Proyecto Final

Nombre	Apellidos	Código
Santiago	Bobadilla Suarez	201820728
Daniela	Ruiz López	201914337
Daniel	Giraldo	201920055

Descripción del Problema 1. Music Year Prediction

Este problema esta basado en el proyecto Million Song Dataset de la universidad de Columbia, el cual consiste en predecir el año en que una canción fue lanzada de acuerdo con 90 atributos. Teniendo en cuenta la naturaleza de numérica de estos 90 atributos se busca realizar diferentes modelos de regresión para estimar de la manera más exacta posible el año en que fue lanzada una canción.

Exploración y visualización de datos

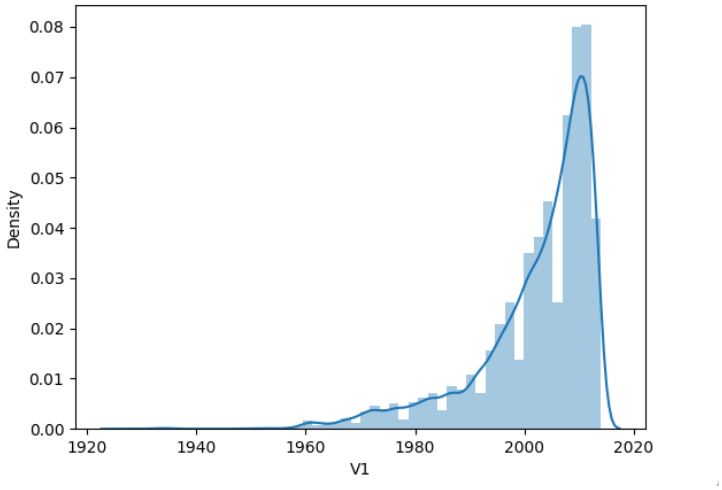

Ilustración 1. Histograma de frecuencia de los años de las canciones.

La figura 1 muestra un histograma combinado con una curva de densidad para la variable V1, que representa el año de lanzamiento de las canciones. La mayoría de las canciones parecen haberse lanzado en un período más reciente, lo cual es evidente por la gran barra en el extremo derecho del histograma, de hecho, el pico en la distribución ocurre cerca al 2020. Hay una pequeña cantidad de canciones distribuidas de manera esporádica en años anteriores, desde alrededor de 1920 hasta el 2000, lo que indica que hay menos canciones de esas épocas en la base de datos. Esto puede deberse a que la disponibilidad y la popularidad de la música grabada ha aumentado con el tiempo, y los métodos de recolección de datos tienden a ser más completos para los años recientes.

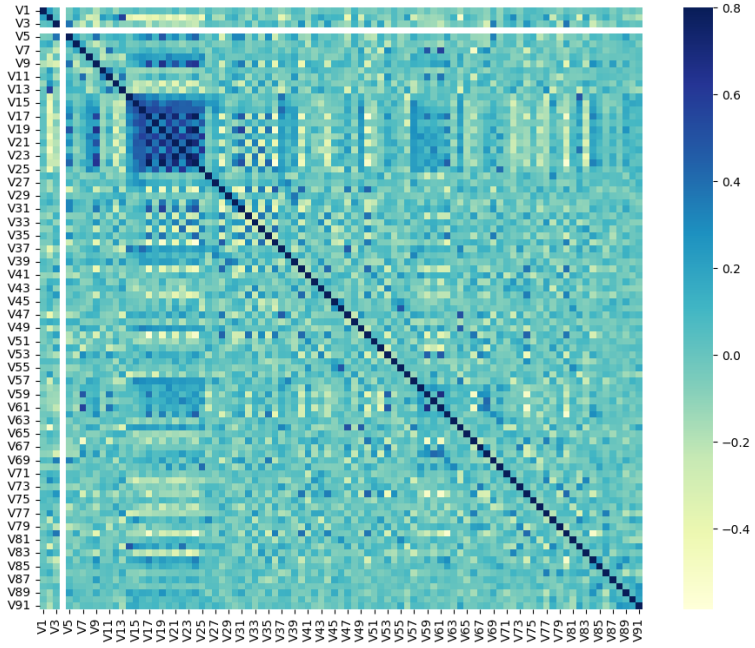

Ilustración 2. Matriz de correlación de todas las variables

Ilustración 3. Matriz de correlación de las 5 variables más correlacionadas con la variable de predicción

En las ilustraciones 2 y 3 podemos ver las correlaciones entre las variables según las escalas de color definidas para cada una. La presencia de cuadrados oscuros en la ilustración 2 indica un grupo de variables que están fuertemente correlacionadas entre sí, las cuales van desde V13-V25. Esto es un indicio de multicolinealidad, lo cual puede ser un problema al momento de realizar predicciones, por lo que se podría considerar eliminar algunas de estas variables para reducir la redundancia. Sin embargo, la mayoría de las variables parecen tener poca relación entre ellas, en especial aquellas que tienen más correlación con la variable de predicción.

