Tema 3: Validación de modelos y selección de características

En el campo del aprendizaje automático y la minería de datos, la capacidad para construir modelos predictivos precisos y generalizables es esencial para una amplia gama de aplicaciones, desde la medicina hasta las finanzas y la ingeniería. Sin embargo, este proceso está plagado de desafíos, especialmente cuando se trata de garantizar que los modelos sean capaces de realizar predicciones precisas sobre datos no vistos, es decir, que generalicen bien. Dos de los problemas más recurrentes que enfrentan los practicantes del aprendizaje automático son el sobreajuste, donde el modelo se ajusta demasiado a los datos de entrenamiento y pierde capacidad de generalización, y el subajuste, donde el modelo es demasiado simplista para capturar la complejidad de los datos subyacentes.

En este contexto, la validación de modelos y la selección de características emergen como componentes críticos del proceso de desarrollo de modelos. En este tema, nos sumergiremos en la comprensión y aplicación de métodos avanzados de validación cruzada y evaluación de modelos, así como en las técnicas de selección de características y reducción de la dimensionalidad. Estos procedimientos no solo nos ayudan a mejorar la precisión y robustez de nuestros modelos, sino que también nos permiten entender mejor la naturaleza de los datos y las relaciones entre las características, lo que es crucial para la interpretación de los resultados.

A lo largo de este tema, exploraremos en detalle cómo los diferentes enfoques de validación cruzada, como la validación cruzada k-fold y la validación cruzada leave-one-out, nos permiten estimar de manera más precisa el rendimiento de nuestros modelos en datos no vistos. También examinaremos técnicas de evaluación de modelos, que nos brindan información valiosa sobre cómo el rendimiento de nuestros modelos varía con respecto al tamaño del conjunto de datos y otros parámetros clave.

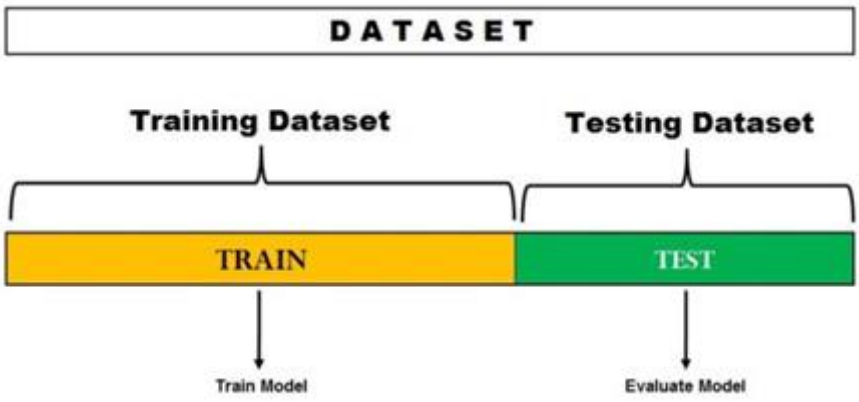
Además, abordaremos la selección de características y la reducción de la dimensionalidad, donde exploraremos métodos como el filtrado de características, y el empaquetado de características entre otros. Estas técnicas nos ayudan a identificar qué características son relevantes para la tarea en cuestión, eliminando la información redundante o ruidosa que puede afectar negativamente el rendimiento del modelo.

Es importante destacar que la validación de modelos desempeña un papel fundamental en la prevención tanto del sobreajuste como del subajuste, asegurando que nuestros modelos sean capaces de generalizar de manera efectiva a datos nuevos y desconocidos. Al comprender y aplicar adecuadamente estas técnicas, podemos desarrollar modelos de aprendizaje automático más robustos, confiables y útiles en una variedad de aplicaciones del mundo real. Por lo tanto, este tema ofrece una oportunidad invaluable para profundizar en la mejora continua de nuestras habilidades en el desarrollo y validación de modelos de aprendizaje automático.

Métodos de validación cruzada y evaluación de modelos

En el proceso de desarrollo de modelos de aprendizaje automático, es fundamental dividir nuestro conjunto de datos en conjuntos separados para entrenamiento, prueba y, en algunos casos, validación. Esta división nos permite evaluar la capacidad de generalización de nuestros modelos y garantizar su eficacia en datos no vistos. Sin embargo, en muchos casos, los conjuntos de datos no están previamente divididos, lo que nos lleva a la necesidad de realizar esta separación de manera adecuada.

Imaginemos que contamos con un archivo que contiene 10,000 registros de datos. Para entrenar y evaluar nuestro modelo de machine learning de manera efectiva, lo primero que debemos hacer es dividir este conjunto de datos inicial en dos partes principales:



- Conjunto de entrenamiento (train): Este conjunto se utilizará para entrenar el modelo, es decir, para ajustar los parámetros del algoritmo de aprendizaje a partir de los datos proporcionados. Por lo general, este conjunto representa el 80% de los datos originales.

- Conjunto de prueba (test): Este conjunto se utilizará para evaluar el rendimiento del modelo entrenado en datos no vistos. Representa el 20% restante de los datos originales.

Para realizar esta división de manera adecuada, es importante que los datos se mezclen de forma aleatoria y no se tomen en secuencia. Esto garantiza que nuestros conjuntos de entrenamiento y prueba sean representativos de la distribución general de los datos.

Por ejemplo, si deseamos realizar una clasificación utilizando un algoritmo supervisado, dividiremos nuestros datos de la siguiente manera:

- X train: Contendrá 8000 registros para el entrenamiento del modelo.
- y train: Contendrá las etiquetas correspondientes a los resultados esperados en X train.
- X test: Contendrá 2000 registros para la evaluación del modelo.
- y test: Contendrá las etiquetas correspondientes a los resultados esperados en X test.

El primer paso es realizar esta división utilizando el método train_test_split, que nos permite dividir fácilmente un conjunto de datos en dos conjuntos aleatorios con un tamaño determinado. Al llamar a esta función, especificamos las variables X e y que representan nuestros datos y etiquetas, y obtenemos como resultados conjuntos de entrenamiento y prueba para ambas. Es importante mencionar que, por defecto, el 75% de los registros se incluyen en el conjunto de entrenamiento y el 25% restante en el conjunto de prueba.

Este proceso de división también ofrece opciones adicionales, como la especificación del tamaño del conjunto de prueba, la semilla aleatoria para la reproducibilidad de los resultados y la estratificación de los datos.

Una vez que hemos separado nuestros datos en conjuntos de entrenamiento y prueba, procedemos a entrenar nuestro modelo utilizando únicamente los datos de entrenamiento mediante el método fit (X train, y train). Luego, evaluamos el rendimiento del modelo utilizando los datos de prueba mediante el método predict (X test), donde el modelo realiza predicciones sobre los datos de prueba sin incorporar nuevos conocimientos.

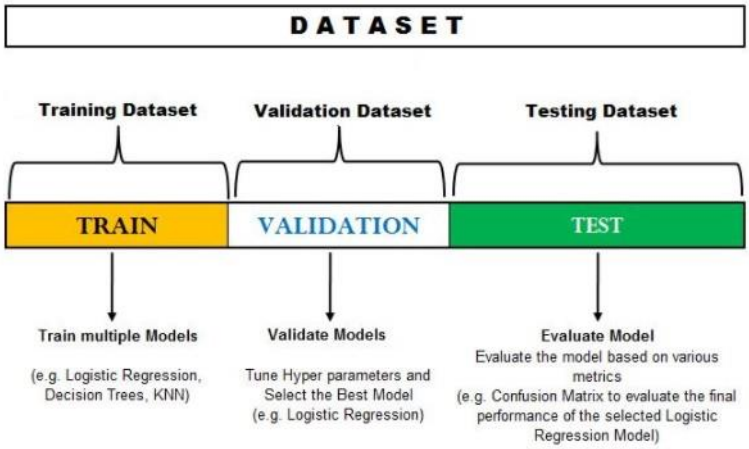
La comparación de las predicciones del modelo con los valores reales nos permite calcular métricas de evaluación, como el accuracy. Si el rendimiento del modelo en el conjunto de prueba se aproxima al rendimiento en el conjunto de entrenamiento, indicando una buena generalización, consideramos que el modelo es válido. Sin embargo, discrepancias significativas entre el rendimiento en entrenamiento y prueba pueden indicar problemas como el sobreajuste, que requieren una evaluación más detallada, posiblemente utilizando un conjunto de validación adicional.

Si observamos una discrepancia significativa entre las métricas obtenidas en los conjuntos de entrenamiento y prueba, es un claro indicador de que nuestro modelo no es satisfactorio. En este punto, es crucial emprender acciones para mejorar su desempeño.

Podemos optar por ajustar los parámetros del modelo, aumentar la cantidad de datos de entrenamiento, revisar y mejorar el preprocesamiento de datos, realizar una limpieza más exhaustiva, abordar el desbalanceo de clases, así como explorar la selección o generación de características adicionales. Incluso, si concluimos que el modelo elegido inicialmente no es adecuado, podemos considerar la exploración de otros modelos como árboles de decisión, redes neuronales o ensamblados.

