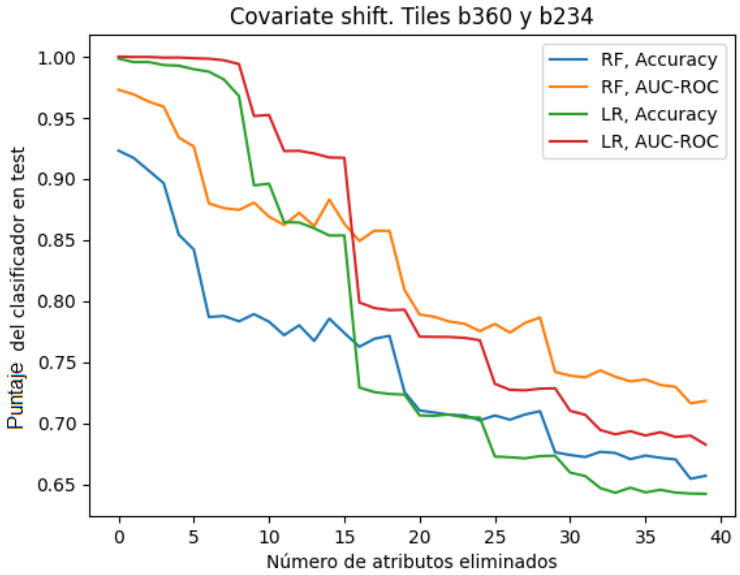


Figura 8-13.: En esta figura se muestra cómo se degrada la capacidad de RF y LR de distinguir entre b234 y b360 al eliminar atributos que contribuyen notoriamente a la presencia de covariate drift.

Para poder aclarar esta situación, se procedió a entrenar un clasificador por cada atributo. La tarea del clasificador nuevamente será intentar distinguir entre elementos de b234 y elementos de b360, pero esta vez utilizando únicamente un atributo. Los resultados se encuentran en la figura **8-14**, y nos permiten apreciar que casi todos los atributos contribuyen a la presencia de covariate drift. Esto se deduce del hecho de que prácticamente todos los clasificadores individuales tienen mejor performance que un clasificador aleatorio. En particular, los atributos de color muestran nuevamente ser los más informativos para separar tiles.

Finalmente, una vez identificados los atributos que más contribuyen a la presencia de covariate drift, se procedió a evaluar si eliminarlos conduce a algún beneficio en el problema de clasificación original. Para ello, se entrenó SVM-RBF en b360, testeando en b234 utilizando los preprocesamientos e hiperparámetros óptimos hallados en la sección 5.3. Antes de entrenar el clasificador, se eliminan los n atributos que más contribuyen a la presencia de covariate drift, omitiendo eliminar aquellos que se encuentran entre los 5 más importantes para distinguir RRLs de noRRLs.

Desafortunadamente, no se observó mejoría alguna en R-AUPRC. Esto puede deberse a que es necesario eliminar demasiados atributos para reducir el covariate drift significativamente, y la pérdida de información asociada es demasiado grande.

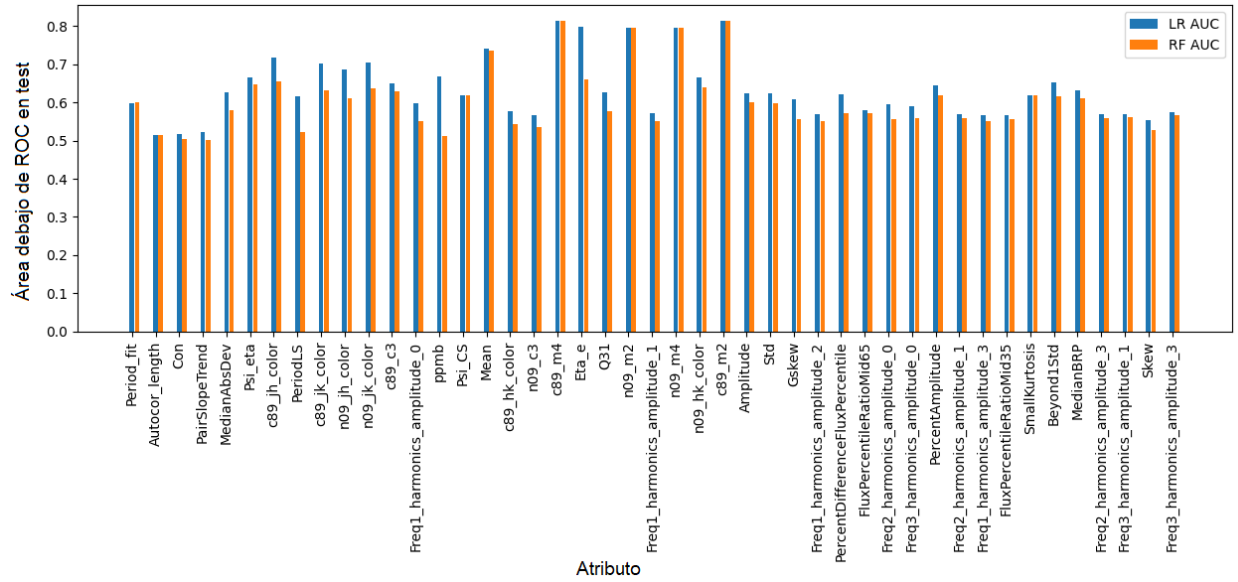


Figura 8-14.: Estas figuras muestran la capacidad de cada atributo a la hora de distinguir entre b360 y b234. Por cada atributo, se entrenó un clasificador que utiliza únicamente ese atributo para distinguir entre b234 y b360.

8.3. Conclusiones

En este capítulo se verificó, utilizando distancia discriminatoria, que el problema de clasificación que se aborda en esta tesina sufre de un fenómeno denominado covariate shift, en el cual las distribuciones de probabilidad de los datos de entrenamiento y de testeo son significativamente distintas.

Si bien en la sección 8.1 se verificó que esto no imposibilita hallar hiperparámetros adecuados para todos los tiles, la presencia de covariate drift se propone como una explicación razonable a la disparidad de performance en clasificación que se ve al utilizar distintos tiles de entrenamiento y testeo.
f Se intentó reducir el impacto de covariate shift eliminando los atributos que más contribuyen a distinguir distintos tiles. Sin embargo, esta técnica no logró mejorar la performance de los clasificadores SVM. Es probable esto se deba a que prácticamente todos los atributos contribuyen a la disparidad entre distintos tiles, y eliminar tan solo un puñado de atributos no logrará corregir la disparidad global de los distintos tiles. Se propone el uso de técnicas más sofisticadas, como importance reweighting, como alternativas interesantes para explorar en trabajos futuros.

Trabajos previos indican que los árboles de decisión se ven menos afectados por covariate shift que SVM. Esto se condice con el hecho de que todos los pares de tiles donde RF supera

en performance a SVM estén entre aquellos que maximizan la presencia de covariate shift.

9. Conclusión y trabajo futuro

9.1. Conclusión

En esta tesina se abordó la problemática de clasificar estrellas variables de tipo RRL utilizando SVM, intentando igualar la performance obtenida por RF en trabajos previos, así como entender por qué los ensambles de árboles parecen funcionar mejor.

A diferencia de trabajos previos, en este trabajo se utilizaron tiles completas (sin undersampling) para calcular la performance de SVM en test; asumiendo el costo computacional. Esto nos permitió comparar fehacientemente la performance de RF y SVM, dado que testear en datos subsampleados produce resultados irreales y optimistas.

Otra contribución interesante de esta tesina es el uso de un aproximador de kernel que permite utilizar SVM con kernel RBF en tiles de gran tamaño. Trabajos previos se limitaron al uso de SVM RBF exacto, que únicamente es capaz de trabajar en datasets de pequeño tamaño.

Con el objeto de mejorar la performance de SVM, se realizó una serie de experimentos agregando preprocesamientos y realizando modificaciones a las SVM, intentando abordar cuestiones que no habían sido consideradas en trabajos previos.

En primer lugar se exploró escalar y discretizar los datos de distintas formas. Se seleccionó binning quantile junto con un escalado estándar como la elección óptima, obteniendo un notable incremento promedio en el área bajo la curva de Precision-Recall. Esto permitió mitigar el efecto de distintas escalas, outliers y ruido en los atributos. Posteriormente se hallaron hiperparámetros C y γ óptimos utilizando cross validation, de forma tal que no se sobreajusten los datos de entrenamiento.

Otra serie de experimentos consistió en evaluar distintas técnicas de selección y extracción de atributos, tendientes a reducir la cantidad de atributos usados para describir cada estrella. Se decidió eliminar 17 (14 en SVM-L) atributos utilizando filtros univariable, lo que permitió deshacerse de atributos no informativos y reducir la complejidad computacional y espacial significativamente, junto con un leve incremento en el área bajo la curva de Precision-Recall.

Se procedió a inspeccionar los datos en detalle, buscando comprender qué atributos son los más informativos para detectar RRLs. También se identificó la presencia de bloques de atributos altamente correlacionados en nuestros tiles, pero se descartó que esto tuviese un impacto negativo en la performance de los clasificadores SVM.

Se experimentó con distintas técnicas para corregir el marcado desbalance de clases presente en los datos. Se concluyó que el uso de oversampling aleatorio conduce a una leve mejoría en SVM Lineal, en tanto que SVM-RBF no mejora su performance al corregir el desbalance.

