Home Credit Default Risk

PySpark Introduction

- Apache Spark is popular in big data due to in-memory computation and parallel processing.
- MLlib, built on Spark, is a scalable Machine Learning library with high-quality algorithms and speed.
- MLlib has APIs for Java, Python, and Scala, making it suitable for Data Analysts, Engineers, and Scientists.
- MLlib includes algorithms for classification, regression, clustering, collaborative filtering, and more.

Problem Statement Introduction

- The article discusses building an end-to-end machine learning model using MLlib in PySpark.
- The dataset used is from the Home Credit Default Risk competition on Kaggle.
- The objective is to determine if loan applicants can repay their loans based on collected data.
- It's a binary classification problem with an imbalanced target label: 0 (applicants who paid back loans) and 1 (applicants who didn't).
- The distribution ratio is approximately 0.91 (applicants who repaid) to 0.09 (applicants who didn't).

1. Data Ingestion and Spark session creation

- Dataset Link: https://www.kaggle.com/c/home-credit-default-risk (https://www.kaggle.com/c/home-credit-default-risk)

In [ ]:

```python
from pyspark.sql import SparkSession
# initiate our session and read the main CSV file, then we print the #dataframe schema
spark = SparkSession.builder.appName('imbalanced_binary_classification').getOrCreate()
new_df = spark.read.csv("../Data/application_train.csv/application_train.csv", header=True, inferSchema=True)
new_df.printSchema()
```

```
root
 |-- SK_ID_CURR: integer (nullable = true)
 |-- TARGET: integer (nullable = true)
 |-- NAME_CONTRACT_TYPE: string (nullable = true)
 |-- CODE_GENDER: string (nullable = true)
 |-- FLAG_OWN_CAR: string (nullable = true)
 |-- FLAG_OWN_REALTY: string (nullable = true)
 |-- CNT_CHILDREN: integer (nullable = true)
 |-- AMT_INCOME_TOTAL: double (nullable = true)
 |-- AMT_CREDIT: double (nullable = true)
 |-- AMT_ANNUITY: double (nullable = true)
 |-- AMT_GOODS_PRICE: double (nullable = true)
 |-- NAME_TYPE_SUITE: string (nullable = true)
 |-- NAME_INCOME_TYPE: string (nullable = true)
 |-- NAME_EDUCATION_TYPE: string (nullable = true)
 |-- NAME_FAMILY_STATUS: string (nullable = true)
 |-- NAME_HOUSING_TYPE: string (nullable = true)
 |-- REGION_POPULATION_RELATIVE: double (nullable = true)
 |-- DAYS_BIRTH: integer (nullable = true)
 |-- DAYS_EMPLOYED: integer (nullable = true)
 |-- DAYS_REGISTRATION: double (nullable = true)
 |-- DAYS_ID_PUBLISH: integer (nullable = true)
 |-- OWN_CAR_AGE: double (nullable = true)
 |-- FLAG_MOBIL: integer (nullable = true)
 |-- FLAG_EMP_PHONE: integer (nullable = true)
 |-- FLAG_WORK_PHONE: integer (nullable = true)
 |-- FLAG_CONT_MOBILE: integer (nullable = true)
 |-- FLAG_PHONE: integer (nullable = true)
 |-- FLAG_EMAIL: integer (nullable = true)
 |-- OCCUPATION_TYPE: string (nullable = true)
 |-- CNT_FAM_MEMBERS: double (nullable = true)
 |-- REGION_RATING_CLIENT: integer (nullable = true)
 |-- REGION_RATING_CLIENT_W_CITY: integer (nullable = true)
 |-- WEEKDAY_APPR_PROCESS_START: string (nullable = true)
 |-- HOUR_APPR_PROCESS_START: integer (nullable = true)
 |-- REG_REGION_NOT_LIVE_REGION: integer (nullable = true)
 |-- REG_REGION_NOT_WORK_REGION: integer (nullable = true)
 |-- LIVE_REGION_NOT_WORK_REGION: integer (nullable = true)
 |-- REG_CITY_NOT_LIVE_CITY: integer (nullable = true)
 |-- REG_CITY_NOT_WORK_CITY: integer (nullable = true)
 |-- LIVE_CITY_NOT_WORK_CITY: integer (nullable = true)
```

```
|-- ORGANIZATION_TYPE: string (nullable = true)
|-- EXT_SOURCE_1: double (nullable = true)
|-- EXT_SOURCE_2: double (nullable = true)
|-- EXT_SOURCE_3: double (nullable = true)
|-- APARTMENTS_AVG: double (nullable = true)
|-- BASEMENTAREA_AVG: double (nullable = true)
|-- YEARS_BEGINEXPLUATATION_AVG: double (nullable = true)
|-- YEARS_BUILD_AVG: double (nullable = true)
|-- COMMONAREA_AVG: double (nullable = true)
|-- ELEVATORS_AVG: double (nullable = true)
|-- ENTRANCES_AVG: double (nullable = true)
|-- FLOORSMAX_AVG: double (nullable = true)
|-- FLOORSMIN_AVG: double (nullable = true)
|-- LANDAREA_AVG: double (nullable = true)
|-- LIVINGAPARTMENTS_AVG: double (nullable = true)
|-- LIVINGAREA_AVG: double (nullable = true)
|-- NONLIVINGAPARTMENTS_AVG: double (nullable = true)
|-- NONLIVINGAREA_AVG: double (nullable = true)
|-- APARTMENTS_MODE: double (nullable = true)
|-- BASEMENTAREA_MODE: double (nullable = true)
|-- YEARS_BEGINEXPLUATATION_MODE: double (nullable = true)
|-- YEARS_BUILD_MODE: double (nullable = true)
|-- COMMONAREA_MODE: double (nullable = true)
|-- ELEVATORS_MODE: double (nullable = true)
|-- ENTRANCES_MODE: double (nullable = true)
|-- FLOORSMAX_MODE: double (nullable = true)
|-- FLOORSMIN_MODE: double (nullable = true)
|-- LANDAREA_MODE: double (nullable = true)
|-- LIVINGAPARTMENTS_MODE: double (nullable = true)
|-- LIVINGAREA_MODE: double (nullable = true)
|-- NONLIVINGAPARTMENTS_MODE: double (nullable = true)
|-- NONLIVINGAREA_MODE: double (nullable = true)
|-- APARTMENTS_MEDI: double (nullable = true)
|-- BASEMENTAREA_MEDI: double (nullable = true)
|-- YEARS_BEGINEXPLUATATION_MEDI: double (nullable = true)
|-- YEARS_BUILD_MEDI: double (nullable = true)
|-- COMMONAREA_MEDI: double (nullable = true)
|-- ELEVATORS_MEDI: double (nullable = true)
|-- ENTRANCES_MEDI: double (nullable = true)
|-- FLOORSMAX_MEDI: double (nullable = true)
|-- FLOORSMIN_MEDI: double (nullable = true)
|-- LANDAREA_MEDI: double (nullable = true)
```

```
|-- LIVINGAPARTMENTS_MEDI: double (nullable = true)
|-- LIVINGAREA_MEDI: double (nullable = true)
|-- NONLIVINGAPARTMENTS_MEDI: double (nullable = true)
|-- NONLIVINGAREA_MEDI: double (nullable = true)
|-- FONDKAPREMONT_MODE: string (nullable = true)
|-- HOUSETYPE_MODE: string (nullable = true)
|-- TOTALAREA_MODE: double (nullable = true)
|-- WALLSMATERIAL_MODE: string (nullable = true)
|-- EMERGENCYSTATE_MODE: string (nullable = true)
|-- OBS_30_CNT_SOCIAL_CIRCLE: double (nullable = true)
|-- DEF_30_CNT_SOCIAL_CIRCLE: double (nullable = true)
|-- OBS_60_CNT_SOCIAL_CIRCLE: double (nullable = true)
|-- DEF_60_CNT_SOCIAL_CIRCLE: double (nullable = true)
|-- DAYS_LAST_PHONE_CHANGE: double (nullable = true)
|-- FLAG_DOCUMENT_2: integer (nullable = true)
|-- FLAG_DOCUMENT_3: integer (nullable = true)
|-- FLAG_DOCUMENT_4: integer (nullable = true)
|-- FLAG_DOCUMENT_5: integer (nullable = true)
|-- FLAG_DOCUMENT_6: integer (nullable = true)
|-- FLAG_DOCUMENT_7: integer (nullable = true)
|-- FLAG_DOCUMENT_8: integer (nullable = true)
|-- FLAG_DOCUMENT_9: integer (nullable = true)
|-- FLAG_DOCUMENT_10: integer (nullable = true)
|-- FLAG_DOCUMENT_11: integer (nullable = true)
|-- FLAG_DOCUMENT_12: integer (nullable = true)
|-- FLAG_DOCUMENT_13: integer (nullable = true)
|-- FLAG_DOCUMENT_14: integer (nullable = true)
|-- FLAG_DOCUMENT_15: integer (nullable = true)
|-- FLAG_DOCUMENT_16: integer (nullable = true)
|-- FLAG_DOCUMENT_17: integer (nullable = true)
|-- FLAG_DOCUMENT_18: integer (nullable = true)
|-- FLAG_DOCUMENT_19: integer (nullable = true)
|-- FLAG_DOCUMENT_20: integer (nullable = true)
|-- FLAG_DOCUMENT_21: integer (nullable = true)
|-- AMT_REQ_CREDIT_BUREAU_HOUR: double (nullable = true)
|-- AMT_REQ_CREDIT_BUREAU_DAY: double (nullable = true)
|-- AMT_REQ_CREDIT_BUREAU_WEEK: double (nullable = true)
|-- AMT_REQ_CREDIT_BUREAU_MON: double (nullable = true)
|-- AMT_REQ_CREDIT_BUREAU_QRT: double (nullable = true)
|-- AMT_REQ_CREDIT_BUREAU_YEAR: double (nullable = true)
```

