



## CIS5560 Term Project Tutorial



**Authors:** [Sapan Shah](#); [Snehil Sarkar](#); [Katherine Belknap](#);

**Instructor:** [Jongwook Woo](#)

**Date:** 05/14/2024

### Lab Tutorial

Sapan Shah([sshah82@calstatela.edu](mailto:sshah82@calstatela.edu))

Snehil Sarkar([ssarkar4@calstatela.edu](mailto:ssarkar4@calstatela.edu))

Katherine Belknap([kbelkna2@calstatela.edu](mailto:kbelkna2@calstatela.edu))

05/14/2024

## Analysis of Anti-Money Laundering (AML) with Spark ML

---

<p><b>Note:</b> As per the feedback, dataset is rebalanced by Under-Sampling and for all algorithms, equal numbers of hyper parameters are used for better comparison.</p>
--

## Objectives:

This dataset is segregated into two categories based on the prevalence of money laundering: high and low ratios. It encompasses information on individuals, companies, and banks engaged in both lawful and illicit financial transactions.

- The aim is to develop a model capable of identifying money laundering patterns.
- Utilizing this dataset with its sample, a fraud detection model can be trained using Databricks, which offers a scalable computing environment.
- PySpark's SQL and DataFrame APIs enable various operations such as aggregation, filtering, and joining.
- Subsequently, Performance Metrics can be assessed through techniques like Cross-Validation or Train-Validation Splits.

The findings of this study hold the potential to significantly impact the financial sector, corporations, and governments. By predicting future occurrences of money laundering activities and visualizing potential trends, it could lead to safer and more secure financial transactions.

## Platform Specifications:

- **Hadoop Version:** 3.3.3
- **CPU Speed:** 1995.312 MHz
- **No of CPU cores:** 8
- **No of nodes:** 5
- **Spark Version:** 3.2.1
- **Total Memory Size:** 860.4 GB

## Dataset Specifications: -

**Dataset Name:** IBM Transactions for Anti Money Laundering (AML)

**Dataset Size:** 2.82 GB

**Dataset URL:** <https://www.kaggle.com/datasets/ealtman2019/ibm-transactions-for-anti-money-laundering-aml>

**Dataset Format:** CSV

**Dataset:** HI\_Medium\_Trans.csv

The screenshot shows the Kaggle dataset page for "IBM Transactions for Anti Money Laundering (AML)". The page includes a search bar, user options (Sign In, Register), and dataset statistics (116 votes, New Notebook button, Download 8 GB button). The dataset description mentions its use for Foundation Models, GNNs, and more. It lists related papers and a GitHub site. The right sidebar shows a usability score of 9.41, a license, and tags like Tabular, Law, Finance, Classification, and Banking.

## Step 1: Get data from the Data source.

### Download the dataset from Kaggle.

1. Log in to Kaggle.
2. It shows 6 files consisting of 3 with HI – Higher Illicit and other 3 with LI- Lower Illicit values files.
3. Out of these files, 1 small HI- Higher Illicit of 0.4 GB file is used for model training in Databricks.
4. And 1 medium HI – Higher Illicit of 2.82 GB file is used to run models for all algorithms on spark-submit.

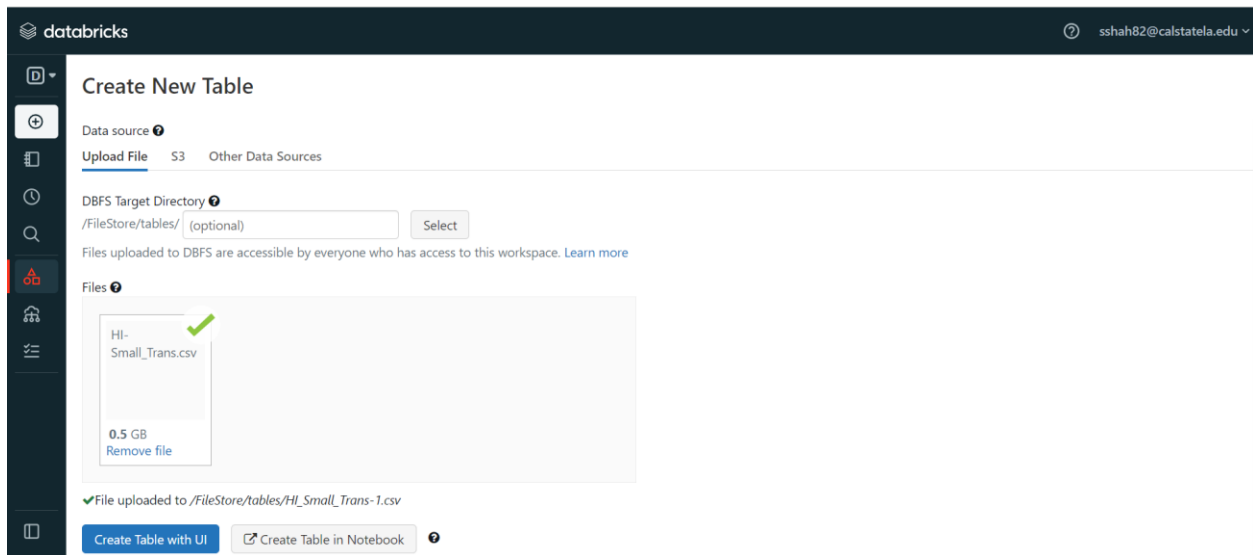
## Step 2: Sign in to Databricks.

Login to databricks using the below URL. <https://community.cloud.databricks.com/login.html>

## Step 3: Create a cluster in the databricks.

The screenshot shows the Databricks cluster configuration page. The left sidebar contains navigation icons. The main area shows the "Configuration" tab for a cluster named "QS". It displays the Databricks Runtime Version as "12.2 LTS (includes Apache Spark 3.3.2, Scala 2.12)". The Driver type is set to "Community Optimized" with "15.3 GB Memory, 2 Cores, 1 DBU". A note mentions "Free 15 GB Memory: As a Community Edition user, your compute will automatically terminate after an idle period of one or two hours." The bottom of the page shows tabs for "Instances", "Spark", and "JDBC/ODBC".

## Step 4: Upload HI\_Small\_Trans.csv by creating new table in notebook.



## Step 5: Import all necessary libraries.

```
# Databricks notebook source
import logging
from pyspark.sql.functions import *
from pyspark.ml import Pipeline
from pyspark.ml.feature import VectorAssembler, StringIndexer, VectorIndexer
from pyspark.ml.tuning import ParamGridBuilder, TrainValidationSplit, CrossValidator
from pyspark.mllib.evaluation import MulticlassMetrics
from pyspark.ml.classification import LogisticRegression, GBTClassifier,
RandomForestClassifier, LinearSVC
from pyspark.ml.evaluation import BinaryClassificationEvaluator
from time import time
from pyspark.context import SparkContext
from pyspark.sql.session import SparkSession
from pyspark.storagelevel import StorageLevel
from pyspark.sql.types import StructType, StructField, IntegerType, StringType, FloatType

# Configure logging
logging.basicConfig(level=logging.INFO, format='%(message)s')
```

```

1 # Databricks notebook source
2 import logging
3 from pyspark.sql.functions import *
4 from pyspark.ml import Pipeline
5 from pyspark.ml.feature import VectorAssembler, StringIndexer, VectorIndexer
6 from pyspark.ml.tuning import ParamGridBuilder, TrainValidationSplit, CrossValidator
7 from pyspark.mllib.evaluation import MulticlassMetrics
8 from pyspark.ml.classification import LogisticRegression, GBTClassifier, RandomForestClassifier, LinearSVC
9 from pyspark.ml.evaluation import BinaryClassificationEvaluator
10 from time import time
11 from pyspark.context import SparkContext
12 from pyspark.sql.session import SparkSession
13 from pyspark.storagelevel import StorageLevel
14 from pyspark.sql.types import StructType, StructField, IntegerType, StringType, FloatType
15
16 # Configure logging
17 logging.basicConfig(level=logging.INFO, format='%(message)s')

```

- Use following code in Spark-Submit with 'True'.