Como resultado final, la performance de SVM logró igualar a la de RF en varios de los tiles estudiados, aunque RF continúa siendo superior en otros (ver figura **8-2**). Numéricamente, se logró un crecimiento el área bajo la curva de precision recall robusta promedio en SVM-Lineal del 22,08 % respecto a trabajos anteriores ([Cabral et al., 2020a]), en tanto que el incremento en SVM-RBF fue del 13,79 %. Adicionalmente, realizar selección de atributos conduce a modelos que son más eficientes en términos de tiempo de cómputo y uso de memoria. En este punto, ya se han abordado todas las razones obvias por las cuales SVM podría funcionar peor que RF: sobreajuste, escalado de los datos, ruido y outliers, atributos no informativos o altamente correlacionados y desbalance de clases.

Finalmente, se profundiza sobre el hecho de que el problema de clasificación estudiado en esta tesina incumple una asunción básica en aprendizaje automatizado clásico: los datasets de entrenamiento y de testeo no están regidos por la misma distribución de probabilidad. Este fenómeno, llamado dataset drift, tiene poca visibilidad en el campo de aprendizaje automatizado a pesar de ser muy común en la práctica y no parece haber sido considerado en trabajos previos.

Se propuso que la marcada diferencia entre las distribuciones de probabilidad subyacentes a cada tile es la causa por la cual la performance en test varía tanto entre distintos tiles. Basándose en resultados de trabajos previos, se conjetura que RF se ve menos afectado que SVM por la presencia de dataset shift, lo cuál explicaría por qué SVM es incapaz de superar a RF en pares de tiles espacialmente distantes.

Como experimento final se intentó corregir, sin éxito, la divergencia entre las distribuciones; eliminando atributos que contribuyen marcadamente a distinguir tiles. Considerando que prácticamente todos los atributos contribuyen a la divergencia, parece ser necesario recurrir a métodos más sofisticados para mitigar los efectos del dataset shift presente.

9.2. Trabajo futuro

Con el objeto de mantener la longitud de este trabajo acotada, se optó por no realizar ciertos experimentos:

- Resultaría interesante evaluar si varias de las técnicas aplicadas en este trabajo, tales como corrección del imbalance de clases, logran mejorar la performance de RF. En esta tesina los esfuerzos se concentraron únicamente en mejorar SVM.
- Puede ser beneficioso utilizar otros algoritmos de aprendizaje automatizado para abordar este problema. En particular, Gradient Boosting [Friedman, 2000] [Friedman, 2002] es una opción muy interesante teniendo en cuenta el alto desempeño obtenido por Adaboost a la hora de clasificar RRLs [Elorrieta López et al., 2016].
- La mayoría de los experimentos de este trabajo se concentraron en agregar preprocesamientos y modificaciones a SVM. Sin embargo, para comprender por qué RF funciona mejor que SVM, puede ser interesante diseñar otro conjunto de experimentos enfocados en RF, intentando entender qué es lo que ocasiona que funcione tan bien.
- Sería interesante realizar un análisis de aquellas estrellas que están siendo clasificadas incorrectamente por SVM, intentando determinar si hay algún patrón en ellas que SVM no es capaz de aprender. También sería de interés evaluar si las estrellas incorrectamente clasificadas por RF son un subconjunto de aquellas incorrectamente clasificadas por SVM.
- Experimentar con distintos tipos de regularización y funciones de pérdida en SVM (ver sección 1.3.8)

Respecto a la presencia de dataset shift en nuestros datos, se propone como trabajo futuro:

- Uilizar importance reweighting ([Kouw and Loog, 2019], [Sugiyama and Kawanabe, 2012] , [GeetaDharani. et al., 2019]) para mitigar el efecto de covariate shift. Esto tiene el potencial de mejorar la performance de los clasificadores. Debido a la alta dimensionalidad de los datos, se deberá buscar métodos aproximados para hallar los coeficientes asociados a cada instancia de entrenamiento.
- Explorar las múltiples técnicas para abordar la presencia de covariate shift que han sido propuestas en trabajos recientes [GeetaDharani. et al., 2019], incluyendo mapeo de subespacios [Kouw and Loog, 2019] y reducción de dimensionalidad extrema [Wang and Rudin, 2018].
- Realizar experimentos para verificar si, efectivamente, RF es más robusto que SVM ante datos que presentan covariate shift.

- Una alternativa es combinar datos de distintos tiles a la hora de entrenar, en vez de utilizar datos de un único tile. Esto permitirá que los clasificadores no estén tan sesgados hacia un área particular de la vía láctea.
- Finalmente, se sugiere modificar el conjunto de atributos utilizado para describir cada estrella, tratando de codificar la misma información de alguna forma que sirva para caracterizar estrellas independientemente del tile al que pertenece.

Otras avenidas de trabajo futuro incluyen:

- Extender los datos recolectados en Carpyncho, agregando más tiles de la VVV y etiquetando RRLs identificadas en otros trabajos.
- Realizar experimentos similares para tratar de clasificar otro tipo de estrellas variables en la VVV.

A. Listado de características

A continuación se enumeran todas las características numéricas que son utilizadas en este trabajo para describir una curva de luz. El primer subgrupo fue generado utilizando la herramienta Feets [Cabral et al., 2018], mientras que el segundo subgrupo son características de color introducidas en [Cabral et al., 2020a]

A.1. Características extraídas utilizando Feets

Las descripciones de este listado están basadas en la documentación de Feets `https://feets.readthedocs.io`:

- **Amplitude**: La mitad de la diferencia entre la mediana del 5 % de los valores mayores y la mediana del 5 % de los valores más pequeños de las magnitudes en la banda K_S.
- **Autocolor_length**: (Auto-Correlation Length) Correlación cruzada de la señal con si misma. Es útil para encontrar patrones, como señales periódicas, ocultas por ruido.
- **Beyond1Std**: Porcentaje de puntos más allá de una desviación estándar de la media ponderada.
- **Con**: (Consecutive points) Cantidad de tres puntos consecutivos mayores o menores que 2σ (normalizado por $N - 200$).
- **Eta_e**: Verifica la dependencia de observaciones con respecto a las anteriores.
- **FluxPercentileRatioMid**: (Proporción del flujo de percentiles centrales) Caracteriza las distribuciones de magnitud ordenadas en base a percentiles de sus flujos[1]. Se calcularon las siguientes cinco features, donde $F_{i,j}$ es la diferencia entre el percentil de flujo j y el percentil de flujo i:
  - FluxPercentileRatioMid20 = $F_{40,60}/F_{5,95}$
  - FluxPercentileRatioMid35 = $F_{32,5,67,5}/F_{5,95}$
  - FluxPercentileRatioMid50 = $F_{25,75}/F_{5,95}$
  - FluxPercentileRatioMid65 = $F_{17,5,82,5}/F_{5,95}$

[1]Flujo o luminosidad, es la cantidad de energía que brinda una estrella en una unidad de tiempo

  - FluxPercentileRatioMid80 = $F_{10,90}/F_{5,95}$

- **Componentes de Fourier**: Las primeras tres componentes de Fourier para los primeros tres períodos candidatos. *Freqk_harmonics_amplitude_i* y *Freqk_harmonics_rel_phase_j* representan la i-ésima amplitud y la j-ésima fase de la señal para el k-ésimo período candidato. (Un total de 18 features)
- **Gskew**: Medida de distorsión basada en la mediana de magnitudes.
- **LinearTrend**: (Tendencia linear) Es la pendiente del ajuste lineal de los datos.
- **MaxSlope**: Pendiente máxima absoluta entre dos observaciones consecutivas.
- **Mean**: Media de magnitudes.
- **MedianAbsDev**: (Mediana de las desviaciones absolutas) La mediana de la diferencia de cada magnitud observada con la mediana de la serie.
- **MedianBRP**: (Median Buffer Range Percentage) Proporción de las magnitudes entre la décima parte del rango total alrededor de la mediana.
- **PairSlopeTrend**: Tendencia de pendiente de a pares.
- **PercentAmplitude**: (Porcentaje de Amplitud) Máximo porcentaje de diferencia entre la máxima o la mínima magnitud y su mediana.
- **PercentDifferenceFluxPercentile**: $F_{5,95}$ convertido a magnitud.
- **PeriodLS**: Período extraído usando el método de Lomb-Scargle.
- **Period_fit**: (Period Fitness) Medida de la confiabilidad de PeriodLS.
- **Psi_CS**: Índice R_CS aplicado a la curva en fase, utilizando el período extraído con el método de Lomb-Scargle.
- **Psi_eta**: Índice η^e calculado con la curva de luz puesta en fase con el período extraído con el método de Lomb-Scargle.
- **Q31**: Diferencia entre el tercer cuartil y el primer cuartil de magnitudes.
- **Rcs**: (Range Cumulative Sum) Rango obtenido luego de calcular la suma acumulada de mediciones.
- **Skew**: La asimetría de las magnitudes.
- **SmallKurtosis**: Kurtosis pequeña de las magnitudes.
- **Std**: La desviación estándar de las magnitudes.

A.2. Características de color

A continuación se encuentra el listado de features relativas al color de una estrella que se utilizaron en este trabajo.

