IBM TRANSACTIONS FOR ANTI MONEY LAUNDERING (AML)

CIS 5560 INTRO TO BIG DATA SCIENCE

Professor: Jongwook Woo

Submitted by:

Lekha Ajit Kumar

Sushmitha Dandu

Dauren Omarov

Navyasree Sriramoju

AGENDA

- Introduction
- Dataset Specifications
- Technical Specifications
- Prediction System Flowchart
- Machine Learning Algorithms
- Feature Importance
- Algorithms Comparison
- Classification
- GitHub Link
- References
- Summary

INTRODUCTION

- The IBM Transactions for Anti Money Laundering (AML) dataset is a synthetic dataset that contains financial transactions involving individuals, companies, and banks.
- The dataset contains both legitimate and laundering transactions that are labeled, making it ideal for training and testing Anti Money Laundering models.
- We chose "LI_Medium_Trans.csv," file from the Dataset which is of 2.98gb and has 11 Columns

DATASET SPECIFICATIONS

DATASET NAME: IBM Transactions for Anti Money Laundering (AML)

TOTAL DATASET SIZE: 2.98 GB

DATASET FORMAT: CSV

DATASET URL:
https://www.kaggle.com/datasets/ealtman2019/ibm-transactions-for-anti-money-laundering-aml?select=LI-Medium_Trans.csv

TECHNICAL SPECIFICATION

Hadoop Version: 3.1.2

No. of CPUs: 8

Py Spark version: 3.0.2

Nodes: 3

Total Storage:

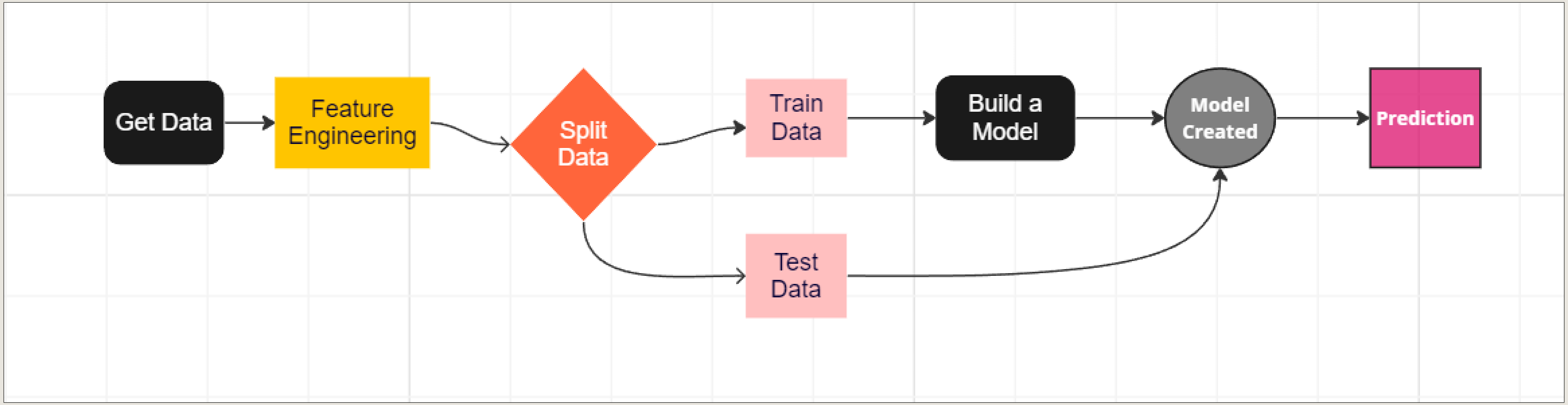
Databricks Community Version: 10.4 LTS (includes Apache Spark 3.1.1, Scala 2.12)

File System: DBFS (Data Bricks File System)

Nodes: 1

Python Version: 3.10.4

PREDICTION SYSTEM FLOWCHART



CLASSIFICATION

The Classification algorithm is a Supervised Learning technique that is used to identify the category of new observations based on training data.

In Classification, a program learns from the given dataset or observations and then classifies new observation into a number of classes or groups. Such as, **Yes or No, 0 or 1**

SPLITTING THE DATASET

Sample Data

```
#Finding the count of training and testing rows
train_rows = train.count()
test_rows = test.count()
print("Training Rows:", train_rows, " Testing Rows:", test_rows)
```

▸ (4) Spark Jobs

```
Training Rows: 3994  Testing Rows: 1714
```

Full Data Set

```
>>> splits = data.randomSplit([0.7, 0.3])
>>> train = splits[0]
>>> test = splits[1].withColumnRenamed("label", "trueLabel")
>>> train_rows = train.count()
>>> test_rows = test.count()
>>> print("Training Rows:", train_rows, " Testing Rows:", test_rows)
Training Rows: 21882083  Testing Rows: 9373309
```

MACHINE LEARNING ALGORITHMS

- Logistic Regression
- Gradient Boost Tree
- Decision Tree
- Random Forest
- Factorization Machine
- Support Vector Machine

LOGISTIC REGRESSION

CV & TVS and their respective AUC Values:

Python

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

evaluator = BinaryClassificationEvaluator(labelCol="trueLabel", rawPredictionCol="prediction", metricName="areaUnderROC")
auc_cv_lr = evaluator.evaluate(prediction_lr_cv)
print("AUC = ", auc_cv_lr)

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

(9) Spark Jobs

```
AUC =  0.5216146888078845
AUC =  0.5216146888078845
AUC =  0.5
```

Command took 1.33 seconds -- by lekha19202@gmail.com at 07/05/2023, 21:19:29 on Qs

LOGISTIC REGRESSION

Precision and recall Values:

```
# Precision and Recall
tp = float(prediction_lr_tvs.filter("prediction == 1.0 AND truelabel == 1").count())
fp = float(prediction_lr_tvs.filter("prediction == 1.0 AND truelabel == 0").count())
tn = float(prediction_lr_tvs.filter("prediction == 0.0 AND truelabel == 0").count())
fn = float(prediction_lr_tvs.filter("prediction == 0.0 AND truelabel == 1").count())
metrics2 = spark.createDataFrame([
 ("TP", tp),
 ("FP", fp),
 ("TN", tn),
 ("FN", fn),
 ("Precision", tp / (tp + fp)),
 ("Recall", tp / (tp + fn))],["metric", "value"])
metrics2.show()
```

(11) Spark Jobs

metrics2: pyspark.sql.dataframe.DataFrame = [metric: string, value: double]

```
+---------+--------------------+
|   metric|               value|
+---------+--------------------+
|       TP|                 2.0|
|       FP|                 0.0|
|       TN|              1669.0|
|       FN|                43.0|
|Precision|                 1.0|
|   Recall|0.044444444444444446|
+---------+--------------------+
```

GRADIENT BOOST TREE

CV & TVS and their respective AUC Values:

Cmd 39

Python

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

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

▸ (6) Spark Jobs

```
AUC =  0.8959160253813969
AUC =  0.8959160253813969
```

Command took 0.98 seconds -- by lekha19202@gmail.com at 07/05/2023, 21:19:29 on Qs

Cmd 40

DECISION TREE

CV & TVS and their respective AUC Values:

Python

```python
evaluator = BinaryClassificationEvaluator(labelCol="trueLabel", rawPredictionCol="prediction", metricName="areaUnderROC")
auc_tvs_dt = evaluator.evaluate(predicted_dt_tvs)
print("AUC = ", auc_tvs_dt)

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

▸ (6) Spark Jobs

```
AUC =  0.5
AUC =  0.5
```

Command took 1.39 seconds -- by lekha19202@gmail.com at 07/05/2023, 22:14:37 on Qs

RANDOM FOREST

CV & TVS and their respective AUC Values:

Cmd 32

Python

```
evaluator = BinaryClassificationEvaluator(labelCol="trueLabel", rawPredictionCol="prediction", metricName="areaUnderROC")
auc_tvs1_rf = evaluator.evaluate(prediction)
print("AUC = ", auc_tvs1_rf)

evaluator = BinaryClassificationEvaluator(labelCol="trueLabel", rawPredictionCol="prediction", metricName="areaUnderROC")
auc_tvs_rf = evaluator.evaluate(predictiontvs)
print("AUC = ", auc_tvs_rf)

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

▸ (9) Spark Jobs

```
AUC =  0.917922235722965
AUC =  0.9269339813689754
AUC =  0.8963142972863507
```

Command took 1.77 seconds -- by lekha19202@gmail.com at 07/05/2023, 21:19:29 on Qs

RANDOM FOREST

Running time for the entire dataset

```
>>> #tvs = TrainValidationSplit(estimator=pipeline, evaluator=BinaryClassificationEvaluator(labelCol="label", rawPredictionCol="pr
UnderROC"), estimatorParamMaps=paramGrid, trainRatio=0.8)
...
>>> model = pipeline.fit(train)

>>> #model = tvs.fit(train)
...
>>> end = time()
>>> phrase = 'Random Forest tvs testing'
>>> print('{} takes {} seconds'.format(phrase, (end - start))) #round(end - start, 2)))
Random Forest tvs testing takes 1861.5901260375977 seconds
>>>
```

```
>>>
>>> end = time()
>>> phrase = 'Random Forest tvs2 testing'
>>> print('{} takes {} seconds'.format(phrase, (end - start))) #round(end - start, 2)))
Random Forest tvs2 testing takes 313.30773091316223 seconds
>>>
```

```
> end = time()
> phrase = 'Random Forest testing'
> print('{} takes {} seconds'.format(phrase, (end - start))) #round(end - start, 2)))
ndom Forest testing takes 301.2368438243866 seconds
>
```

FACTORIZATION MACHINE

CV & TVS and their respective AUC Values:

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

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

▸ (6) Spark Jobs

```
AUC =  0.7959497772377667
AUC =  0.7959497772377667
```

Command took 1.13 seconds -- by lekha19202@gmail.com at 07/05/2023, 21:19:29 on Qs