- printSchema() only shows us the column names and its data type. we are going to drop the SK_ID_CURR column, rename the “TARGET” column to “label” and see the distribution of our target variable:

2. Analysis of the Data

In [ ]:
```
# Sk_ID_Curr is the id column which we dont need it in the process #so we get rid of it. and we rename the name of our
# target variable to "label"
drop_col = ['SK_ID_CURR']
new_df = new_df.select([column for column in new_df.columns if column not in drop_col])
new_df = new_df.withColumnRenamed('TARGET', 'label')
new_df.groupby('label').count().toPandas()
```

Out[ ]:

	label	count
0	1	24825
1	0	282686

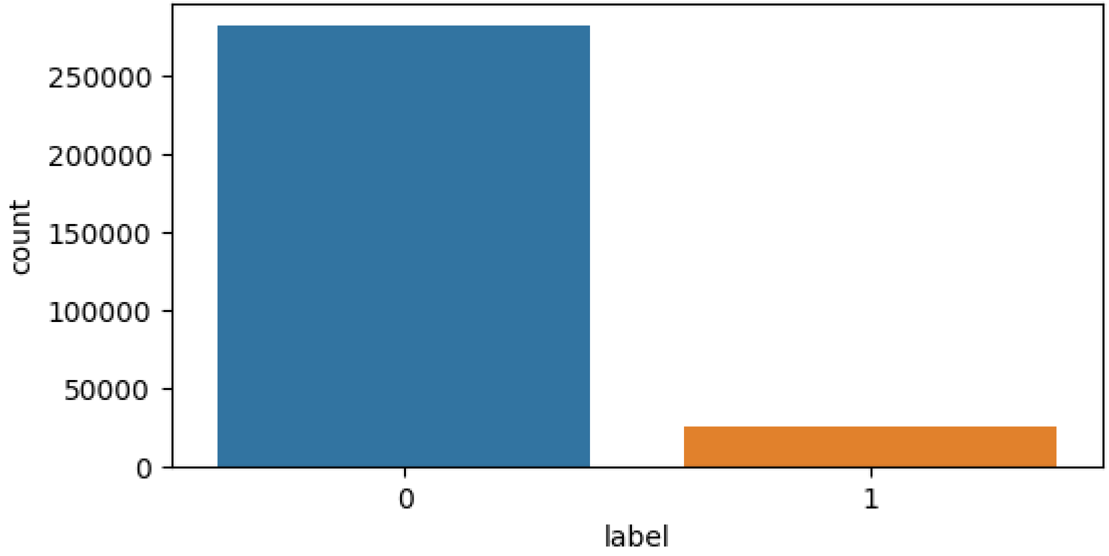
- Dataset is highly imbalanced

In [ ]:
```
# Let's have a look at the distribution of our target variable:
# to make it look better, we first convert our spark df to a Pandas
# Import the updated library
import pandas as pd
# Rest of your code
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
df_pd = new_df.toPandas()
print(len(df_pd))
plt.figure(figsize=(6,3))
sns.countplot(x='label', data=df_pd, order=df_pd['label'].value_counts().index)
```

```
307511
```

```
d:\Personal Files\Projects\Credit_Risk_Modelling_using_PySpark\venv\lib\site-packages\seaborn\_oldcore.py:1498: Futur
eWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, Categoric
alDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
d:\Personal Files\Projects\Credit_Risk_Modelling_using_PySpark\venv\lib\site-packages\seaborn\_oldcore.py:1498: Futur
eWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, Categoric
alDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
```

```
Out[ ]: <Axes: xlabel='label', ylabel='count'>
```



Data Wrangling can be summarized in the following points:

1. Understanding Dataset Structure: Begin by gaining insights into the dataset's general structure.
2. Identifying Feature Types: Determine the count of Categorical and Numerical features within the dataset.
3. Handling Missing Values: Develop a function to extract crucial information regarding missing values in the dataset.