- **c89_c3**: C3 Pseudo-color usando la Ley de Extinción de Cardelli-89.
- **c89_ab_color**: Diferencia de magnitud en la primera época entre la banda a y la banda b, usando la Ley de Extinción de Cardelli-89. a y b pueden ser las bandas H, J y Ks.
- **c89_m2** y **c89_m4**: Pseudo-magnitudes m2 y m4 usando la Ley de Extinción de Cardelli-89.
- **n09_c3**: C3 Pseudo-color usando la Ley de Extinción de Nishiyama-09.
- **n09_ab_color**: Diferencia de magnitud en la primera época entre la banda a y la banda b, usando la Ley de Extinción de Nishiyama-09. a y b pueden ser las bandas H, J y Ks.
- **n09_m2** y **n09_m4**: Pseudo-magnitudes m2 y m4 usando la Ley de Extinción de Nishiyama-09.
- **ppmb** (Pseudo-Phase Multi-Band): Este índice setea el primer instante en fase con respecto al tiempo promedio en todas las bandas, usando el período calculado por Feets:

  $$ppmb = frac(\frac{|mean(HJD_H, HJD_J, HJD_{K_s}) - T_0|}{P})$$

  Donde HJD_H, HJD_J y HJD_K son el tiempo de las observaciones en las bandas H, J y K_S; y T_0 es el tiempo de observación de la máxima magnitud en la banda K_S. P es el período extraído. $frac$ retorna sólo la parte decimal del número.

Para más información acerca de la Ley de Extinción de Cardelli-89, consultar [Cardelli et al., 1989]. Para más información acerca de la Ley de Extinción de Nishiyama-09, consultar [Nishiyama et al., 2009]. Para más información acerca de pseudo colores, consultar [Minniti et al., 2010]

A.3. Índice de atributos

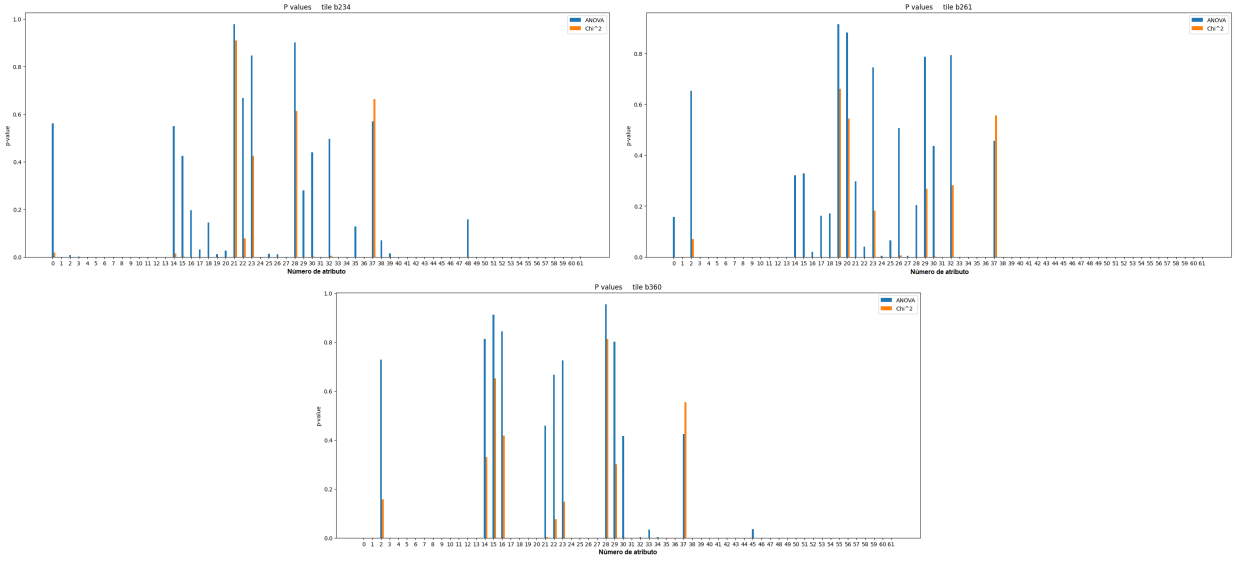
A continuación se enumeran todos los atributos nuevamente, así como un número identificador para cada uno. Estos identificadores serán utilizados para referirse al nombre de cada atributo en distintas secciones del trabajo.

0. Amplitude
1. Autocor_length
2. Beyond1Std
3. Con
4. Eta_e
5. FluxPercentileRatioMid20
6. FluxPercentileRatioMid35
7. FluxPercentileRatioMid50
8. FluxPercentileRatioMid65
9. FluxPercentileRatioMid80
10. Freq1_harmonics_amplitude_0
11. Freq1_harmonics_amplitude_1
12. Freq1_harmonics_amplitude_2
13. Freq1_harmonics_amplitude_3
14. Freq1_harmonics_rel_phase_1
15. Freq1_harmonics_rel_phase_2
16. Freq1_harmonics_rel_phase_3
17. Freq2_harmonics_amplitude_0
18. Freq2_harmonics_amplitude_1
19. Freq2_harmonics_amplitude_2
20. Freq2_harmonics_amplitude_3
21. Freq2_harmonics_rel_phase_1
22. Freq2_harmonics_rel_phase_2
23. Freq2_harmonics_rel_phase_3
24. Freq3_harmonics_amplitude_0
25. Freq3_harmonics_amplitude_1
26. Freq3_harmonics_amplitude_2
27. Freq3_harmonics_amplitude_3
28. Freq3_harmonics_rel_phase_1
29. Freq3_harmonics_rel_phase_2
30. Freq3_harmonics_rel_phase_3
31. Gskew
32. LinearTrend
33. MaxSlope
34. Mean
35. MedianAbsDev
36. MedianBRP
37. PairSlopeTrend
38. PercentAmplitude
39. PercentDifferenceFluxPercentile
40. PeriodLS
41. Period_fit
42. Psi_CS
43. Psi_eta
44. Q31
45. Rcs
46. Skew
47. SmallKurtosis
48. Std
49. c89_c3
50. c89_hk_color
51. c89_jh_color
52. c89_jk_color
53. c89_m2
54. c89_m4
55. n09_c3
56. n09_hk_color
57. n09_jh_color
58. n09_jk_color
59. n09_m2
60. n09_m4
61. ppmb

B. Gráficas

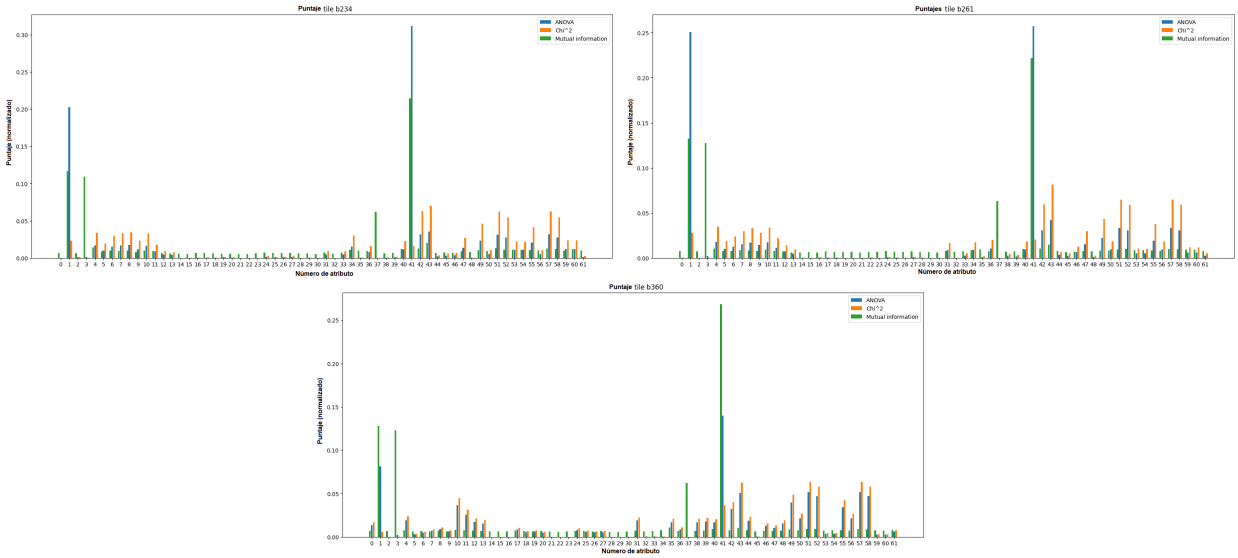
B.1. Tests estadísticos - pvalues

En esta sección se muestran los p-values de los tests χ^2 y ANOVA, complementando la figura **6-4**.