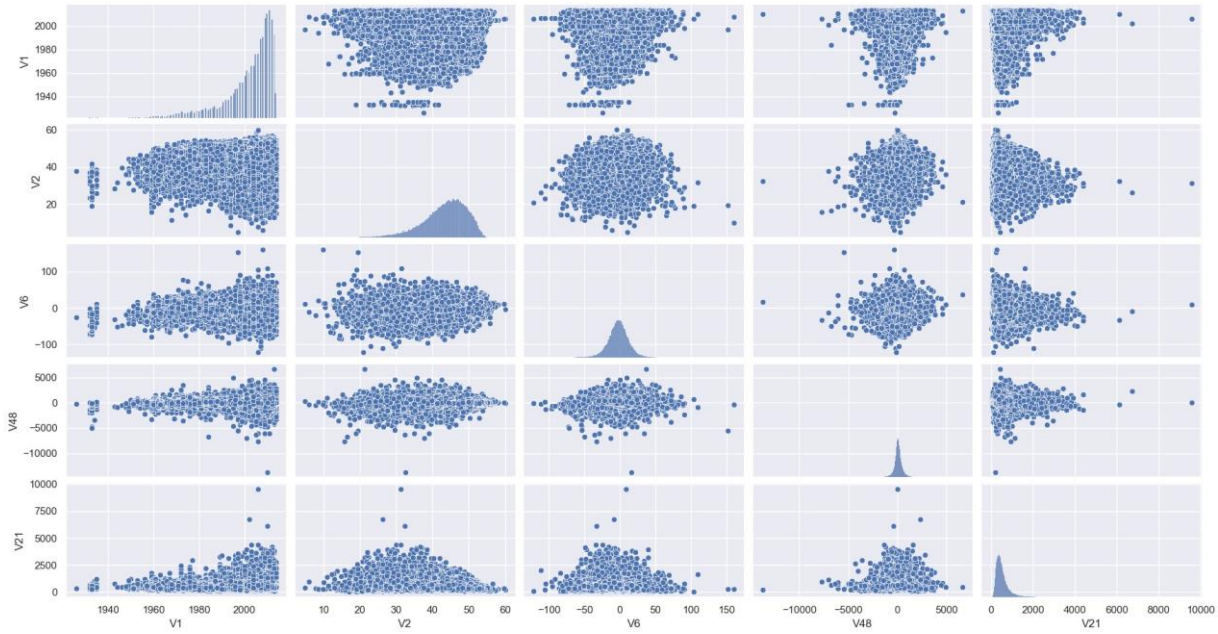

Ilustración 4. Gráfico de pares entre las variables con mayor relación al año de lanzamiento

Realizamos un gráfico de pares para explorar mejor las relaciones entre las variables más correlacionadas. Cada gráfico individual muestra la relación entre dos variables, con una variable asignada al eje X y la otra al eje Y. Los histogramas en la diagonal muestran la distribución de cada variable independiente.

Al revisar las distribuciones de las relaciones, notamos que la mayoría muestran una nube de puntos dispersa sin un patrón discernible, lo que sugiere una falta de relación lineal o una relación muy débil entre las variables comparadas. Muchas de las variables parecen tener una distribución en forma de cono (o embudo), lo que indica heterocedasticidad, es decir, la variabilidad de una variable es desigual a lo largo de los valores de la otra variable.

En cuanto a las distribuciones individuales, tenemos que la distribución asimétrica de V1 indica un sesgo en los datos. También hay presencia de valores atípicos, como se puede ver en los extremos de algunas distribuciones. Sin embargo, la mayoría parecen tener una distribución normal. Por otro lado, las variables tienen rangos muy diferentes, como se puede ver en los ejes (por ejemplo, V1 va de 1940 a algo más allá de 2000, mientras que V48 va de -10000 a 5000). Esto puede afectar el análisis, por lo que demuestra que se requiere realizar normalización o estandarización de los datos antes de emplearlos en los modelos predictivos.

Preparación y limpieza de datos

Para la preparación de los datos primero se analizaron las variables que componen a cada canción, donde se encontró que la variable "V4" es 100% ceros, no tiene valores distintos por lo que no aporta nada al modelo de regresión, decidiendo extraerla para la aplicación de los modelos. En cuanto a las demás variables, no se evidenciaron datos faltantes, por lo que se decidió continuar con ellos el proceso de selección y extracción de variables.

Selección y Extracción de variables.

Se utilizó selección de variables para identificar el número de componentes principales en el que la acumulación de varianza explicada comienza a nivelarse. Ese número puede servir como una guía para seleccionar el número de componentes principales que se conservarán en el análisis PCA. El número de componentes que representan el 95% de la varianza se empieza a estabilizar en la variable 68 como se puede apreciar en la Ilustración **¡Error! No se encuentra el origen de la referencia.**.

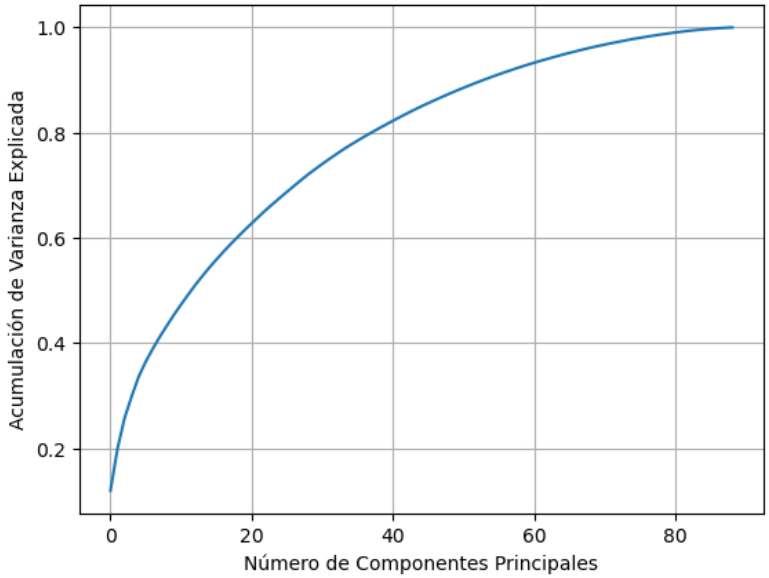

Ilustración 5. Varianza acumulada de las variables para PCA

Para la selección de variables se probaron varias técnicas de selección de características. Entre ellas está Recursive Feature Elimination (RFE), la cual ajusta el modelo y elimina las características menos importantes de manera recursiva hasta que se alcanza el número deseado de características. Por otro lado, SelectKBest que selecciona las k características con puntuaciones más altas en pruebas estadísticas como ANOVA y Chi-cuadrado. Finalmente, SelectFromModel selecciona características a través de un estimador (en este caso Lasso) que asigna coeficientes a las características y a partir de un umbral se seleccionan las variables.