La técnica de validación cruzada surge como una herramienta valiosa en este proceso, ya que nos permite evaluar el rendimiento de nuestros modelos de manera más robusta y encontrar rápidamente la mejor configuración. Es importante destacar que, antes de iniciar el proceso de validación cruzada, contamos con dos conjuntos de datos claves: el conjunto de entrenamiento y el conjunto de pruebas.

El conjunto de validación no constituye un tercer conjunto separado, sino que se encuentra dentro del conjunto de entrenamiento. Durante las iteraciones de entrenamiento y evaluación con el conjunto de entrenamiento, utilizaremos este conjunto de validación para ajustar los parámetros del modelo y evaluar su desempeño en cada paso.



Una de las técnicas de validación cruzada más comunes es K-Folds, donde el conjunto de entrenamiento se divide en K subconjuntos o "pliegues".

Cada pliegue se utiliza como conjunto de validación una vez, mientras que los restantes se utilizan para entrenar el modelo. Este proceso se repite K veces, alternando los pliegues de validación y de entrenamiento en cada iteración, lo que nos proporciona K estimaciones del rendimiento del modelo.

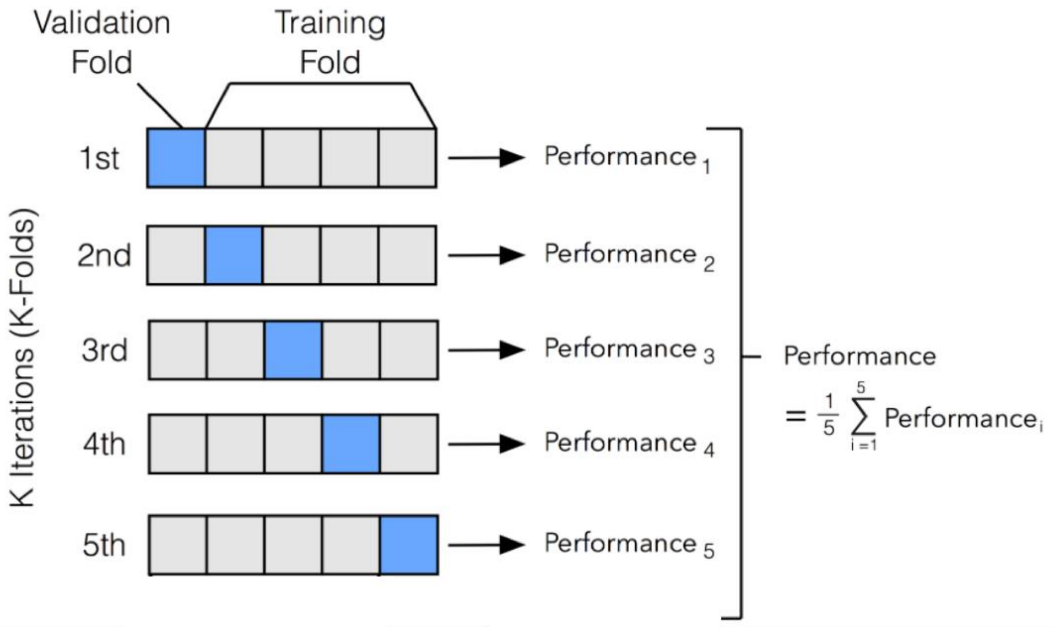
Otras técnicas de validación cruzada incluyen Stratified K-Folds, Leave P Out y ShuffleSplit, cada una con sus propias particularidades y aplicaciones específicas. En este contexto, profundizaremos en la técnica K-Folds debido a su amplia utilización y comprensión generalizada, y revisaremos el resto de las técnicas con una menor profundidad

Validación cruzada K-Folds

La validación cruzada K-Folds es una técnica utilizada en el aprendizaje automático para evaluar el rendimiento de un modelo de manera más robusta y precisa. Su objetivo principal es maximizar la utilización de los datos disponibles para entrenamiento y prueba, al tiempo que se reduce el sesgo en la evaluación del modelo.

La idea fundamental detrás de la validación cruzada K-Folds es dividir el conjunto de datos en K grupos (o "pliegues") de aproximadamente el mismo tamaño. Luego, se entrena y evalúa el modelo K veces, utilizando cada grupo como conjunto de prueba una vez y los restantes como conjunto de entrenamiento en cada iteración.

Por ejemplo, si utilizamos K = 5, dividimos nuestros datos en 5 grupos. En la primera iteración, el primer grupo se utiliza como conjunto de prueba y los otros cuatro como conjunto de entrenamiento. En la segunda iteración, el segundo grupo se utiliza como conjunto de prueba y los demás como conjunto de entrenamiento, y así sucesivamente hasta que cada grupo haya sido utilizado una vez como conjunto de prueba.



Al finalizar las K iteraciones, se obtiene una estimación del rendimiento del modelo al promediar los resultados de las K evaluaciones. Esto proporciona una medida más confiable del rendimiento del modelo, ya que se ha evaluado en diferentes conjuntos de datos y se han utilizado todos los datos disponibles tanto para entrenamiento como para prueba.

La validación cruzada K-Folds es especialmente útil cuando se dispone de un conjunto de datos limitado y se desea maximizar su utilización, así como cuando se quiere evitar la dependencia de una única división de los datos que puede llevar a conclusiones sesgadas sobre el rendimiento del modelo. Además, ayuda a identificar si el modelo está sobreajustado o subajustado al conjunto de datos, ya que evalúa su rendimiento en diferentes subconjuntos de datos.

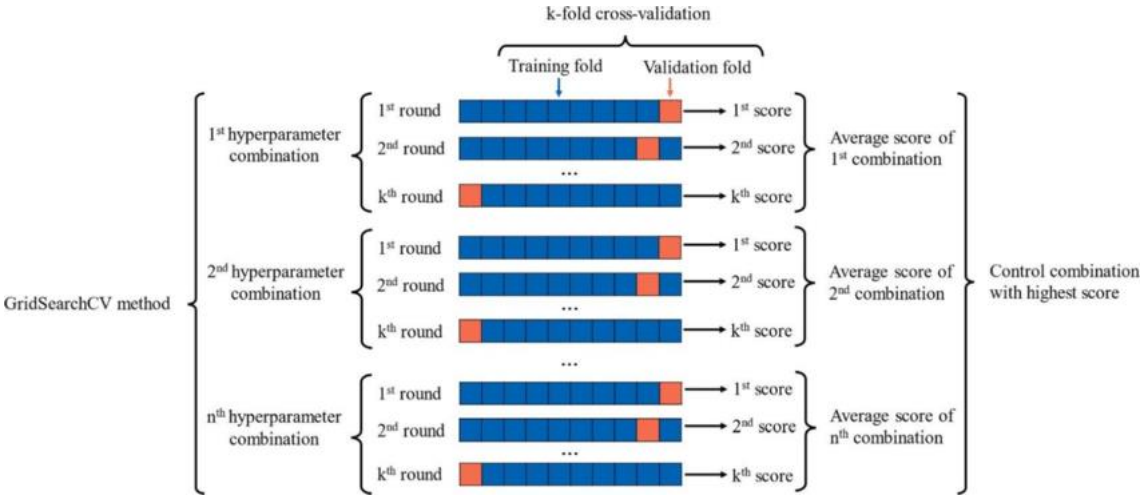
Revisa el siguiente notebook para tener una noción completa de como implementar CV utilizando Python y Scikit Learn: https://colab.research.google.com/drive/1VQoSKrlsz4F_qajp-mQ65vK2mh7d-4tA?usp=sharing

La validación cruzada no solo se utiliza para estimar la calidad de un modelo ya construido, sino que también es una herramienta valiosa para encontrar los mejores hiperparámetros para un algoritmo de aprendizaje automático.

Los hiperparámetros son configuraciones ajustables que no se aprenden directamente del conjunto de datos durante el entrenamiento del modelo, como la tasa de aprendizaje en un algoritmo de redes neuronales o el número de vecinos en un algoritmo de vecinos más cercanos.

La validación cruzada también puede utilizarse para estimar los hiperparámetros de nuestros modelos, para ello en primer lugar debemos elegir los hiperparámetros que se desea ajustar. Por ejemplo, si estamos utilizando un algoritmo de máquinas de vectores de soporte (SVM), podríamos querer ajustar el valor del parámetro de regularización C y el tipo de kernel.

Una vez tenemos esto claro, debemos definir un espacio de búsqueda para cada hiperparámetro. Este espacio puede ser discreto o continuo, dependiendo del tipo de hiperparámetro. Por ejemplo, podríamos querer probar valores de C en el rango [0.1, 1, 10, 100] y diferentes tipos de kernel como lineal, polinómico y radial.