Identifying Categorical & Numerical features

In [ ]:
```
# now let's see how many categorical and numerical features we have:
cat_cols = [item[0] for item in new_df.dtypes if item[1].startswith('string')]
print(str(len(cat_cols)) + '  categorical features')
num_cols = [item[0] for item in new_df.dtypes if item[1].startswith('int') | item[1].startswith('double')][1:]
print(str(len(num_cols)) + '  numerical features')
```

```
16  categorical features
104  numerical features
```

Identifying missing info

In [ ]:
```
# we use the below function to find more information about the #missing values
def info_missing_table(df_pd):
    """Input pandas dataframe and Return columns with missing value and percentage"""
    mis_val = df_pd.isnull().sum() #count total of null in each columns in dataframe
    #count percentage of null in each columns
    mis_val_percent = 100 * df_pd.isnull().sum() / len(df_pd)
    mis_val_table = pd.concat([mis_val, mis_val_percent], axis=1)
    #join to left (as column) between mis_val and mis_val_percent
    mis_val_table_ren_columns = mis_val_table.rename(
    columns = {0 : 'Missing Values', 1 : '% of Total Values'})
    #rename columns in table
    mis_val_table_ren_columns = mis_val_table_ren_columns[
    mis_val_table_ren_columns.iloc[:,1] != 0].sort_values('% of Total Values', ascending=False).round(1)

    print ("Your selected dataframe has " + str(df_pd.shape[1]) + " columns.\n"    #.shape[1] : just view total column
s in dataframe
    "There are " + str(mis_val_table_ren_columns.shape[0]) +
    " columns that have missing values.") #.shape[0] : just view total rows in dataframe
    return mis_val_table_ren_columns

missings = info_missing_table(df_pd)
missings
```

```
Your selected dataframe has 121 columns.
There are 67 columns that have missing values.
```

Out[ ]:

	Missing Values	% of Total Values
COMMONAREA_MEDI	214865	69.9
COMMONAREA_AVG	214865	69.9
COMMONAREA_MODE	214865	69.9
NONLIVINGAPARTMENTS_MEDI	213514	69.4
NONLIVINGAPARTMENTS_MODE	213514	69.4
...	...	...
EXT_SOURCE_2	660	0.2
AMT_GOODS_PRICE	278	0.1
AMT_ANNUITY	12	0.0
CNT_FAM_MEMBERS	2	0.0
DAYS_LAST_PHONE_CHANGE	1	0.0

67 rows × 2 columns

- There are 67 columns out of 121 that has missing values in them. and it doesn't show all of them in the image, but overall, most of these 67 columns have more than 50 percent missing values. so we are dealing with a lot of missing values. we are going to fill the numerical missing values with the average of each column and the categorical missing values with the most frequent category of each column.

In [ ]:

```
# so this function deals with the a spark dataframe directly to find more about the missing values
def count_missings(spark_df):
    null_counts = []
    for col in spark_df.dtypes:
        cname = col[0]
        ctype = col[1]
        nulls = spark_df.where( spark_df[cname].isNull()).count() #check count of null in column name
        result = tuple([cname, nulls])  #new tuple, (column name, null count)
        null_counts.append(result)      #put the new tuple in our result list
    null_counts=[(x,y) for (x,y) in null_counts if y!=0]  #view just columns that have missing values
    return null_counts
```

In [ ]:

```
# counts =missings.index
# miss_counts = []
# for column, miss_count in zip(missings.index, missings['Missing Values']):
#     miss_counts.append((column, miss_count))

miss_counts = count_missings(new_df)
miss_counts
```

```
Out[ ]:  [('AMT_ANNUITY', 12),
          ('AMT_GOODS_PRICE', 278),
          ('NAME_TYPE_SUITE', 1292),
          ('OWN_CAR_AGE', 202929),
          ('OCCUPATION_TYPE', 96391),
          ('CNT_FAM_MEMBERS', 2),
          ('EXT_SOURCE_1', 173378),
          ('EXT_SOURCE_2', 660),
          ('EXT_SOURCE_3', 60965),
          ('APARTMENTS_AVG', 156061),
          ('BASEMENTAREA_AVG', 179943),
          ('YEARS_BEGINEXPLUATATION_AVG', 150007),
          ('YEARS_BUILD_AVG', 204488),
          ('COMMONAREA_AVG', 214865),
          ('ELEVATORS_AVG', 163891),
          ('ENTRANCES_AVG', 154828),
          ('FLOORSMAX_AVG', 153020),
          ('FLOORSMIN_AVG', 208642),
          ('LANDAREA_AVG', 182590),
          ('LIVINGAPARTMENTS_AVG', 210199),
          ('LIVINGAREA_AVG', 154350),
          ('NONLIVINGAPARTMENTS_AVG', 213514),
          ('NONLIVINGAREA_AVG', 169682),
          ('APARTMENTS_MODE', 156061),
          ('BASEMENTAREA_MODE', 179943),
          ('YEARS_BEGINEXPLUATATION_MODE', 150007),
          ('YEARS_BUILD_MODE', 204488),
          ('COMMONAREA_MODE', 214865),
          ('ELEVATORS_MODE', 163891),
          ('ENTRANCES_MODE', 154828),
          ('FLOORSMAX_MODE', 153020),
          ('FLOORSMIN_MODE', 208642),
          ('LANDAREA_MODE', 182590),
          ('LIVINGAPARTMENTS_MODE', 210199),
          ('LIVINGAREA_MODE', 154350),
          ('NONLIVINGAPARTMENTS_MODE', 213514),
          ('NONLIVINGAREA_MODE', 169682),
          ('APARTMENTS_MEDI', 156061),
          ('BASEMENTAREA_MEDI', 179943),
          ('YEARS_BEGINEXPLUATATION_MEDI', 150007),
          ('YEARS_BUILD_MEDI', 204488),
```

```
('COMMONAREA_MEDI', 214865),
('ELEVATORS_MEDI', 163891),
('ENTRANCES_MEDI', 154828),
('FLOORSMAX_MEDI', 153020),
('FLOORSMIN_MEDI', 208642),
('LANDAREA_MEDI', 182590),
('LIVINGAPARTMENTS_MEDI', 210199),
('LIVINGAREA_MEDI', 154350),
('NONLIVINGAPARTMENTS_MEDI', 213514),
('NONLIVINGAREA_MEDI', 169682),
('FONDKAPREMONT_MODE', 210295),
('HOUSETYPE_MODE', 154297),
('TOTALAREA_MODE', 148431),
('WALLSMATERIAL_MODE', 156341),
('EMERGENCYSTATE_MODE', 145755),
('OBS_30_CNT_SOCIAL_CIRCLE', 1021),
('DEF_30_CNT_SOCIAL_CIRCLE', 1021),
('OBS_60_CNT_SOCIAL_CIRCLE', 1021),
('DEF_60_CNT_SOCIAL_CIRCLE', 1021),
('DAYS_LAST_PHONE_CHANGE', 1),
('AMT_REQ_CREDIT_BUREAU_HOUR', 41519),
('AMT_REQ_CREDIT_BUREAU_DAY', 41519),
('AMT_REQ_CREDIT_BUREAU_WEEK', 41519),
('AMT_REQ_CREDIT_BUREAU_MON', 41519),
('AMT_REQ_CREDIT_BUREAU_QRT', 41519),
('AMT_REQ_CREDIT_BUREAU_YEAR', 41519)]
```