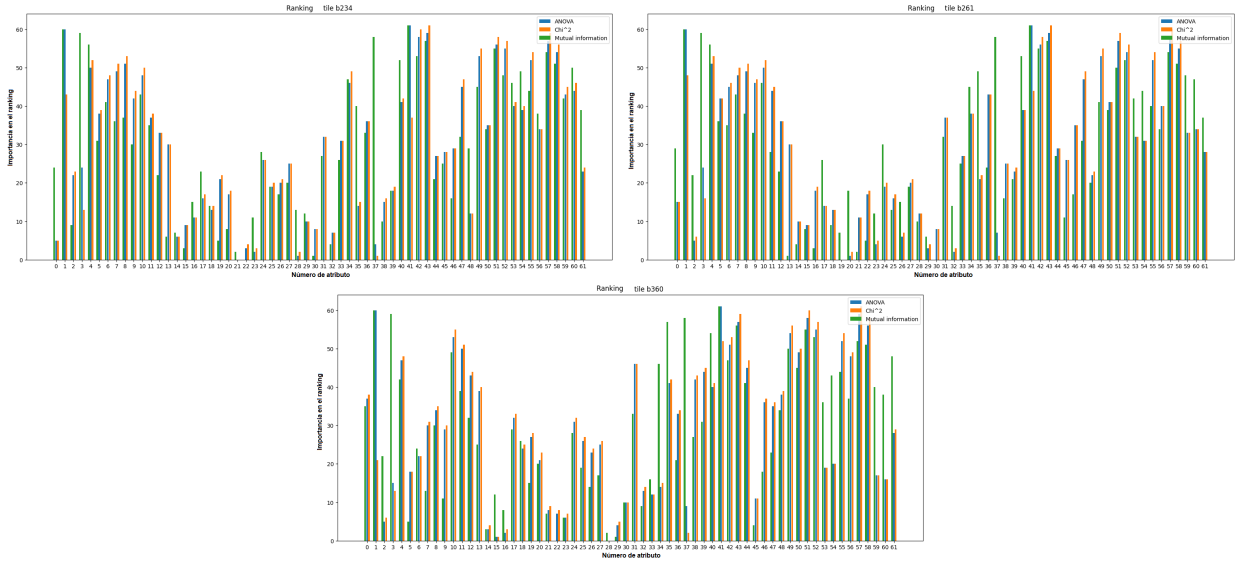
B.2. Tests estadísticos - métricas

En esta sección se muestran los puntajes de los tests χ^2 y ANOVA, así como la información mutua, complementando la figura **6-5**.



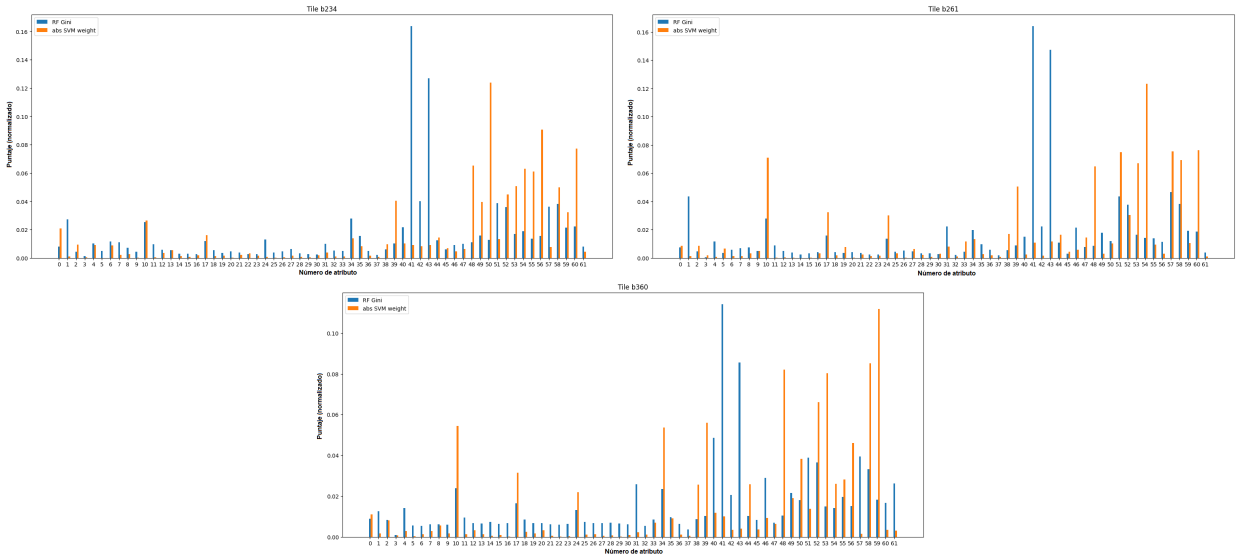
B.3. Tests estadísticos - rankings

En esta sección se muestran los rankings de atributos generados por los tests χ^2 y ANOVA, así como por información mutua, complementando la figura **6-6**.



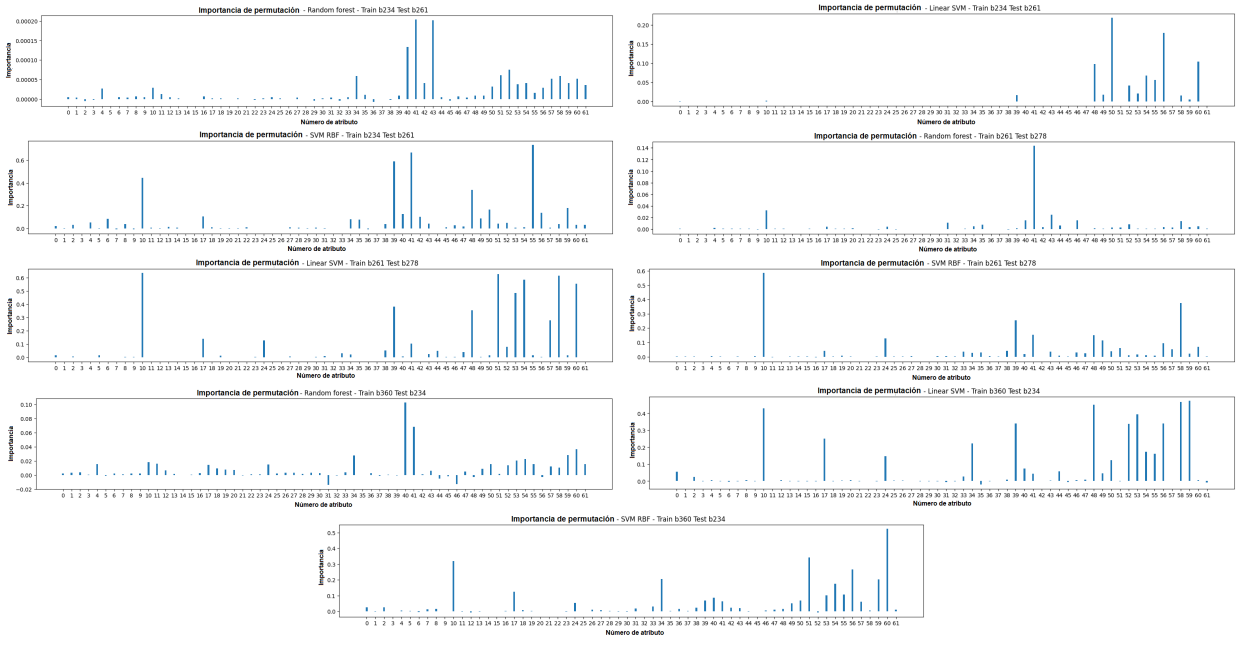
B.4. Importancia de atributos basada en clasificadores

En esta sección se muestra la importancia asignada por SVM-Lineal y RF en distintos tiles, y complementando la figura **6-7**.



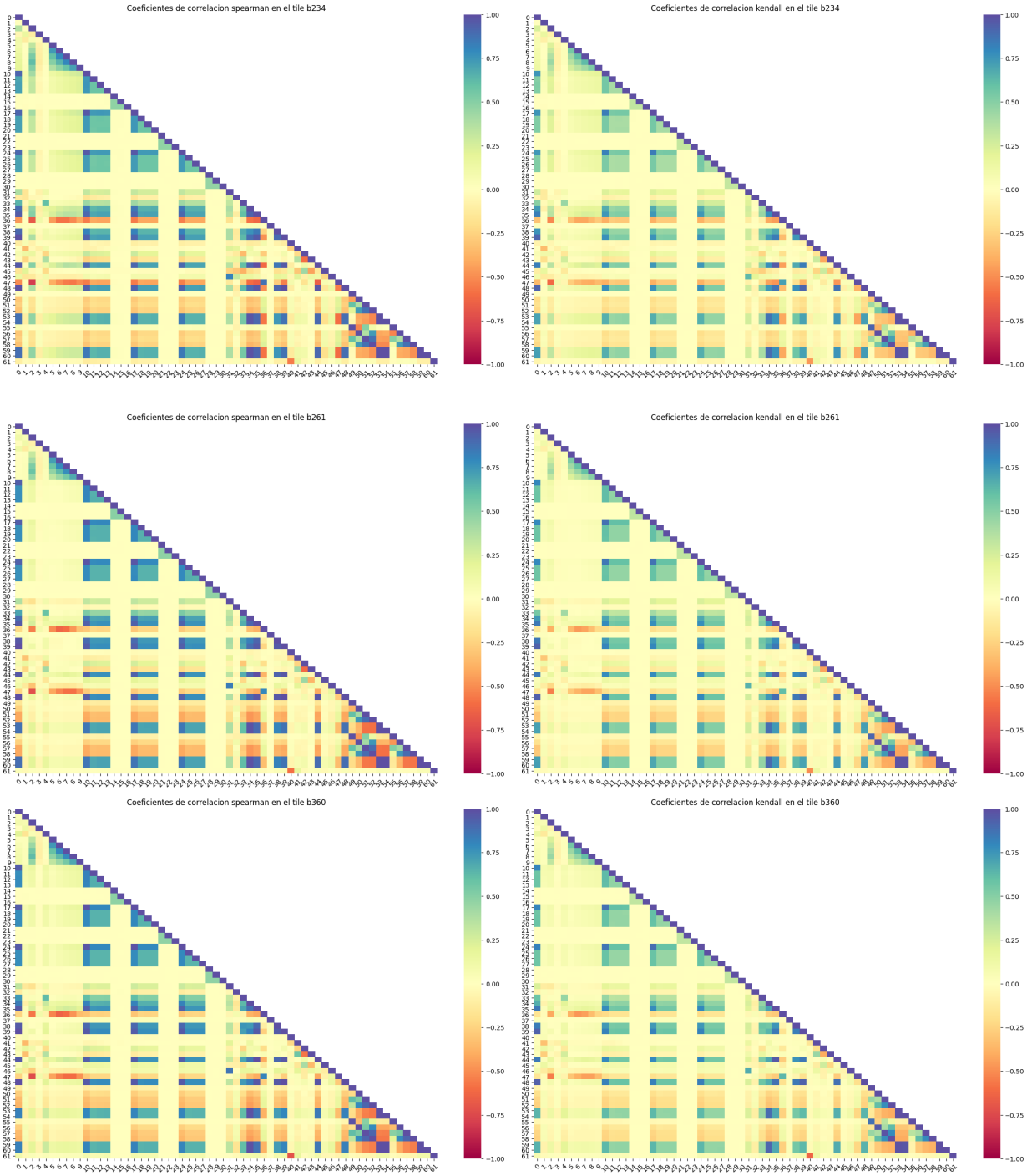
B.5. Importancia de atributos basada en permutaciones

En esta sección se muestra la importancia basada en permutaciones estimada por RF, SVM Lineal y SVM RBF en distintos tiles, y complementando la figura **6-8**.