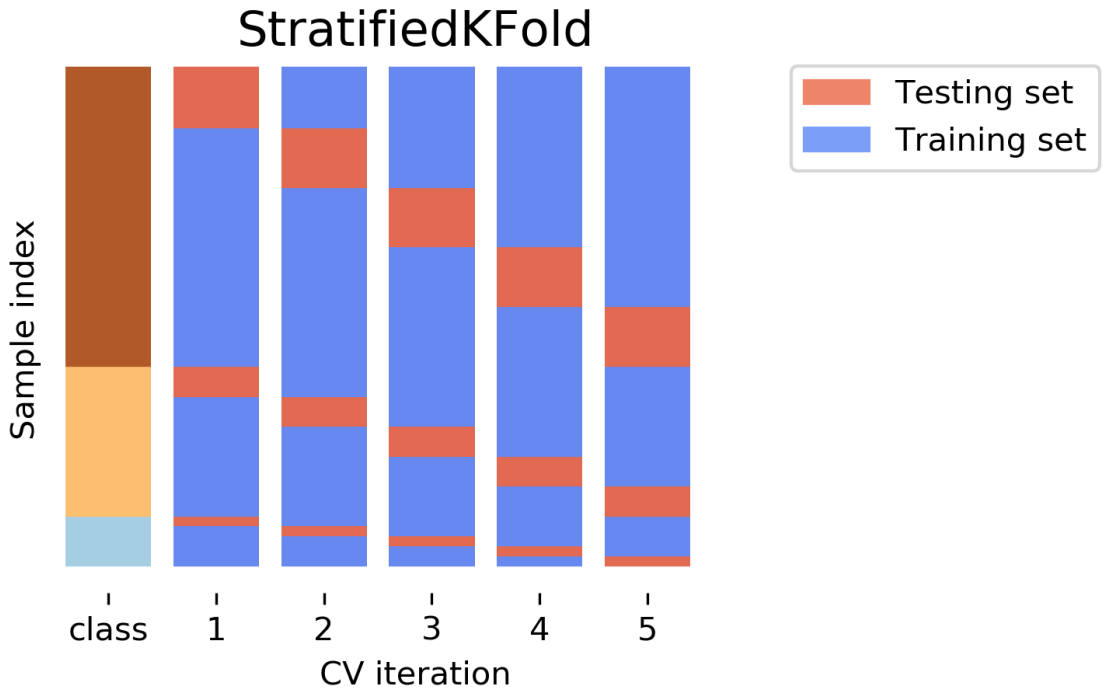
Tras esto utilizamos la validación cruzada para evaluar el rendimiento del modelo con diferentes combinaciones de hiperparámetros. Por ejemplo, se podrían realizar múltiples iteraciones de validación cruzada utilizando diferentes valores de C y tipos de kernel. En cada iteración, se entrena el modelo utilizando el conjunto de entrenamiento y se evalúa en el conjunto de validación. El rendimiento del modelo se registra para cada combinación de hiperparámetros.

Una vez que se han evaluado todas las combinaciones de hiperparámetros, se selecciona el conjunto que produce el mejor rendimiento en términos de alguna métrica de evaluación, como precisión, F1-score o área bajo la curva ROC.

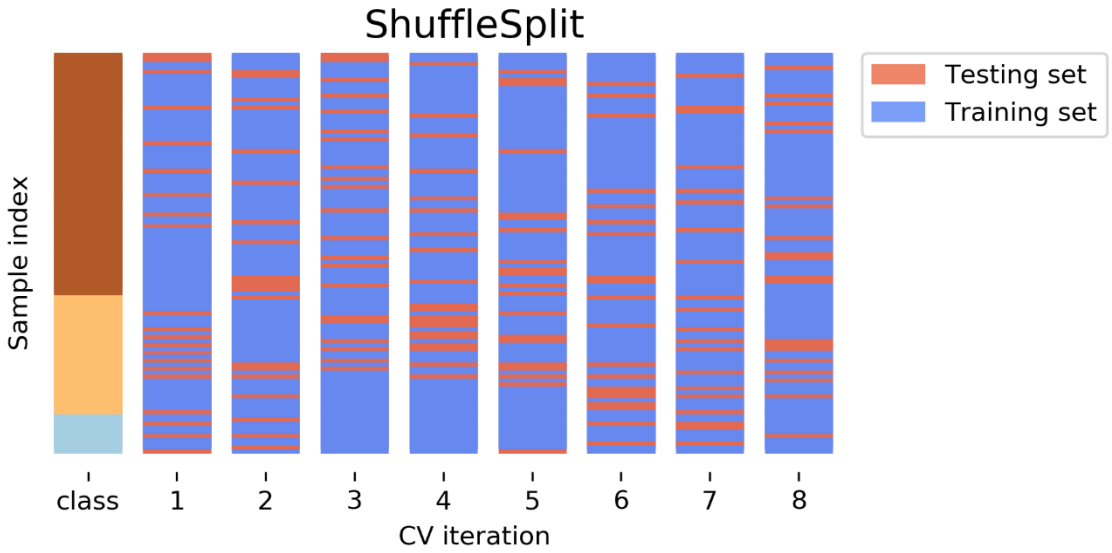
Finalmente, se entrena un modelo utilizando el mejor conjunto de hiperparámetros en todo el conjunto de datos (o en una combinación de entrenamiento y validación más grande) para obtener el modelo final.

Otras técnicas de validación

Stratified K-Fold es una variante mejorada de K-Fold que considera la distribución equilibrada de las clases al realizar las divisiones del conjunto de entrenamiento. Esto es importante para evitar sesgos en el modelo, especialmente en casos donde las clases están desequilibradas. Por ejemplo, si estamos clasificando datos en "SI" o "NO", si una iteración de K-Fold normal contiene solo muestras etiquetadas como "SI", el modelo podría aprender a generalizar incorrectamente. Stratified K-Fold resuelve este problema asegurando que cada división mantenga una proporción equilibrada de clases.



Leave P Out selecciona un número determinado P de muestras para ser excluidas como conjunto de validación en cada iteración. Por ejemplo, si elegimos P = 100, separamos 100 muestras para validar y realizamos el proceso de entrenamiento y evaluación en el resto de los datos, repitiendo este proceso como se explicó anteriormente. Es importante tener en cuenta que, si P es muy pequeño, esto resultará en un gran número de iteraciones de entrenamiento, lo que puede ser costoso computacionalmente y llevar mucho tiempo. Por otro lado, si P es muy grande, el conjunto de validación podría contener más muestras que el conjunto de entrenamiento, lo que no sería práctico. Por lo tanto, es crucial elegir un valor sensato para P que equilibre los puntajes de validación con el tiempo de entrenamiento.



ShuffleSplit primero mezcla los datos antes de dividirlos y nos permite especificar la cantidad de divisiones (splits) que deseamos realizar, así como el tamaño del conjunto de validación en cada división. Esta técnica es útil cuando queremos realizar múltiples iteraciones independientes de entrenamiento y evaluación, y nos permite controlar el tamaño del conjunto de validación en cada iteración.

Selección de características y reducción de la dimensionalidad

En el ámbito del Machine Learning y la ciencia de datos, el principal objetivo sigue siendo identificar las características más relevantes que influyen de manera significativa en los resultados de nuestros modelos en producción. Sin embargo, en muchos casos, los conjuntos de datos están saturados con numerosas características, lo que conduce al sobreajuste y aumenta los costos de entrenamiento, ralentizando considerablemente el proceso.

A lo largo del tiempo, se han desarrollado algoritmos para abordar estos problemas fundamentales, que incluyen:

- Reducción de la dimensionalidad del conjunto de datos para conservar la información relevante mientras se reduce la varianza.
- Reducción del tiempo y costo de entrenamiento.
- Desarrollo de formas efectivas de visualización.

Es crucial comprender la importancia de la reducción de la dimensionalidad antes de analizar los datos y extraer conclusiones. En muchos casos, la visualización del conjunto de datos es necesaria para comprenderlo, pero la presencia de numerosas variables dificulta esta tarea. Es aquí donde entra en juego la reducción de la dimensionalidad.

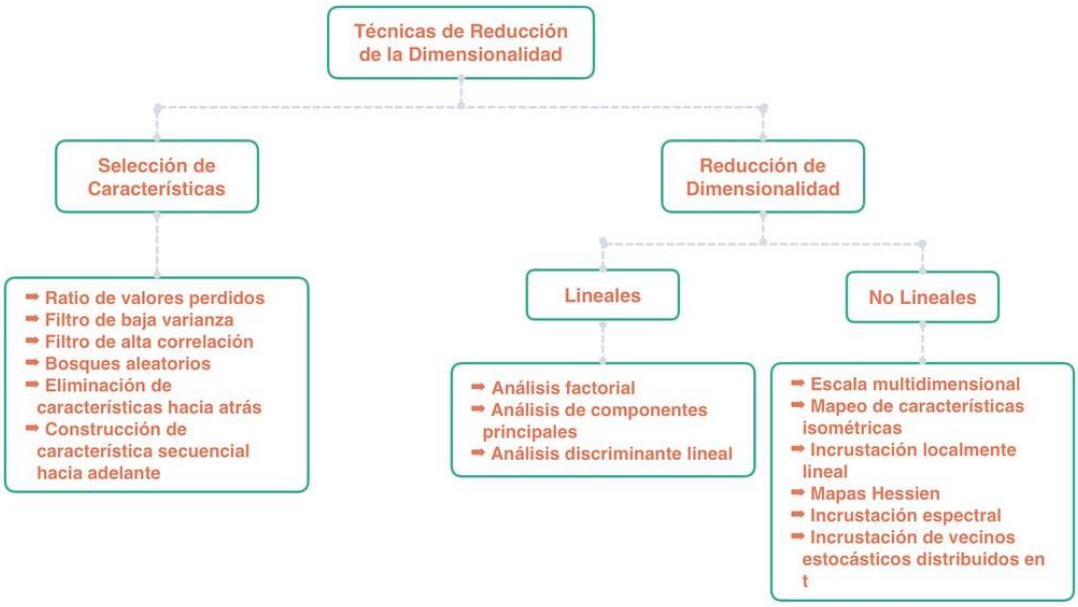
La reducción de la dimensionalidad implica la disminución del número de variables aleatorias en el conjunto de datos, obteniendo un conjunto de variables principales.