Evaluación de modelos.

Para la evaluación y desempeño de los modelos se optó por probar desde los modelos más sencillos partiendo desde regresión lineal, aplicando penalización (lasso) y ver el desempeño, a partir de los resultados evaluar si modelos más complejos podían tener mejor desempeño con el conjunto completo o a partir de la transformación de variables se podía disminuir la métrica de RMSE.

Entre los modelos que se utilizaron se encuentran regresión lineal con selección de variables, regresión Lasso de grado dos con Cross-validation, Bosques aleatorios, elastic net, gradient boosting, Xgboost y redes neuronales.

Durante los intentos de modelado, utilizamos el método SelectKBest para seleccionar características en el modelo de regresión lineal. SelectKBest selecciona características basándose en una prueba estadística, generalmente la más fuertemente asociadas con la variable objetivo, obteniendo 30 caracteristicas principales las cuales son V2, V6, V9, V12, V15, V21, V22, V25, V26, V30, V32, V34, V37, V39, V41, V42, V47, V48, V54, V58, V60, V64, V68, V69, V70, V73, V74, V75, V79, V87. Las características seleccionadas son aquellas que tienen la asociación más fuerte con la variable objetivo según la prueba estadística. Esto no significa necesariamente que estas características sean causalmente importantes, sino que están estadísticamente asociadas con la variable objetivo. Con este modelo el resultado de RMSE evaluado en test fue de RMSE: 9.50113.

EL modelo de regresión lasso con grado de polinomio 2 después de evaluar en test RMSE: 9.52027, los coeficientes de penalización se pueden ver apropiadamente dentro del repositorio.

Gradient boosting es una técnica de aprendizaje automático para problemas de regresión y clasificación, que produce un modelo de predicción en forma de un ensamble de modelos predictivos débiles, típicamente árboles de decisión. Para este modelo se probaron diferentes configuraciones de los métodos de selección de variables que se listan en la tabla siguiente junto con el reporte de la métrica MAE:

Gradient boosting

Feature selection method	n features	MAE
None	90	7.44966572
RFE	20	8.01407817
RFE	45	7.4031242
RFE	60	7.32386218
RFE	70	7.692
Kbest	60	7.4192
select from model	80	7.31672666

De acuerdo con los resultados anteriores, cuando no se realiza selección de características, el modelo usa todas las 90 características disponibles y obtiene un MAE de 7.4497. Este es el punto de partida para la comparación con otros métodos de selección de características.

El método RFE con diferentes números de características muestra una tendencia interesante; a medida que el número de características aumenta de 20 a 60, el MAE disminuye, indicando un mejor rendimiento. Sin embargo, al pasar a 70 características, el MAE aumenta ligeramente, lo que sugiere que pueden haberse incluido características menos predictivas o ruido en el modelo. Debido a esto, se decidió probar el método Kbest con 60 características, obteniendo un MAE de 7.4192, que es ligeramente peor que el mejor RFE con 60 características, pero aun así mejor que no seleccionar ninguna. Por último, "SelectFromModel" seleccionó 80 características al utilizar un umbral de 0.2 y obtuvo el MAE más bajo de 7.3167, lo que sugiere que este método ha sido capaz de identificar el subconjunto de características que proporcionan la mejor capacidad predictiva para el modelo de Gradient Boosting.

Posteriormente decidimos probar el desempeño de XgBoost, el cual sigue el mismo principio del modelo anterior, pero utiliza más regularización para controlar el sobreajuste y por ende, suele reportar mejores resultados,

XGBoost

Feature selection method	n_features	MAE
None	90	7.3169838
select from model	80	7.3169838
RFE	60	7.30007714

Usando todas las 90 características disponibles, el modelo XGBoost obtuvo un MAE de 7.3169838. Al aplicar el método 'select from model' se seleccionaron las mismas 80 características que anteriormente y se mantuvo un MAE idéntico a cuando no se realizó selección de características. Esto sugiere que las 10 características eliminadas no tenían un impacto significativo en el desempeño del modelo o que el modelo era capaz de manejar la inclusión de todas las características sin perder rendimiento. Usando RFE se redujo el número de características a 60, lo que resultó en un MAE ligeramente mejor de 7.300077141. Esto indica que eliminar las 30 características menos importantes (según lo determinado por el proceso RFE) ayudó a mejorar el rendimiento del modelo. La falta de cambio en el MAE cuando se utiliza 'select from model' sugiere que la selección de características no siempre resulta en una mejora del rendimiento, especialmente si el modelo original ya es robusto frente a características irrelevantes o redundantes.