Code:

```

IS_SPARK_SUBMIT_CLI = True
if IS_SPARK_SUBMIT_CLI:
    sc = SparkContext.getOrCreate()
    spark = SparkSession(sc)

```

```

1 IS_SPARK_SUBMIT_CLI = True
2 if IS_SPARK_SUBMIT_CLI:
3     sc = SparkContext.getOrCreate()
4     spark = SparkSession(sc)

```

-Following Code is automatically generated after successfully uploaded file in new notebook.

Code:

```

# File location and type
file_location = "/FileStore/tables/HI_Small_Trans.csv"

file_type = "csv"

# CSV options
infer_schema = "true"
first_row_is_header = "true"
delimiter = ","

laundryingSchema = StructType([
    StructField("TimeStamp", StringType(), False),
    StructField("From_Bank", IntegerType(), False),
    StructField("From_acc", StringType(), False),

```

```

StructField("To_Bank", IntegerType(), False),
StructField("To_acc", StringType(), False),
StructField("Amount_Received", FloatType(), False),
StructField("Receiving_Currency", StringType(), False),
StructField("Amount_Paid", FloatType(), False),
StructField("Payment_Currency", StringType(), False),
StructField("Payment_Format", StringType(), False),
StructField("Is_Laundering", IntegerType(), False)
])

```

# The applied options are for CSV files. For other file types, these will be ignored.

```

df = spark.read.format(file_type) \
.schema(launderingSchema) \
.option("header", first_row_is_header) \
.option("sep", delimiter) \
.load(file_location)

```

```
display(df)
```

```
#print(df)
```

```

1  # File location and type
2  file_location = "/FileStore/tables/HI_Small_Trans.csv"
3  file_type = "csv"
4
5  # CSV options
6  infer_schema = "true"
7  first_row_is_header = "true"
8  delimiter = ","
9
10 launderingSchema = StructType([
11   StructField("TimeStamp", StringType(), False),
12   StructField("From_Bank", IntegerType(), False),
13   StructField("From_acc", StringType(), False),
14   StructField("To_Bank", IntegerType(), False),
15   StructField("To_acc", StringType(), False),
16   StructField("Amount_Received", FloatType(), False),
17   StructField("Receiving_Currency", StringType(), False),
18   StructField("Amount_Paid", FloatType(), False),
19   StructField("Payment_Currency", StringType(), False),
20   StructField("Payment_Format", StringType(), False),
21   StructField("Is_Laundering", IntegerType(), False)
22 ])

```

```

24 # The applied options are for CSV files. For other file types, these will be ignored.
25 df = spark.read.format(file_type) \
26 .schema(launderingSchema) \
27 .option("header", first_row_is_header) \
28 .option("sep", delimiter) \
29 .load(file_location)
30
31 display(df)

```

## Output:

	TimeStamp	From_Bank	From_acc	To_Bank	To_acc	Amount_Received	Receiving_Currency	Amount_Paid	Paym
1	2022/09/01 00:20	10	8000EBD30	10	8000EBD30	3697.34	US Dollar	3697.34	US
2	2022/09/01 00:20	3208	8000F4580	1	8000F5340	0.01	US Dollar	0.01	US
3	2022/09/01 00:00	3209	8000F4670	3209	8000F4670	14675.57	US Dollar	14675.57	US
4	2022/09/01 00:02	12	8000F5030	12	8000F5030	2806.97	US Dollar	2806.97	US
5	2022/09/01 00:06	10	8000F5200	10	8000F5200	36682.97	US Dollar	36682.97	US
6	2022/09/01 00:03	1	8000F5AD0	1	8000F5AD0	6162.44	US Dollar	6162.44	US
7									

10,000+ rows | Truncated data | 17.83 seconds runtime | Refreshed 8 days ago

## Step 6: Define Schema & get count of laundering data.

```
df = spark.read.format(file_type) \  
.schema(launderingSchema) \  
.option("header", first_row_is_header) \  
.option("sep", delimiter) \  
.load(file_location)
```

```
#print(df)  
display(df)
```

```
#Display the schema  
df.printSchema()
```

```
1 df = spark.read.format(file_type) \  
2 .schema(launderingSchema) \  
3 .option("header", first_row_is_header) \  
4 .option("sep", delimiter) \  
5 .load(file_location)  
6  
7 print(df)  
8  
9 # COMMAND -----  
10  
11 #Display the schema  
12 df.printSchema()
```

```
print(df.count())  
Laundering_df = df.filter(col("Is_Laundering") == 1)  
print(Laundering_df.count())  
test_df = df.filter(col("Is_Laundering") == 0)  
print(test_df.count())
```

```
1 print(df.count())  
2 Laundering_df = df.filter(col("Is_Laundering") == 1)  
3 print(Laundering_df.count())  
4 test_df = df.filter(col("Is_Laundering") == 0)  
5 print(test_df.count())
```

## Step 7: Use of Indexer to transform string data into numeric with new column

```
indexer1 = StringIndexer(inputCol="Receiving_Currency",
outputCol="Receiving_CurrencyIndex")
indexer2 = StringIndexer(inputCol="Payment_Currency",
outputCol="Payment_CurrencyIndex")
indexer3 = StringIndexer(inputCol="Payment_Format", outputCol="Payment_FormatIndex")

df = indexer1.fit(df).transform(df)
df = indexer2.fit(df).transform(df)
df = indexer3.fit(df).transform(df)
```

```
1 indexer1 = StringIndexer(inputCol="Receiving_Currency", outputCol="Receiving_CurrencyIndex")
2 indexer2 = StringIndexer(inputCol="Payment_Currency", outputCol="Payment_CurrencyIndex")
3 indexer3 = StringIndexer(inputCol="Payment_Format", outputCol="Payment_FormatIndex")
4
5
6 df = indexer1.fit(df).transform(df)
7 df = indexer2.fit(df).transform(df)
8 df = indexer3.fit(df).transform(df)
```

Create new dataframe with defining 'Label'

```
df3 = df.select("From_Bank", "To_Bank", "Amount_Received", "Receiving_CurrencyIndex",
"Amount_Paid", "Payment_CurrencyIndex", "Payment_FormatIndex",
col("Is_Laundering").alias("label"))
df3.show(2)
```

```
1 df3 = df.select("From_Bank", "To_Bank", "Amount_Received", "Receiving_CurrencyIndex", "Amount_Paid", "Payment_CurrencyIndex",
"Payment_FormatIndex", col("Is_Laundering").alias("label"))
2 df3.show(2)
```

▶ (1) Spark Jobs

▶ df3: pyspark.sql.dataframe.DataFrame = [From\_Bank: integer, To\_Bank: integer ... 6 more fields]

From_Bank	To_Bank	Amount_Received	Receiving_CurrencyIndex	Amount_Paid	Payment_CurrencyIndex	Payment_FormatIndex	label
10	10	3697.34	0.0	3697.34	0.0	4.0	0
3208	1	0.01	0.0	0.01	0.0	0.0	0

only showing top 2 rows



## Step 8: Data balancing with UnderSampling

```
#Balancing by undersampling
Laundering_df2 = df3.filter(col("label") == 1)
print(Laundering_df2.count())
test_df2 = df3.filter(col("label") == 0)
print(test_df2.count())
balanced_ratio = Laundering_df2.count() / test_df2.count()
print(balanced_ratio)
undersampled_df2 = test_df2.sample(withReplacement=False, fraction=balanced_ratio)
print(undersampled_df2.count())
df3 = Laundering_df2.union(undersampled_df2)
print(df3.count())
```

```
1 #Balancing by undersampling
2 Laundering_df2 = df3.filter(col("label") == 1)
3 print(Laundering_df2.count())
4 test_df2 = df3.filter(col("label") == 0)
5 print(test_df2.count())
6 balanced_ratio = Laundering_df2.count() / test_df2.count()
7 print(balanced_ratio)
8 undersampled_df2 = test_df2.sample(withReplacement=False, fraction=balanced_ratio)
9 print(undersampled_df2.count())
10 df3 = Laundering_df2.union(undersampled_df2)
11 print(df3.count())
```