¿Por qué es relevante la reducción de la dimensionalidad? Imagina un escenario en el que se requieren numerosas variables indicadoras para obtener un modelo de Machine Learning preciso. Inicialmente, se pueden agregar tantas características como sea posible, pero llegará un punto en el que agregar más características no mejorará el rendimiento del modelo, sino que lo empeorará. Este fenómeno se conoce como "la maldición de la dimensionalidad".

La maldición de la dimensionalidad se manifiesta cuando la densidad de la muestra disminuye exponencialmente con el aumento de la dimensionalidad. A medida que se agregan más características sin aumentar el número de muestras de entrenamiento, el espacio de características se vuelve más disperso, lo que facilita el sobreajuste.

Para superar la maldición de la dimensionalidad y evitar el sobreajuste, especialmente cuando se tienen muchas características y pocas muestras de entrenamiento, se utilizan técnicas de reducción de la dimensionalidad. Estas técnicas se dividen principalmente en dos clases: eliminación de características y extracción de características.

- La eliminación de características implica descartar variables redundantes o que no aportan información nueva al conjunto de datos, lo que puede reducir la cantidad de datos necesarios para el análisis.
- La extracción de características implica la creación de nuevas variables a partir de las existentes, lo que puede ayudar a resumir la información contenida en el conjunto de datos.



A continuación, vamos a presentar cada una de las técnicas que se pueden observar en la imagen anterior.

Selección de características

La selección de características es un proceso fundamental en el campo del aprendizaje automático y la minería de datos que consiste en identificar y elegir las características más relevantes y significativas de un conjunto de datos para construir modelos predictivos más precisos y eficientes.

Cuando trabajamos con conjuntos de datos que contienen múltiples características o variables, no todas estas características contribuyen de manera igual a la capacidad predictiva de un modelo. Algunas características pueden ser redundantes, irrelevantes o incluso perjudiciales para el rendimiento del modelo, mientras que otras pueden ser altamente informativas y esenciales para hacer predicciones precisas.

La selección de características aborda este desafío al buscar identificar y retener solo las características más útiles para el problema en cuestión, descartando aquellas que no contribuyen significativamente o que pueden introducir ruido en el modelo.

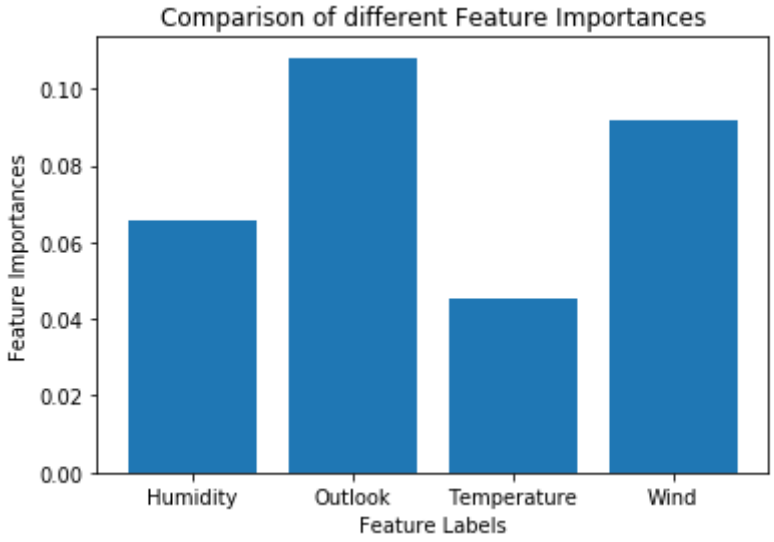
El proceso de selección de características puede llevarse a cabo mediante una variedad de técnicas y métodos, que incluye

- **Ratio de valores perdidos**: Es poco probable que las columnas de datos con un alto número de valores faltantes contengan información relevante. Por lo tanto, se pueden descartar las columnas con una proporción de valores perdidos que supere cierto umbral. Cuanto mayor sea este umbral, más agresiva será la reducción de características. Si en un conjunto de datos sobre pacientes médicos, una columna que registra la presión arterial tiene más del 50% de sus valores faltantes, es probable que esta columna no sea útil para el análisis y podría ser eliminada.

- **Filtro de baja varianza**: Las columnas de datos con poca variabilidad ofrecen poca información. Por tanto, se pueden eliminar aquellas columnas cuya desviación estándar sea inferior a un cierto umbral después de la normalización. En un conjunto de datos de precios de productos en una tienda, si una columna que indica descuentos tiene una desviación estándar muy baja (pocos cambios en los descuentos), podría eliminarse ya que no contribuye significativamente a las diferencias entre los productos.

- **Filtro de alta correlación**: Columnas de datos con una alta correlación entre sí probablemente contengan información redundante. Para evitar la duplicación de información, se pueden conservar solo una de las columnas altamente correlacionadas. En un conjunto de datos de ventas de productos, las columnas que registran el número de unidades vendidas y el total de ingresos pueden estar altamente correlacionadas. En este caso, podríamos conservar solo una de estas columnas para evitar redundancias en el análisis.

- **Bosques aleatorios**: Los bosques aleatorios no solo son efectivos para clasificar datos, sino también para identificar las columnas más relevantes. Al construir muchos árboles y analizar qué columnas se utilizan con mayor frecuencia para las divisiones, se pueden identificar las características más informativas. En un conjunto de datos de marketing digital, los bosques aleatorios podrían identificar qué métricas, como el número de clics o la tasa de conversión, son las más importantes para predecir el éxito de una campaña publicitaria.



Los modelos de árboles de decisión pueden utilizarse tanto para la selección de características como para la construcción de modelos predictivos. En el contexto de la selección de características, los árboles de decisión ofrecen una ventaja única debido a su capacidad intrínseca para evaluar la importancia de cada característica en la toma de decisiones.

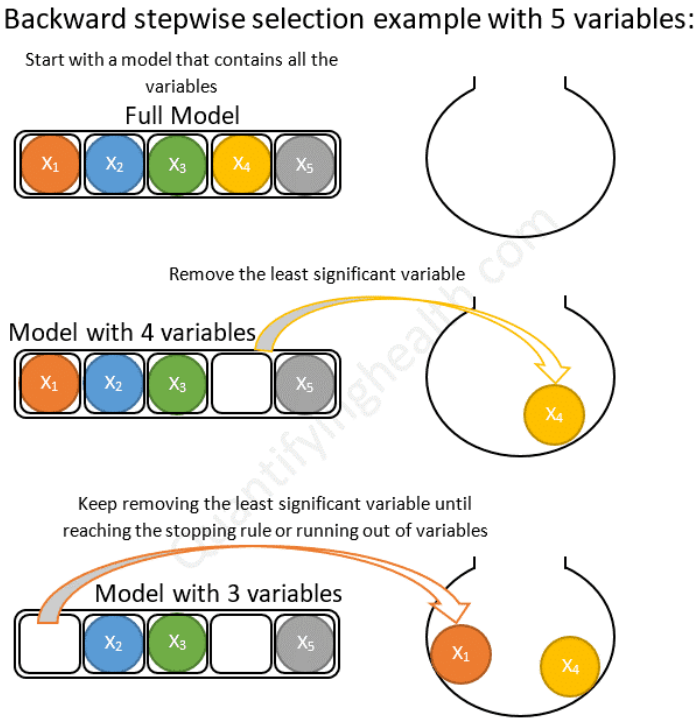
Durante el proceso de construcción del árbol de decisión, cada nodo del árbol se divide utilizando una característica específica del conjunto de datos. La importancia de cada característica se evalúa según cómo contribuye a la reducción de la impureza en los nodos del árbol. Cuanto más arriba en el árbol se utilice una característica para hacer divisiones, más importante se considera esa característica en la toma de decisiones del modelo.

Después de construir el árbol de decisión, se puede evaluar la importancia relativa de cada característica utilizando las métricas mencionadas anteriormente. Las características que contribuyen significativamente a la reducción de la impureza en los nodos del árbol se consideran importantes y pueden ser seleccionadas como las características más relevantes para el problema en cuestión.