3. Data Pre-processing

Separate categorical and numerical columns with missing values

In [ ]:
```
# here we seperate missing columns in our new_df based on #categorical and numerical types
list_cols_miss=[x[0] for x in miss_counts]
list_cols_miss
df_miss= new_df.select(*list_cols_miss)
# categorical columns
catcolums_miss=[item[0] for item in df_miss.dtypes if item[1].startswith('string')]  #will select name of column with string data type
print("cateogrical columns_miss:", catcolums_miss)
### numerical columns
numcolumns_miss = [item[0] for item in df_miss.dtypes if item[1].startswith('int') | item[1].startswith('double')] #will select name of column with integer or double data type
print("numerical columns_miss:", numcolumns_miss)
```

```
cateogrical columns_miss: ['NAME_TYPE_SUITE', 'OCCUPATION_TYPE', 'FONDKAPREMONT_MODE', 'HOUSETYPE_MODE', 'WALLSMATERIAL_MODE', 'EMERGENCYSTATE_MODE']
numerical columns_miss: ['AMT_ANNUITY', 'AMT_GOODS_PRICE', 'OWN_CAR_AGE', 'CNT_FAM_MEMBERS', 'EXT_SOURCE_1', 'EXT_SOURCE_2', 'EXT_SOURCE_3', 'APARTMENTS_AVG', 'BASEMENTAREA_AVG', 'YEARS_BEGINEXPLUATATION_AVG', 'YEARS_BUILD_AVG', 'COMMONAREA_AVG', 'ELEVATORS_AVG', 'ENTRANCES_AVG', 'FLOORSMAX_AVG', 'FLOORSMIN_AVG', 'LANDAREA_AVG', 'LIVINGAPARTMENTS_AVG', 'LIVINGAREA_AVG', 'NONLIVINGAPARTMENTS_AVG', 'NONLIVINGAREA_AVG', 'APARTMENTS_MODE', 'BASEMENTAREA_MODE', 'YEARS_BEGINEXPLUATATION_MODE', 'YEARS_BUILD_MODE', 'COMMONAREA_MODE', 'ELEVATORS_MODE', 'ENTRANCES_MODE', 'FLOORSMAX_MODE', 'FLOORSMIN_MODE', 'LANDAREA_MODE', 'LIVINGAPARTMENTS_MODE', 'LIVINGAREA_MODE', 'NONLIVINGAPARTMENTS_MODE', 'NONLIVINGAREA_MODE', 'APARTMENTS_MEDI', 'BASEMENTAREA_MEDI', 'YEARS_BEGINEXPLUATATION_MEDI', 'YEARS_BUILD_MEDI', 'COMMONAREA_MEDI', 'ELEVATORS_MEDI', 'ENTRANCES_MEDI', 'FLOORSMAX_MEDI', 'FLOORSMIN_MEDI', 'LANDAREA_MEDI', 'LIVINGAPARTMENTS_MEDI', 'LIVINGAREA_MEDI', 'NONLIVINGAPARTMENTS_MEDI', 'NONLIVINGAREA_MEDI', 'TOTALAREA_MODE', 'OBS_30_CNT_SOCIAL_CIRCLE', 'DEF_30_CNT_SOCIAL_CIRCLE', 'OBS_60_CNT_SOCIAL_CIRCLE', 'DEF_60_CNT_SOCIAL_CIRCLE', 'DAYS_LAST_PHONE_CHANGE', 'AMT_REQ_CREDIT_BUREAU_HOUR', 'AMT_REQ_CREDIT_BUREAU_DAY', 'AMT_REQ_CREDIT_BUREAU_WEEK', 'AMT_REQ_CREDIT_BUREAU_MON', 'AMT_REQ_CREDIT_BUREAU_QRT', 'AMT_REQ_CREDIT_BUREAU_YEAR']
```

Fill the missing values

In [ ]:

```
# now that we have seperated the columns based on categorical and #numerical types, we will fill the missing categirac
l
# values with the most frequent category
from pyspark.sql.functions import rank,sum,col
df_Nomiss=new_df.na.drop()
for x in catcolums_miss:
    mode=df_Nomiss.groupBy(x).count().sort(col("count").desc()).collect()[0][0]
    print(x, mode) #print name of columns and it's most categories
    new_df = new_df.na.fill({x:mode})
# and we fill the missing numerical values with the average of each #column
from pyspark.sql.functions import mean, round
for i in numcolumns_miss:
    meanvalue = new_df.select(round(mean(i))).collect()[0][0]
    print(i, meanvalue)
    new_df=new_df.na.fill({i:meanvalue})
```

NAME_TYPE_SUITE Unaccompanied
OCCUPATION_TYPE Laborers
FONDKAPREMONT_MODE reg oper account
HOUSETYPE_MODE block of flats
WALLSMATERIAL_MODE Panel
EMERGENCYSTATE_MODE No
AMT_ANNUITY 27109.0
AMT_GOODS_PRICE 538396.0
OWN_CAR_AGE 12.0
CNT_FAM_MEMBERS 2.0
EXT_SOURCE_1 1.0
EXT_SOURCE_2 1.0
EXT_SOURCE_3 1.0
APARTMENTS_AVG 0.0
BASEMENTAREA_AVG 0.0
YEARS_BEGINEXPLUATATION_AVG 1.0
YEARS_BUILD_AVG 1.0
COMMONAREA_AVG 0.0
ELEVATORS_AVG 0.0
ENTRANCES_AVG 0.0
FLOORSMAX_AVG 0.0
FLOORSMIN_AVG 0.0
LANDAREA_AVG 0.0
LIVINGAPARTMENTS_AVG 0.0
LIVINGAREA_AVG 0.0
NONLIVINGAPARTMENTS_AVG 0.0
NONLIVINGAREA_AVG 0.0
APARTMENTS_MODE 0.0
BASEMENTAREA_MODE 0.0
YEARS_BEGINEXPLUATATION_MODE 1.0
YEARS_BUILD_MODE 1.0
COMMONAREA_MODE 0.0
ELEVATORS_MODE 0.0
ENTRANCES_MODE 0.0
FLOORSMAX_MODE 0.0
FLOORSMIN_MODE 0.0
LANDAREA_MODE 0.0
LIVINGAPARTMENTS_MODE 0.0
LIVINGAREA_MODE 0.0
NONLIVINGAPARTMENTS_MODE 0.0
NONLIVINGAREA_MODE 0.0

```
APARTMENTS_MEDI 0.0
BASEMENTAREA_MEDI 0.0
YEARS_BEGINEXPLUATATION_MEDI 1.0
YEARS_BUILD_MEDI 1.0
COMMONAREA_MEDI 0.0
ELEVATORS_MEDI 0.0
ENTRANCES_MEDI 0.0
FLOORSMAX_MEDI 0.0
FLOORSMIN_MEDI 0.0
LANDAREA_MEDI 0.0
LIVINGAPARTMENTS_MEDI 0.0
LIVINGAREA_MEDI 0.0
NONLIVINGAPARTMENTS_MEDI 0.0
NONLIVINGAREA_MEDI 0.0
TOTALAREA_MODE 0.0
OBS_30_CNT_SOCIAL_CIRCLE 1.0
DEF_30_CNT_SOCIAL_CIRCLE 0.0
OBS_60_CNT_SOCIAL_CIRCLE 1.0
DEF_60_CNT_SOCIAL_CIRCLE 0.0
DAYS_LAST_PHONE_CHANGE -963.0
AMT_REQ_CREDIT_BUREAU_HOUR 0.0
AMT_REQ_CREDIT_BUREAU_DAY 0.0
AMT_REQ_CREDIT_BUREAU_WEEK 0.0
AMT_REQ_CREDIT_BUREAU_MON 0.0
AMT_REQ_CREDIT_BUREAU_QRT 0.0
AMT_REQ_CREDIT_BUREAU_YEAR 2.0
```