▶ (12) Spark Jobs

- ▶ Laundering\_df2: pyspark.sql.dataframe.DataFrame = [From\_Bank: integer, To\_Bank: integer ... 6 more fields]
- ▶ test\_df2: pyspark.sql.dataframe.DataFrame = [From\_Bank: integer, To\_Bank: integer ... 6 more fields]
- ▶ undersampled\_df2: pyspark.sql.dataframe.DataFrame = [From\_Bank: integer, To\_Bank: integer ... 6 more fields]
- ▶ df3: pyspark.sql.dataframe.DataFrame = [From\_Bank: integer, To\_Bank: integer ... 6 more fields]

5177  
5073168  
0.0010204668956360207  
5201  
10378

Vector Assembler to form arrays of data for features & Label

```
assembler = VectorAssembler(inputCols = ["From_Bank", "To_Bank", "Amount_Received",  
"Receiving_CurrencyIndex", "Amount_Paid", "Payment_CurrencyIndex",  
"Payment_FormatIndex"], outputCol="features")
```

```
1 assembler = VectorAssembler(inputCols = ["From_Bank", "To_Bank", "Amount_Received", "Receiving_CurrencyIndex", "Amount_Paid",  
"Payment_CurrencyIndex", "Payment_FormatIndex"], outputCol="features")
```

## **Step 9: Split the data into Train(70%) & Test Dataset(30%).**

```
splits = df3.randomSplit([0.7,0.3])
train = splits[0]
test = splits[1].withColumnRenamed("label", "trueLabel")
print ("Training Rows:", train.count(), " Testing Rows:", test.count())
```

```
1 splits = df3.randomSplit([0.7,0.3])
2 train = splits[0]
3 test = splits[1].withColumnRenamed("label", "trueLabel")
4 print ("Training Rows:", train.count(), " Testing Rows:", test.count())
```

► (4) Spark Jobs

►  train: pyspark.sql.dataframe.DataFrame = [From\_Bank: integer, To\_Bank: integer ... 6 more fields]

►  test: pyspark.sql.dataframe.DataFrame = [From\_Bank: integer, To\_Bank: integer ... 6 more fields]

Training Rows: 7205 Testing Rows: 3173

## **Step 10: Defining ALL Algorithms with pipeline & Hyper parameter**

### **1) Logistic Regression: -**

# Define Logistic Regression

```
lr = LogisticRegression(labelCol="label",featuresCol="features",maxIter=10,regParam=0.3)
```

# Combine stages into pipeline

```
pipeline = Pipeline(stages=[assembler, lr])
```

#Define paramGrid

```
paramGrid = ParamGridBuilder() \
.addGrid(lr.regParam, [0.01, 0.1, 1.0]) \
.addGrid(lr.elasticNetParam, [0.0, 0.5, 1.0]) \
.build()
```

# Define Logistic Regression

```
lr = LogisticRegression(labelCol="label",featuresCol="features",maxIter=10,regParam=0.3)
```

# Combine stages into pipeline

```
pipeline = Pipeline(stages=[assembler, lr])
```

#Define paramGrid

```
paramGrid = ParamGridBuilder() \
.addGrid(lr.regParam, [0.01, 0.1, 1.0]) \
.addGrid(lr.elasticNetParam, [0.0, 0.5, 1.0]) \
.build()
```

## 2) LinearSVC: -

```
# Define LinerSVC
lsvc = LinearSVC(labelCol="label",featuresCol="features")

# Combine stages into pipeline
pipeline_lsvc = Pipeline(stages=[assembler, lsvc])

# Define paramGrid
paramGrid_lsvc = (ParamGridBuilder()
.addGrid(lsvc.regParam, [0.01, 0.1, 1.0]) \
.addGrid(lsvc.maxIter, [10, 20, 30]) \
.build())
```

```
1  # Define LinerSVC
2  lsvc = LinearSVC(labelCol="label",featuresCol="features")
3
4  # Combine stages into pipeline
5  pipeline_lsvc = Pipeline(stages=[assembler, lsvc])
6
7  # Define paramGrid
8  paramGrid_lsvc = (ParamGridBuilder()
9  .addGrid(lsvc.regParam, [0.01, 0.1, 1.0]) \
10 .addGrid(lsvc.maxIter, [10, 20, 30]) \
11 .build())
```

## 3) Random Forest: -

```
# Define Random Forest
rf = RandomForestClassifier(labelCol="label",featuresCol="features",numTrees=10, maxDepth=5)

# Combine stages into pipeline
pipeline = Pipeline(stages=[assembler, rf])

# Define paramGrid
paramGrid = ParamGridBuilder() \
.addGrid(rf.numTrees, [10, 20, 30]) \
.addGrid(rf.maxDepth, [3, 5, 8]) \
.build()
```

```
# Define Random Forest
rf = RandomForestClassifier(labelCol="label",featuresCol="features",numTrees=10, maxDepth=5)

# Combine stages into pipeline
pipeline = Pipeline(stages=[assembler, rf])

# Define paramGrid
paramGrid = ParamGridBuilder() \
.addGrid(rf.numTrees, [10, 20, 30]) \
.addGrid(rf.maxDepth, [3, 5, 8]) \
.build()
```

#### 4) Gradient Boosted Tree (GBT): -

```
# Define Gradient Boosted Tree
gbt = GBTClassifier(featuresCol="features", labelCol="label", maxIter=10)

# Combine stages into pipeline
pipeline = Pipeline(stages=[assembler, gbt])

# Define paramGrid
paramGrid = ParamGridBuilder() \
    .addGrid(gbt.maxDepth, [3, 5, 8]) \
    .addGrid(gbt.maxIter, [10, 20, 30]) \
    .build()
```

```
1  # Define Gradient Boosted Tree
2  gbt = GBTClassifier(featuresCol="features", labelCol="label", maxIter=10)
3
4  # Combine stages into pipeline
5  pipeline = Pipeline(stages=[assembler, gbt])
6
7  # Define paramGrid
8  paramGrid = ParamGridBuilder() \
9      .addGrid(gbt.maxDepth, [3, 5, 8]) \
10     .addGrid(gbt.maxIter, [10, 20, 30]) \
11     .build()
12
```

#### Step 11: Model train, calculating time and prediction (Same Code apply to all algorithms)

i) With Train-validation Split

```
# Create a TrainValidator
tv = TrainValidationSplit(estimator=pipeline, evaluator=BinaryClassificationEvaluator(),
    estimatorParamMaps=paramGrid, trainRatio=0.8)
```

```
# Create a TrainValidator
tv = TrainValidationSplit(estimator=pipeline, evaluator=BinaryClassificationEvaluator(), estimatorParamMaps=paramGrid, trainRatio=0.8)
```

#### #Training the model and Calculating its time

```
import time
start_time = time.time() # Start time

tvModel = tv.fit(train)

end_time = time.time()# End time
```

```

print("Model trained!")
# Calculate training time
training_time = end_time - start_time
# Calculate minutes and seconds
minutes = int(training_time // 60)
seconds = int(training_time % 60)

logging.info("Training time: %02d:%02d" % (minutes, seconds))

#Prediction:

prediction = tvModel.transform(test)
predicted = prediction.select("features", "prediction", "probability", "trueLabel")

predicted.show(100, truncate=False)