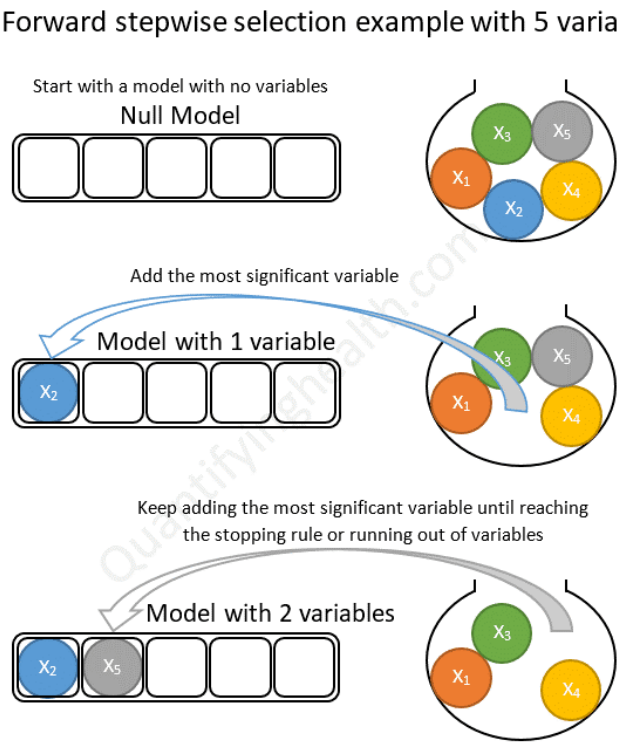
Una vez que se han evaluado las importancias de las características, se pueden tomar decisiones sobre qué características mantener y cuáles descartar. Las características menos importantes pueden ser eliminadas del conjunto de datos, lo que simplifica el modelo y reduce el riesgo de sobreajuste.

Además de evaluar la importancia de las características, la estructura misma del árbol de decisión puede proporcionar información sobre cómo interactúan las características entre sí y cómo se relacionan con la variable objetivo.

- **Eliminación de características hacia atrás**: Este método de selección de característica consiste en entrenar un modelo con todas las características y luego eliminar de forma iterativa la menos relevante. Se repite este proceso hasta alcanzar el rendimiento deseado con un número mínimo de características. En un conjunto de datos de diagnósticos médicos, podríamos usar la eliminación de características hacia atrás para identificar qué síntomas tienen menos impacto en la predicción de una enfermedad específica.



- **Construcción de características secuenciales hacia adelante**: Contrario a la eliminación de características hacia atrás, aquí comenzamos con una sola característica y agregamos progresivamente las más relevantes. Este método puede ser costoso en tiempo y recursos computacionales. En un conjunto de datos de análisis de crédito, podríamos usar la construcción de características secuenciales hacia adelante para identificar qué variables, como el historial crediticio o los ingresos mensuales, son las más importantes para predecir la probabilidad de pago de un préstamo.



Al realizar el proceso de selección de características mediante los métodos forward stepwise y backward stepwise comentados previamente, es común utilizar criterios como el AIC (Criterio de Información de Akaike) y el BIC (Criterio de Información Bayesiano) para evaluar y comparar los modelos obtenidos en cada paso.

A continuación, podemos observar la fórmula que nos permitiría calcular el valor de estos dos estadísticos:

$$\mathrm{AIC}_i = -2\log L_i + 2p_i$$

$$\mathrm{BIC}_i = -2\log L_i + p_i \log n$$

El AIC y el BIC son medidas de la calidad del ajuste de un modelo estadístico, que tienen en cuenta tanto la bondad de ajuste del modelo como su complejidad. En el contexto de la selección de características, estos criterios pueden utilizarse para evaluar diferentes modelos que difieren en la inclusión o exclusión de ciertas características.

En el primero de los notebooks con contenido sobre Scikit Lean y Python puedes observar como utilizamos el AIC para la construcción de modelos de regresión lineal y la selección de sus características.

- **AIC** (Criterio de Información de Akaike): Este criterio penaliza la complejidad del modelo al agregar un término que aumenta con el número de parámetros del modelo. Por lo tanto, un modelo con un valor de AIC más bajo se considera preferible, ya que proporciona un buen ajuste con una menor complejidad.

- **BIC** (Criterio de Información Bayesiano): Similar al AIC, el BIC también penaliza la complejidad del modelo, pero de una manera más estricta al incluir un término que aumenta más rápidamente con el número de parámetros del modelo. Por lo tanto, el BIC tiende a favorecer modelos más simples que el AIC.

El uso del AIC y el BIC en los métodos forward y backward stepwise permite una selección más objetiva y automatizada de características, ayudando a encontrar el conjunto óptimo de características que proporciona un buen ajuste del modelo con una complejidad adecuada.

Métodos de reducción de la dimensionalidad

La reducción de la dimensionalidad es un proceso fundamental en el análisis de datos y el aprendizaje automático, que consiste en disminuir el número de variables o dimensiones de un conjunto de datos. El objetivo principal de la reducción de la dimensionalidad es simplificar la representación de los datos mientras se conserva la mayor cantidad posible de información relevante.

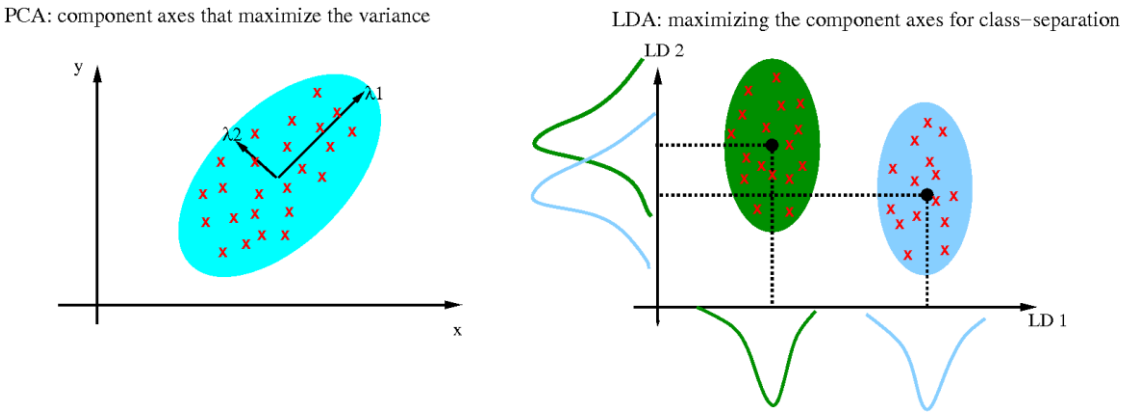
Los métodos que aplican transformaciones lineales son comunes en la reducción de dimensionalidad. Algunos ejemplos significativos incluyen:

- **Análisis factorial**: Esta técnica busca reducir un conjunto extenso de variables a un número menor de factores significativos. Se basa en la idea de que las observaciones pueden expresarse como combinaciones lineales de causas subyacentes. Se supone que estas observaciones son el resultado de una transformación lineal de factores latentes de menor dimensión, junto con un componente de ruido gaussiano. En un estudio socioeconómico, podemos reducir diversas medidas de estatus económico, educativo y laboral a un número menor de factores, como "nivel socioeconómico" o "calidad de vida".

- **Análisis de componentes principales** (PCA): Este procedimiento estadístico transforma las dimensiones originales de un conjunto de datos en un nuevo conjunto de dimensiones denominadas componentes principales. El primer componente principal captura la mayor varianza posible en los datos, y cada componente subsiguiente captura la máxima varianza no explicada por los componentes anteriores.

Conservar solo los primeros componentes principales reduce la dimensionalidad mientras se retiene la mayor parte de la información de los datos. Sin embargo, la interpretación de los resultados puede ser complicada, ya que las nuevas coordenadas no representan directamente las variables originales. En un conjunto de datos de imágenes, PCA podría usarse para reducir la dimensión de píxeles de la imagen mientras se conservan las características más relevantes para el reconocimiento de objetos o rostros.

- **Análisis discriminante lineal** (LDA): Este método busca proyectar los datos de manera que se maximice la separación entre clases. Los ejemplos de la misma clase se agrupan estrechamente en la proyección, mientras que los ejemplos de diferentes clases se separan lo más posible. LDA es útil cuando el objetivo es clasificar o separar datos en diferentes categorías. En un problema de reconocimiento facial, LDA podría usarse para encontrar una proyección que maximice la separación entre las imágenes de diferentes personas, facilitando así la clasificación precisa de nuevas imágenes en función de quién aparece en ellas.



Importancia de la validación para evitar sobreajuste y subajuste

El overfitting (sobreajuste) y el underfitting (subajuste) son dos problemas principales que pueden llevar a obtener resultados deficientes en el aprendizaje automático. Durante el entrenamiento de un modelo, buscamos ajustar los datos de entrada a la salida deseada. El overfitting ocurre cuando el modelo se adapta demasiado a los datos de entrenamiento, capturando el ruido y las fluctuaciones irrelevantes, lo que resulta en una falta de capacidad para generalizar correctamente sobre datos nuevos. Por otro lado, el underfitting ocurre cuando el modelo es demasiado simple para capturar la estructura subyacente de los datos, lo que resulta en un rendimiento deficiente tanto en los datos de entrenamiento como en los nuevos.

Para entender mejor estos conceptos, consideremos el siguiente ejemplo. En esencia, tanto las máquinas de aprendizaje como los seres humanos necesitan poder generalizar conceptos. Por ejemplo, cuando vemos un perro labrador por primera vez y nos dicen que es un perro, generalizamos esa información para reconocer a otros perros en el futuro, incluso si son de diferentes razas. Este proceso de generalización nos permite identificar nuevos ejemplos como "perros" o distinguirlos de otras entidades, como gatos.