Finalmente, probamos la implementación de una red neuronal. Para esta red se probaron varias configuraciones de capas e hiperparámetros, pero aquella que reportó el mejor desempeño según la métrica del MAE se compone de tres capaz con regularización de Dropout. De manera mas especifica, La primera capa es una capa totalmente conectada con 64 neuronas unidades. Esta capa recibe el número de características del conjunto de datos como su dimensión de entrada. Se utiliza la función de activación ReLU (Rectified Linear Unit) y se aplica la regularización de Dropout con una tasa de 5%, lo que significa que al azar el 5% de las neuronas se "apagarán" durante el entrenamiento. La segunda capa es otra capa densa con 16 neuronas. Se inicializa de la misma manera que la primera capa y también utiliza la activación ReLU. Por último, la capa de salida tiene una sola neurona. Se utilizó el optimizador Adam con una tasa de aprendizaje de 0.01.

Clasificación Global: Esto proporciona una visión general del rendimiento general de los modelos en comparación a la muestra de prueba en la competición de Kaggle.

1. Regresión Lasso (Mejor RSME: 9.50)
2. Bosque Aleatorio (Mejor RMSE: 9.76)
3. Redes Neuronales Mejor RMSE: 10.20
4. Redes Elásticas (Mejor RMSE. 10.55)
5. XgBoosting (Mejor RMSE. 10.72)
6. Gradient Boosting (Mejor RMSE 10.80)

El RSME mide la magnitud de los errores entre los valores predichos por un modelo y los valores reales. Se calcula tomando la raíz cuadrada del promedio de los cuadrados de estos errores.

Se esperaba un mejor desempeño de los modelos en el test, debido a la calibración y elección de las variables, en conclusión, aunque la combinación de técnicas de selección y regresión de los modelos presentados anteriormente estaba destinada a mejorar el rendimiento mediante la reducción de la varianza y el sobreajuste, los resultados obtenidos sugieren que, en este caso específico, la configuración de hiperparámetros no logró capturar patrones significativos en los datos.

Descripción del Problema 2: Bankruptcy.

Cada una de las secciones presentan el resumen de la experimentación y el resumen de todos los resultados, para más detalles remitirse a: https://github.com/danielaruizl1/Statistical_project. Acá no se presenta los resultados de las corridas, ni detalles específicos de los hiperparámetros de cada modelo, eso se encuentra en cada archivo del repositorio.

Exploración y Visualización de datos

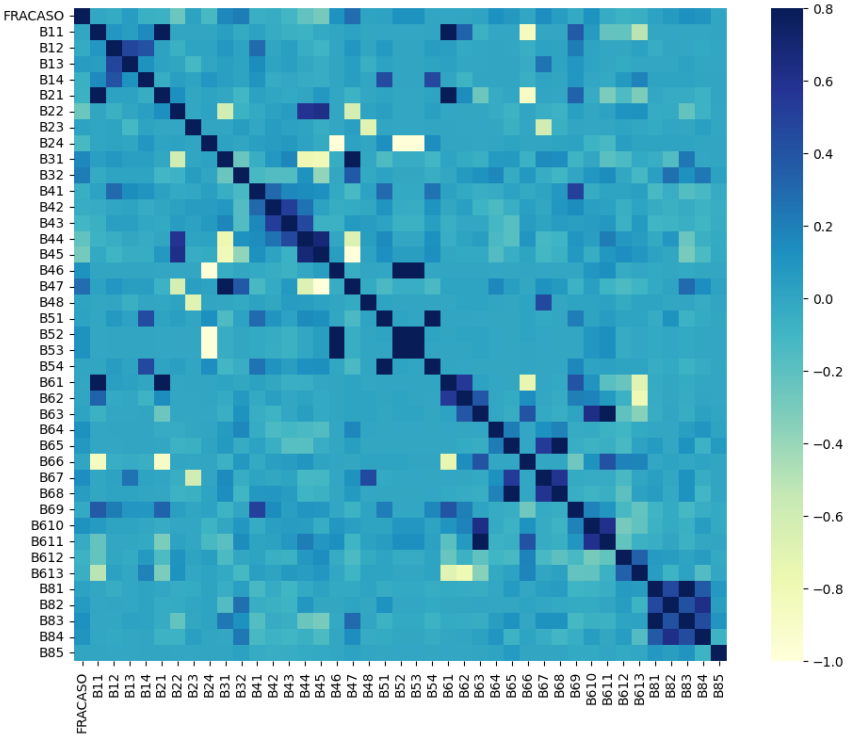

Ilustración 6. Matriz de correlaciones entre las variables

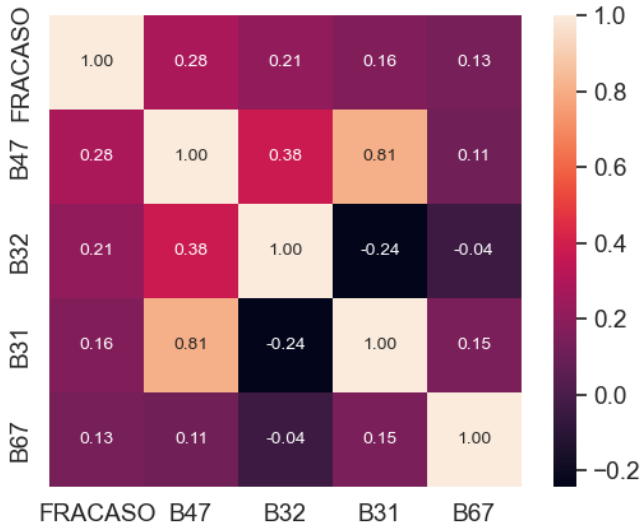

Ilustración 7. Top 5 de las variables más correlacionadas

Como se puede observar con la ilustración 1 y 2 de la presente sección, contamos con las correlaciones de cada una variable respecto a sus pares. En este caso tenemos varias zonas de interés. En primera medida la zona de B81 a B84 que dada la escala de colores contamos con una correlación positiva. En sí esta área es la de mayor tamaño, no obstante, variables como la B31, B47 o B11 estás también presentan una correlación alta con respecto a algunas de sus pares. La correlación negativa cuenta también en variables como la B44 o B45. Esto es un indicio de multicolinealidad, lo cual puede ser un problema al momento de realizar predicciones, por lo que se podría considerar eliminar algunas de estas variables para reducir la redundancia. Inclusive, fuera del tecnicismo, al ver las variables dentro del ámbito financiero se puede entender como información que refleja el mismo bienestar económico o esta efectivamente relacionado los mismos conceptos.