```

```

1  #Training the model and Calculating its time
2  import time
3  start_time = time.time() # Start time
4
5  tvModel = tv.fit(train)
6
7  end_time = time.time()# End time
8
9  print("Model trained!")
10 # Calculate training time
11 training_time = end_time - start_time
12 # Calculate minutes and seconds
13 minutes = int(training_time // 60)
14 seconds = int(training_time % 60)
15
16 logging.info("Training time: %02d:%02d" % (minutes, seconds))
17
18 #Performance Evaluation:
19 prediction = tvModel.transform(test)
20 predicted = prediction.select("features", "prediction", "probability", "trueLabel")
21
22 predicted.show(100, truncate=False)

```

## ii) With Cross-validation Split

```

# Create a CrossValidator
cv = CrossValidator(estimator=pipeline, evaluator=BinaryClassificationEvaluator(),
estimatorParamMaps=paramGrid, numFolds=3)

```

```

# Create a CrossValidator
cv = CrossValidator(estimator=pipeline, evaluator=BinaryClassificationEvaluator(), estimatorParamMaps=paramGrid, numFolds=3)

```

### #Training the model and Calculating its time

```
import time
start_time = time.time() # Start time

cvModel = cv.fit(train)

end_time = time.time()# End time

print("Model trained!")
# Calculate training time
training_time = end_time - start_time
# Calculate minutes and seconds
minutes = int(training_time // 60)
seconds = int(training_time % 60)

logging.info("Training time: %02d:%02d" % (minutes, seconds))
```

### #Prediction:

```
prediction = cvModel.transform(test)
predicted = prediction.select("features", "prediction", "probability", "trueLabel")

predicted.show(100, truncate=False)
```

```
1  #Training the model and Calculating its time
2  import time
3  start_time = time.time() # Start time
4
5  cvModel = cv.fit(train)
6
7  end_time = time.time()# End time
8
9  print("Model trained!")
10 # Calculate training time
11 training_time = end_time - start_time
12 # Calculate minutes and seconds
13 minutes = int(training_time // 60)
14 seconds = int(training_time % 60)
15
16 logging.info("Training time: %02d:%02d" % (minutes, seconds))
17
18 #Performance Evaluation:
19 prediction = cvModel.transform(test)
20 predicted = prediction.select("features", "prediction", "probability", "trueLabel")
21
22 predicted.show(100, truncate=False)
```

## **Step 12: Evaluator & Metrics Code ( Recall, Precision, AUC) for TVS/CV**

```
tp = float(predicted.filter("prediction == 1.0 AND truelabel == 1").count())
fp = float(predicted.filter("prediction == 1.0 AND truelabel == 0").count())
tn = float(predicted.filter("prediction == 0.0 AND truelabel == 0").count())
fn = float(predicted.filter("prediction == 0.0 AND truelabel == 1").count())
metrics = spark.createDataFrame([
    ("TP", tp),
    ("FP", fp),
    ("TN", tn),
    ("FN", fn),
    ("Precision", tp / (tp + fp)),
    ("Recall", tp / (tp + fn)),["metric", "value"])

metrics.show()

evaluator = BinaryClassificationEvaluator(labelCol="trueLabel", rawPredictionCol="rawPrediction",
metricName="areaUnderROC")
auc = evaluator.evaluate(prediction)
print("AUC = ", auc)
```

```
1  tp = float(predicted.filter("prediction == 1.0 AND truelabel == 1").count())
2  fp = float(predicted.filter("prediction == 1.0 AND truelabel == 0").count())
3  tn = float(predicted.filter("prediction == 0.0 AND truelabel == 0").count())
4  fn = float(predicted.filter("prediction == 0.0 AND truelabel == 1").count())
5  metrics = spark.createDataFrame([
6      ("TP", tp),
7      ("FP", fp),
8      ("TN", tn),
9      ("FN", fn),
10     ("Precision", tp / (tp + fp)),
11     ("Recall", tp / (tp + fn)),["metric", "value"])
12
13  metrics.show()
14
15  evaluator = BinaryClassificationEvaluator(labelCol="trueLabel", rawPredictionCol="rawPrediction", metricName="areaUnderROC")
16  auc = evaluator.evaluate(prediction)
17  print("AUC = ", auc)
```

## **Step 13: Feature Importance: (for example GBT-TVS in our case)**

```
# Get the best model from TrainValidationSplit
best_model = tvModel.bestModel
gbtModel = best_model.stages[-1]

# Feature importance
import pandas as pd
featureImp = pd.DataFrame(list(zip(assembler.getInputCols(), gbtModel.featureImportances)), columns=["feature",
"importance"])
featureImp = featureImp.sort_values(by="importance", ascending=False)
```

```
# Get the best model from TrainValidationSplit
best_model = tvModel.bestModel
gbtModel = best_model.stages[-1]

# Feature importance
import pandas as pd
featureImp = pd.DataFrame(list(zip(Assembler.getInputCols(), gbtModel.featureImportances)), columns=["feature", "importance"])
featureImp = featureImp.sort_values(by="importance", ascending=False)
```

## Steps to run code at Spark ML:

- 1) Download Big Datafile- HI\_Medium\_Trans.csv on GitBash through Linux command.

```
$ scp C:/Users/sshah82/Desktop/BigData5560/IBMDData/archive/HI_Medium_Trans.csv
sshah82@129.153.214.22:
```

```
AD+sshah82@STU-PF2XCGW9 MINGW64 /
$ scp C:/Users/sshah82/Desktop/BigData5560/IBMDData/archive/HI
_Medium_Trans.csv sshah82@129.153.214.22:
sshah82@129.153.214.22's password:
HI_Medium_Trans.csv      100% 2891MB   1.2MB/s   39:59
AD+sshah82@STU-PF2XCGW9 MINGW64 /
```

- 2) Move the .csv file to Hadoop file system.

```
bash-4.2$ hdfs dfs -put HI_Medium_Trans.csv
```

```
AD+sshah82@STU-PF2XCGW9 MINGW64 /
$ ssh sshah82@129.153.214.22
sshah82@129.153.214.22's password:
Last login: Sat May 11 18:17:39 2024 from 130.182.24.105
-bash-4.2$ hdfs dfs -put HI_Medium_Trans.csv
-bash-4.2$ hdfs dfs -ls
Found 1 items
-rw-r--r--  3 sshah82 hdfs 3031783420 2024-05-14 15:41 HI_Medium_Trans.csv
```

- 3) Store all databricks source files in .py format on local system.

For Ex.

```
scp
C:/Users/sshah82/Desktop/BigData5560/Final_Project/UnderSampling/AML_LogisticRe
gression_TVFinal_UnderSample.py sshah82@129.153.214.22:
```

```
AD+sshah82@STU-PF2XCGW9 MINGW64 /
$ scp C:/Users/sshah82/Desktop/BigData5560/Final_Project/UnderSampling/AML_LogisticReg
ression_TVFinal_UnderSample.py sshah82@129.153.214.22:
sshah82@129.153.214.22's password:
AML_LogisticRegression_TVFinal_UnderSample.py      100% 5902    59.3kB/s   00:00
```



- 4) Get all .py files on GitBash through linux commands.
- 5) All .py files has given path to fetch .csv file from Hadoop file system.
- 6) Run all .py files on spark-submit.

```
-bash-4.2$ spark-submit AML_LogisticRegression_TVFinal_UnderSample.py
```

```
-bash-4.2$ spark-submit AML_LogisticRegression_TVFinal_UnderSample.py
```