De manera similar, al entrenar modelos de aprendizaje automático, buscamos que puedan generalizar patrones y características relevantes de los datos de entrenamiento para hacer predicciones precisas sobre datos nuevos y desconocidos. Si el modelo se sobreajusta, capturará demasiado los detalles específicos de los datos de entrenamiento, perdiendo la capacidad de generalización. Por otro lado, si el modelo está subajustado, no será capaz de capturar adecuadamente la complejidad de los datos y, por lo tanto, tendrá un rendimiento deficiente incluso en los datos de entrenamiento.

Tanto el overfitting como el underfitting son problemas que afectan la capacidad de un modelo para generalizar correctamente sobre datos nuevos y desconocidos, lo que puede llevar a resultados deficientes en el aprendizaje automático.

Debemos encontrar un equilibrio en el proceso de aprendizaje de nuestro modelo para evitar tanto el subajuste como el sobreajuste, lo cual puede ser desafiante. Una forma de reconocer estos problemas es dividir nuestro conjunto de datos en dos partes: una para entrenamiento y otra para pruebas, que el modelo no haya visto previamente. Usualmente, se reserva el 80% para entrenamiento y el 20% para pruebas, asegurándonos de que el conjunto de pruebas sea diverso y lo suficientemente grande como para evaluar los resultados después del entrenamiento.

Durante el entrenamiento, ajustamos diversos parámetros del algoritmo, como el número de iteraciones o la tasa de aprendizaje, y comparamos los resultados con el conjunto de pruebas para encontrar el mejor rendimiento sin caer en problemas de ajuste.

Para prevenir el sobreajuste y el subajuste, podemos tomar varias medidas:

- Garantizar una cantidad mínima de muestras tanto para el entrenamiento como para la validación.
- Mantener un equilibrio de clases en los datos de entrenamiento, especialmente en el aprendizaje supervisado.
- Reservar un conjunto de datos de prueba para evaluar el rendimiento del modelo de manera imparcial.
- Considerar la reducción de características si hay muchas dimensiones con pocas muestras, utilizando técnicas como PCA.
- Identificar problemas de sobreajuste si hay una gran discrepancia entre el rendimiento en el conjunto de entrenamiento y el conjunto de pruebas.

- En el caso de modelos de aprendizaje profundo, evitar el sobreajuste limitando el número de capas ocultas para evitar la memorización de salidas.

- Reconocer el subajuste si el modelo tiene dificultades para generalizar y solo puede predecir un tipo de clase o produce resultados constantes y poco variados en el conjunto de pruebas.

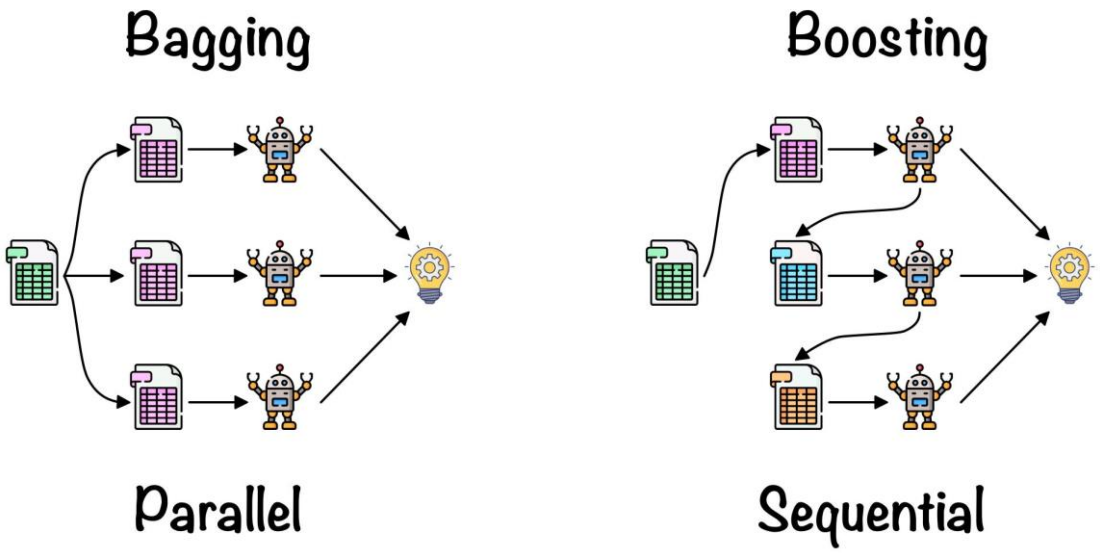
Ensemble. Bagging y Boosting

Por último, vamos a hablar de un concepto realmente interesante que nos permite combinar múltiples modelos de aprendizaje para mejorar el rendimiento predictivo y la generalización del sistema, este concepto recibe el nombre de ensemble.

Los métodos de ensambling logran esto al combinar las predicciones de varios modelos base, lo que puede reducir el sesgo y la varianza del modelo final. Al combinar múltiples modelos, el ensemble puede capturar una gama más amplia de patrones y características en los datos, lo que conduce a predicciones más precisas y robustas.

Además, al promediar las predicciones de varios modelos, el ensemble puede mitigar el efecto del sobreajuste, especialmente si los modelos base tienen diferentes sesgos y varianzas y tiende a ser más estable y robusto frente a cambios en los datos de entrenamiento, ya que no dependen tanto de un solo modelo.

Existen diferentes técnicas de ensemble, como el bagging y el boosting, que pueden adaptarse a diferentes tipos de datos y problemas de aprendizaje automático.



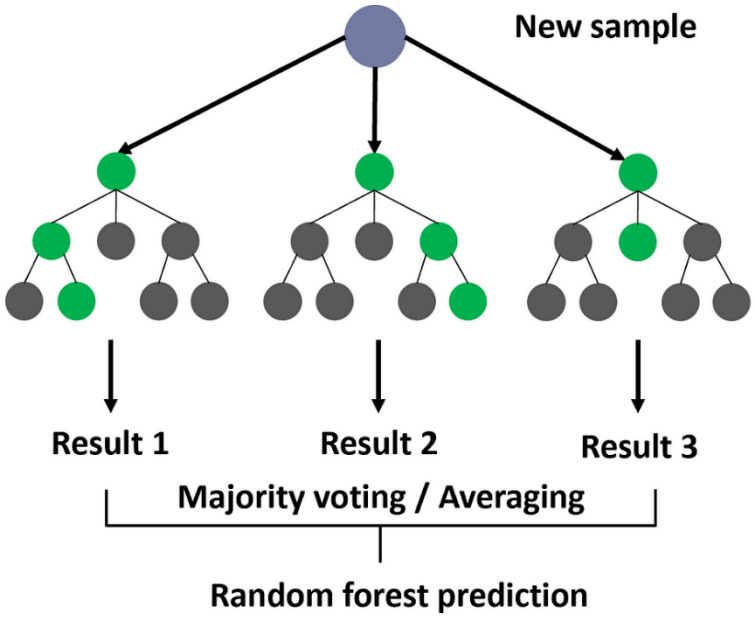
Tanto el bagging como el boosting son técnicas de ensemble que combinan múltiples modelos para mejorar el rendimiento predictivo. Sin embargo, difieren en cómo se construyen y cómo combinan los modelos base.

Por un lado, Bagging es una técnica de ensemble que se basa en la idea de entrenar múltiples modelos base de forma independiente y luego combinar sus predicciones mediante votación o promedio. La característica principal del bagging es que cada modelo base se entrena en una submuestra aleatoria del conjunto de datos de entrenamiento con reemplazo (bootstrap), lo que significa que algunas instancias pueden aparecer múltiples veces en una muestra y otras pueden no aparecer en absoluto. Esto ayuda a reducir la varianza al promediar las predicciones de los modelos base, lo que puede llevar a un mejor rendimiento general.

El algoritmo de bagging más conocido es el Random Forest (del que hablaremos posteriormente), que utiliza árboles de decisión como modelos base y promedia sus predicciones para obtener una predicción final.

Por otro lado, boosting es otra técnica de ensemble que se centra en mejorar la precisión de un modelo combinando de manera secuencial múltiples modelos base débiles. A diferencia del bagging, en boosting los modelos base se construyen de manera iterativa, y cada modelo se enfoca en mejorar la predicción de las instancias mal clasificadas por los modelos anteriores. En cada iteración, se da más peso a las instancias mal clasificadas para que los modelos subsiguientes se centren en corregir esos errores.

AdaBoost (Adaptive Boosting) es uno de los algoritmos de boosting más populares. Otros ejemplos incluyen Gradient Boosting Machine (GBM), XGBoost y LightGBM.



Como hemos comentado previamente, Random Forest es un algoritmo de aprendizaje supervisado utilizado tanto para problemas de clasificación como de regresión. Es una extensión de los árboles de decisión que se basa en la técnica de bagging (bootstrap aggregating) para mejorar la precisión y la generalización del modelo.

