Integrantes

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

Progreso alcanzado

Se llevaron a cabo 4 procesos que están distribuidos en distintos notebooks, debido a que a través de los notebook se van creando nuevos archivos que se emplean en los siguientes notebooks, los datasets empleados fueron almacenados en un bucket de S3, esto con el fin de que no sea necesario descargar los datos en la máquina local y subirlos al Colab, de esta forma los notebook podrán ser reproducibles. Los distintos procesos son los que se presentan a continuación:

Simulación de datos

En el archivo *01 - Data Simulation.ipynb* se evidencia que el dataset original (6819 x 96) no contaba con variables categóricas y fue necesario realizar un procedimiento para convertir variables numéricas a categóricas. Esto se hizo a través de la función column_to_discrete() que se muestra a continuación:

```
def column_to_discrete(column_df):
    for i in range(len(column_df)):
        if column_df[i]<=0.45:
            column_df.loc[i] = "Low"
        elif column_df[i] >= 0.55:
            column_df.loc[i] = "High"
        else:
            column_df.loc[i] = "Medium"
```

Esto se llevó a cabo con 10 columnas con valores menores a 1. De esta manera se cumple con el requisito de tener al menos el 10% de las columnas categóricas.

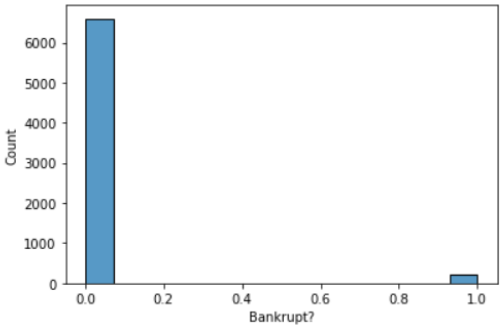
Por otro lado, como el dataset no tenía valores nulos se llevó a cabo una eliminación de 501 filas de **tres** columnas aleatorias. Por lo cual, se cumple el requisito de al menos tener un 5% de datos faltantes en al menos 3 columnas (el 7.34% de datos faltantes en 3 columnas). Una vez realizado, las siguientes columnas fueron afectadas:

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

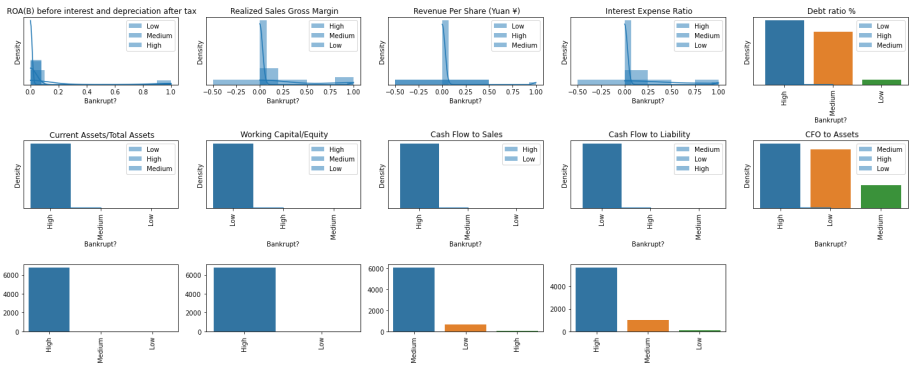
Exploración de datos

Tras haber realizado la simulación de datos se obtiene como salida el archivo *raw_data.parquet* con el cual se realiza una exploración de datos para conocer los datos, identificar patrones en los datos y graficarlos, 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:



Preprocesamiento de datos

En el archivo *03 - Data Preprocessing.ipynb* se puede observar que partiendo del archivo *raw_data.parquet,* se realiza un preprocesado de los datos. Se llevan a cabo los siguientes procesos:

- **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)
```

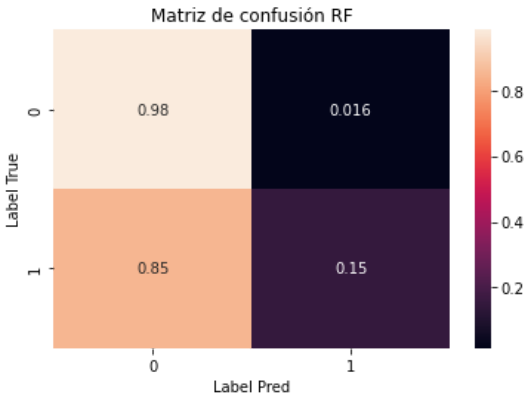
Tras el preprocesado, se generará un nuevo archivo llamado *clean_data.parquet* que se empleará para realizar los primeros experimentos con los modelos.

Modelos

Para la etapa actual del proyecto 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.ipynb. Los modelos probados son los siguientes:

RF (Random Forest)

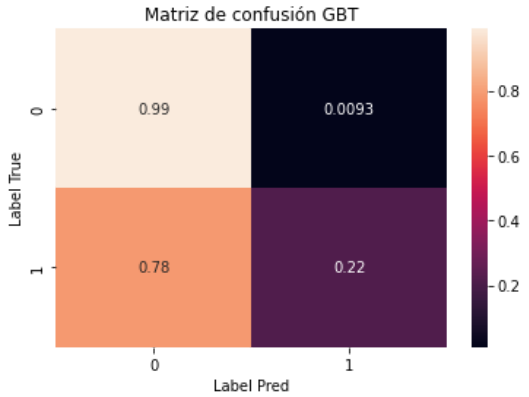
Los bosques aleatorios (random forest) son una modificación de los árboles de decisión que crean una gran colección de árboles descorrelacionados para mejorar aún más el rendimiento predictivo. Para este modelo, se realizaron corridas con árboles [5,10,20,50,100, 150] y variables para la selección del 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 Tree

Un modelo Gradient Boosting está formado por un conjunto de árboles de decisión individuales, entrenados de forma secuencial de forma que cada nuevo árbol trata de mejorar los errores de los árboles anteriores. La predicción de una nueva observación se obtiene agregando las predicciones de todos los árboles individuales que forman el modelo.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:

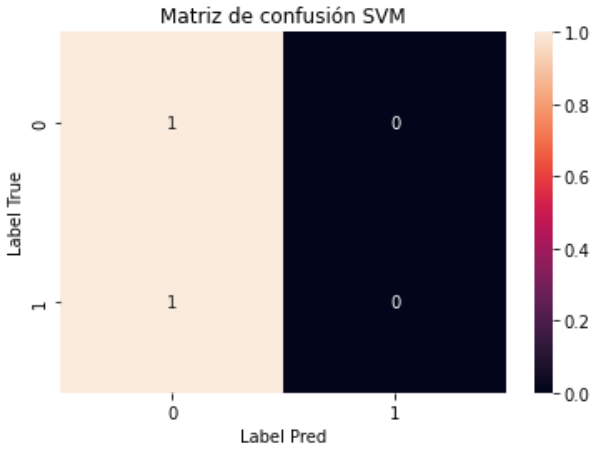


En cuanto a los verdaderos negativos (TN) se observa que el modelo clasifica correctamente la clase 0 (la empresa no entrará en bancarrota) con el 99% de los datos pertenecientes a esa clase. A nivel de verdaderos positivos (TP) no se obtiene un resultado muy favorable, ya que se

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)

El objetivo del algoritmo SVM es encontrar un hiperplano que separe de la mejor forma posible dos clases diferentes de puntos de datos. 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. Lo mejor sería probar con distintos hiperparámetros para determinar si se obtiene un mejor resultado o si definitivamente no se mejora el rendimiento del modelo, esto quiere decir que el modelo no es el apropiado para este caso.

Análisis

En general se obtienen eficiencias (accuracy) de prueba mayores a 96%, lo cual es un valor que es bastante alto sin embargo no es la mejor manera de validar el modelo ya que los datos se encuentran desbalanceados.

Los resultados de la matriz de confusión pueden deberse a que el dataset está desbalanceado y esto genera que la mayoría de predicciones generadas por el modelo sean el valor de 0, es decir que la empresa no entrará en bancarrota. Una posible solución es implementar el uso de técnicas de sobremuestreo como SMOTE lo cual se llevará a cabo para la próxima entrega.