## Results in SPARK-SUBMIT:

### Logistic Regression

Train Validation Split

Model Train time:

```

MINGW64/c:/Users/ssah82
mory (size: 33.7 KiB, free: 366.0 MiB)
24/05/09 12:52:19 INFO BlockManagerInfo: Removed broadcast_436_piece0 on bigdaiwn0.sub03291929060.trainingvcn.oraclevcn.com:19781 in me
mory (size: 33.7 KiB, free: 366.0 MiB)
24/05/09 12:52:19 INFO MemoryStore: Block broadcast_445_piece0 stored as bytes in memory (estimated size 56.9 KiB, free 359.6 MiB)
24/05/09 12:52:19 INFO BlockManagerInfo: Added broadcast_445_piece0 in memory on bigdaimn0.sub03291929060.trainingvcn.oraclevcn.com:132
85 (size: 56.9 KiB, free: 365.7 MiB)
24/05/09 12:52:19 INFO SparkContext: Created broadcast 445 from rdd at ClassificationSummary.scala:191
24/05/09 12:52:19 INFO FileSourceScanExec: Planning scan with bin packing, max size: 134217728 bytes, open cost is considered as scanni
ng 4194304 bytes.
24/05/09 12:52:19 INFO MemoryStore: Block broadcast_446 stored as values in memory (estimated size 568.0 KiB, free 359.0 MiB)
24/05/09 12:52:19 INFO MemoryStore: Block broadcast_446_piece0 stored as bytes in memory (estimated size 56.9 KiB, free 359.0 MiB)
24/05/09 12:52:19 INFO BlockManagerInfo: Added broadcast_446_piece0 in memory on bigdaimn0.sub03291929060.trainingvcn.oraclevcn.com:132
85 (size: 56.9 KiB, free: 365.6 MiB)
24/05/09 12:52:19 INFO SparkContext: Created broadcast 446 from rdd at ClassificationSummary.scala:191
24/05/09 12:52:19 INFO FileSourceScanExec: Planning scan with bin packing, max size: 134217728 bytes, open cost is considered as scanni
ng 4194304 bytes.
Model trained!
Training time: 02:26
24/05/09 12:52:19 INFO FileSourceStrategy: Pushed Filters:
24/05/09 12:52:19 INFO FileSourceStrategy: Post-Scan Filters:
24/05/09 12:52:19 INFO FileSourceStrategy: Output Data Schema: struct<From_Bank: int, To_Bank: int, Amount_Received: float, Receiving_C

```

Metrics:

```

24/05/09 12:54:18 INFO CodeGenerator: Code generated in 8.472779 ms
+-----+
| metric | value |
+-----+
| TP | 8752.0 |
| FP | 2976.0 |
| TN | 7359.0 |
| FN | 1704.0 |
| Precision | 0.7462482946793997 |
| Recall | 0.8370313695485846 |
+-----+
*****TrainValidator-Results-LR*****
24/05/09 12:54:18 INFO FileSourceStrategy: Pushed Filters:
24/05/09 12:54:18 INFO FileSourceStrategy: Post-Scan Filters:
24/05/09 12:54:19 INFO DAGScheduler: Killing all running tasks in stage 303: Stage finished
24/05/09 12:54:45 INFO DAGScheduler: Job 189 finished: collect at AreaUnderCurve.scala:44, took 0.144032 s
24/05/09 12:54:45 INFO MapPartitionsRDD: Removing RDD 1195 from persistence list
AUC = 0.708159693496631
24/05/09 12:54:45 INFO BlockManager: Removing RDD 1195
24/05/09 12:54:45 INFO SparkContext: Invoking stop() from shutdown hook
24/05/09 12:54:45 INFO AbstractConnector: Stopped Spark@3753d2ff4[UTTP/1.1 (http/1.1)]@0.0.0.0:4041

```

## CrossValidation Split –

### Model Train time:

```
MINGW64/c:/Users/ssshah82
emory (size: 33.3 KiB, free: 365.9 MiB)
24/05/09 13:13:27 INFO BlockManagerInfo: Removed broadcast_1391_piece0 on bigdaiwn1.sub03291929060.trainingvcn.oraclevcn.com:18191 in m
emory (size: 33.3 KiB, free: 365.9 MiB)
24/05/09 13:13:27 INFO BlockManagerInfo: Removed broadcast_1391_piece0 on bigdaiwn2.sub03291929060.trainingvcn.oraclevcn.com:22803 in m
emory (size: 33.3 KiB, free: 365.9 MiB)
24/05/09 13:13:27 INFO BlockManagerInfo: Removed broadcast_1391_piece0 on bigdaiwn2.sub03291929060.trainingvcn.oraclevcn.com:30379 in m
emory (size: 33.3 KiB, free: 365.9 MiB)
24/05/09 13:13:27 INFO BlockManagerInfo: Removed broadcast_1391_piece0 on bigdaiwn2.sub03291929060.trainingvcn.oraclevcn.com:19199 in m
emory (size: 33.3 KiB, free: 365.9 MiB)
24/05/09 13:13:27 INFO BlockManagerInfo: Removed broadcast_1391_piece0 on bigdaiwn0.sub03291929060.trainingvcn.oraclevcn.com:11823 in m
emory (size: 33.3 KiB, free: 365.9 MiB)
24/05/09 13:13:27 INFO BlockManagerInfo: Removed broadcast_1391_piece0 on bigdaiwn0.sub03291929060.trainingvcn.oraclevcn.com:31215 in m
emory (size: 33.3 KiB, free: 365.9 MiB)
Model trained!
24/05/09 13:13:27 INFO BlockManagerInfo: Removed broadcast_1391_piece0 on bigdaiwn0.sub03291929060.trainingvcn.oraclevcn.com:33389 in m
emory (size: 33.3 KiB, free: 365.9 MiB)
Training time: 05:18
24/05/09 13:13:27 INFO FileSourceStrategy: Pushed Filters:
24/05/09 13:13:27 INFO FileSourceStrategy: Post-Scan Filters:
24/05/09 13:13:27 INFO FileSourceStrategy: Output Data Schema: struct<From Bank: int, To Bank: int, Amount Received: float, Receiving C
```

### Metrics:

```
24/05/09 13:15:26 INFO YarnScheduler: Killing all running tasks in stage 1116: stage finished
24/05/09 13:15:26 INFO DAGScheduler: Job 529 finished: showString at NativeMethodAccessorImpl.java:0, took 0.898113 s
24/05/09 13:15:26 INFO CodeGenerator: Code generated in 8.097895 ms
+-----+
| metric | value |
+-----+
| TP     | 8622.0 |
| FP     | 3567.0 |
| TN     | 7055.0 |
| FN     | 1899.0 |
| Precision | 0.7073590942653212 |
| Recall  | 0.8195038494439693 |
+-----+
*****CrossValidator-Results-LR*****
24/05/09 13:15:26 INFO FileSourceStrategy: Pushed Filters:
24/05/09 13:15:26 INFO FileSourceStrategy: Post-Scan Filters:
24/05/09 13:15:26 INFO FileSourceStrategy: Output Data Schema: struct<From Bank: int, To Bank: int, Amount Received: float, Receiving C
24/05/09 13:15:54 INFO DAGScheduler: Job 533 is finished. Cancelling potential speculative or zombie tasks for this job
24/05/09 13:15:54 INFO YarnScheduler: Killing all running tasks in stage 1127: stage finished
24/05/09 13:15:54 INFO DAGScheduler: Job 533 finished: collect at AreaUnderCurve.scala:44, took 0.123263 s
24/05/09 13:15:54 INFO MapPartitionsRDD: Removing RDD 3307 from persistence list
AUC = 0.7085023719316798
24/05/09 13:15:54 INFO BlockManager: Removing RDD 3307
24/05/09 13:15:54 INFO SparkContext: Invoking stop() from shutdown hook
24/05/09 13:15:54 INFO AbstractConnector: Stopped spark@752d3ff4[HTTP/1.1, (http/1.1)]{0.0.0.0:4041}
```

