Inteligencia Artificial para las Ciencias e Ingenierías
Proyecto de semestre - Informe final

Responsables

Juan Esteban Gutiérrez Zuluaga - CC. 1152225177 - Ingeniería de Sistemas

Andrés Quintero Bedoya - CC. 1216727950 - Ingeniería de Sistemas

Andrés Felipe Grajales Acevedo - CC. 1037640035 - Ingeniería Civil

Medellín - Antioquia

2022

Introducción

Descripción del problema

La empresa Taiwan Economic Journal, la cual es la empresa local de información financiera más grande de Taiwán proporciona a las empresas la información necesaria para el análisis fundamental del mercado financiero, servicios y soluciones de consultoría sobre la gestión del riesgo crediticio/mercado, la valoración de activos y el análisis de inversiones, en base a datos tomados de esta empresa los cuales van desde el año 1999 al 2009. Se desea generar un modelo predictivo acerca de cuándo una empresa estaría en quiebra, teniendo presente que la quiebra de la empresa se definió con base en las regulaciones comerciales de la bolsa de valores de Taiwán, para esto se tomarán las bases de datos suministradas en la competencia de Kaggle y se les hará el debido análisis y organización para posteriormente utilizar las métricas requeridas con la intención de obtener un porcentaje de acierto alto que de una visión sobre si una empresa puede entrar en bancarrota o no.

Dataset

El dataset empleado es Company Bankruptcy Prediction (https://www.kaggle.com/datasets/fedesoriano/company-bankruptcy-prediction?datasetId=1111894&sortBy=voteCount) el cual consta de 6.819 instancias y 96 columnas.

Métricas de desempeño

Comprendiendo que el problema que se está trabajando es de clasificación binaria, se toman en cuenta métricas de desempeño que permitan un entendimiento del comportamiento del modelo. Las métricas definidas para medir el desempeño del modelo son:

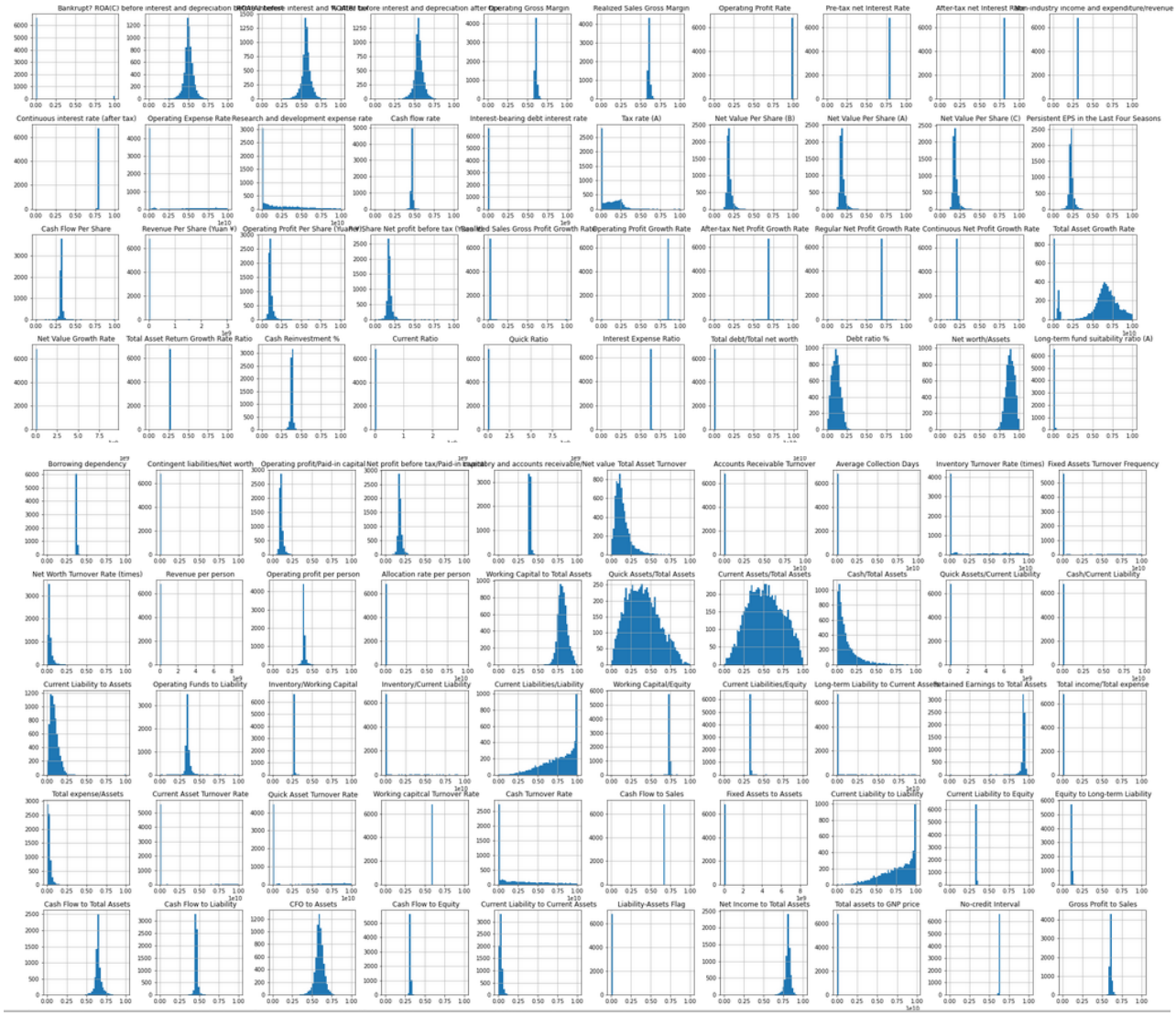
- Accuracy
- Matriz de confusión
- Precision
- Recall
- F1
- Curvas ROC (AUC)

