**Metodología**

1. Decidí usar la librería **pyspark** porque me permite crear un *pipeline*  (una estructura del modelo desarrollado) y genera un código más limpio. Sin embargo, debo decir que genera unos WARNINGS del estilo: *An illegal reflective access operation has occurred* que serán omitidos en la próxima versión de la librería. Estos warnings no afecta el resultado.
2. Hago un EDA (Exploratory Data Analysis) y observo:
3. Los datos no están balanceados y por esta razón decido usar la métrica *f1score* que se usa para datos no balanceados ya que usa dos métricas adicionales: *precision, recall* las cuales me dicen qué tan exacto es la predicción de una clase específica y me dicen también la habilidad del modelo para identificar una clase. Si *f1score* tiene un valor cercano a 1 me dice que a pesar de tener los datos no balanceados el modelo lograr predecir correctamente las dos clases.
4. Creo una columna en cada dataframe, la llamo *label* donde 0 representa al dataset positivo y 1 representa al dataset negativo.
5. Estos dataframe son objetos de la librería spark y tienen métodos similares a los de pandas, de hecho se puede convertir de spark a pandas y viceversa.
6. Solo tengo en cuenta a las columnas *nomProy* y *label* porque la *codBPIN* tiene muchos valores nulos en los dos dataframe y además no me da información sobre los datos.
7. Concateno los dataframe positivo y negativo para formar uno solo.
8. Dado que los datos a analizar son texto entonces el problema a resolver es un problema de clasificación de texto y se usa Procesamiento de Lenguaje Natural procesar los datos y encontrar su equivalente númerico para cada texto. Lo hice generando 4 variables que serán introducidas en el pipeline del modelo:
9. *tokenizer: Separar el texto en partes más pequeñas: palabras, caracteres, …*
10. *stop\_words\_remover: Remover palabras que no aporta información al modelo como los pronombre Yo, artículos definidos lo, la, …*
11. *vectorizer: Transformando los tokens en vectores numéricos*
12. *idf: Nos da la importancia de cada término en un texto.*
13. Separo los datos en trainDF y testDF donde trainDF corresponde al 80% de los datos y testDF al 20%.
14. Dado que ya tenemos el equivalente numérico de cada texto y tenemos una etiqueta de 0 o 1 para cada texto entonces tenemos un problema supervisado de clasificación y por esa razón decido usar una regresión logística. El modelo lo almaceno en la variable *lr*
15. Creo la variable *pipeline*  que contendrá las cuatro variables explicadas en el punto 3 y el modelo de regresión logística del punto 4.
16. El pipeline da orden al código y se ejecuta en una sola línea.
17. Entreno el modelo con los datos de entrenamiento *trainDF*
18. Testeo el modelo con el dataset *testDF* y guardo las predicciones en la variable *predictions*
19. Evalúo en modelo con la clase *MulticlassClassificationEvaluator* de *pyspark*. Creo una instancia de esa clase y la guardo en la variable *evaluator*
20. Uso las métrica *accuracy* y *f1score* (la cuál expliqué en el punto 2.a la razón de mi escogencia de esta métrica).
21. Observo que el desempeño con la métrica *accuracy* es de *0.96357.* El desempeño con la métrica *f1score* es de 0.96461.
22. El desempeño de la métrica *f1score*  me dice que a pesar de tener datos desbalanceados no está afectando de manera significativa la predicción de cada clase (0 y 1).
23. Por último muestro una matriz de confusión para mirar los Verdaderos positivos, Verdaderos negativos, Falsos positivos y Falsos negativos.
24. Observo la distribución de datos en el dataset testDF. Me percato de que los falsos positivos son 17 en total representando el 13.6% del total de datos: 125 en total. Que es un poco alto y se podría arreglar si se balancean los datos.
25. Observo que los falsos negativos son 34 de un total de 1275 representando solo el 0.03% del total de los datos.
26. Los puntos 15 y 16 muestran que el desbalanceo de los datos afecta la *precisión* del modelo a pesar de tener un *f1score* muy bueno.
27. Realizo un oversampling de la clase con menor cantidad de datos que es la clase con etiqueta 0, es decir los datos positivos.
28. Vuelvo a ejecutar el modelo con este conjunto de datos
29. Repito los pasos del 4 al 12
30. Observo que las métricas *accuracy* y *f1score* mejoraron y tienen el mismo valor 0.984
31. Calculo nuevamente la matriz de confusión y observo que la cantidad de Falsos positivos es 0 y la cantidad de Falsos negativos es de 39 ligeramente mayor al modelo sin balancear los datos.
32. Concluyo que el modelo mejoró al balancear los datos con el oversampling.