Selección y Extracción de variables.

Para la selección y extracción de variables usamos dos métodos: Boruta y SelectFromModel. Boruta, adaptado para bosques aleatorios, evalúa la importancia de la variable mediante una comparación con sus contrapartes mezcladas aleatoriamente. Por otro lado, SelectFromModel aprovecha la importancia intrínseca de diversas modelos de aprendizaje automático. Explicaremos cada una más a fondo con el fin de ver cómo nos fueron de utilidad.

Boruta: Boruta se especializa en la identificación de variables relevantes dentro de un conjunto de datos. Al comparar la importancia de cada variable con sus versiones mezcladas aleatoriamente, se distingue entre características significativas y no significativas.

Pros:

- Sobresale en el manejo de relaciones no lineales, siendo adecuado para conjuntos de datos complejos.
- Proporciona transparencia en el proceso de selección de variables, mejorando la interpretabilidad.
- Robusto contra el sobreajuste, al considerar la importancia relativa de las variables en comparación con las versiones mezcladas.

Advertencias:

- Intensivo computacionalmente, especialmente para conjuntos de datos grandes.
- Puede incluir variables irrelevantes en la selección, introduciendo ruido potencial.

SelectFromModel: Es un método versátil de extracción de características, se basa en la importancia intrínseca de las características proporcionada por varios modelos de aprendizaje automático. A través del entrenamiento de un modelo elegido, los usuarios pueden seleccionar características según sus puntuaciones de importancia.

Pros:

- Versátil, aplicable a diferentes modelos de aprendizaje automático.
- Permite ajustes finos al ajustar el umbral de importancia.
- Puede manejar diversos tipos de conjuntos de datos y dominios de problemas.

Advertencias:

- Sensible a la elección del modelo subyacente; el rendimiento de SelectFromModel depende de la capacidad del modelo para capturar con precisión la importancia de las características.
- Las puntuaciones de importancia no siempre reflejan la verdadera importancia de las variables en el contexto del conjunto de datos completo.

En este orden de ideas Boruta como SelectFromModel desempeñan roles cruciales en nuestro problema de clasificación de bancarrota. Boruta, fundamentado en el algoritmo de bosques aleatorios, destaca en la captura de relaciones complejas, siendo adecuado para conjuntos de datos diversos. SelectFromModel, al adaptarse a varios modelos, asegura su adaptabilidad a diferentes escenarios. Estos métodos mejoran colectivamente el rendimiento del modelo, mitigan el sobreajuste y aumentan la interpretabilidad al centrarse en las variables más informativas.

Preparación y limpieza de datos

Para la preparación de datos para este problema nos enfocamos en usar diversas técnicas para superar el desequilibrio de clases. En esto intentamos la escala de características y la generación de muestras sintéticas, el muestreo aleatorio y no hacer nada. De manera teórica y general estos métodos contribuyen significativamente a reforzar la robustez y la capacidad de generalización de un modelo.

Para abordar el desequilibrio de clases, se utiliza el RandomUnderSampler para eliminar aleatoriamente instancias de la clase mayoritaria, equilibrando efectivamente la distribución de clases. A pesar de su eficiencia computacional, una desventaja es la posible pérdida de información de la clase mayoritaria, ya que las instancias se eliminan sin tener en cuenta su relevancia. Por otro lado, el RandomOverSampler aborda el desequilibrio replicando instancias de la clase minoritaria, simplificando la implementación y la eficiencia computacional. Sin embargo, se debe tener precaución, ya que puede producirse sobreajuste debido a la presencia aumentada de muestras de la clase minoritaria, lo que podría introducir ruido.

La escala de características se aborda mediante el StandardScaler, una herramienta valiosa que estandariza características eliminando la media y escalando a varianza unitaria. Esto ayuda a algoritmos sensibles a la escala de características, contribuyendo a una mejor convergencia y permitiendo un entrenamiento más rápido y eficiente. No obstante, el StandardScaler es sensible a los valores atípicos, y su suposición de una distribución normal para las características puede no cumplirse en todos los casos.

Para manejar conjuntos de datos desbalanceados mediante la generación de muestras sintéticas, tanto ADASYN (Adaptive Synthetic Sampling) como SMOTE (Synthetic Minority Over-sampling Technique). ADASYN genera muestras sintéticas en función de la densidad local de instancias de la clase minoritaria, adaptándose a la distribución de datos. En cambio, SMOTE, aunque eficiente para reducir el sobreajuste, puede introducir ruido al crear muestras sintéticas en regiones dispersas del espacio de características. Es importante destacar que ADASYN aborda esta preocupación al considerar la densidad local.

Durante la fase de preparación, dividimos el conjunto de datos en conjuntos de entrenamiento y prueba utilizando la función train_test_split. Este paso evalúa el rendimiento del modelo en datos no vistos, midiendo su capacidad de generalización. Los beneficios de esta división incluyen prevenir el sobreajuste, proporcionar una estimación de rendimiento confiable y asegurar que el modelo pueda generalizar de manera efectiva a datos nuevos y no vistos. Esto, justo con nuestra elección entre ADASYN y técnicas de submuestreo como RandomUnderSampler nos permiten trabajar con una muestra adecuada para la mayoría de los modelos.