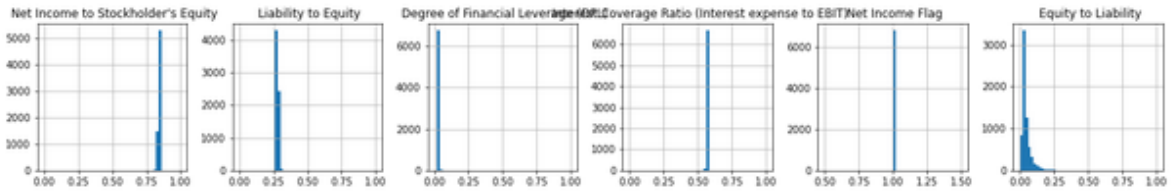
Métricas del negocio

La bancarrota es un estado de un negocio que tiene impacto a nivel empresarial y a nivel de la economía global. La predicción de la bancarrota es muy importante en tareas relacionadas a instituciones financieras. El objetivo del trabajo es realizar predicciones sobre una empresa o negocio que puede caer en bancarrota. Al realizar un análisis de predicción de bancarrota de las compañías de forma efectiva y eficiente se mejoran las decisiones de préstamos por parte de la organización.

Exploración descriptiva del dataset

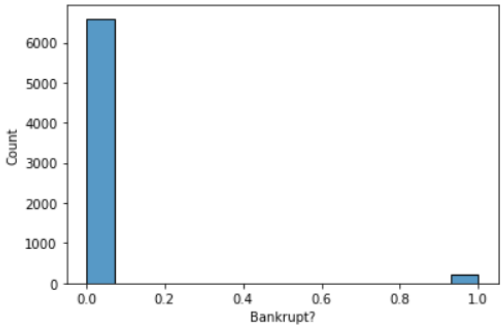
En el archivo *01 - Data Simulation.ipynb* es posible evidenciar las distribuciones que presentan los datos:



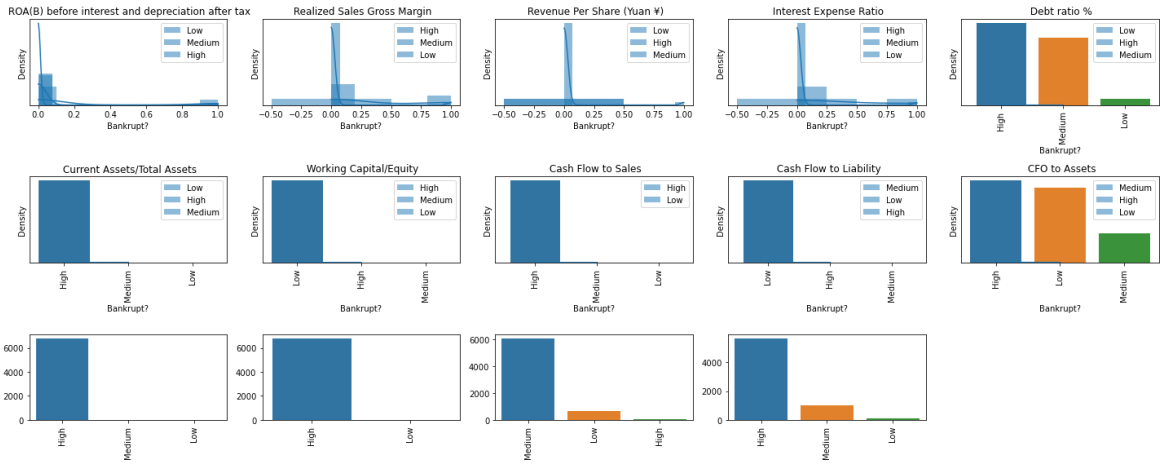


En el notebook *02 - Data Exploration.ipynb* es posible evidenciar que a partir del archivo *raw_data.parquet* se realiza una exploración de datos para conocer los datos, identificar patrones en los datos y graficarlos, y se encontró lo siguiente:

- **Tamaño del dataset:** El tamaño del dataset es (6819, 96).
- **Valores nulos:** Las columnas *Operating Gross Margin*, *Total Asset Return Growth Rate Ratio*, *Accounts Receivable Turnover* tienen 501 valores nulos.
- **Variable objetivo:** Se grafica variable de salida, se encuentra que el valor 0 (empresa no entrará en bancarrota) es el más presente en el dataset.