SUPPORT VECTOR MACHINE

CV & TVS and their respective AUC Values:

```python
evaluatorSVM_tvs = BinaryClassificationEvaluator(labelCol="trueLabel", rawPredictionCol="prediction", metricName="areaUnderROC")
auc_SVM_tvs = evaluatorSVM_tvs.evaluate(predictionSVM_tvs)
print("AUC = ", auc_SVM_tvs)

evaluatorSVM_cv = BinaryClassificationEvaluator(labelCol="trueLabel", rawPredictionCol="prediction", metricName="areaUnderROC")
auc_SVM_cv = evaluatorSVM_cv.evaluate(predictionSVM_cv)
print("AUC = ", auc_SVM_cv)
```

Python

▸ (6) Spark Jobs

```
AUC =  0.5
AUC =  0.5
```

Command took 0.91 seconds -- by lekha19202@gmail.com at 07/05/2023, 21:19:29 on Qs

FEATURE IMPORTANCE

Running command 27. Go to

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

	feature	importance
0	TimestampIdx	0.354191
1	From Bank	0.258357
3	To Bank	0.211514
2	Account2	0.175938

0	From Bank	0.354191
1	To Bank	0.258357
3	Amount Paid	0.211514
2	Amount Received	0.175938

Feature Importance indicates how much each feature contributes to the model prediction. Basically, it determines the degree of usefulness of a specific variable for a current model and prediction.

ALGORITHMS COMPARISON

Algorithms	AUC		Computation Time	
	CV	TVS	CV	TVS
Gradient Boost Tree	0.895916	0.8959160	195.254653sec	71.486958sec
Random Forest	0.896314	0.926933	57.922800sec	5.795582Sec(Tvs) 24.588072Sec(Tvs2)
Factorization Machine	0.795949	0.795949	32.444634sec	14.9555480 sec
Decision Tree	0.5	0.5	448.105305sec	140.117408 sec
Logistic Regression	0.521614	0.521614	1566.377578sec	497.938576 sec
Support Vector Machine	0.5	0.5	138.414037sec	57.775793sec

IMPLEMENTATION IN SPARK ML

lekhaajit — ssh lajitku@144.24.53.159 — 128×27

dataset.csv

Done

```
-rw-r--r--   3 lajitku hdfs    2438266 2023-04-19 19:53 ratings.csv
-rw-r--r--   3 lajitku hdfs      96368 2023-04-05 14:22 tweets.csv
-bash-4.2$ spark-submit Python_project.py
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/odh/1.1.5/spark/jars/slf4j-log4j12-1.7.30.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/odh/1.1.5/hadoop/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
23/05/08 07:59:22 INFO SparkContext: Running Spark version 3.0.2
23/05/08 07:59:22 INFO ResourceUtils: ==============================================================
23/05/08 07:59:22 INFO ResourceUtils: Resources for spark.driver:

23/05/08 07:59:22 INFO ResourceUtils: ==============================================================
23/05/08 07:59:22 INFO SparkContext: Submitted application: Python_project.py
23/05/08 07:59:22 INFO SecurityManager: Changing view acls to: lajitku
23/05/08 07:59:22 INFO SecurityManager: Changing modify acls to: lajitku
23/05/08 07:59:22 INFO SecurityManager: Changing view acls groups to:
23/05/08 07:59:22 INFO SecurityManager: Changing modify acls groups to:
23/05/08 07:59:22 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users  with view permissions
: Set(lajitku); groups with view permissions: Set(); users  with modify permissions: Set(lajitku); groups with modify permission
s: Set()
23/05/08 07:59:23 INFO Utils: Successfully started service 'sparkDriver' on port 32531.
23/05/08 07:59:23 INFO SparkEnv: Registering MapOutputTracker
23/05/08 07:59:23 INFO SparkEnv: Registering BlockManagerMaster
23/05/08 07:59:23 INFO BlockManagerMasterEndpoint: Using org.apache.spark.storage.DefaultTopologyMapper for getting topology inf
ormation
23/05/08 07:59:23 INFO BlockManagerMasterEndpoint: BlockManagerMasterEndpoint up
```

GITHUB LINK

https://github.com/Lekha19202/CIS-5560-big-data-science-project

REFERENCES

- Li, Susan. “Machine Learning with Pyspark and MLlib - Solving a Binary Classification Problem.” *Medium*, Towards Data Science, 7 May 2018, https://towardsdatascience.com/machine-learning-with-pyspark-and-mllib-solving-a-binary-classification-problem-96396065d2aa.
- “Regression Analysis.” *Corporate Finance Institute*, 3 May 2023, https://corporatefinanceinstitute.com/resources/data-science/regression-analysis/.
- Jagdeesh. “PySpark Decision Tree – How to Build and Evaluate Decision Tree Model for Classification Using PySpark MLlib.” *Machine Learning Plus*, 1 May 2023, https://www.machinelearningplus.com/pyspark/pyspark-decision-tree/.

1 We learned how to compare the accuracy of different models based on AUC, Presession and Recall

2 How to optimize computation time by changing the paramgrid parameters.

3 How to increase the accuracy of the model.

4 Building pipeline

SUMMARY

THANK YOU