B.6. Matrices de correlación entre atributos

En esta sección se muestra el coeficiente de correlación para cada par de atributos en los tiles b234, b261 y b360.



B.7. Distribuciones de probabilidad de atributos

En esta sección se muestran las distribuciones de probabilidad de los atributos de los tiles b234, b261 y b360.

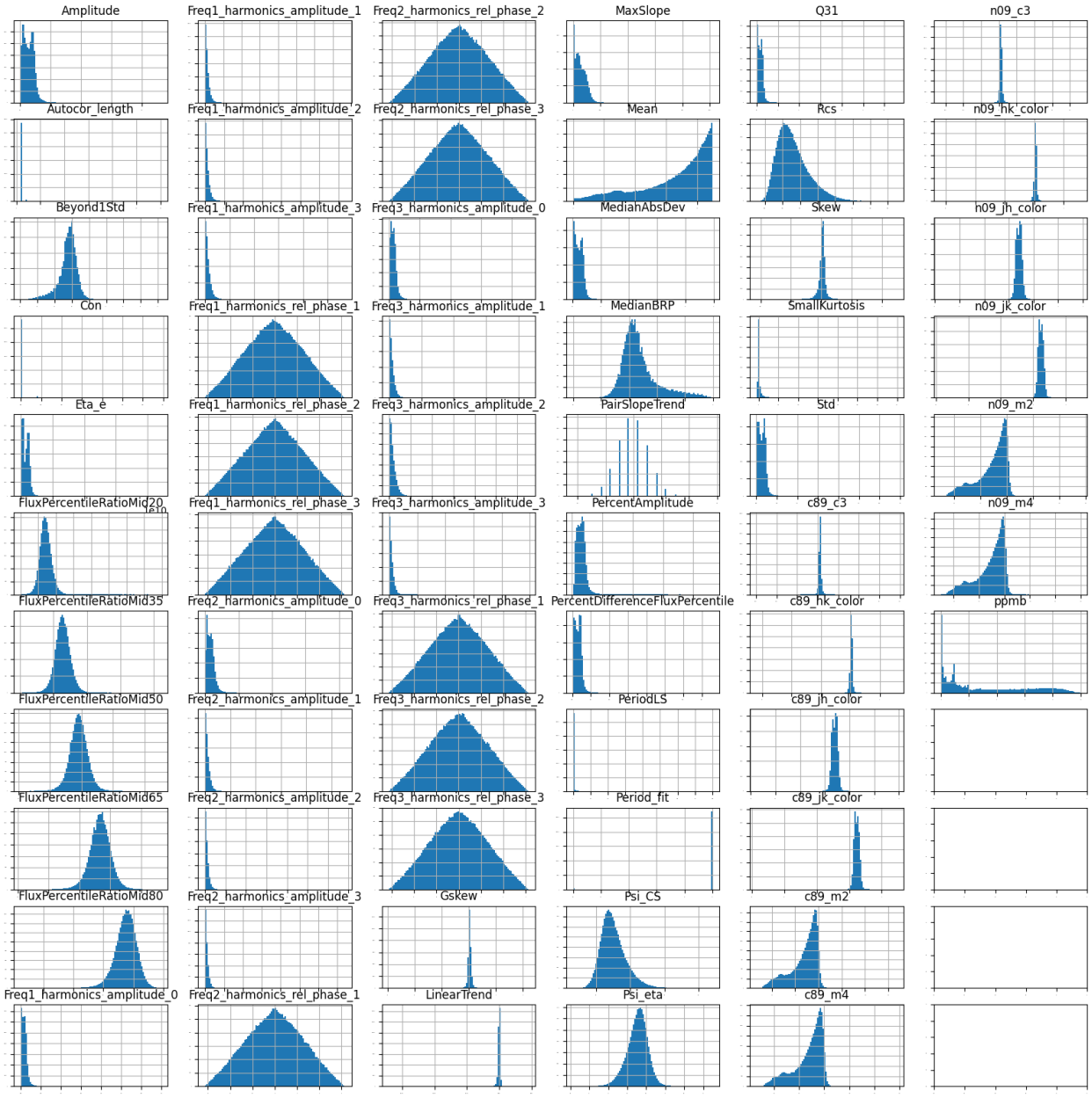


Figura B-1.: Histogramas de frecuencia para los atributos del tile b234

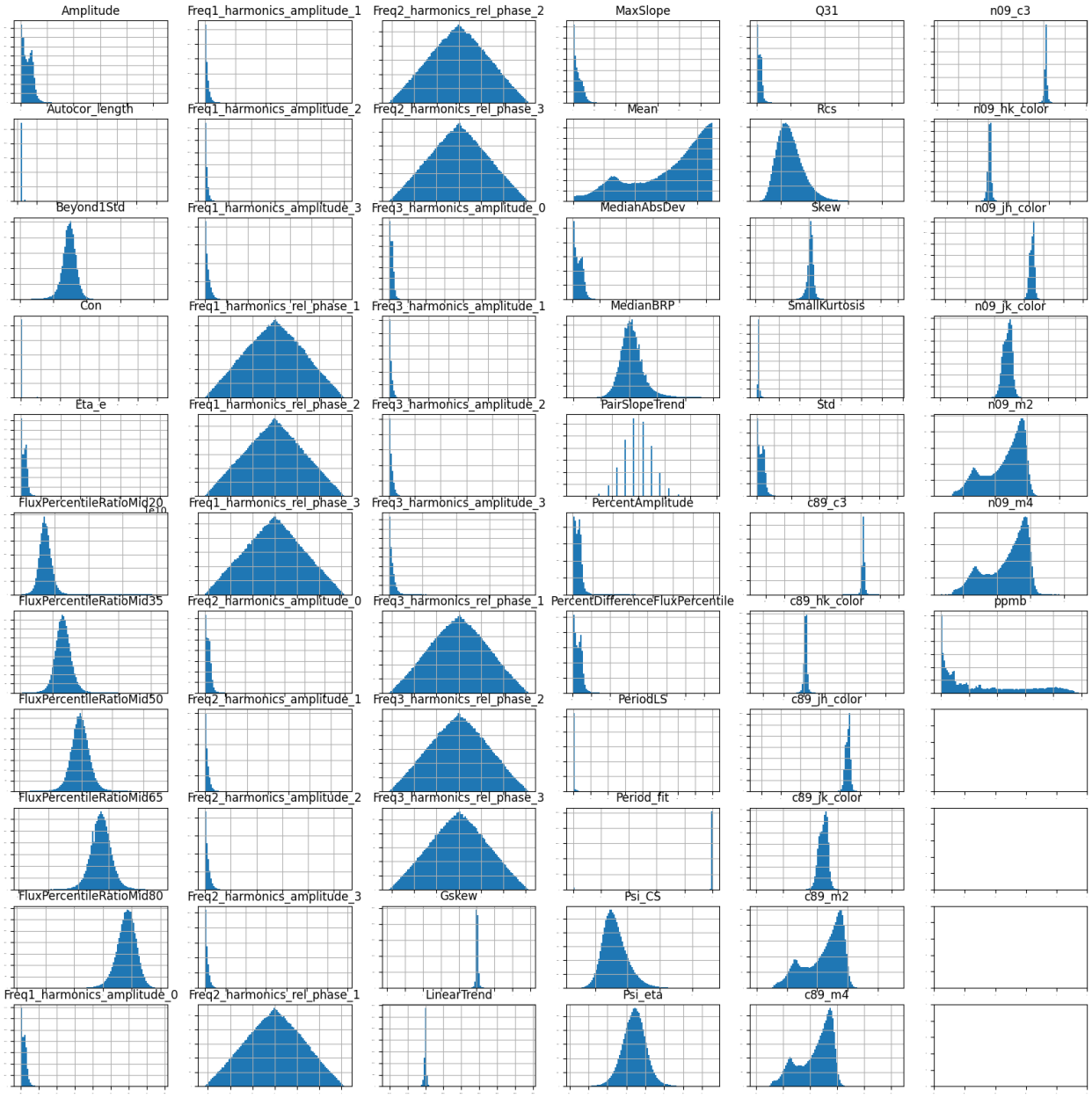


Figura B-2.: Histogramas de frecuencia para los atributos del tile b261

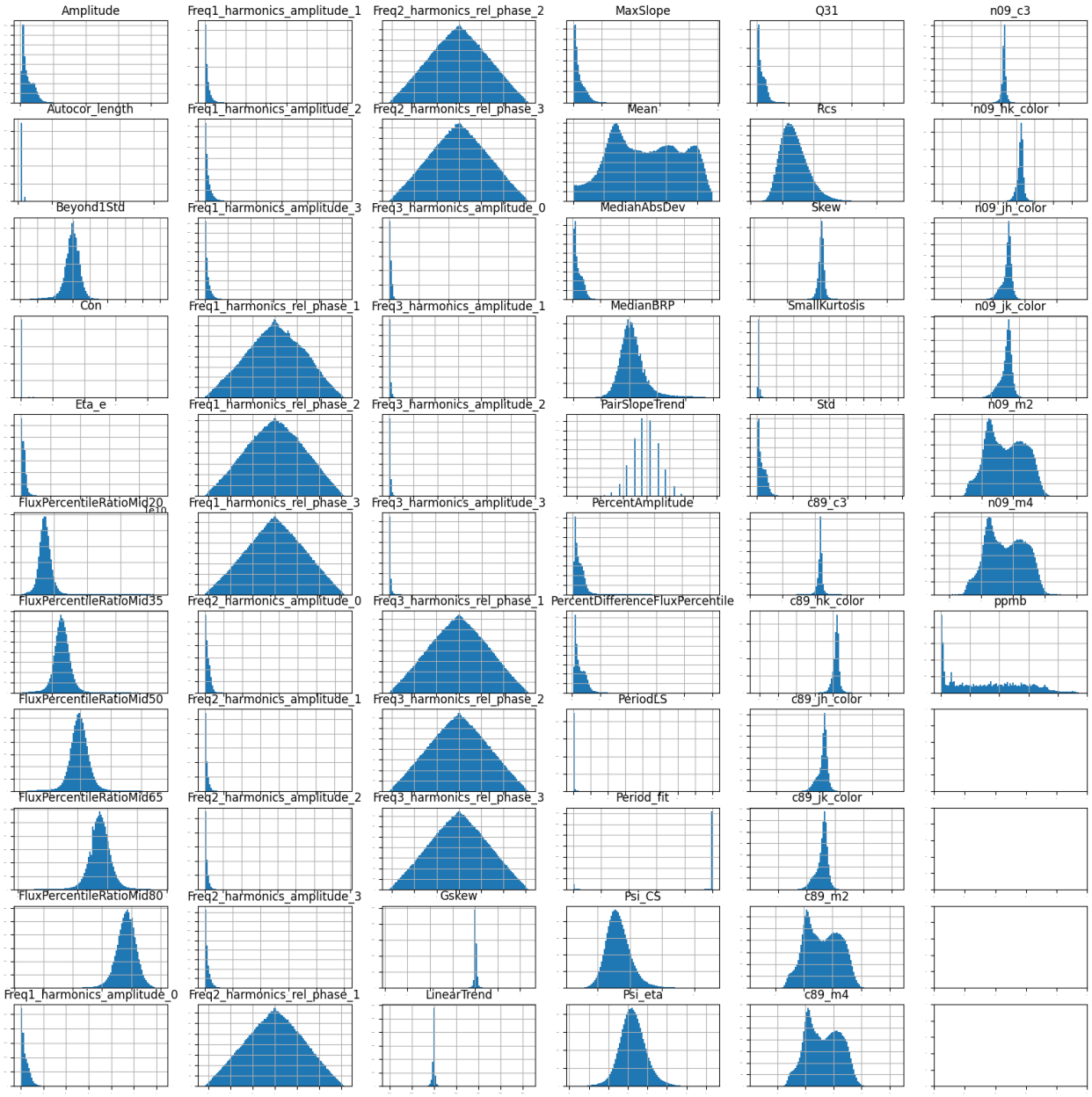


Figura B-3.: Histogramas de frecuencia para los atributos del tile b2360

C. Glosario de siglas

TP: Número de verdaderos positivos (True positives) en un conjunto de test. Ver 1.3.3
TN: Número de verdaderos negativos (True negatives) en un conjunto de test. Ver 1.3.3
FP: Número de falsos positivos (False positives) en un conjunto de test. Ver 1.3.3
FN: Número de falsos negativos (False negatives) en un conjunto de test. Ver 1.3.3
AUCPR: Área debajo de la curva de precision-recall. Ver 1.3.5.
R-AUCPR: Área debajo de la curva de precision-recall restringida a recall [0.35,1]. Ver 1.3.5.
ROC (Curva): (Del inglés, receiver operating characteristic). Dado un clasificador que retorna un puntaje en $[0, 1]$ para cada instancia a clasificar, la curva ROC se obtiene graficando la proporción de falsos positivos ($\frac{FP}{FP+TN}$) en el eje x y el recall en el eje y, para cada valor de corte t en $[0, 1]$. (análogamente a la curva de precision recall definida en la sección 1.3.5).
AUCROC: Area debajo de la curva ROC.

Bibliografía

[Akbani et al., 2004] Akbani, R., Kwek, S., and Japkowicz, N. (2004). Applying support vector machines to imbalanced data sets. volume 3201, pages 39–50.

[Alaiz and Japkowicz, 2008] Alaiz, R. and Japkowicz, N. (2008). Assessing the impact of changing environments on classifier performance. volume 5032, pages 13–24.

[Armstrong et al., 2015] Armstrong, D., Kirk, J., Lam, K., McCormac, J., Walker, S., Brown, D., Osborn, H., Pollacco, D., and Spake, J. (2015). K2 variable catalogue: Variable stars and eclipsing binaries in k2 campaigns 1 and 0. *Astronomy and Astrophysics*.

[Baade, 1946] Baade, W. (1946). A search for the nucleus of our galaxy. *Publications of the Astronomical Society of the Pacific*, 58:249.

[Bailey, 1902] Bailey, S. I. (1902). A discussion of variable stars in the cluster Omega Centauri. *Annals of the Astronomical Observatory of Harvard College*, 38.

[Ball and Brunner, 2010] Ball, N. and Brunner, R. (2010). Data mining and machine learning in astronomy. *International Journal of Modern Physics D*, 19(7):1049–1106.

[Batista et al., 2003] Batista, G. E. A. P. A., Bazzan, A., and Monard, M. C. (2003). Balancing training data for automated annotation of keywords: a case study. In *WOB*.