- **Tipos de datos:** Los tipos de datos que están presentes son float64 en su mayoría, algunos int64 y otros object en las variables que se convirtieron a categóricas.
- **Inspección de datos numéricos:** Se calcula el promedio, la desviación estándar, valor mínimo, máximo y percentiles de las columnas numéricas además se graficó la matriz de correlaciones.
- **Inspección de variables categóricas:** En las variables transformadas a categóricas se realizó un conteo de las filas correspondientes a cada categoría (High, Medium y Low). Se incluyen también histogramas para encontrar relaciones entre la variable de salida (Bankrupt?) y cada una de las variables categóricas:



Iteraciones de desarrollo

Todas las iteraciones realizadas se hicieron con base en el archivo *clean_data.parquet,* archivo de salida generado tras haber realizado un preprocesamiento de datos previo (ver notebook *03 - Data Preprocessing.ipynb*) que consistió en lo siguiente:

- **Llenado de datos faltantes:** Debido a que las 3 columnas con datos faltantes son numéricas, el proceso se realiza con la media. Esto a través de la función fillna().

```
list_na = df.columns[df.isna().any()].tolist()
list_na
```

```
[' Operating Gross Margin',
 ' Total Asset Return Growth Rate Ratio',
 ' Accounts Receivable Turnover']
```

```
for i in list_na:
    df[i].fillna(df[i].mean(),inplace=True)
```

- **Conversión variables categóricas a numéricas:** Se utiliza la estrategia One Hot Encoding, la cual consiste en la creación de una una columna para cada valor distinto que exista en la característica que estamos codificando y, para cada registro se marca con un 1 la columna a la que pertenezca dicho registro y se dejan las demás con un valor de 0. Esto es posible gracias a la función get_dummies():

```
for i in list_cat:
    df_aux = pd.get_dummies(df[i], prefix=i)
    df.drop(i,inplace=True,axis=1)
    df = df.join(df_aux)
```

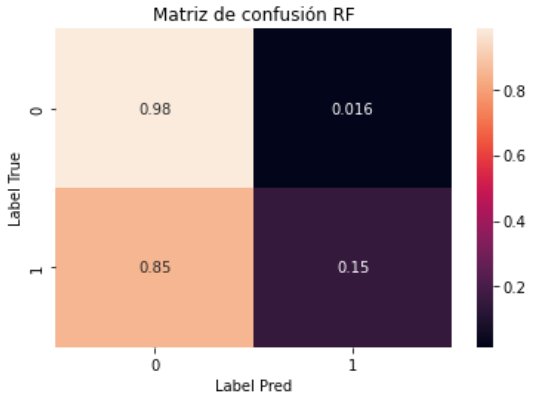
Teniendo esto en cuenta, se enuncian las distintas iteraciones:

Iteración 1: Modelos sin SMOTE

En la primera iteración se llevaron a cabo experimentos dividiendo los datos mediante StratifiedKFold, esto con el fin de que los experimentos se realicen conservando la estratificación de los datos, lo cual es posible visualizarlo en el archivo *04 - Models without SMOTE.ipynb* Los modelos probados fueron los siguientes:

RF (Random Forest)

Para este modelo, se realizaron corridas con árboles [5,10,20,50,100, 150] y variables para la selección del mejor umbral [5,10,15,20,25] a través de combinaciones entre sí. Por ejemplo, con un número de árboles 50 y variables para selección de mejor umbral 5 se obtuvo una eficiencia (accuracy) de prueba de 96,3485%, y en cuanto a la matriz de confusión se obtuvo lo siguiente:

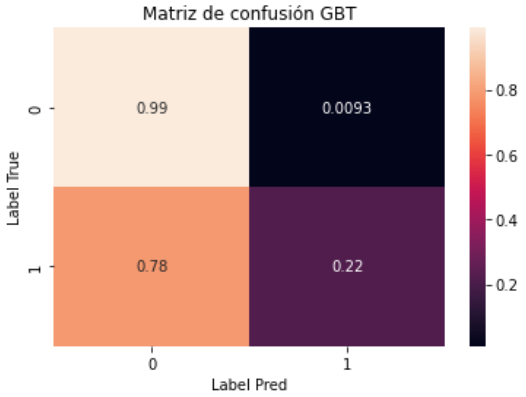


Para la matriz de confusión se emplean porcentajes para visualizar mejor los resultados de las predicciones del modelo. En cuanto a los verdaderos negativos (TN) se observa que el modelo clasifica correctamente la clase 0 (la empresa no entrará en bancarrota) con el 98% de los datos pertenecientes a esa clase. A nivel de verdaderos positivos (TP) no se obtiene un resultado muy favorable, ya que se observa que el modelo predice correctamente el 15% de las veces cuando una empresa sí entrará en quiebra. En los falsos negativos se obtiene una tasa del 85% mientras que en los falsos positivos se visualiza una tasa del 1.6%.