- The dataset no longer contains missing values.

Handling Imbalanced Classess

- The next step is addressing the issue of imbalanced classes.
- Various methods can be used to mitigate this problem.
- One approach involves under-sampling the majority class or over-sampling the minority class to achieve a more balanced outcome.
- Another approach is assigning weights to each class to penalize the majority class with lower weights and boost the minority class with higher weights.
- To implement this, a new column called "weights" is created in the dataset.
- The weights are assigned based on the inverse ratio of each class.

In [ ]:
```
# adding the new column weights and fill it with ratios
from pyspark.sql.functions import when
ratio = 0.91
def weight_balance(labels):
    return when(labels == 1, ratio).otherwise(1*(1-ratio))
new_df = new_df.withColumn('weights', weight_balance(col('label')))
```

In [ ]:
```
new_df.show(5)
```

```
+-----+------------------+-----------+------------+---------------+------------+----------------+----------+---------
--+---------------+---------------+----------------+--------------------+--------------------+-----------------+-----
---------------------+----------+-------------+-----------------+---------------+-----------+----------+-------------
-+---------------+----------------+----------+----------+---------------+---------------+--------------------+-------
--------------------+--------------------------+-----------------------+--------------------------+------------------
--------+---------------------------+----------------------+----------------------+-----------------------+----------
----------+-------------------+------------------+-------------------+--------------+----------------+---------------
------------+------------------+--------------+-------------+-------------+-------------+-------------+------------+-
-------------------+--------------+-----------------------+-----------------+---------------+-----------------+------
----------------------+----------------+---------------+--------------+--------------+--------------+--------------+-
------------+---------------------+---------------+------------------------+------------------+---------------+------
-----------+----------------------------+----------------+---------------+--------------+--------------+-------------
-+--------------+-------------+---------------------+---------------+------------------------+------------------+----
--------------+--------------+--------------+------------------+-------------------+------------------------+--------
----------------+------------------------+------------------------+----------------------+---------------+-----------
----+---------------+---------------+---------------+---------------+---------------+---------------+----------------
+----------------+----------------+----------------+----------------+----------------+----------------+--------------
--+----------------+----------------+----------------+----------------+--------------------------+-------------------
------+--------------------------+-------------------------+-------------------------+--------------------------+----
---------------+
|label|NAME_CONTRACT_TYPE|CODE_GENDER|FLAG_OWN_CAR|FLAG_OWN_REALTY|CNT_CHILDREN|AMT_INCOME_TOTAL|AMT_CREDIT|AMT_ANNUI
TY|AMT_GOODS_PRICE|NAME_TYPE_SUITE|NAME_INCOME_TYPE| NAME_EDUCATION_TYPE|  NAME_FAMILY_STATUS|NAME_HOUSING_TYPE|REGIO
N_POPULATION_RELATIVE|DAYS_BIRTH|DAYS_EMPLOYED|DAYS_REGISTRATION|DAYS_ID_PUBLISH|OWN_CAR_AGE|FLAG_MOBIL|FLAG_EMP_PHON
E|FLAG_WORK_PHONE|FLAG_CONT_MOBILE|FLAG_PHONE|FLAG_EMAIL|OCCUPATION_TYPE|CNT_FAM_MEMBERS|REGION_RATING_CLIENT|REGION_
RATING_CLIENT_W_CITY|WEEKDAY_APPR_PROCESS_START|HOUR_APPR_PROCESS_START|REG_REGION_NOT_LIVE_REGION|REG_REGION_NOT_WOR
K_REGION|LIVE_REGION_NOT_WORK_REGION|REG_CITY_NOT_LIVE_CITY|REG_CITY_NOT_WORK_CITY|LIVE_CITY_NOT_WORK_CITY|   ORGANIZ
ATION_TYPE|       EXT_SOURCE_1|      EXT_SOURCE_2|       EXT_SOURCE_3|APARTMENTS_AVG|BASEMENTAREA_AVG|YEARS_BEGINEXPL
UATATION_AVG|   YEARS_BUILD_AVG|COMMONAREA_AVG|ELEVATORS_AVG|ENTRANCES_AVG|FLOORSMAX_AVG|FLOORSMIN_AVG|LANDAREA_AVG|L
IVINGAPARTMENTS_AVG|LIVINGAREA_AVG|NONLIVINGAPARTMENTS_AVG|NONLIVINGAREA_AVG|APARTMENTS_MODE|BASEMENTAREA_MODE|YEARS_
BEGINEXPLUATATION_MODE|YEARS_BUILD_MODE|COMMONAREA_MODE|ELEVATORS_MODE|ENTRANCES_MODE|FLOORSMAX_MODE|FLOORSMIN_MODE|L
ANDAREA_MODE|LIVINGAPARTMENTS_MODE|LIVINGAREA_MODE|NONLIVINGAPARTMENTS_MODE|NONLIVINGAREA_MODE|APARTMENTS_MEDI|BASEME
NTAREA_MEDI|YEARS_BEGINEXPLUATATION_MEDI|YEARS_BUILD_MEDI|COMMONAREA_MEDI|ELEVATORS_MEDI|ENTRANCES_MEDI|FLOORSMAX_MED
I|FLOORSMIN_MEDI|LANDAREA_MEDI|LIVINGAPARTMENTS_MEDI|LIVINGAREA_MEDI|NONLIVINGAPARTMENTS_MEDI|NONLIVINGAREA_MEDI|FOND
KAPREMONT_MODE|HOUSETYPE_MODE|TOTALAREA_MODE|WALLSMATERIAL_MODE|EMERGENCYSTATE_MODE|OBS_30_CNT_SOCIAL_CIRCLE|DEF_30_C
NT_SOCIAL_CIRCLE|OBS_60_CNT_SOCIAL_CIRCLE|DEF_60_CNT_SOCIAL_CIRCLE|DAYS_LAST_PHONE_CHANGE|FLAG_DOCUMENT_2|FLAG_DOCUME
NT_3|FLAG_DOCUMENT_4|FLAG_DOCUMENT_5|FLAG_DOCUMENT_6|FLAG_DOCUMENT_7|FLAG_DOCUMENT_8|FLAG_DOCUMENT_9|FLAG_DOCUMENT_10
|FLAG_DOCUMENT_11|FLAG_DOCUMENT_12|FLAG_DOCUMENT_13|FLAG_DOCUMENT_14|FLAG_DOCUMENT_15|FLAG_DOCUMENT_16|FLAG_DOCUMENT_
17|FLAG_DOCUMENT_18|FLAG_DOCUMENT_19|FLAG_DOCUMENT_20|FLAG_DOCUMENT_21|AMT_REQ_CREDIT_BUREAU_HOUR|AMT_REQ_CREDIT_BURE
AU_DAY|AMT_REQ_CREDIT_BUREAU_WEEK|AMT_REQ_CREDIT_BUREAU_MON|AMT_REQ_CREDIT_BUREAU_QRT|AMT_REQ_CREDIT_BUREAU_YEAR|
weights|
+-----+------------------+-----------+------------+---------------+------------+----------------+----------+---------
```

```
--+----------------+---------------+-----------------+--------------------+---------------------+-----------------+-----
----------------------+----------+--------------+-----------------+---------------+------------+----------+-------------
-+----------------+----------------+----------+-----------+---------------+----------------+--------------------+-------
---------------------+---------------------------+-----------------------+---------------------------+------------------
---------+---------------------------+-----------------------+-----------------------+-----------------------+----------
-----------+-------------------+-------------------+-------------------+---------------+----------------+---------------
-------------+------------------+---------------+-------------+-------------+--------------+-------------+------------+-
--------------------+--------------+------------------------+-----------------+----------------+-----------------+------
-----------------------+----------------+----------------+--------------+---------------+--------------+--------------+-
-------------+---------------------+----------------+------------------------+-------------------+---------------+------
------------+----------------------------+-----------------+---------------+---------------+--------------+-------------
-+---------------+-------------+----------------------+---------------+-------------------------+------------------+----
---------------+--------------+---------------+------------------+--------------------+------------------------+--------
-----------------+------------------------+-------------------------+-----------------------+---------------+-----------
----+----------------+---------------+----------------+---------------+----------------+---------------+----------------
+-----------------+----------------+-----------------+----------------+-----------------+----------------+--------------
--+-----------------+----------------+-----------------+----------------+---------------------------+-------------------
------+---------------------------+--------------------------+--------------------------+--------------------------+----
---------------+
|    1|        Cash loans|          M|           N|               Y|           0|        202500.0|  406597.5|    2470
0.5|       351000.0|  Unaccompanied|         Working|Secondary / secon...|Single / not married|House / apartment|
0.018801|     -9461|         -637|          -3648.0|           -2120|       12.0|          1|             1|
0|               1|          1|         0|        Laborers|            1.0|                      2|
2|                   WEDNESDAY|                     10|                          0|                        0|
0|                    0|                      0|                          0|Business Entity T...|0.08303696739132256|0.26
29485927471776|0.13937578009978951|        0.0247|           0.0369|                          0.9722|             0.6192|
0.0143|          0.0|         0.069|       0.0833|         0.125|      0.0369|                   0.0202|         0.019|
0.0|                0.0|           0.0252|           0.0383|                           0.9722|           0.6341|          0.0144
|           0.0|          0.069|          0.0833|          0.125|        0.0377|                   0.022|          0.0198|
0.0|                0.0|            0.025|           0.0369|                           0.9722|           0.6243|          0.014
4|           0.0|          0.069|        0.0833|          0.125|         0.0375|                 0.0205|          0.0193|
0.0|                0.0|  reg oper account|block of flats|        0.0149|      Stone, brick|                    No|
2.0|                   2.0|                        2.0|                        2.0|                   -1134.0|              0
|              1|              0|              0|               0|              0|               0|             0|
0|               0|                0|                0|                0|                0|                0|
0|               0|                0|                0|                0|                        0.0|
0.0|                         0.0|                         0.0|                         0.0|                         1.0|
0.91|