## Linear SVC:

### TrainValidation Split –

### Model Train time:

```
MINGW64/c:/Users/ssshah82
24/05/09 14:23:30 INFO FileSourceScanExec: Planning scan with bin packing, max size: 134217728 bytes, open cost is con
sidered as scanning 4194304 bytes.
24/05/09 14:23:30 INFO MemoryStore: Block broadcast_1541 stored as values in memory (estimated size 567.9 KiB, free 35
8.9 MiB)
24/05/09 14:23:30 INFO MemoryStore: Block broadcast_1541_piece0 stored as bytes in memory (estimated size 56.9 KiB, fr
ee 358.9 MiB)
24/05/09 14:23:30 INFO BlockManagerInfo: Added broadcast_1541_piece0 in memory on bigdaiwn0.sub03291929060.trainingvcn
.oraclevcn.com:15835 (size: 56.9 KiB, free: 365.6 MiB)
24/05/09 14:23:30 INFO SparkContext: Created broadcast 1541 from rdd at classificationSummary.scala:191
24/05/09 14:23:30 INFO FileSourceScanExec: Planning scan with bin packing, max size: 134217728 bytes, open cost is con
sidered as scanning 4194304 bytes.
24/05/09 14:23:30 INFO Instrumentation: [3fe22ce8] training finished
Model trained!
Training time: 03:20
24/05/09 14:23:31 INFO FileSourceStrategy: Pushed Filters:
24/05/09 14:23:31 INFO FileSourceStrategy: Post-Scan Filters:
```

## Metrics:

```
24/05/09 14:25:29 INFO DAGScheduler: Job 550 finished: showString at NativeMethodAccessorImpl.java:0, took 0.698264 s
24/05/09 14:25:29 INFO codeGenerator: code generated in 11.724141 ms
+-----+
| metric | value |
+-----+
| TP | 9235.0 |
| FP | 4191.0 |
| TN | 6336.0 |
| FN | 1459.0 |
| Precision | 0.6878444808580366 |
| Recall | 0.8635683560875257 |
+-----+
*****TrainValidator-Results-LinearSVC*****
24/05/09 14:25:29 INFO BlockManagerInfo: Removed broadcast_1551_piece0 on bigdaimn0.sub03291929060.trainingvcn.oraclelevcn.com:12423 (size: 56.9 KiB, free: 364.5 MiB)
24/05/09 14:25:56 INFO MapPartitionsRDD: Removing RDD 2658 from persistence list
24/05/09 14:25:56 INFO BlockManager: Removing RDD 2658
AUC = 0.7046126271008766
24/05/09 14:25:56 INFO SparkContext: Invoking stop() from shutdown hook
24/05/09 14:25:56 INFO AbstractConnector: stopped spark@752d3ff4{HTTP/1.1, (http/1.1)}{0.0.0.0:4041}
```

## CrossValidation Split –

### Model Train time:

```
MINGW64/c:/Users/ssah82
24/05/09 14:40:00 INFO BlockManagerInfo: Added broadcast_4665_piece0 in memory on bigdaimn0.sub03291929060.trainingvcn.oraclelevcn.com:12423 (size: 56.9 KiB, free: 364.5 MiB)
24/05/09 14:40:00 INFO SparkContext: Created broadcast 4665 from rdd at ClassificationSummary.scala:191
24/05/09 14:40:00 INFO FileSourceScanExec: Planning scan with bin packing, max size: 134217728 bytes, open cost is considered as scanning 4194304 bytes.
24/05/09 14:40:00 INFO MemoryStore: Block broadcast_4666 stored as values in memory (estimated size 567.9 KiB, free 351.4 MiB)
24/05/09 14:40:00 INFO MemoryStore: Block broadcast_4666_piece0 stored as bytes in memory (estimated size 56.9 KiB, free 351.4 MiB)
24/05/09 14:40:00 INFO BlockManagerInfo: Added broadcast_4666_piece0 in memory on bigdaimn0.sub03291929060.trainingvcn.oraclelevcn.com:12423 (size: 56.9 KiB, free: 364.5 MiB)
24/05/09 14:40:00 INFO SparkContext: Created broadcast 4666 from rdd at ClassificationSummary.scala:191
24/05/09 14:40:00 INFO FileSourceScanExec: Planning scan with bin packing, max size: 134217728 bytes, open cost is considered as scanning 4194304 bytes.
24/05/09 14:40:00 INFO Instrumentation: [e3c68a77] training finished
Model trained!
Training time: 08:05
24/05/09 14:40:01 INFO FileSourceStrategy: Pushed Filters:
24/05/09 14:40:01 INFO FileSourceStrategy: Post-Scan Filters:
```

## Metrics:

```
24/05/09 14:42:03 INFO DAGScheduler: Job 1619 finished: showString at NativeMethodAccessorImpl.java:0, took 0.865177 s
24/05/09 14:42:03 INFO codeGenerator: code generated in 9.164766 ms
+-----+
| metric | value |
+-----+
| TP | 9174.0 |
| FP | 4169.0 |
| TN | 6339.0 |
| FN | 1358.0 |
| Precision | 0.6875515251442704 |
| Recall | 0.8710596278009874 |
+-----+
*****CrossValidator-Results-LinearSVC*****
24/05/09 14:42:03 INFO FileSourceStrategy: Pushed Filters:
24/05/09 14:42:03 INFO FileSourceStrategy: Post-Scan Filters:
24/05/09 14:42:30 INFO MapPartitionsRDD: Removing RDD 7676 from persistence list
24/05/09 14:42:30 INFO BlockManager: Removing RDD 7676
AUC = 0.7121164877399397
24/05/09 14:42:30 INFO BlockManagerInfo: Removed broadcast_4684_piece0 on bigdaimn0.sub03291929060.trainingvcn.oraclelevcn.com:12423 in memory (size: 56.9 KiB, free: 365.9 MiB)
24/05/09 14:42:30 INFO BlockManagerInfo: Removed broadcast_4684_piece0 on bigdaimn1.sub03291929060.trainingvcn.oraclelevcn.com:16575 in m
```

## Random Forest

TrainValidation Split –

Model Train time:

```
MINGW64/c/Users/ssah82
24/05/09 13:31:51 INFO FileSourceStrategy: Output Data Schema: struct<From_Bank: int, To_Bank: int, Amount_Received: float, Receiving_Currency: string, Amount_Paid: float ... 6 more fields>
24/05/09 13:31:51 INFO CodeGenerator: Code generated in 26.913732 ms
24/05/09 13:31:51 INFO MemoryStore: Block broadcast_351 stored as values in memory (estimated size 568.0 KiB, free 358.6 MiB)
24/05/09 13:31:51 INFO MemoryStore: Block broadcast_351_piece0 stored as bytes in memory (estimated size 56.8 KiB, free 358.5 MiB)
24/05/09 13:31:51 INFO BlockManagerInfo: Added broadcast_351_piece0 in memory on bigdaimn0.sub03291929060.trainingvcn.oraclevcn.com:12853 (size: 56.8 KiB, free: 365.4 MiB)
24/05/09 13:31:51 INFO SparkContext: Created broadcast 351 from rdd at ClassificationSummary.scala:191
24/05/09 13:31:51 INFO FileSourceScanExec: Planning scan with bin packing, max size: 134217728 bytes, open cost is considered as scanning 4194304 bytes.
24/05/09 13:31:51 INFO MemoryStore: Block broadcast_352 stored as values in memory (estimated size 568.0 KiB, free 358.0 MiB)
24/05/09 13:31:51 INFO MemoryStore: Block broadcast_352_piece0 stored as bytes in memory (estimated size 56.8 KiB, free 357.9 MiB)
24/05/09 13:31:51 INFO BlockManagerInfo: Added broadcast_352_piece0 in memory on bigdaimn0.sub03291929060.trainingvcn.oraclevcn.com:12853 (size: 56.8 KiB, free: 365.4 MiB)
24/05/09 13:31:51 INFO SparkContext: Created broadcast 352 from rdd at ClassificationSummary.scala:191
24/05/09 13:31:51 INFO FileSourceScanExec: Planning scan with bin packing, max size: 134217728 bytes, open cost is considered as scanning 4194304 bytes.
24/05/09 13:31:51 INFO Instrumentation: [80e32abc] training finished
Model trained!
Training time: 03:54
24/05/09 13:31:51 INFO FileSourceStrategy: Pushed Filters:
```