Gradient Boosting

Con la función GradientBoostingClassifier de sklearn se varía el n_estimators entre los valores [20,50,100,200,300], por ejemplo, empleando un valor de n_estimators en 300,

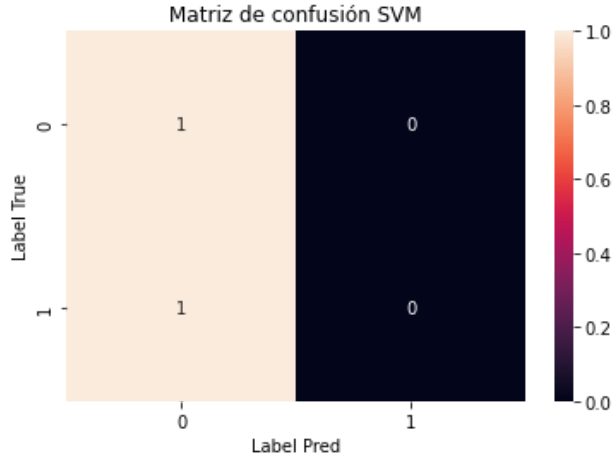
obtenemos una eficiencia de prueba de 96.5221% y con relación a la matriz de confusión se observa lo siguiente:



En cuanto a los verdaderos negativos (TN) se observa que el modelo clasifica correctamente la clase 0 (la empresa no entrará en bancarrota) con el 99% de los datos pertenecientes a esa clase. A nivel de verdaderos positivos (TP) no se obtiene un resultado muy favorable, ya que se observa que el modelo predice correctamente el 22% de las veces cuando una empresa sí entrará en quiebra. En los falsos negativos se obtiene una tasa del 78% mientras que en los falsos positivos se visualiza una tasa del 0.93%.

SVM (Support Vector Machine)

Para este caso se realizó un experimento con kernel ‘rbf’, gamma en 0.01 y C en 0.01 y se obtuvo una eficiencia de prueba de 96.7737% y la siguiente matriz de confusión:

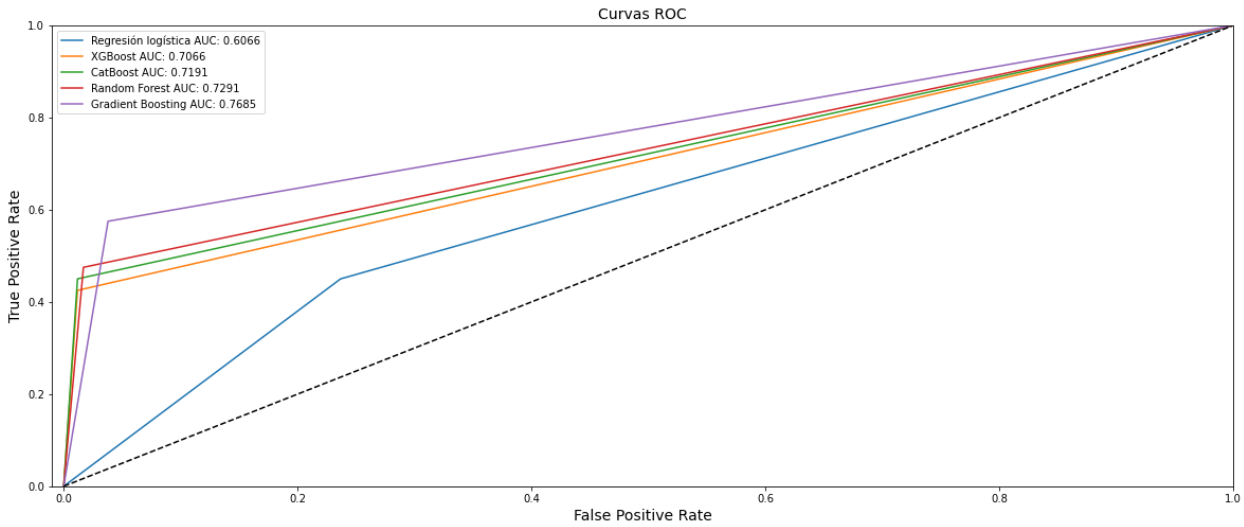


Para este caso, la matriz de confusión muestra que el 100% el modelo realizó predicciones con el valor de 0 (empresa no entrará en bancarrota), un resultado no muy esperado.