|    0|        Cash loans|          F|           N|               N|           0|        270000.0| 1293502.5|    3569
8.5|      1129500.0|          Family|   State servant|    Higher education|             Married|House / apartment|
0.0035409999999999|    -16765|       -1188|          -1186.0|            -291|       12.0|          1|             1
|              0|              1|         1|          0|     Core staff|           2.0|                    1|
```

```
1|                    MONDAY|                     11|                      0|                      0|
0|                    0|                    0|                      0|          School| 0.3112673113812225|0.62
22457752555098|              1.0|        0.0959|          0.0529|                    0.9851|0.7959999999999999|
0.0605|         0.08|        0.0345|        0.2917|        0.3333|         0.013|             0.0773|        0.0549|
0.0039|            0.0098|            0.0924|            0.0538|                      0.9851|          0.804|        0.0
497|        0.0806|        0.0345|        0.2917|        0.3333|        0.0128|                0.079|        0.0554|
0.0|            0.0|            0.0968|            0.0529|                       0.9851|          0.7987|        0.060
8|          0.08|        0.0345|        0.2917|        0.3333|        0.0132|               0.0787|        0.0558|
0.0039|               0.01|  reg oper account|block of flats|        0.0714|             Block|                No|
1.0|                0.0|                    1.0|                   0.0|                -828.0|                0
|             1|             0|             0|             0|             0|             0|             0|
0|              0|               0|               0|               0|               0|               0|
0|              0|               0|               0|               0|                 0.0|
0.0|                      0.0|                      0.0|                      0.0|                      0.0|0.08999
999999999997|
|    0|   Revolving loans|         M|          Y|              Y|           0|        67500.0|  135000.0|     675
0.0|       135000.0|  Unaccompanied|        Working|Secondary / secon...|Single / not married|House / apartment|
0.010032|    -19046|         -225|          -4260.0|          -2531|      26.0|         1|             1|
1|              1|         1|        0|       Laborers|            1.0|                   2|
2|                    MONDAY|                      9|                      0|                      0|
0|                    0|                    0|                      0|       Government|                1.0|0.55
59120833904428| 0.7295666907060153|          0.0|            0.0|                      1.0|               1.0|
0.0|           0.0|          0.0|          0.0|          0.0|          0.0|                0.0|          0.0|
0.0|             0.0|             0.0|              0.0|                        1.0|              1.0|          0.0
|           0.0|          0.0|          0.0|          0.0|          0.0|                0.0|          0.0|
0.0|               0.0|              0.0|              0.0|                       1.0|            1.0|          0.
0|          0.0|          0.0|          0.0|          0.0|          0.0|                0.0|          0.0|
0.0|               0.0|  reg oper account|block of flats|          0.0|             Panel|              No|
0.0|                0.0|                    0.0|                   0.0|                -815.0|                0
|             0|             0|             0|             0|             0|             0|             0|
0|              0|               0|               0|               0|               0|               0|
0|              0|               0|               0|               0|                 0.0|
0.0|                      0.0|                      0.0|                      0.0|                      0.0|0.08999
999999999997|
|    0|        Cash loans|         F|          N|              Y|           0|       135000.0|  312682.5|    2968
6.5|       297000.0|  Unaccompanied|        Working|Secondary / secon...|      Civil marriage|House / apartment|
0.008019|    -19005|        -3039|          -9833.0|          -2437|      12.0|         1|             1|
0|              1|         0|        0|       Laborers|            2.0|                   2|
2|                 WEDNESDAY|                     17|                      0|                      0|
0|                    0|                    0|                      0|Business Entity T...|                1.0|0.65
04416904014653|              1.0|          0.0|            0.0|                      1.0|               1.0|
0.0|           0.0|          0.0|          0.0|          0.0|          0.0|                0.0|          0.0|
```

```
0.0|             0.0|           0.0|             0.0|                      1.0|            1.0|           0.0
|          0.0|          0.0|          0.0|          0.0|          0.0|                0.0|           0.0|
0.0|            0.0|           0.0|             0.0|                      1.0|            1.0|           0.
0|          0.0|          0.0|          0.0|          0.0|          0.0|                0.0|           0.0|
0.0|            0.0|  reg oper account|block of flats|         0.0|            Panel|                 No|
2.0|                0.0|                  2.0|                  0.0|                 -617.0|              0
|            1|            0|            0|            0|            0|            0|            0|
0|            0|            0|            0|            0|            0|            0|
0|            0|            0|            0|            0|                0.0|
0.0|                    0.0|                    0.0|                    0.0|                    2.0|0.08999
999999999997|
|    0|       Cash loans|          M|          N|             Y|           0|        121500.0|  513000.0|    2186
5.5|       513000.0|  Unaccompanied|        Working|Secondary / secon...|Single / not married|House / apartment|
0.028663|    -19932|       -3038|          -4311.0|          -3458|      12.0|         1|           1|
0|            1|        0|        0|     Core staff|          1.0|                  2|
2|                THURSDAY|                     11|                      0|                     0|
0|                  0|                  1|                  1|          Religion|                 1.0|0.32
27382869704046|              1.0|          0.0|          0.0|                  1.0|           1.0|
0.0|          0.0|          0.0|          0.0|          0.0|          0.0|               0.0|          0.0|
0.0|             0.0|           0.0|             0.0|                      1.0|            1.0|           0.0
|          0.0|          0.0|          0.0|          0.0|          0.0|                0.0|           0.0|
0.0|            0.0|           0.0|             0.0|                      1.0|            1.0|           0.
0|          0.0|          0.0|          0.0|          0.0|          0.0|                0.0|           0.0|
0.0|            0.0|  reg oper account|block of flats|         0.0|            Panel|                 No|
0.0|                0.0|                  0.0|                  0.0|                -1106.0|              0
|            0|            0|            0|            0|            0|            1|            0|
0|            0|            0|            0|            0|            0|            0|
0|            0|            0|            0|            0|                0.0|
0.0|                    0.0|                    0.0|                    0.0|                    0.0|0.08999
999999999997|
+-----+------------------+----------+------------+---------------+-----------+----------------+----------+---------
--+--------------+---------------+-----------------+-------------------+-------------------+----------------+-----
--------------------+----------+------------+----------------+--------------+----------+----------+----------
-+--------------+---------------+---------+---------+--------------+--------------+-------------------+-------
-------------------+-------------------------+-----------------------+-------------------------+-----------------
--------+-------------------------+--------------------+--------------------+----------------------+----------
----------+------------------+-----------------+------------------+-------------+---------------+-------------
------------+-----------------+-------------+------------+------------+------------+------------+-------------+-
-------------------+-------------+---------------------+---------------+--------------+---------------+------
---------------------+---------------+--------------+-------------+-------------+-------------+--------------+-
------------+-------------------+--------------+----------------------+----------------+--------------+-----
----------+--------------------------+---------------+--------------+-------------+-------------+------------
```

```
-+--------------+-------------+---------------------+---------------+------------------------+------------------+----
--------------+--------------+--------------+------------------+-------------------+------------------------+--------
----------------+------------------------+------------------------+----------------------+---------------+-----------
----+---------------+---------------+---------------+---------------+---------------+---------------+----------------
+----------------+----------------+----------------+----------------+----------------+----------------+--------------
--+----------------+----------------+----------------+----------------+--------------------------+-------------------
------+--------------------------+-------------------------+-------------------------+--------------------------+----
---------------+
only showing top 5 rows
```