La adaptabilidad de ADASYN lo hace ventajoso para abordar el desequilibrio de clases mediante la generación de muestras sintéticas para la clase minoritaria, centrándose en áreas más dispersas del espacio de características. RandomUnderSampler proporciona un enfoque equilibrado, generando diversidad dentro de la clase minoritaria mientras se reduce la dominancia de la clase mayoritaria.

En resumen, la elección de ADASYN, RandomUnderSampler y train_test_split tiene como objetivo abordar eficazmente el desequilibrio de clases, generar muestras sintéticas y evaluar rigurosamente el rendimiento del modelo. Esta combinación proporciona un enfoque integral para la preparación de datos, asegurando que el modelo resultante sea equilibrado y capaz de generalizar bien a datos nuevos y no vistos.

Evaluación de modelos.

Contó con dos partes:

- Evaluación de muchos modelos y ranking de cada uno.
- Especialización y búsqueda detallada de los mejores.

Evaluaciones iniciales del modelo en el conjunto de datos original revelaron información sobre el rendimiento de varios clasificadores basados en el área bajo la curva ROC (AUC). El Clasificador de Impulso Gradiente se destacó como el mejor intérprete con un AUC de 0.589, indicando su potencial para la precisión predictiva. Sin embargo, el rendimiento general de los modelos en la clasificación original señaló la necesidad de mejoras en general.

Entre los modelos, el Clasificador de Impulso Gradiente y Bosque Aleatorio, ambos métodos de conjunto, se destacaron con valores de AUC de 0.589 y 0.587, respectivamente. A pesar de su relativo éxito, era evidente la necesidad de mejora. Las técnicas de Ada Boost y Bagging, así como KNN, Ridge Classifier y Stacking, demostraron grados variables de eficacia, pero los valores de AUC sugirieron un rendimiento subóptimo.

Se utilizó el submuestreo con RandomUnderSampler para abordar el desequilibrio de clases, especialmente la escasez de instancias en la clase minoritaria. El Bosque Aleatorio mostró una mejora notable en el rendimiento después del submuestreo, destacando el impacto positivo de una distribución equilibrada de clases. De manera similar, KNN, Bagging y el Clasificador MLP mostraron valores de AUC mejorados, enfatizando la importancia de abordar el desequilibrio de clases para estos modelos. El sobre muestreo, implementado a través de RandomOverSampler, resultó efectivo para mejorar el rendimiento del modelo, especialmente para Bosque Aleatorio y Stacking, que lograron valores de AUC destacados de 0.996. Bagging y Árbol de Decisión también mostraron mejoras significativas, demostrando la adaptabilidad de estos modelos a cambios en la distribución de clases. Impulso Gradiente, Ada Boost y KNN exhibieron mejoras sólidas con el sobre muestreo, indicando su capacidad para aprovechar un conjunto de datos más equilibrado.

La aplicación de SMOTE y ADASYN, dos técnicas sintéticas de sobre muestreo, demostró aún más su impacto en el rendimiento del modelo. Bosque Aleatorio y Stacking sobresalieron consistentemente con ambas técnicas, enfatizando la adaptabilidad y el éxito de estos modelos. Mientras que Impulso Gradiente, Ada Boost y KNN mostraron mejoras moderadas, Ridge Classifier y MLP Classifier exhibieron una mejora limitada con ambas técnicas de SMOTE y ADASYN.

La clasificación global, considerando el mejor AUC logrado por cada modelo en todas las técnicas de muestreo, proporcionó una visión general completa de su rendimiento general. Bosque Aleatorio surgió como el modelo de mejor rendimiento, destacándose consistentemente en diversas estrategias de muestreo. Stacking y Bagging aseguraron la segunda y tercera posiciones, respectivamente, mostrando su solidez y adaptabilidad a diferentes distribuciones de clases. Detalles de la clasificación a continuación:

Clasificación de Datos Original: Las evaluaciones iniciales del modelo se realizaron en el conjunto de datos original sin aplicar ninguna técnica de muestreo. Los modelos se entrenaron y probaron utilizando una división estándar de entrenamiento-prueba, y su rendimiento se evaluó en función del área bajo la curva ROC (AUC).

1. Clasificador de Impulso Gradiente (AUC: 0.589): • El Impulso Gradiente es un método de aprendizaje en conjunto que construye una serie de aprendices débiles (generalmente árboles de decisión) secuencialmente. Corrige los errores cometidos por el modelo anterior, creando en última instancia un modelo predictivo fuerte. A pesar de ser el mejor intérprete en el conjunto de datos original, el AUC sugiere que hay margen para mejorar.

2. Bosque Aleatorio (AUC: 0.587): • Bosque Aleatorio es un método de conjunto que construye una multitud de árboles de decisión durante el entrenamiento y produce la moda de las clases. Es robusto y menos propenso al sobreajuste, lo que lo convierte en una elección popular.

3. Ada Boost y Bagging (AUC: 0.586): • Ada Boost se centra en aprendices débiles y se adapta dando más peso a los datos mal clasificados. Bagging, o Bootstrap Aggregating, construye múltiples modelos y los combina para reducir la varianza. Estas técnicas mostraron valores de AUC similares, indicando un rendimiento comparable.