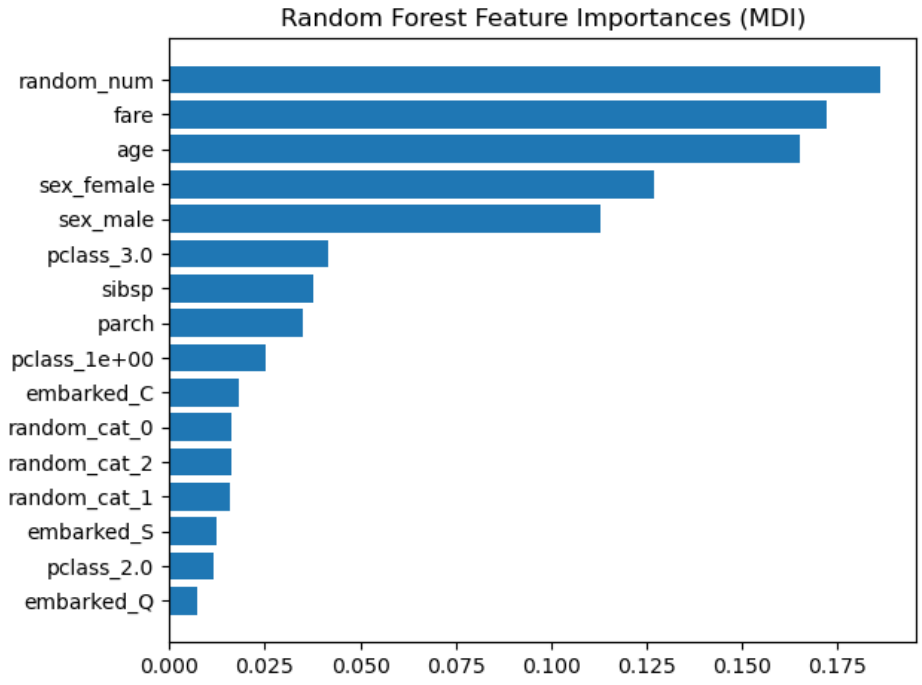
Random Forest se compone de un conjunto (ensemble) de árboles de decisión. Cada árbol se construye de forma independiente utilizando una submuestra aleatoria del conjunto de datos de entrenamiento y un subconjunto aleatorio de características en cada división del árbol.

La aleatoriedad es una característica clave de Random Forest. En cada división de un árbol, se selecciona un subconjunto aleatorio de características, lo que ayuda a reducir la correlación entre los árboles y a mejorar la diversidad del ensemble. Además, cada árbol se entrena en una submuestra aleatoria del conjunto de datos de entrenamiento, lo que contribuye a la generalización del modelo.

Una vez que se han entrenado todos los árboles en el conjunto, las predicciones de clasificación se determinan por votación mayoritaria, mientras que en el caso de regresión se promedian las predicciones de todos los árboles.

Debido a la aleatoriedad en la construcción de los árboles y al promedio de múltiples modelos, Random Forest tiende a ser robusto frente al sobreajuste, especialmente en conjuntos de datos ruidosos o con alta dimensionalidad.

Aunque los modelos de Random Forest pueden ser más difíciles de interpretar que un solo árbol de decisión, aún es posible evaluar la importancia de las características en la clasificación o regresión gracias a métricas como la importancia de las características (feature importance), que indican cuánto contribuye cada característica a la precisión del modelo.



El feature importance (importancia de las características) es una métrica muy interesante que podemos obtener al implementar el modelo de Random Forest. Esta es una medida que indica cuánto contribuye cada característica o variable en el modelo al hacer predicciones. Esta medida es útil para comprender qué características son más relevantes para la tarea de predicción y puede ayudar en la selección de características, la interpretación del modelo y la detección de posibles sesgos.

El feature importance en Random Forest se calcula mediante la evaluación de cuánto disminuye la precisión del modelo cuando se permutan los valores de una característica en el conjunto de datos de prueba. Si la precisión disminuye significativamente, entonces se considera que esa característica es importante para el modelo, ya que su presencia o ausencia influye en las predicciones.

Las características con una mayor importancia son aquellas que tienen un mayor impacto en las predicciones del modelo. Las características con una alta importancia son aquellas que tienen una fuerte relación con la variable objetivo y son más informativas para hacer predicciones precisas.

Comprender qué características son más importantes puede ayudarte a interpretar el modelo de manera más profunda y a comprender qué aspectos del conjunto de datos están influyendo más en las predicciones.

Es importante tener en cuenta que el feature importance en Random Forest no necesariamente indica causalidad, sino más bien una relación estadística entre las características y las predicciones del modelo. Por lo tanto, siempre se debe interpretar con cautela y considerar el contexto del problema específico.

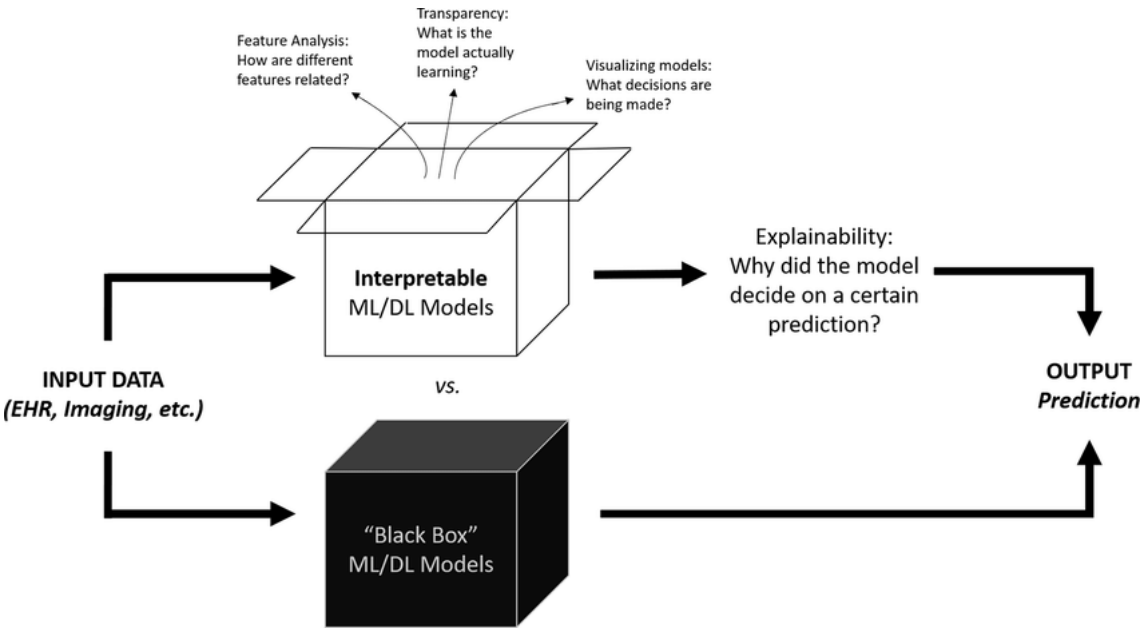
Para entender cómo se determina qué variables son más influyentes, se puede observar cuántas veces una característica es seleccionada para dividir un nodo en los árboles individuales que componen el bosque. Si una variable aparece con mayor frecuencia en los niveles superiores de los árboles, es indicativo de que tiene más importancia en la predicción del modelo. En otras palabras, si una característica se usa con frecuencia para dividir los nodos y esto conduce a una mejora en la precisión del modelo, entonces se considerará más importante. Esto se debe a que las características que se utilizan en los nodos superiores de los árboles tienen un mayor impacto en la separación de las clases objetivo. Por lo tanto, las variables que se utilizan en estos nodos superiores se consideran más influyentes en el proceso de toma de decisiones del modelo.

Por último, comentar que el análisis del "feature importance" puede ser crucial en la detección de sesgos éticos en el aprendizaje automático, ya que permite examinar qué factores están siendo considerados de manera desproporcionada en la toma de decisiones.

Por ejemplo, si ciertas variables relacionadas con la raza, el género o el origen socioeconómico tienen una importancia excesiva en el modelo, esto podría indicar la presencia de sesgos discriminatorios. Este tipo de análisis es fundamental para abordar cuestiones éticas en el machine learning, ya que permite una mayor transparencia y comprensión de cómo se están tomando las decisiones automáticas y cómo pueden estar sesgadas. En futuros temas, exploraremos más a fondo cómo estos hallazgos pueden ayudar a mitigar y eliminar sesgos injustos en los sistemas de inteligencia artificial.

Shapley Values

Estamos acostumbrados a desarrollar modelos para resolver los diferentes desafíos que se nos plantean, pero en ocasiones, estos modelos son bastante complejos y entender su comportamiento resulta difícil. Este es el caso, por ejemplo, de las redes neuronales u otros modelos que funcionan como "cajas negras".



Ante esta situación, nos enfrentamos a la pregunta de si debemos cambiar nuestro modelo sofisticado por uno más simple, como una regresión, que pueda ser más fácil de explicar pero que posiblemente no ofrezca resultados tan buenos. O tal vez nos conformamos con la mentalidad de "si funciona, no lo toques".