Feature Engineering

- The next step in the process is Feature Engineering.
- pySpark simplifies feature extraction, making it easier.
- The steps involved in feature extraction are as follows:
    1. Apply StringIndexer() to assign indices to each category in our categorical columns.
    2. Use OneHotEncoderEstimator() to convert categorical columns into one-hot encoded vectors.
    3. Apply VectorAssembler() to create a feature vector that combines all categorical and numerical features, referred to as "features."

In [ ]:
```
# We use the OneHotEncoderEstimator from MLlib in Spark to convert each categorical feature into one-hot vectors.
# Next, we use VectorAssembler to combine the resulting one-hot vectors and the rest of the numerical features into a
single vector column.
# We append every step of the process in a stages array.

from pyspark.ml.feature import OneHotEncoder, StringIndexer, VectorAssembler

stages = []

for categoricalCol in cat_cols:
    stringIndexer = StringIndexer(inputCol=categoricalCol, outputCol=categoricalCol + 'Index')
    encoder = OneHotEncoder(inputCols=[stringIndexer.getOutputCol()], outputCols=[categoricalCol + "classVec"])
    stages += [stringIndexer, encoder]

assemblerInputs = [c + "classVec" for c in cat_cols] + num_cols
assembler = VectorAssembler(inputCols=assemblerInputs, outputCol="features")
stages += [assembler]
```

1. We will now create a pipeline.
2. This pipeline will perform a sequence of transformations.
3. The purpose is to apply all these transformations at once.
4. This approach simplifies the process of applying multiple transformations in a sequence.

In [ ]:
```
# we use a pipeline to apply all the stages of transformation
from pyspark.ml import Pipeline
cols = new_df.columns
pipeline = Pipeline(stages = stages)
pipelineModel = pipeline.fit(new_df)
new_df = pipelineModel.transform(new_df)
selectedCols = ['features']+cols
new_df = new_df.select(selectedCols)
pd.DataFrame(new_df.take(5), columns=new_df.columns)
```

Out[ ]:

	features	label	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME
0	(1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 0.0, ...	1	Cash loans	M	N	Y	0	202500.0
1	(1.0, 1.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, ...	0	Cash loans	F	N	N	0	270000.0
2	(0.0, 0.0, 1.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0, ...	0	Revolving loans	M	Y	Y	0	67500.0
3	(1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 0.0, 0.0, 0.0, ...	0	Cash loans	F	N	Y	0	135000.0
4	(1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 0.0, ...	0	Cash loans	M	N	Y	0	121500.0

5 rows × 123 columns

4. Model Building

Training and Hyper-parameter Tuning

- Split the dataset into training and testing sets for training.
- Begin training with Logistic Regression.
- Choose Logistic Regression for its performance in binary classification problems.