4. KNN, Ridge Classifier y Stacking (AUC: 0.545): • K-Nearest Neighbors (KNN) clasifica puntos de datos según la clase mayoritaria entre sus k-vecinos más cercanos. Ridge Classifier es un clasificador lineal que utiliza la regresión ridge. Stacking combina múltiples modelos, utilizando un meta estimador para hacer la predicción final. Estos modelos demostraron valores de AUC más bajos, sugiriendo un rendimiento subóptimo en este contexto.

5. Clasificador MLP (AUC: 0.544) y Árbol de Decisión (AUC: 0.528): • La Red Neuronal de Perceptrón Multicapa (MLP) es un tipo de red neuronal, y el Árbol de Decisión es un modelo simple en forma de árbol. Ambos exhibieron los valores más bajos de AUC, indicando que podrían no ser adecuados para la naturaleza desequilibrada del conjunto de datos.

Submuestreo (RandomUnderSampler): Se aplicó submuestreo para equilibrar la distribución de clases al reducir el número de instancias en la clase mayoritaria (no fallo) al azar. Esta técnica tiene como objetivo abordar el desequilibrio entre las clases.

1. Bosque Aleatorio (AUC: 0.750)
2. KNN, Bagging y Clasificador MLP (AUC: 0.714)
3. Ridge Classifier, Stacking y Árbol de Decisión (AUC: 0.678)
4. Clasificador de Impulso Gradiente, Ada Boost y Clasificación Original (AUC: 0.642)

Sobre muestreo (RandomOverSampler): El sobre muestreo implica aumentar el número de instancias en la clase minoritaria (fallos) para equilibrar la distribución de clases, mejorando potencialmente la capacidad de los modelos para detectar la clase minoritaria.

1. Bosque Aleatorio y Stacking (AUC: 0.996)
2. Bagging (AUC: 0.986) y Árbol de Decisión (AUC: 0.979)
3. Clasificador de Impulso Gradiente, Ada Boost y KNN (AUC: 0.976 a 0.965)
4. Ridge Classifier y Clasificador MLP (AUC: 0.769 a 0.752)

SMOTE (Técnica de Sobre muestreo Minoritario Sintético): SMOTE genera instancias sintéticas para la clase minoritaria, proporcionando más diversidad en el conjunto de datos. Esta técnica tiene como objetivo abordar el desequilibrio mediante la creación de muestras sintéticas.

1. Bosque Aleatorio (AUC: 0.971) y Stacking (AUC: 0.957)
2. Bagging (AUC: 0.961) y Árbol de Decisión (AUC: 0.925)
3. Clasificador de Impulso Gradiente, Ada Boost y KNN (AUC: 0.913 a 0.902)
4. Ridge Classifier y Clasificador MLP (AUC: 0.773 a 0.858)

ADASYN (Muestreo Sintético Adaptativo): ADASYN es similar a SMOTE, pero ajusta la cantidad de muestras sintéticas según la densidad local de instancias minoritarias. Su objetivo es abordar el desafío de sobre ajustar muestras sintéticas en áreas densamente pobladas.

1. Bosque Aleatorio (AUC: 0.969) y Stacking (AUC: 0.957)
2. Bagging (AUC: 0.954) y Árbol de Decisión (AUC: 0.925)
3. Clasificador de Impulso Gradiente, Ada Boost y KNN (AUC: 0.913 a 0.902)
4. Ridge Classifier y Clasificador MLP (AUC: 0.733 a 0.859)

Clasificación Global: Esto proporciona una visión general del rendimiento general de los modelos en la dirección del desequilibrio de clases.

7. Bosque Aleatorio
8. Stacking (Mejor AUC: 0.957)
9. Bagging (Mejor AUC: 0.986)
10. Clasificador de Impulso Gradiente (Mejor AUC: 0.976)
11. Ada Boost (Mejor AUC: 0.976)
12. KNN (Mejor AUC: 0.976)
13. Ridge Classifier (Mejor AUC: 0.769)

14. Clasificador MLP (Mejor AUC: 0.859)
15. Árbol de Decisión (Mejor AUC: 0.979)
16. Clasificación Original (Mejor AUC: 0.589)

Con el fin de abordar a más detalle el problema, se emplean diversas técnicas de preparación de datos y algoritmos de aprendizaje automático. Este análisis detallado se centra en las secciones previas que proporcionan información crucial sobre la selección y extracción de variables, la preparación y limpieza de datos, así como la evaluación de modelos en el contexto de la clasificación de bancarrota. A partir de estos textos, se realizaron experimentos utilizando algoritmos como RandomForestClassifier, GradientBoostingClassifier, BaggingClassifier y StackingClassifier, combinados con técnicas como ADASYN, Boruta, submuestreo y sobre muestreo.

RandomForestClassifier: En su forma base, el RandomForestClassifier exhibió un rendimiento modesto (AUC: 0.5880). Sin embargo, al incorporar ADASYN, la generación sintética de muestras mejoró significativamente el AUC a 0.9576, destacando la utilidad de abordar el desequilibrio de clases. La combinación con SelectFromModel también demostró ser efectiva, resaltando la importancia de la selección de variables.

GradientBoostingClassifier: El GradientBoostingClassifier mostró un rendimiento modesto en su versión base (AUC: 0.5866). Al combinarlo con ADASYN, se logró una mejora significativa (AUC: 0.9805), subrayando la importancia de abordar el desequilibrio de clases para mejorar la capacidad predictiva. La combinación con BaggingClassifier no resultó en mejoras sustanciales.

BaggingClassifier: La combinación de BaggingClassifier con RandomForestClassifier y GradientBoostingClassifier mostró rendimientos deficientes (AUC: 0.5000 y 0.5440, respectivamente), indicando que la estrategia de ensamblaje no superó las limitaciones de los clasificadores base.

