Fraud Detection on Financial Transactions

Name: Avinash Betha
Student Id: 2176522

Project Plan:

1. Introduction:

Project Title:

Fraud Detection on Financial Transactions

Objective:

Develop an end-to-end big data pipeline that:

- Ingests a large dataset of financial transactions.
- Processes and transforms the data using Apache Spark.
- Stores the transformed data in AWS Athena for efficient querying.
- Uses Spark MLlib to build and evaluate a machine learning model for fraud detection.
- Provides comprehensive visualizations and reporting of findings.

2. Dataset Selection and Staging in S3

Dataset Choice:

- A synthetic dataset containing financial transaction details with columns such as:
    - step, type, amount, nameOrig, oldbalanceOrg, newbalanceOrig, nameDest, oldbalanceDest, newbalanceDest, isFraud, isFlaggedFraud

Staging in S3:

- The raw CSV file is uploaded directly to the S3 bucket (e.g., abhi-financial-data-bucket).

3. Data Processing and Transformation (Apache Spark)

Spark Environment Setup

- Use EMR Studio to run the processing code.
- Install necessary Python packages on the driver using methods such as sc.install_pypi_package.

Data Ingestion

- Read the CSV file from S3 directly:

Code:

```
raw_data_path = "s3a://abhi-financial-data-
bucket/Synthetic_Financial_datasets_log.csv"
df = spark.read.csv(raw_data_path, header=True, inferSchema=True)
```

Data Cleaning

- Remove duplicates, handle missing values, and generate a unique transaction ID:

```
df_cleaned = df.dropDuplicates()

df_cleaned = df_cleaned.filter(df_cleaned["amount"].isNotNull())

df_cleaned = df_cleaned.withColumn("transaction_id", monotonically_increasing_id())
```

Feature Engineering

- Use Spark's window functions to compute additional features such as:
  - Transaction frequency per originator (nameOrig)
  - Total, average, and standard deviation of transaction amounts
  - Difference between original and new balance

```
from pyspark.sql.functions import count, sum as _sum, avg, stddev, expr

from pyspark.sql.window import Window


windowSpec = Window.partitionBy("nameOrig")

df_features = df_cleaned.withColumn("transaction_frequency",
count("transaction_id").over(windowSpec)) \

  .withColumn("total_amount", _sum("amount").over(windowSpec)) \

  .withColumn("avg_amount", avg("amount").over(windowSpec)) \

  .withColumn("std_dev_amount", stddev("amount").over(windowSpec)) \

  .withColumn("balance_diff", expr("oldbalanceOrg - newbalanceOrig"))
```

Save Transformed Data

- Wrote the transformed DataFrame to S3 in Parquet format. This format is efficient for both querying in Athena and loading into Spark for ML.

```
transformed_data_path = "s3a://abhi-financial-data-bucket/transformed-data/"

df_features.write.mode("overwrite").parquet(transformed_data_path)
```

4. AWS Athena Integration

- External Table Creation:
  Create an external table in Athena that points directly to the Parquet data:

```
CREATE EXTERNAL TABLE fraud_data (
 step INT,
 type STRING,
 amount DOUBLE,
 nameOrig STRING,
 oldbalanceOrg DOUBLE,
 newbalanceOrig DOUBLE,
 nameDest STRING,
 oldbalanceDest DOUBLE,
 newbalanceDest DOUBLE,
 isFraud INT,
 isFlaggedFraud INT
 -- Add engineered features as needed
)
STORED AS PARQUET
LOCATION 's3://abhi-financial-data-bucket/transformed-data/';
```

Querying:
Use Athena's SQL interface to explore the data

```
SELECT type, COUNT(*) AS num_transactions

FROM fraud_data

GROUP BY type;
```

5. Machine Learning with Spark MLlib

Load Data for ML

- Load the transformed data into Spark:

```
df_features = spark.read.parquet("s3a://abhi-financial-data-bucket/transformed-data/")
```

Prepare Data & Feature Assembly

- Use VectorAssembler to create a feature vector:

```
from pyspark.ml.feature import VectorAssembler

feature_cols = ["amount", "transaction_frequency", "total_amount", "avg_amount", "std_dev_amount", "balance_diff"]

assembler = VectorAssembler(inputCols=feature_cols, outputCol="features", handleInvalid="skip")

data = assembler.transform(df_features)

data = data.filter(data.isFraud.isNotNull())
```

Model Training & Evaluation

- Split data, train a Logistic Regression model, and evaluate:

```
train_data, test_data = data.randomSplit([0.7, 0.3], seed=42)

from pyspark.ml.classification import LogisticRegression

lr = LogisticRegression(labelCol="isFraud", featuresCol="features", maxIter=10)
```

```
model = lr.fit(train_data)

predictions = model.transform(test_data)


from pyspark.ml.evaluation import BinaryClassificationEvaluator,
MulticlassClassificationEvaluator

roc_evaluator = BinaryClassificationEvaluator(labelCol="isFraud",
metricName="areaUnderROC")

roc_auc = roc_evaluator.evaluate(predictions)

print("Test ROC-AUC:", roc_auc)


accuracy_evaluator = MulticlassClassificationEvaluator(labelCol="isFraud",
predictionCol="prediction", metricName="accuracy")

accuracy = accuracy_evaluator.evaluate(predictions)

print("Test Accuracy:", accuracy)


precision_evaluator = MulticlassClassificationEvaluator(labelCol="isFraud",
predictionCol="prediction", metricName="weightedPrecision")

precision = precision_evaluator.evaluate(predictions)

print("Test Precision:", precision)


recall_evaluator = MulticlassClassificationEvaluator(labelCol="isFraud",
predictionCol="prediction", metricName="weightedRecall")

recall = recall_evaluator.evaluate(predictions)

print("Test Recall:", recall)


f1_evaluator = MulticlassClassificationEvaluator(labelCol="isFraud",
predictionCol="prediction", metricName="f1")

f1_score = f1_evaluator.evaluate(predictions)

print("Test F1 Score:", f1_score)
```

6. Visualization & Reporting

Visualizations for Data Exploration

- Histograms, Boxplots, Count Plots, and Correlation Heatmaps:
  Use seaborn and matplotlib on a Pandas DataFrame (converted from Spark DataFrame):

df_pd = df_features.toPandas()

Visualizations for Model Evaluation

- ROC Curve, Precision-Recall Curve, and Confusion Matrix:
  Convert the predictions DataFrame to Pandas and plot these metrics using matplotlib and seaborn.

7. Conclusion

This project plan demonstrates a complete big data solution for fraud detection by integrating multiple technologies. Apache Spark is used for distributed processing, performing data cleaning, transformation, and feature engineering on a large financial transactions dataset. The transformed data is stored in Parquet format on S3, enabling efficient querying through AWS Athena. A robust machine learning model is then built and evaluated using Spark MLlib to identify fraudulent transactions, with its performance measured via metrics such as ROC-AUC, accuracy, precision, recall, and F1 score. Comprehensive visualizations are generated to explore the data and assess model performance, and these plots are uploaded to S3 for further analysis and reporting.