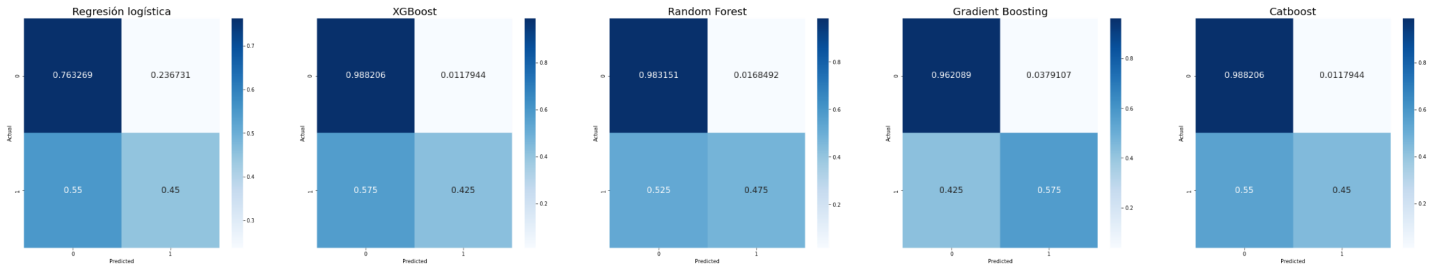
Iteración 2: Modelos con SMOTE

Esta iteración puede ser observada en el notebook *05 - Models with SMOTE.ipynb* en el cual inicialmente se llevó a cabo una división de datos de train y de test. Además, para esta se buscaba mejorar el desempeño de los modelos a través de nuevas técnicas como **SMOTE** la cual es una técnica estadística de sobremuestreo de minorías sintéticas para aumentar el número de casos de un conjunto de datos de forma equilibrada [1]. Además de esto también se dividieron los datos mediante **StratifiedKFold** (n_splits en 5), para que los experimentos se realicen conservando la estratificación de los datos.

Posterior a esto se realizaron experimentos con los modelos regresión logística, XGBoost, Random Forest, Gradient Boosting y Catboost haciendo uso de la función **RandomizedSearchCV** para encontrar los parámetros óptimos de cada modelo mencionado. A continuación se observan las curvas ROC de los cinco modelos con parámetros óptimos:

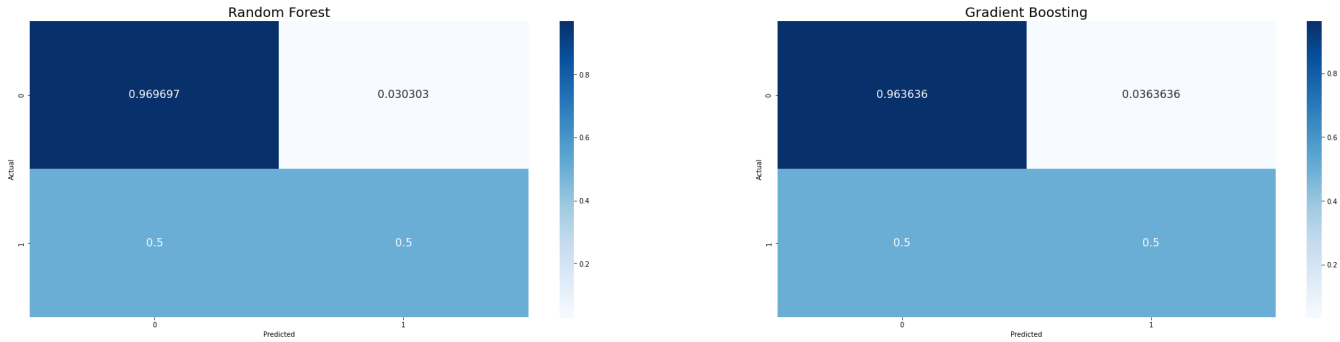


Es posible evidenciar el modelo con mejor AUC es Gradient Boosting Tree, y de segundo se encuentra Random Forest con un valor de 0.7291. Además de esto se graficaron las matrices de confusión a través del parámetro 'normalize' en true, para así observar los porcentajes y poder comparar con los resultados de la anterior iteración:



Lo que se buscaba mejorar para esta iteración eran los verdaderos positivos (TP) ya que esta era una falencia para nosotros en la anterior iteración, por lo cual, optamos por elegir los modelos Random Forest (con los parámetros 'random_state': 42, 'max_features': 'sqrt', 'criterion': 'entropy', 'class_weight': 'balanced', 'bootstrap': False) en el que se observa que el modelo predice correctamente el 47.5% de las veces cuando una empresa sí entrará en quiebra además del modelo Gradient Boosting (con los parámetros 'random_state': 42, 'n_estimators': 100, 'max_features': 'auto') con el cual se evidencia que el modelo predice correctamente el 57.5% de las veces cuando una empresa sí entrará en bancarrota.