Metrics & Feature Importance:

```
24/05/09 13:34:00 INFO CodeGenerator: Code generated in 11.979997 ms
+-----+-----+
| metric | value |
+-----+-----+
| TP | 9762.0 |
| FP | 1506.0 |
| TN | 9113.0 |
| FN | 644.0 |
| Precision | 0.8663471778487752 |
| Recall | 0.9381126273303864 |
+-----+-----+

*****TrainValidator-Results-RF*****
24/05/09 13:34:00 INFO BlockManagerInfo: Removed broadcast_367_piece0 on bigdaimn0.sub03291929060.trainingvcn.oraclevcn.com:12853 in memory (size: 701.3 KiB, free: 365.4 MiB)
24/05/09 13:34:29 INFO BlockManager: Removing RDD 1091
AUC = 0.964778717709856
+-----+-----+
| feature | importance |
+-----+-----+
| 6 | Payment_FormatIndex | 0.641433 |
| 4 | Amount_Paid | 0.117367 |
| 0 | From_Bank | 0.113267 |
| 2 | Amount_Received | 0.067685 |
| 5 | Payment_CurrencyIndex | 0.021178 |
| 1 | To_Bank | 0.020979 |
| 3 | Receiving_CurrencyIndex | 0.018091 |
+-----+-----+
24/05/09 13:34:30 INFO SparkContext: Invoking stop() from shutdown hook
24/05/09 13:34:30 INFO AbstractConnector: stopped spark@752d3ff4[HTTP/1.1 - (http/1.1)](0.0.0.0:4041)
```

CrossValidation Split –

Model Train time:

```
MINGW64/c/Users/ssah82
24/05/09 13:54:40 INFO BlockManagerInfo: Added broadcast_881_piece0 in memory on bigdaimn0.sub03291929060.trainingvcn.oraclevcn.com:28815 (size: 56.8 KiB, free: 365.0 MiB)
24/05/09 13:54:40 INFO SparkContext: Created broadcast 881 from rdd at ClassificationSummary.scala:191
24/05/09 13:54:40 INFO FileSourceScanExec: Planning scan with bin packing, max size: 134217728 bytes, open cost is considered as scanning 4194304 bytes.
24/05/09 13:54:40 INFO MemoryStore: Block broadcast_882 stored as values in memory (estimated size 568.0 KiB, free 352.2 MiB)
24/05/09 13:54:40 INFO MemoryStore: Block broadcast_882_piece0 stored as bytes in memory (estimated size 56.8 KiB, free 352.2 MiB)
24/05/09 13:54:40 INFO BlockManagerInfo: Added broadcast_882_piece0 in memory on bigdaimn0.sub03291929060.trainingvcn.oraclevcn.com:28815 (size: 56.8 KiB, free: 364.9 MiB)
24/05/09 13:54:40 INFO SparkContext: Created broadcast 882 from rdd at ClassificationSummary.scala:191
24/05/09 13:54:40 INFO FileSourceScanExec: Planning scan with bin packing, max size: 134217728 bytes, open cost is considered as scanning 4194304 bytes.
24/05/09 13:54:40 INFO Instrumentation: [81031fc9] training finished
Model trained!
Training time: 06:47
24/05/09 13:54:40 INFO FileSourceStrategy: Pushed Filters:
```



## Metrics & Feature Importance:

```
24/05/09 13:56:45 INFO CodeGenerator: Code generated in 8.162877 ms
+-----+
| metric | value |
+-----+
| TP | 9966.0 |
| FP | 1528.0 |
| TN | 9138.0 |
| FN | 676.0 |
| Precision | 0.8670610753436575 |
| Recall | 0.9364781056192445 |
+-----+

*****CrossValidator-Results-RF*****
24/05/09 13:56:46 INFO FileSourceStrategy: Pushed Filters:
24/05/09 13:57:14 INFO MapPartitionsRDD: Removing RDD 2685 from persistence list
24/05/09 13:57:14 INFO BlockManager: Removing RDD 2685
AUC = 0.9655287093974663
+-----+
| feature | importance |
+-----+
| 6 Payment_FormatIndex | 0.670477 |
| 0 From_Bank | 0.109920 |
| 4 Amount_Paid | 0.090222 |
| 2 Amount_Received | 0.078054 |
| 1 To_Bank | 0.018127 |
| 3 Receiving_CurrencyIndex | 0.017692 |
| 5 Payment_CurrencyIndex | 0.015508 |
+-----+
24/05/09 13:57:14 INFO SparkContext: Invoking stop() from shutdown hook
24/05/09 13:57:14 INFO AbstractConnector: Stopped Spark@752d3ff4(HTTP/1.1, (http/1.1)){0.0.0.0:4041}
```

## Gradient Boosted Tree

TrainValidation Split –

Model Train time:

```
MINGW64/c:/Users/ssahab2
emory (size: 1122.0 B, free: 360.1 MiB)
24/05/09 14:57:55 INFO MapPartitionsRDD: Removing RDD 5290 from persistence list
24/05/09 14:57:55 INFO BlockManagerInfo: Removed broadcast_2858_piece0 on bigdaiwn1.sub03291929060.trainingvcn.oraclevcn.com:13865 in m
emory (size: 1122.0 B, free: 360.1 MiB)
24/05/09 14:57:55 INFO BlockManagerInfo: Removed broadcast_2858_piece0 on bigdaiwn1.sub03291929060.trainingvcn.oraclevcn.com:35761 in m
emory (size: 1122.0 B, free: 360.1 MiB)
24/05/09 14:57:55 INFO BlockManagerInfo: Removed broadcast_2858_piece0 on bigdaiwn1.sub03291929060.trainingvcn.oraclevcn.com:36951 in m
emory (size: 1122.0 B, free: 360.1 MiB)
24/05/09 14:57:55 INFO BlockManagerInfo: Removed broadcast_2858_piece0 on bigdaiwn1.sub03291929060.trainingvcn.oraclevcn.com:12877 in m
emory (size: 1122.0 B, free: 360.1 MiB)
24/05/09 14:57:55 INFO BlockManagerInfo: Removed broadcast_2858_piece0 on bigdaiwn0.sub03291929060.trainingvcn.oraclevcn.com:18261 in m
emory (size: 1122.0 B, free: 359.9 MiB)
24/05/09 14:57:55 INFO BlockManagerInfo: Removed broadcast_2858_piece0 on bigdaiwn0.sub03291929060.trainingvcn.oraclevcn.com:33991 in m
emory (size: 1122.0 B, free: 359.8 MiB)
24/05/09 14:57:55 INFO BlockManagerInfo: Removed broadcast_2858_piece0 on bigdaiwn2.sub03291929060.trainingvcn.oraclevcn.com:31913 in m
emory (size: 1122.0 B, free: 360.0 MiB)
24/05/09 14:57:55 INFO BlockManagerInfo: Removed broadcast_2858_piece0 on bigdaiwn2.sub03291929060.trainingvcn.oraclevcn.com:17991 in m
emory (size: 1122.0 B, free: 360.0 MiB)
24/05/09 14:57:55 INFO BlockManager: Removing RDD 5290
24/05/09 14:57:55 INFO BlockManagerInfo: Removed broadcast_2858_piece0 on bigdaiwn0.sub03291929060.trainingvcn.oraclevcn.com:17833 in m
emory (size: 1122.0 B, free: 360.2 MiB)
24/05/09 14:57:55 INFO MapPartitionsRDD: Removing RDD 5318 from persistence list
24/05/09 14:57:55 INFO MapPartitionsRDD: Removing RDD 5346 from persistence list
24/05/09 14:57:55 INFO BlockManager: Removing RDD 5346
24/05/09 14:57:55 INFO Instrumentation: [46b0396c] {"numFeatures":7}
24/05/09 14:57:55 INFO Instrumentation: [46b0396c] training finished
Model trained!
Training time: 07:14
24/05/09 14:57:55 INFO BlockManagerInfo: Removed broadcast_3479_piece0 on bigdaimn0.sub03291929060.trainingvcn.oraclevcn.com:19535 in m
```