[Benjamin et al., 2003] Benjamin, R., Churchwell, E., Babler, B., Bania, T., Clemens, D., Cohen, M., Dickey, J., Indebetouw, R., Jackson, J., Kobulnicky, H., and et al. (2003). Glimpse. i. ansirtflegacy project to map the inner galaxy. *Publications of the Astronomical Society of the Pacific*, 115(810):953–964.

[Berrar, 2018] Berrar, D. (2018). *Cross-Validation*.

[Bickel et al., 2007] Bickel, S., Brückner, M., and Scheffer, T. (2007). Discriminative learning for differing training and test distributions. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, page 81–88, New York, NY, USA. Association for Computing Machinery.

[Boser et al., 1996] Boser, B., Guyon, I., and Vapnik, V. (1996). A training algorithm for optimal margin classifier. *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, 5.

[Breiman, 1996] Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24(2):123–140.

[Breiman, 2001] Breiman, L. (2001). Random forests. *Machine Learning*, 45:5–32.

[Brown et al., 2018] Brown, A. G. A., Vallenari, A., Prusti, T., de Bruijne, J. H. J., Babusiaux, C., Bailer-Jones, C. A. L., Biermann, M., Evans, D. W., Eyer, L., and et al. (2018). Gaia data release 2. *Astronomy and Astrophysics*, 616:A1.

[Buitinck et al., 2013] Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B., and Varoquaux, G. (2013). API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122.

[Cabral et al., 2020a] Cabral, J. B., Ramos, F., Gurovich, S., and Granitto, P. (2020a). Automatic Catalog of RRLyrae from $\sim$ 14 million VVV Light Curves: How far can we go with traditional machine-learning? *arXiv e-prints*, page arXiv:2005.00220.

[Cabral et al., 2020b] Cabral, J. B., Ramos, F., Gurovich, S., and Granitto, P. (2020b). Carpyncho: VVV Catalog browser toolkit.

[Cabral et al., 2018] Cabral, J. B., Sánchez, B., Ramos, F., Gurovich, S., Granitto, P., and Vanderplas, J. (2018). From fats to feets: Further improvements to an astronomical feature extraction tool based on machine learning.

[Cardelli et al., 1989] Cardelli, J., Clayton, G., and Mathis, J. (1989). The relationship between infrared, optical, and ultraviolet extinction. *The Astrophysical Journal*, 345.

[Carreira-perpinan, 1997] Carreira-perpinan, M. A. (1997). A review of dimension reduction techniques. Technical report.

[Catelan et al., 2014] Catelan, M., Dekany, I., Hempel, M., and Minniti, D. (2014). Stellar variability in the vvv survey: An update.

[Cejnek and Bukovsky, 2018] Cejnek, M. and Bukovsky, I. (2018). Concept drift robust adaptive novelty detection for data streams. *Neurocomputing*, 309.

[Chawla et al., 2002] Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357.

[Chen et al., 2016] Chen, X., Monfort, M., Liu, A., and Ziebart, B. D. (2016). Robust covariate shift regression. In Gretton, A. and Robert, C. C., editors, *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pages 1270–1279, Cadiz, Spain. PMLR.