Con esta elección, se emplean los datos de testing separados inicialmente para validar estos dos modelos, obteniéndose así las matrices de confusión:



Por último, se hace un reporte de las métricas de ambos:

Metricas de Random Forest

	precision	recall	f1-score	support
Fin.Stable	0.98	0.97	0.98	660
Fin.Unstable	0.35	0.50	0.42	22
accuracy			0.95	682
macro avg	0.67	0.73	0.70	682
weighted avg	0.96	0.95	0.96	682

Metricas de Gradient Boosting

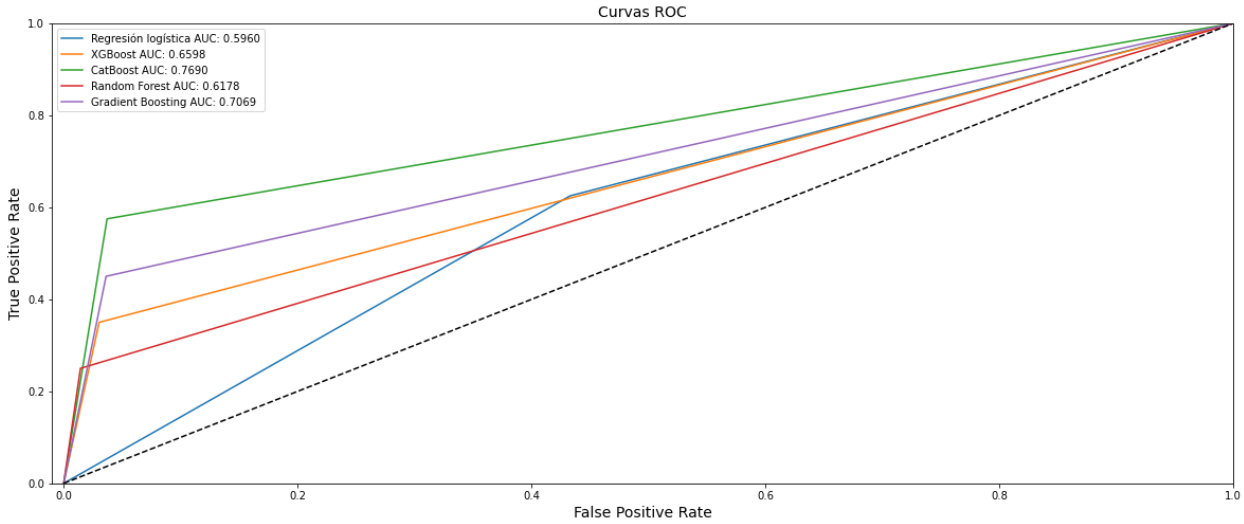
	precision	recall	f1-score	support
Fin.Stable	0.98	0.96	0.97	660
Fin.Unstable	0.31	0.50	0.39	22
accuracy			0.95	682
macro avg	0.65	0.73	0.68	682
weighted avg	0.96	0.95	0.95	682

Ambos modelos tuvieron resultados sumamente similares, sin embargo por una mínima diferencia, el modelo Random Forest con los parámetros 'random_state': 42, 'max_features': 'sqrt', 'criterion': 'entropy', 'class_weight': 'balanced', 'bootstrap': False es el mejor modelo.

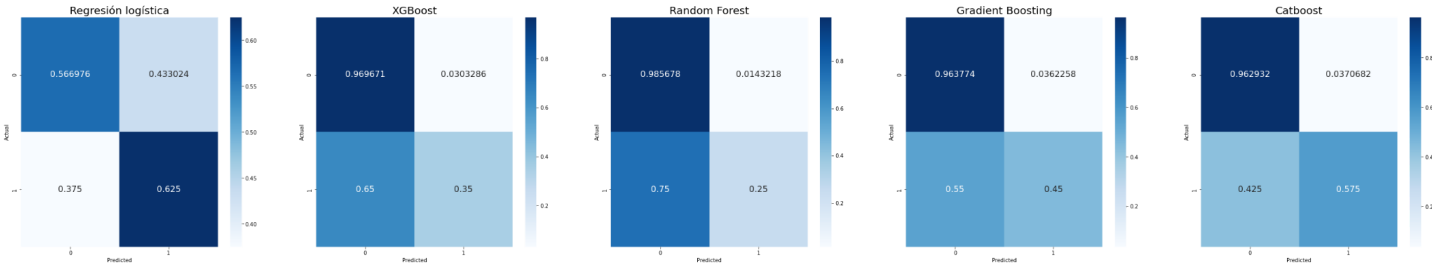
Iteración 3: Modelos con SMOTE y PCA

La presente iteración puede ser observada en el notebook *06 - Models with SMOTE and PCA.ipynb,* la iteración se realizó de la misma manera que en el notebook *05 - Models with SMOTE.ipynb* pero incluyéndose PCA (Análisis de componentes principales), el cual es un método estadístico que permite simplificar la complejidad de espacios muestrales con muchas dimensiones a la vez que conserva su información [2]. Internamente se realizaron varias corridas con diversos números de componentes (n_components) y se encontró que el número óptimo de componentes es **65** por lo cual en el notebook se observa la corrida con este número.