Train-Test Split

In [ ]:
```
# split the data into trainign and testin sets
train, test = new_df.randomSplit([0.80, 0.20], seed = 42)
print(train.count())
print(test.count())
```

```
246240
61271
```

Logistics Regression

In [ ]:
```
# first we check how LogisticRegression perform
from pyspark.ml.classification import LogisticRegression
LR = LogisticRegression(featuresCol = 'features', labelCol = 'label', maxIter=15)
LR_model = LR.fit(train)
```

Gradient Boosting Trees (GBT)

In [ ]:
```
# next we checkout gradient boosting trees
from pyspark.ml.classification import GBTClassifier
gbt = GBTClassifier(maxIter=15)
GBT_Model = gbt.fit(train)
```

- Achieved a significantly improved result of 0.732 using GBT.

Hyperparameter Tuning - Gradient Boosting Trees (GBT)

- Final strategy includes hyper-parameter tuning through grid search.
- Planning to follow up with cross-validation to further enhance GBT performance.

In [ ]:
```
from pyspark.ml.tuning import ParamGridBuilder, CrossValidator
from pyspark.ml.evaluation import BinaryClassificationEvaluator  # Import BinaryClassificationEvaluator

paramGrid = (ParamGridBuilder().addGrid(gbt.maxDepth, [2, 4, 6]).addGrid(gbt.maxBins, [20, 30]).addGrid(gbt.maxIter,
[10, 15]).build())
cv = CrossValidator(estimator=gbt, estimatorParamMaps=paramGrid, evaluator = BinaryClassificationEvaluator(), numFolds
=5)
# Run cross validations.
cvModel = cv.fit(train)
```

- The result showed a slight improvement.
- There is room for further enhancement by experimenting with hyper-parameter tuning.

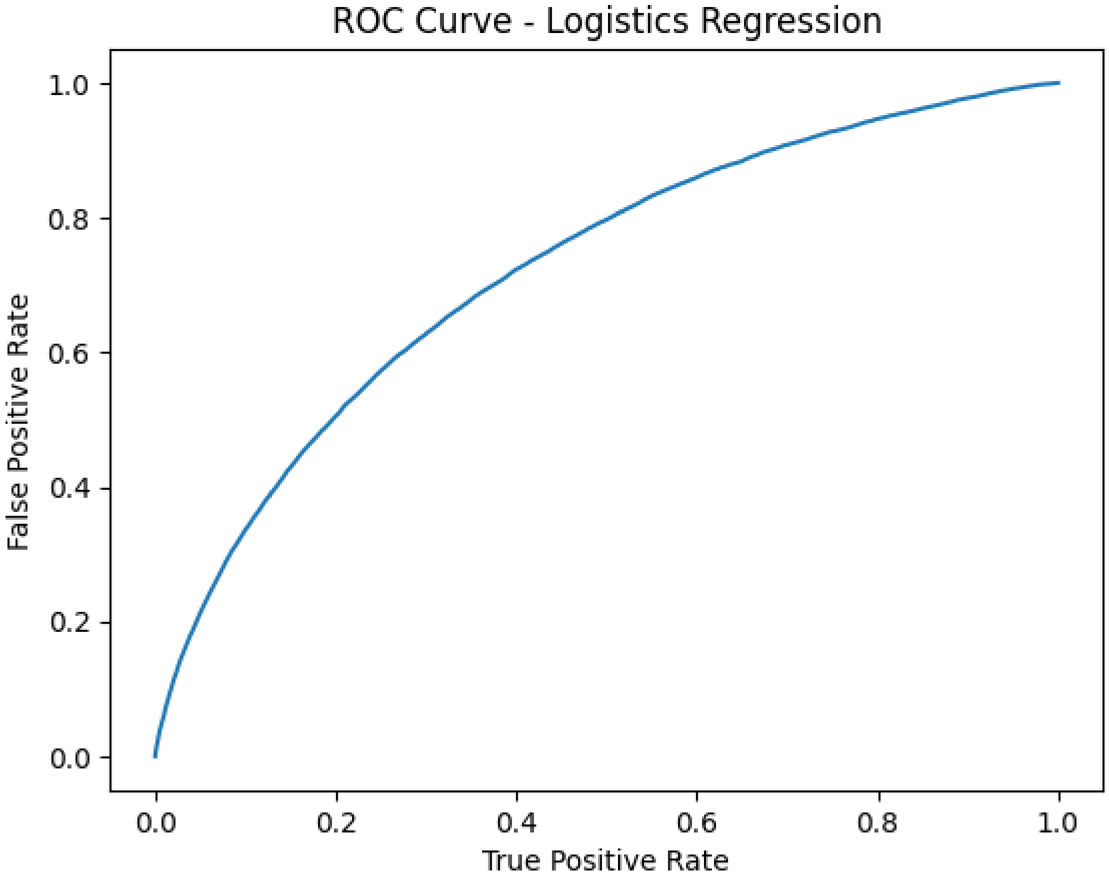
5. Model Evaluation

1. We will create an ROC curve for the training data.
2. The purpose is to assess the performance of Logistic Regression.
3. We will utilize the Area Under the ROC Curve (AUC-ROC).
4. AUC-ROC is a common metric for evaluating binary classification models.

In [ ]:
```
from pyspark.ml.evaluation import BinaryClassificationEvaluator
evaluator = BinaryClassificationEvaluator()

predictions_LR = LR_model.transform(test)
print("Test_SET Logistic Regression (Area Under ROC): " + str(evaluator.evaluate(predictions_LR, {evaluator.metricNam
e: "areaUnderROC"})))

gbt_predictions = GBT_Model.transform(test)
print("Test_SET GBT (Area Under ROC): " + str(evaluator.evaluate(gbt_predictions, {evaluator.metricName: "areaUnderRO
C"})))

gbt_cv_predictions = cvModel.transform(test)
print("Test_SET GBT_Tuned (Area Under ROC): " + str(evaluator.evaluate(gbt_cv_predictions, {evaluator.metricName: "are
aUnderROC"})))
```

```
Test_SET Logistic Regression (Area Under ROC): 0.7215699765501389
Test_SET GBT (Area Under ROC): 0.7298618196435773
Test_SET GBT_Tuned (Area Under ROC): 0.7321482798175072
```

In [ ]:
```
#plotting the ROC Curve
trainingSummary = LR_model.summary
roc_LR = trainingSummary.roc.toPandas()
plt.plot(roc_LR['FPR'],roc_LR['TPR'])
plt.ylabel('False Positive Rate')
plt.xlabel('True Positive Rate')
plt.title('ROC Curve - Logistics Regression')
plt.show()
print('Training set ROC: ' + str(trainingSummary.areaUnderROC))
```




```
Training set ROC: 0.7229899807041205
```

THE END