[Cochran, 1952] Cochran, W. G. (1952). The chi-squared test of goodness of fit. *Annals of Mathematical Statistics*, 23(3):315–345.

[Cortes and Vapnik, 1995] Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Mach. Learn.*, 20(3):273–297.

[Dash and Liu, 1997] Dash, M. and Liu, H. (1997). Feature selection for classification. *Intelligent Data Analysis*, 1(1):131–156.

[Debosscher et al., 2007] Debosscher, J., Sarro, L., Aerts, C., Cuypers, J., Vandenbussche, B., Garrido, R., and Solano, E. (2007). Automated supervised classification of variable stars i. methodology. *Astronomy and Astrophysics*, 475.

[Dubath et al., 2011] Dubath, P., Rimoldini, L., Süveges, M., Blomme, J., López, M., Sarro, L., De Ridder, J., Cuypers, J., Guy, L., Lecoeur, I., Nienartowicz, K., Jan, A., Beck, M., Mowlavi, N., Cat, P., Lebzelter, T., and Eyer, L. (2011). Random forest automated supervised classification of hipparcos periodic variable stars. *Monthly Notices of The Royal Astronomical Society - MON NOTIC ROY ASTRON SOC*, 414.

[Elorrieta López et al., 2016] Elorrieta López, F., Eyheramendy, S., Jordán, A., and Dekany, I. (2016). A machine learned classifier for rr lyrae in the vvv survey. *Astronomy & Astrophysics*, 595.

[Emerson et al., 2004] Emerson, J., Irwin, M., Lewis, J., Hodgkin, S., Evans, D., Bunclark, P., Mcmahon, R., Hambly, N., Mann, R., Bond, I., Sutorius, E., Read, M., Williams, P., Lawrence, A., and Stewart, M. (2004). Vista data flow system: overview. *Proceedings of SPIE - The International Society for Optical Engineering*, 5493.

[Fan et al., 2008] Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. (2008). Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9.

[Fisher et al., 2018] Fisher, A., Rudin, C., and Dominici, F. (2018). Model class reliance: Variable importance measures for any machine learning model class, from the rashomon"perspective.

[Freund and Schapire, 1997] Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *JournalSamus, N. N., Durlevich, O. V., et al of Computer and System Sciences*, 55(1):119 – 139.

[Friedman, 2000] Friedman, J. H. (2000). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232.

[Friedman, 2002] Friedman, J. H. (2002). Stochastic gradient boosting. *Comput. Stat. Data Anal.*, 38(4):367–378.

[GeetaDharani. et al., 2019] GeetaDharani., Y., Nair, N. G., Satpathy, P., and Christopher, J. (2019). Covariate shift: A review and analysis on classifiers. *2019 Global Conference for Advancement in Technology (GCAT)*, pages 1–6.

[Ghojogh et al., 2019] Ghojogh, B., Samad, M., Mashhadi, S., Kapoor, T., Ali, W., Karray, F., and Crowley, M. (2019). Feature selection and feature extraction in pattern analysis: A literature review.

[Gran et al., 2015] Gran, F., Minniti, D., Saito, R. K., Navarrete, C., Dékány, I., McDonald, I., Contreras Ramos, R., and Catelan, M. (2015). Bulge rr lyrae stars in the vvv tile b201. *Astronomy and Astrophysics*, 575:A114.

[Gran et al., 2016] Gran, F., Minniti, D., Saito, R. K., Zoccali, M., Gonzalez, O. A., Navarrete, C., Catelan, M., Contreras Ramos, R., Elorrieta, F., Eyheramendy, S., and et al. (2016). Mapping the outer bulge with rrab stars from the vvv survey. *Astronomy and Astrophysics*, 591:A145.

[Gray et al., 2007] Gray, J., Szalay, A., Budavari, T., Lupton, R., Nieto-Santisteban, M., and Thakar, A. (2007). Cross-matching multiple spatial observations and dealing with missing data. *CoRR*, abs/cs/0701172.

[Guyon and Elisseeff, 2003] Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3(null):1157–1182.

[Han et al., 2012] Han, J., Kamber, M., and Pei, J. (2012). *Data mining concepts and techniques, third edition*. Morgan Kaufmann Publishers, Waltham, Mass.

[Hastie et al., 2001] Hastie, T., Tibshirani, R., and Friedman, J. (2001). *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA.

[He et al., 2008] He, H., Bai, Y., Garcia, E., and Li, S. (2008). Adasyn: Adaptive synthetic sampling approach for imbalanced learning. pages 1322 – 1328.

[He and Garcia, 2009] He, H. and Garcia, E. (2009). Learning from imbalanced data. *Knowledge and Data Engineering, IEEE Transactions on*, 21:1263 – 1284.

[Hsu et al., 2003] Hsu, C.-w., Chang, C.-c., and Lin, C.-J. (2003). A practical guide to support vector classification chih-wei hsu, chih-chung chang, and chih-jen lin.

[Jain and Dubes, 1988] Jain, A. K. and Dubes, R. C. (1988). *Algorithms for clustering data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

[Japkowicz, 2000] Japkowicz, N. (2000). The class imbalance problem: Significance and strategies. *Proceedings of the 2000 International Conference on Artificial Intelligence ICAI*.

[Jovic et al., 2015] Jovic, A., Brkić, K., and Bogunovic, N. (2015). A review of feature selection methods with applications. pages 1200–1205.

[Khamis, 2008] Khamis, H. (2008). Measures of association: How to choose? *Journal of Diagnostic Medical Sonography*, 24(3):155–162.

[Kouw and Loog, 2019] Kouw, W. M. and Loog, M. (2019). An introduction to domain adaptation and transfer learning.

[Kraskov et al., 2004] Kraskov, A., Stogbauer, H., and Grassberger, P. (2004). Estimating mutual information. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 69:066138.

[Krzysztofowicz, 1997] Krzysztofowicz, R. (1997). Transformation and normalization of variates with specified distributions. *Journal of Hydrology*, 197(1):286–292.

[Kuhn and Johnson, 2013] Kuhn, M. and Johnson, K. (2013). Applied predictive modeling.

[Kuperman et al., 2016] Kuperman, V., Matsuki, K., and Van Dyke, J. (2016). The random forests statistical technique: An examination of its value for the study of reading. *Scientific Studies of Reading*, 20.

[Liu and Ziebart, 2017] Liu, A. and Ziebart, B. D. (2017). Robust covariate shift prediction with general losses and feature views.

[Liu et al., 2010] Liu, H., Motoda, H., Setiono, R., and Zhao, Z. (2010). Feature selection: An ever evolving frontier in data mining. *Journal of Machine Learning Research - Proceedings Track*, 10:4–13.

[Lomb, 1976] Lomb, N. R. (1976). Least - squares frequency analysis of unequally spaced data. *Astrophys. Space Sci.*, 39:447–462.

[Lopez-Martinez, 2017] Lopez-Martinez, D. (2017). Regularization approaches for support vector machines with applications to biomedical data.

[Louppe, 2015] Louppe, G. (2015). Understanding random forests: From theory to practice.

[López et al., 2014] López, V., Fernández, A., and Herrera, F. (2014). On the importance of the validation technique for classification with imbalanced datasets: Addressing covariate shift when data is skewed. *Information Sciences*, 257:1–13.

[Manning et al., 2008] Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press, USA.

[McWilliam, 2011] McWilliam, A. (2011). Rr lyrae stars, metal-poor stars, and the galaxy.

[Menardi and Torelli, 2012] Menardi, G. and Torelli, N. (2012). Training and assessing classification rules with imbalanced data. *Data Mining and Knowledge Discovery*, 28:92–122.

[Meyer, 1970] Meyer, P. (1970). *Introductory Probability and Statistical Applications*. Addison-Wesley series in statistics. Addison-Wesley Publishing Company.

[Minniti et al., 2010] Minniti, D., Lucas, P., Emerson, J., Saito, R., Hempel, M., Pietrukowicz, P., Ahumada, A., Alonso, M., Alonso-García, J., Arias, J., Bandyopadhyay, R., Barbá, R., Barbuy, B., Bedin, L., Bica, E., Borissova, J., Bronfman, L., Carraro, G., Catelan, M., and Zoccali, M. (2010). Vista variables in the via lactea (vvv): The public eso near-ir variability survey of the milky way. *New Astronomy*, 15:433.

[Mitchell, 1997] Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill, New York.

[Molnar, 2019] Molnar, C. (2019). *Interpretable Machine Learning*. `https://christophm.github.io/interpretable-ml-book/`.

[Moore and McCabe, 1989] Moore, D. S. and McCabe, G. P. (c1989.). *Introduction to the practice of statistics /*. W.H. Freeman,, New York :.

[Moreno-Torres et al., 2012] Moreno-Torres, J. G., Raeder, T., Alaiz-Rodríguez, R., Chawla, N. V., and Herrera, F. (2012). A unifying view on dataset shift in classification. *Pattern Recognition*, 45(1):521–530.

[Márcio Catelan, 2015] Márcio Catelan, H. A. S. (2015). *Pulsating Stars*. Wiley-VCH.

[Nishiyama et al., 2009] Nishiyama, S., Tamura, M., Hatano, H., Kato, D., Tanabé, T., Sugitani, K., and Nagata, T. (2009). Interstellar extinction law toward the galactic center iii: J, h, ks bands in the 2mass and the mko systems, and 3.6, 4.5, 5.8, 8.0 micron in the spitzer/irac system. *Astrophysical Journal - ASTROPHYS J*, 696:1407–1417.