Se llevaron a cabo también experimentos con los modelos regresión logística, XGBoost, Random Forest, Gradient Boosting y Catboost haciendo uso de la función **RandomizedSearchCV** para encontrar los parámetros óptimos de cada modelo mencionado. A continuación se observan las curvas ROC de los cinco modelos con parámetros óptimos:

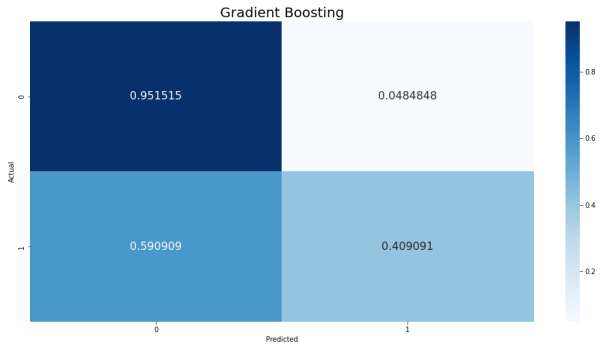


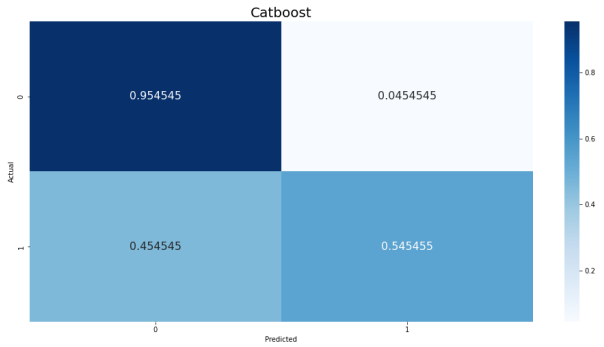
Se puede ver que el modelo con mejor AUC es CatBoost con un valor de 0.769, y de segundo se encuentra Gradient Boosting Tree con un valor de 0.7069. A continuación se muestran las matrices de confusión:



Se opta por elegir los modelos CatBoost (con los parámetros 'random_seed': 42, 'learning_rate': 0.01, 'iterations': 1000, 'eval_metric': 'F1', 'auto_class_weights': 'Balanced) en el que se observa que el modelo predice correctamente el 57.5% de las veces cuando una empresa sí entrará en quiebra además del modelo Gradient Boosting (con los parámetros 'random_state': 42, 'n_estimators': 300, 'max_features': 'log2') con el cual se encuentra que el modelo predice correctamente el 45% de las veces cuando una empresa sí entrará en bancarrota.

Con esta elección, se emplean los datos de testing separados inicialmente para validar estos dos modelos, obteniéndose así las matrices de confusión:





Por último, se hace un reporte de las métricas de ambos:

Metricas de CatBoost

	precision	recall	f1-score	support
Fin.Stable	0.98	0.95	0.97	660
Fin.Unstable	0.29	0.55	0.37	22
accuracy			0.94	682
macro avg	0.64	0.75	0.67	682
weighted avg	0.96	0.94	0.95	682

Metricas de Gradient Boosting

	precision	recall	f1-score	support
Fin.Stable	0.98	0.95	0.97	660
Fin.Unstable	0.22	0.41	0.29	22
accuracy			0.93	682
macro avg	0.60	0.68	0.63	682
weighted avg	0.96	0.93	0.94	682

Para este caso que se aplicó PCA, el modelo CatBoost con los parámetros 'random_seed': 42, 'learning_rate': 0.01, 'iterations': 1000, 'eval_metric': 'F1', 'auto_class_weights': 'Balanced es el mejor modelo.

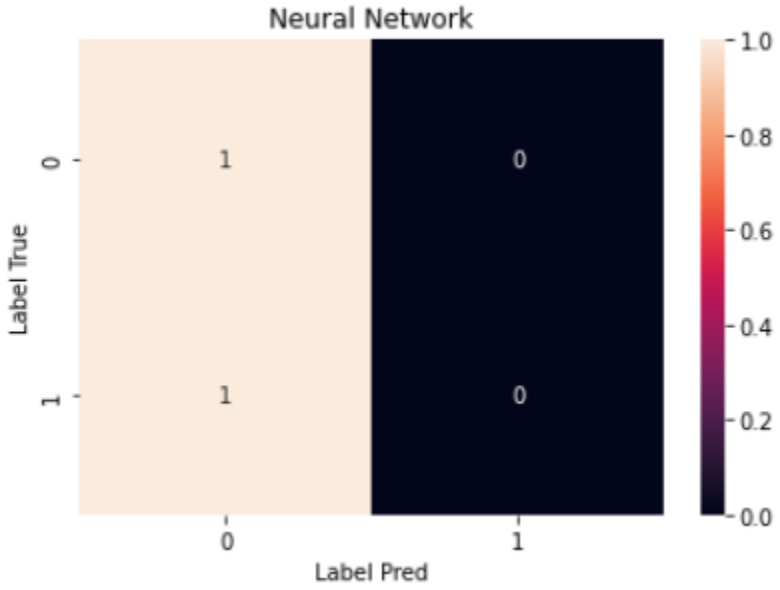
Iteración 4: Redes neuronales

En la cuarta iteración, que se puede observar en el notebook *07 - Models with Neural Network* se tiene una red neuronal con la siguiente configuración:

- Una capa densa de 128 neuronas y activación ReLu
- Una capa de Dropout de 0.3
- Una capa densa de 64 neuronas y activación ReLu
- Una capa densa de 32 neuronas y activación ReLu
- Una capa densa de 16 neuronas y activación ReLu
- Una capa de Dropout de 0.3
- Una capa densa de salida con 1 neurona y activación sigmoide.

Para esta configuración se utilizó el optimizador Nadam con un learning rate de 0.001, se utilizó como función de pérdida Categorical Cross Entropy. El entrenamiento se realizó con un batch size de 256 durante 30 épocas y un split de validación del 20%.

El resultado de la matriz de confusión es el siguiente:



Se observa un gran sobreajuste debido al desbalance de los datos, esto dando a entender que todos los datos se han clasificado en la clase mayoritaria (clase 0), mientras que en la clase minoritaria (clase 1) no se ha clasificado nada.

Retos y consideraciones de despliegue

Los retos que se tenían al principio del proyecto en cuanto al rendimiento del modelo eran:

- Tener una tasa de acierto del 85%.
- Valor de sensibilidad mayor al 80%.

- Valor de especificidad mayor al 90%.

La tasa de acierto se observa mayor al 85% debido a que los modelos están clasificando de forma desbalanceada siendo la clase mayoritaria la que nos indica que se tiene un rendimiento mucho mayor al 85%, por lo tanto no es una métrica que tenga mucha influencia a la hora de desplegar el modelo. Los valores de sensibilidad de todos los modelos son inferiores al 80% por lo tanto no se cumple con el requerimiento asociado a esta métrica que nos indica la fracción de los verdaderos positivos clasificados. Los valores de especificidad de todos los modelos son inferiores al 90% por lo tanto no se cumple con el requerimiento asociado a esta métrica que nos indica la fracción de los verdaderos negativos. Se concluye que no se pueden desplegar los modelos debido a que no se cumplieron los requerimientos mínimos para que se tenga un buen rendimiento a nivel de producción.

Conclusiones

- La mejor manera de obtener un buen modelo de machine learning es aplicando diferentes algoritmos predictivos y combinación de hiperparámetros, en este caso de clasificación binaria sobre el mismo conjunto de datos, debido a que los algoritmos más comunes como regresión logística no siempre ofrecen el mejor resultado. Análogamente es recomendable evaluar el desempeño de cada modelo con diversas métricas de validación y no tomar decisiones solo con la exactitud (accuracy), sino también analizar otras métricas como el AUC, curva ROC, precisión y matriz de confusión.

- A través de técnicas de reducción de dimensionalidad como PCA es posible obtener modelos de machine learning más simplificados incluso con un mejor rendimiento tanto en cuanto a resultados como en tiempo de ejecución, en nuestro caso inicialmente contábamos con 114 variables de entrada, que se transformaron en 65 (número de componentes).

- El data cleaning es un proceso que demanda gran parte del tiempo en un proyecto de machine learning y es un proceso crucial para poder obtener modelos con buenos resultados.

- El modelo empleado de redes neuronales utilizado no se adecúa bien a los datos y por lo tanto no es útil para el problema en cuestión, esto se puede deber a que el modelo probablemente es demasiado complejo para los datos utilizados.

- En datos que se encuentran desbalanceados, como lo es el caso del dataset empleado, es sumamente importante utilizar técnicas de sobremuestreo como

SMOTE para obtener mejores resultados. Esto se evidenció en el desarrollo del proyecto debido a que a nivel de verdaderos positivos (TP) sin SMOTE el modelo Random Forest predijo correctamente el 15% de las veces cuando una empresa sí entrará en quiebra, y con la técnica SMOTE se mejoró considerablemente los verdaderos positivos (TP) con el modelo CatBoost prediciendo correctamente el 54.54% de las veces cuando una empresa sí entrará en bancarrota.

- La realización de este estudio permitió enriquecer los conocimientos sobre gran parte de lo relacionado al machine learning, lo cual genera cierta experticia afrontar proyectos de la industria.

Referencias

[1] likebupt, «SMOTE - Azure Machine Learning». https://learn.microsoft.com/es-es/azure/machine-learning/component-reference/smote (accedido 27 de septiembre de 2022).

[2] «Análisis de Componentes Principales (Principal Component Analysis, PCA) y t-SNE». https://www.cienciadedatos.net/documentos/35_principal_component_analysis (accedido 27 de septiembre de 2022).