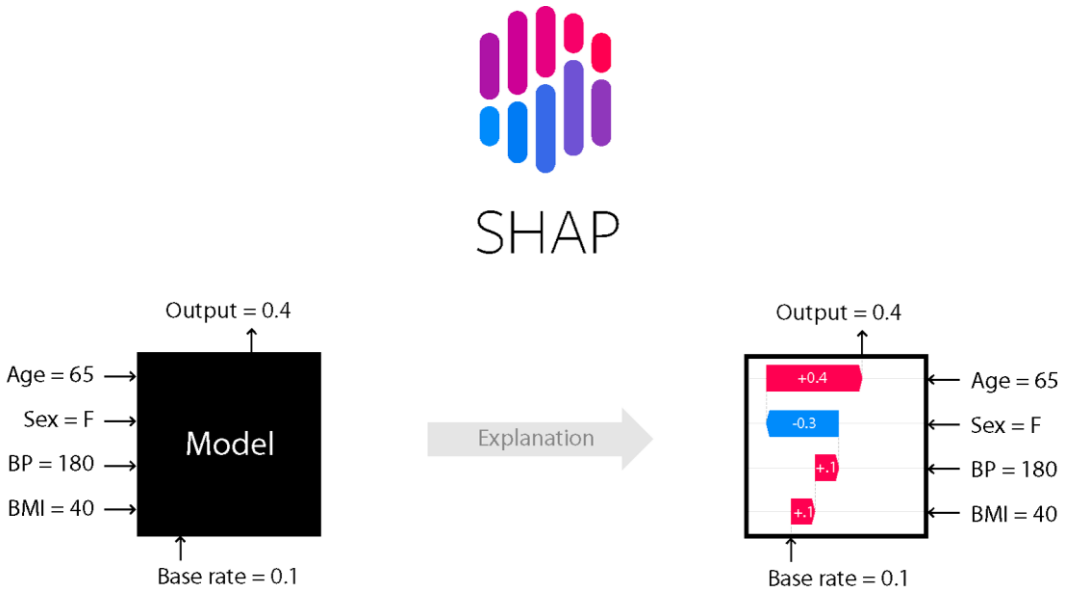
Afortunadamente, existen alternativas para comprender mejor el funcionamiento de estos modelos. Una de las más populares en la actualidad es SHAP. Pero antes de profundizar en su implementación, es importante entender sus fundamentos teóricos.

Los Shapley Values son conceptos introducidos por Lloyd Shapley en 1951 y se centran en asignar de manera óptima los "puntos" ganados en un juego entre todos los jugadores involucrados. Podemos visualizarlo como un juego donde varios jugadores contribuyen a un resultado común, y los Shapley Values determinan cuánto aporta cada jugador de manera justa.

Para entender cómo se calculan los Shapley Values, imaginemos un partido con 4 jugadores donde se han anotado 2 puntos. ¿Cuánto ha contribuido cada jugador? Este enfoque se puede extrapolar al análisis de modelos, donde el "partido" sería nuestro modelo y los "jugadores" serían las diferentes variables que lo componen.

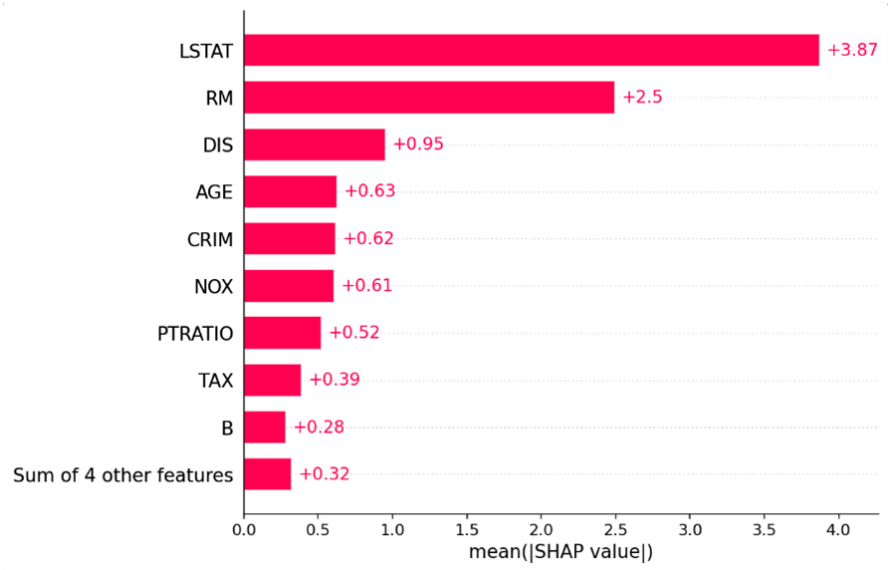
El cálculo de los Shapley Values implica la creación de todas las posibles combinaciones de jugadores, evaluando cómo cambia el resultado al agregar cada uno de ellos, teniendo en cuenta todas las permutaciones posibles para garantizar la justicia en la asignación de contribuciones.

SHAP es una librería ampliamente utilizada para calcular los Shapley Values de manera eficiente, basada en la teoría de los mismos y con herramientas para visualizar los resultados. Está disponible en Python y en R.

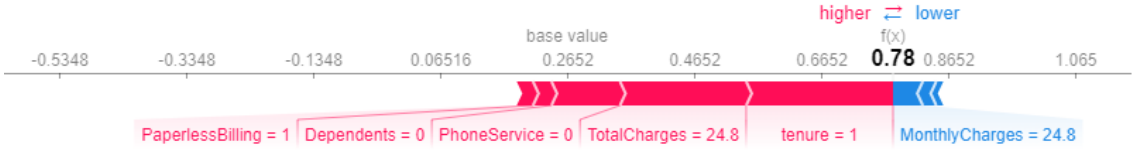


Como hemos comentado previamente, SHAP proporciona diversas gráficas out-of-the-box, que resultan muy útiles para analizar y comprender los resultados. Algunas de las más relevantes son:

- **Bar plot**: la gráfica de importancia de variables de toda la vida, pero basada en la suma de los valores SHAP absolutos para cada variable y todas las muestras.

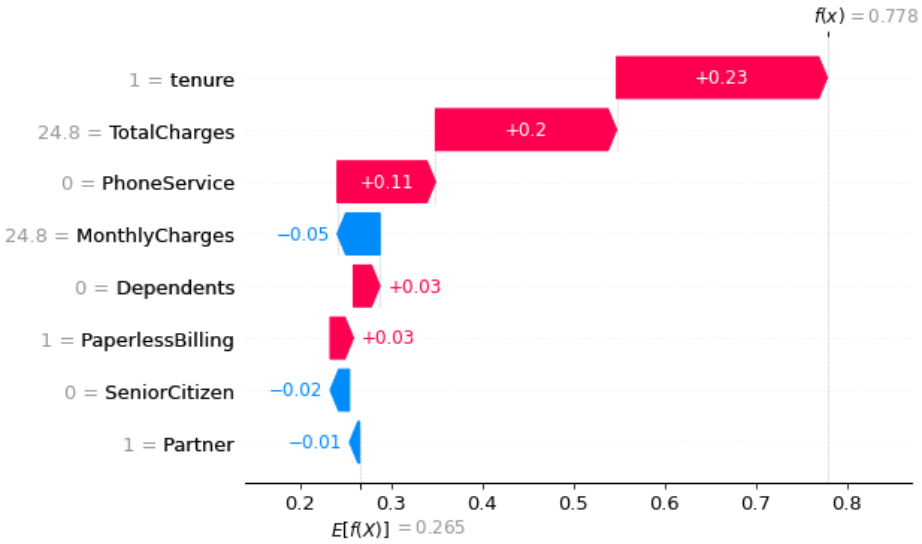


- **Force plot:** Los diagramas de fuerza se pueden utilizar para describir los efectos de las variables en un conjunto de muestras, pero también para instancias específicas. Este segundo uso es perfecto para poder explicarle a alguien exactamente cómo llegó su modelo a la predicción que hizo para una observación específica.



El 0,78 en negrita es el score final del modelo para esta observación. El base value de 0,2652 es la predicción que obtendríamos si el resto de las variables no contuviesen información, y es por tanto la predicción "por defecto". A partir de ahí, cada variable va sumando (en rojo) o restando (en azul) a dicha predicción. La magnitud con la que contribuye se aprecia por el tamaño de la barra. Así, podemos ver que, para este individuo, su predicción de abandono se ha incrementado mucho debido a su poca antigüedad, sus bajos cargos totales y a no tener contratado el teléfono. Aunque tiene variables que le restan riesgo, como el hecho de ser cliente Senior o tener los cargos mensuales bajos, esto no es suficiente para evitar que el cliente aparezca como "peligroso" para el modelo.

- **Waterfall plot:** Los diagramas de cascada están diseñados para mostrar explicaciones de predicciones individuales de forma muy parecida al force plot. Al igual que en el force plot, partimos del intercept (valor esperado de la predicción), y en cada fila se representa el valor de shap de cada variable, que contribuye positiva o negativamente en el resultado final de la predicción. Adicionalmente, en el eje Y vemos tanto el nombre de la variable como su valor original.



En este gráfico se aprecia muy bien cómo se relacionan los valores de shap y el intercept con la predicción final. Así, vemos que, si sumamos el intercept y los valores shap de todas las variables, obtenemos la predicción final de 0.778.