[Ochsenbein, F. et al., 2000] Ochsenbein, F., Bauer, P., and Marcout, J. (2000). The vizier database of astronomical catalogues. *Astron. Astrophys. Suppl. Ser.*, 143(1):23–32.

[Paegert et al., 2014] Paegert, M., Stassun, K., and Burger, D. (2014). The eb factory project i. a fast, neural net based, general purpose light curve classifier optimized for eclipsing binaries. *The Astronomical Journal*, 148:31.

[Pagano, 2010] Pagano, R. (2010). *Understanding Statistics in the Behavioral Sciences*. Wadsworth, Cengage Learning.

[Pedregosa et al., 2011] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.

[Pomerleau, 1989] Pomerleau, D. (1989). Alvinn: An autonomous land vehicle in a neural network. In Touretzky, D., editor, *Proceedings of Advances in Neural Information Processing Systems 1*, pages 305 –313. Morgan Kaufmann.

[Puth et al., 2015] Puth, M.-T., Neuhäuser, M., and Ruxton, G. D. (2015). Effective use of spearman's and kendall's correlation coefficients for association between two measured traits. *Animal Behaviour*, 102:77–84.

[Quinlan, 1986] Quinlan, J. R. (1986). Induction of decision trees. *Mach. Learn.*, 1(1):81–106.

[Quinlan, 1993] Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[Quinonero-Candela et al., 2009] Quinonero-Candela, J., Sugiyama, M., Lawrence, N., and Schwaighofer, A. (2009). *Dataset Shift in Machine Learning*. Neural information processing series. MIT Press.

[Rabanser et al., 2019] Rabanser, S., Günnemann, S., and Lipton, Z. C. (2019). Failing loudly: An empirical study of methods for detecting dataset shift.

[Rahimi and Recht, 2008] Rahimi, A. and Recht, B. (2008). Random features for large-scale kernel machines. In Platt, J., Koller, D., Singer, Y., and Roweis, S., editors, *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc.

[Raileanu and Stoffel, 2004] Raileanu, L. and Stoffel, K. (2004). Theoretical comparison between the gini index and information gain criteria. *Annals of Mathematics and Artificial Intelligence*, 41:77–93.

[Rakotomamonjy, 2003] Rakotomamonjy, A. (2003). Variable selection using svm based criteria. *J. Mach. Learn. Res.*, 3(null):1357–1370.

[Raza et al., 2016] Raza, H., Cecotti, H., Li, Y., and Prasad, G. (2016). Adaptive learning with covariate shift-detection for motor imagery based brain-computer interface. *Soft Computing*, 20.

[Raza et al., 2014] Raza, H., Prasad, G., and Li, Y. (2014). Adaptive learning with covariate shift-detection for non-stationary environments. pages 1–8.

[Richards et al., 2011] Richards, J. W., Starr, D. L., Butler, N. R., Bloom, J. S., Brewer, J. M., Crellin-Quick, A., Higgins, J., Kennedy, R., and Rischard, M. (2011). On machine-learned classification of variable starts with sparse and noisy time-series data. *The Astrophysical Journal*, 733(1):10.

[Rokach and Maimon, 2005] Rokach, L. and Maimon, O. (2005). *Clustering Methods*, pages 321–352.

[Russell and Norvig, 2009] Russell, S. and Norvig, P. (2009). *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, USA, 3rd edition.

[Samus et al., 2009] Samus, N. N., Kazarovets, E. V., Durlevich, O. V., Kireeva, N. N., and Pastukhova, E. N. (2009). VizieR Online Data Catalog: General Catalogue of Variable Stars (Samus+, 2007-2017). *VizieR Online Data Catalog*, page B/gcvs.

[Scargle, 1983] Scargle, J. (1983). Studies in astronomical time series analysis. ii - statistical aspects of spectral analysis of unevenly spaced data. *The Astrophysical Journal*, 263.

[Schwarzchild, 1938] Schwarzchild, M. (1938). On the Light Curve of Delta Cephei. *Harvard College Observatory Circular*, 431:1–13.

[Shapley, 1918] Shapley, H. (1918). Studies based on the colors and magnitudes in stellar clusters. vii. the distances, distribution in space, and dimensions of 69 globular clusters. 48:154–181.

[Skrutskie et al., 2006] Skrutskie, M. F., Cutri, R. M., Stiening, R., Weinberg, M. D., Schneider, S., Carpenter, J. M., Beichman, C., Capps, R., Chester, T., Elias, J., Huchra, J., Liebert, J., Lonsdale, C., Monet, D. G., Price, S., Seitzer, P., Jarrett, T., Kirkpatrick, J. D., Gizis, J. E., Howard, E., Evans, T., Fowler, J., Fullmer, L., Hurt, R., Light, R., Kopan, E. L., Marsh, K. A., McCallon, H. L., Tam, R., Dyk, S. V., and Wheelock, S. (2006). The two micron all sky survey (2mass). *The Astronomical Journal*, 131(2):1163–1183.

[Smith, 2004] Smith, H. A. (2004). *RR Lyrae Stars*.

[Soszyński et al., 2019] Soszyński, I., Udalski, A., Wrona, M., Szymański, M., Pietrukowicz, P., Skowron, J., Skowron, D., Poleski, R., Kozlowski, S., Mróz, P., Ulaczyk, K., Rybicki, K., Iwanek, P., and Gromadzki, M. (2019). Over 78 000 rr lyrae stars in the galactic bulge and disk from the ogle survey.

[Sugiyama and Kawanabe, 2012] Sugiyama, M. and Kawanabe, M. (2012). *Machine Learning in Non-Stationary Environments: Introduction to Covariate Shift Adaptation*. The MIT Press.

[Sugiyama et al., 2007] Sugiyama, M., Krauledat, M., and Müller, K.-R. (2007). Covariate shift adaptation by importance weighted cross validation. *J. Mach. Learn. Res.*, 8:985–1005.

[Tesauro, 1995] Tesauro, G. (1995). *TD-Gammon: A Self-Teaching Backgammon Program*, pages 267–285. Springer US, Boston, MA.

[Udalski, 2004] Udalski, A. (2004). The optical gravitational lensing experiment. real time data analysis systems in the ogle-iii survey. *Acta Astronomica - ACTA ASTRONOM*, 53.

[Udalski et al., 2015] Udalski, A., Szymański, M., and Szymański, G. (2015). Ogle-iv: Fourth phase of the optical gravitational lensing experiment. *Acta Astronomica*, 65.

[VanderPlas, 2018] VanderPlas, J. T. (2018). Understanding the lomb–scargle periodogram. *The Astrophysical Journal Supplement Series*, 236(1):16.

[Vapnik and Chervonenkis, 1974] Vapnik, V. and Chervonenkis, A. (1974). *Theory of Pattern Recognition*. Nauka, Moscow. (German Translation: W. Wapnik & A. Tscherwonenkis, *Theorie der Zeichenerkennung*, Akademie–Verlag, Berlin, 1979).

[Vapnik and Chervonenkis, 1971] Vapnik, V. N. and Chervonenkis, A. Y. (1971). On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264–280.

[Wang and Rudin, 2018] Wang, F. and Rudin, C. (2018). Extreme dimension reduction for handling covariate shift.

[Ward, 1963] Ward, J. H. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58(301):236–244.

[Weir et al., 1995] Weir, N., Fayyad, U., and Djorgovski, G. (1995). Automated star/galaxy classification for digitized poss-ii. *The Astronomical Journal*, 109:2401.

[Widmer, 1998] Widmer, G. (1998). *Special Issue on Context Sensitivity and Concept Drift*. Machine learning. Kluwer.

[Widmer and Kubat, 1994] Widmer, G. and Kubat, M. (1994). Learning in the presence of concept drift and hidden contexts. *Machine Learning*, 23.

[Williams and Seeger, 2001] Williams, C. and Seeger, M. (2001). Using the nyström method to speed up kernel machines. In Leen, T., Dietterich, T., and Tresp, V., editors, *Advances in Neural Information The relationship between infrared, optical, and ultraviolet extinction.Processing Systems*, volume 13, pages 682–688. MIT Press.

[Wu and Chang, 2003] Wu, G. and Chang, E. Y. (2003). Class-boundary alignment for imbalanced dataset learning. In *In ICML 2003 Workshop on Learning from Imbalanced Data Sets*, pages 49–56.

[Wyner et al., 2015] Wyner, A., Olson, M., Bleich, J., and Mease, D. (2015). Explaining the success of adaboost and random forests as interpolating classifiers. *Journal of Machine Learning Research*, 18.

[Yang et al., 2012] Yang, T., Li, Y.-f., Mahdavi, M., Jin, R., and Zhou, Z.-H. (2012). Nyström method vs random fourier features: A theoretical and empirical comparison. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc.

[Yeo and Johnson, 2000] Yeo, I. and Johnson, R. A. (2000). A new family of power transformations to improve normality or symmetry. *Biometrika*, 87(4):954–959.

[Yu et al., 2015] Yu, X., Yu, M., Xu, L.-x., Yang, J., and Xie, Z.-q. (2015). Training classifiers under covariate shift by constructing the maximum consistent distribution subset. *Mathematical Problems in Engineering*, 2015:1–9.

[Zadrozny, 2004] Zadrozny, B. (2004). Learning and evaluating classifiers under sample selection bias. In *Proceedings of the Twenty-First International Conference on Machine Learning*, ICML '04, page 114, New York, NY, USA. Association for Computing Machinery.