## Metrics & Feature Importance:

```
24/05/09 15:00:03 INFO CodeGenerator: Code generated in 16.561249 ms
+-----+
| metric | value |
+-----+
| TP | 10139.0 |
| FP | 1522.0 |
| TN | 9023.0 |
| FN | 389.0 |
| Precision | 0.8694794614527056 |
| Recall | 0.9630509118541033 |
+-----+

*****TrainValidator-Results-GB*****
24/05/09 15:00:03 INFO FileSourceStrategy: Pushed Filters:
24/05/09 15:00:33 INFO DAGScheduler: Job 1221 finished: collect at AreaUnderCurve.scala:44, took 0.126374 s
24/05/09 15:00:33 INFO MapPartitionsRDD: Removing RDD 5431 from persistence list
24/05/09 15:00:33 INFO BlockManager: Removing RDD 5431
AUC = 0.970287326099896
24/05/09 15:00:33 INFO BlockManagerInfo: Removed broadcast_3511_piece0 on bigdaimn0.sub03291929060.trainingvcn.oraclevcn.com:19535 in m
emory (size: 3.4 KiB, free: 362.1 MiB)
+-----+
| feature | importance |
+-----+
| 6 Payment_FormatIndex | 0.618812 |
| 0 From_Bank | 0.137167 |
| 2 Amount_Received | 0.062771 |
| 3 Receiving_CurrencyIndex | 0.060838 |
| 4 Amount_Paid | 0.048353 |
| 5 Payment_CurrencyIndex | 0.036077 |
| 1 To_Bank | 0.035981 |
+-----+
24/05/09 15:00:33 INFO SparkContext: Invoking stop() from shutdown hook
24/05/09 15:00:33 INFO AbstractConnector: Stopped Spark@752d3ff4(HTTP/1.1, (http/1.1)){0.0.0.0:4041}
```

## CrossValidation Split –

### Model Train time:

```
24/05/09 15:29:09 INFO BlockManagerInfo: Removed broadcast_8464_piece0 on bigdaiwn2.sub03291929060.trainingvcn.oraclevcn.com:22267 in m
emory (size: 1104.0 B, free: 340.3 MiB)
24/05/09 15:29:09 INFO BlockManagerInfo: Removed broadcast_8464_piece0 on bigdaiwn0.sub03291929060.trainingvcn.oraclevcn.com:30909 in m
emory (size: 1104.0 B, free: 340.4 MiB)
24/05/09 15:29:09 INFO BlockManagerInfo: Removed broadcast_8464_piece0 on bigdaiwn0.sub03291929060.trainingvcn.oraclevcn.com:24863 in m
emory (size: 1104.0 B, free: 340.4 MiB)
24/05/09 15:29:09 INFO BlockManagerInfo: Removed broadcast_8464_piece0 on bigdaiwn2.sub03291929060.trainingvcn.oraclevcn.com:14603 in m
emory (size: 1104.0 B, free: 340.3 MiB)
24/05/09 15:29:09 INFO BlockManagerInfo: Removed broadcast_8464_piece0 on bigdaiwn0.sub03291929060.trainingvcn.oraclevcn.com:34141 in m
emory (size: 1104.0 B, free: 340.3 MiB)
24/05/09 15:29:09 INFO BlockManagerInfo: Removed broadcast_8464_piece0 on bigdaiwn0.sub03291929060.trainingvcn.oraclevcn.com:20509 in m
emory (size: 1104.0 B, free: 340.3 MiB)
24/05/09 15:29:09 INFO MapPartitionsRDD: Removing RDD 14042
24/05/09 15:29:09 INFO MapPartitionsRDD: Removing RDD 14070
24/05/09 15:29:09 INFO Instrumentation: [dfb0c6d5] {"numFeatures":7}
24/05/09 15:29:09 INFO Instrumentation: [dfb0c6d5] training finished
Model trained!
Training time: 17:17
24/05/09 15:29:09 INFO FileSourceStrategy: Pushed Filters:
24/05/09 15:29:09 INFO FileSourceStrategy: Post-Scan Filters:
```

### Metrics & Feature Importance:

```
24/05/09 15:31:18 INFO BlockScheduler: Job 315 finished: showString at NativeMethodAccessorImpl.java:0, took 0.309730 s
24/05/09 15:31:28 INFO CodeGenerator: Code generated in 8.913699 ms

+-----+
| metric | value |
+-----+
| TP | 10049.0 |
| FP | 1563.0 |
| TN | 8960.0 |
| FN | 501.0 |
| Precision | 0.8653978642783328 |
| Recall | 0.9525118483412323 |
+-----+

*****CrossValidator-Results-GBT*****
24/05/09 15:31:28 INFO FileSourceStrategy: Pushed Filters:
24/05/09 15:31:28 INFO FileSourceStrategy: Post-Scan Filters:

24/05/09 15:32:00 INFO BlockManager: Removing RDD 14155
AUC = 0.9684044248819894
+-----+
| feature | importance |
+-----+
| 6 | Payment_FormatIndex | 0.609271 |
| 0 | From_Bank | 0.138384 |
| 2 | Amount_Received | 0.081831 |
| 3 | Receiving_CurrencyIndex | 0.060288 |
| 4 | Amount_Paid | 0.038671 |
| 5 | Payment_CurrencyIndex | 0.037427 |
| 1 | To_Bank | 0.034127 |
+-----+
24/05/09 15:32:00 INFO SparkContext: Invoking stop() from shutdown hook
```

## Summary:

AML- Algorithms with UnderSampling								
Algorithm	TP	FP	TN	FN	Precision	Recall	AUC	Time
Train Validation Split								
Logistic Regression	8752	2976	7359	1704	0.7462	0.8370	0.7082	2.26m
LinearSVC	9235	4191	6336	1459	0.6878	0.8635	0.7046	3.20m
Random Forest	9762	1506	9113	644	0.8663	0.9381	0.9648	3.54m
Gradient Boosted Tree	10139	1522	9023	389	0.8695	0.9631	0.9703	7.14m
Cross Validation Split								
Logistic Regression	8622	3567	7055	1899	0.7074	0.8195	0.7085	5.18m
LinearSVC	9174	4169	6339	1358	0.6876	0.8711	0.7121	8.05m
Random Forest	9966	1528	9138	676	0.8671	0.9365	0.9655	6.47m
Gradient Boosted Tree	10049	1563	8960	501	0.8654	0.9525	0.9684	17.17m

### Feature Importance for GBT-Train Validation Split

feature	importance
Payment_FormatIndex	0.618812
From_Bank	0.137167
Amount_Received	0.062771
Receiving_CurrencyIndex	0.060838
Amount_Paid	0.048353
Payment_CurrencyIndex	0.036077
To_Bank	0.035981

## Conclusion:

- Best Model is **Gradient Boosted Tree (GBT) with Train Validation Split (TVS)** due to highest Recall and AUC values with less time.
- ‘Payment Format’, specifically ACH type, is the most important feature responsible for money laundering instances.

	GBT - TV
Recall	0.9631
AUC	0.9703
Time	7.14m

## References:

1. URL of Data Source:

<https://www.kaggle.com/datasets/ealtman2019/ibm-transactions-for-anti-money-laundering-aml>

2. URL of GitHub:

[https://github.com/ssarkar4/AntiMoneyLaundering\\_BigData](https://github.com/ssarkar4/AntiMoneyLaundering_BigData)

3. URL of References:

<https://spark.apache.org/docs/2.2.0/ml-pipeline.html>

<https://spark.apache.org/docs/latest/ml-tuning.html>