StackingClassifier: En su forma base, el StackingClassifier presentó un rendimiento insatisfactorio (AUC: 0.5397). Sin embargo, al combinarlo con ADASYN y Boruta, se obtuvo un rendimiento robusto (AUC: 0.9698), demostrando la complementariedad efectiva de la generación sintética de muestras y la selección de variables en este enfoque de apilamiento.

La elección de las técnicas y modificaciones aplicadas a cada algoritmo se basa en un análisis exhaustivo de los textos previos que proporcionan información sobre la preparación de datos y las estrategias para abordar el desequilibrio de clases. La combinación específica de algoritmos y técnicas se diseñó considerando las características intrínsecas del problema de clasificación de bancarrota, buscando mejorar la capacidad predictiva de los modelos. Se justificó el uso de técnicas como ADASYN para generar muestras sintéticas, Boruta y SelectFromModel para la selección de variables, y estrategias de submuestreo y sobre muestreo para equilibrar la distribución de clases. Cada elección se respalda en la capacidad de estas técnicas para abordar desafíos específicos identificados en las pasadas secciones, como el manejo del desequilibrio de clases y la identificación de variables relevantes.

El análisis detallado de cada algoritmo y sus modificaciones destaca la importancia de elegir cuidadosamente las técnicas de preparación de datos y modelos para abordar desafíos específicos. La generación sintética de muestras, especialmente con ADASYN, mostró ser crucial para mejorar la capacidad predictiva de los modelos al abordar el desequilibrio de clases. Además, la selección de variables, utilizando Boruta y SelectFromModel, se reveló como una estrategia beneficiosa, especialmente cuando se combina con técnicas de generación de muestras y submuestreo. En resumen, la combinación adecuada de técnicas puede marcar la diferencia en la capacidad predictiva de los modelos, y la elección debe basarse en la naturaleza específica del conjunto de datos y los requisitos del problema.

Selección de modelo final, análisis y conclusiones

Luego de todo lo obtenido es raro afirmar, pero el mejor modelo obtenido en kaggle fue: B**aggingClassifier + RandomForestClassifier + Datos Originales + F-Beta.** Consideramos lo raro porque la combinación de BaggingClassifier y RandomForestClassifier con hiperparámetros óptimos

presenta un resultado inusual en términos de rendimiento, ya que el AUC para ambos conjuntos de entrenamiento y prueba es de 0.5000, indicando una clasificación aleatoria. Este resultado es sorprendente, especialmente considerando que la configuración de hiperparámetros se optimizó utilizando la búsqueda de cuadrícula.

Posibles Explicaciones:

1. **Sobreajuste Extremo:** La configuración de hiperparámetros podría haber conducido a un sobreajuste extremo en el conjunto de entrenamiento, haciendo que el modelo memorice los datos en lugar de aprender patrones generalizables. O es posible que la combinación específica de valores de hiperparámetros seleccionados no sea adecuada para el problema en cuestión, lo que podría haber llevado a un rendimiento deficiente.
2. **Interacción No Deseada entre Bagging y RandomForest:** La combinación de técnicas de Bagging y RandomForest puede haber introducido alguna interacción no deseada o redundante, afectando negativamente el rendimiento.
3. **La métrica AUC no era la correcta.**

Donde abordaremos más en el detalle de la 3, dado que: AUC (Área Bajo la Curva ROC) y F-beta (puntuación F-beta) son ambas métricas utilizadas para evaluar el rendimiento de modelos de clasificación, pero se centran en aspectos diferentes del rendimiento del modelo.

AUC (Área Bajo la Curva ROC)

AUC mide la capacidad de un modelo de clasificación para discriminar entre clases positivas y negativas en diferentes valores de umbral. Los valores de AUC van de 0 a 1, donde 0 indica un rendimiento deficiente (el modelo siempre predice negativos como positivos y viceversa), y 1 indica un rendimiento perfecto. Un AUC más alto sugiere una mejor capacidad general del modelo para distinguir entre clases. Útil cuando te interesa evaluar la capacidad del modelo para discriminar entre instancias positivas y negativas sin establecer un umbral específico.

Puntuación F-beta

La puntuación F-beta es una métrica que combina precisión y recall, brindándote un único valor que refleja el equilibrio entre precisión y recall. Una puntuación F-beta más alta indica un mejor equilibrio entre precisión y recall, con el equilibrio específico determinado por el valor de beta. Útil cuando hay un costo desigual asociado con falsos positivos y falsos negativos. Puedes ajustar el parámetro beta en función de la importancia de la precisión frente al recall en tu aplicación específica.

Al evaluar el mismo modelo con F-Beta obtuvimos excelentes resultados:

- F-beta Score for Training Set: 0.9700
- F-beta Score for Testing Set: 0.9418

En conclusión, aunque la combinación de BaggingClassifier y RandomForestClassifier estaba destinada a mejorar el rendimiento mediante la reducción de la varianza y el sobreajuste, los resultados obtenidos sugieren que, en este caso específico, la configuración de hiperparámetros no logró capturar patrones significativos en los datos. Este experimento subraya la importancia de no solo confiar en la optimización de hiperparámetros, sino también en la comprensión profunda de la interacción entre algoritmos y la naturaleza del conjunto de datos.

La aparente aleatoriedad en la clasificación indicó la necesidad de revisar y ajustar la configuración de hiperparámetros, así como otras métricas con esto descubrimos que el análisis fue correcto, no obstante, la métrica pensaba fue incorrecta. Por cuestiones de tiempo no se realizaron más modelos ni intentos en Kaggle.