
Project Overview: Real-Time Fraud Detection System for Financial Transactions

Objective:

Participants will build a system that monitors financial transactions in real-time to detect fraudulent activities. The system will process large amounts of transaction data, apply machine learning models to identify fraud patterns, and provide real-time alerts. The project will involve creating data pipelines for real-time data ingestion, data preprocessing, building a fraud detection model, and deploying the solution for real-time inference and monitoring.

Week 1: Data Warehousing and SQL for Transactional Data

Topics Covered:

- Introduction to Data Warehousing for financial data
- SQL for creating tables, querying, and managing large transactional data

Capstone Project Milestone:

- **Objective**: Design a Data Warehouse schema to store transactional data, user profiles, and fraud labels (e.g., legitimate or fraudulent transactions).

Tasks:

1. **Design Schema**: Create tables to store user profiles, transactional data (e.g., amount, location, time), and fraud labels (whether a transaction was marked as fraudulent).
2. **Querying Data**: Write SQL queries to analyze transaction patterns, detect anomalies, and calculate metrics like average transaction amount.

Example Code:

```sql
-- Create schema for fraud detection
CREATE TABLE user_dim (
    user_id INT PRIMARY KEY,
    user_name VARCHAR(255),
    location VARCHAR(255),
    age_group VARCHAR(50)
);

CREATE TABLE transaction_fact (
    transaction_id INT PRIMARY KEY,
    user_id INT,
    amount DECIMAL(10, 2),
    transaction_time TIMESTAMP,
    location VARCHAR(255),
    fraud_label BOOLEAN
);

-- Query to detect suspicious transactions above a certain threshold
SELECT user_id, amount, transaction_time
FROM transaction_fact
WHERE amount > 10000
ORDER BY transaction_time DESC;
```

Outcome: By the end of Week 1, participants will have set up the Data Warehouse schema to store and query transactional data, including initial queries to identify anomalies.

Week 2: Python for Data Preprocessing and Feature Engineering

Topics Covered:

- Python for extracting and preprocessing transactional data
- Feature engineering for fraud detection (e.g., time between transactions, frequency of transactions, location changes)

Capstone Project Milestone:

- **Objective**: Preprocess the transactional data using Python and engineer features that will be used in the fraud detection model.

Tasks:

1. **Data Preprocessing**: Extract data from the Data Warehouse and clean it (e.g., handle missing values, normalize amounts).
2. **Feature Engineering**: Engineer features that are crucial for fraud detection, such as the time difference between consecutive transactions, transaction frequency, location changes, and deviation from usual transaction behavior.

Example Code:

```
import pandas as pd

# Load transactional data
transactions = pd.read_sql("SELECT * FROM transaction_fact", con=engine)

# Feature engineering: calculate time difference between consecutive transactions
transactions['transaction_time'] = pd.to_datetime(transactions['transaction_time'])
transactions['time_diff'] = transactions.groupby('user_id')
['transaction_time'].diff().dt.total_seconds()

# Feature engineering: detect large jumps in transaction amounts
transactions['amount_diff'] = transactions.groupby('user_id')
['amount'].diff().fillna(0)

# Handle missing values by filling with median values
transactions = transactions.fillna(transactions.median())
```

Outcome: By the end of Week 2, participants will have preprocessed the transactional data and engineered relevant features for fraud detection.

Week 3: Real-Time Data Processing and Anomaly Detection with Apache Spark

Topics Covered:

- Introduction to Apache Spark for large-scale data processing
- Real-time anomaly detection using PySpark for streaming transactional data

Capstone Project Milestone:

- **Objective**: Implement real-time anomaly detection using Apache Spark and PySpark, flagging potentially fraudulent transactions in real-time.

Tasks:

1. **Set Up Spark Streaming**: Configure Apache Spark for real-time streaming of transactional data.
2. **Anomaly Detection**: Use PySpark to monitor transactions and detect anomalies, such as unusual amounts, rapid transactions, or suspicious location changes.

Example Code:

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col

# Create Spark session for streaming transactions
spark = SparkSession.builder.appName("FraudDetection").getOrCreate()

# Read streaming data from Kafka or a similar source
transaction_stream = spark.readStream.format("kafka").option("subscribe",
"transactions").load()

# Anomaly detection: flag transactions over $10,000
anomalies = transaction_stream.filter(col("amount") > 10000)

# Write anomalies to console or a database
query = anomalies.writeStream.outputMode("append").format("console").start()

query.awaitTermination()
```

Outcome: By the end of Week 3, participants will have implemented real-time anomaly detection using PySpark to identify potential fraudulent transactions as they occur.

Week 4: Building a Machine Learning Model for Fraud Detection with Azure Databricks

Topics Covered:

- Azure Databricks for building machine learning pipelines
- Training and deploying a fraud detection model using supervised learning (e.g., logistic regression, decision trees)

Capstone Project Milestone:

- **Objective**: Build, train, and deploy a machine learning model for fraud detection using Azure Databricks.

Tasks:

1. **Model Building**: Use Azure Databricks to train a machine learning model on historical transactional data labeled as fraudulent or legitimate.
2. **Deploy the Model**: Deploy the fraud detection model in Azure Databricks and use it to score real-time transactions.

Example Code:

```
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.feature import VectorAssembler
from pyspark.sql import SparkSession

# Create Spark session
spark = SparkSession.builder.appName("FraudDetectionModel").getOrCreate()

# Load historical transaction data
transaction_df = spark.read.csv('/mnt/data/transactions.csv', header=True,
inferSchema=True)

# Feature engineering
assembler = VectorAssembler(inputCols=["amount", "time_diff", "location_diff"],
outputCol="features")
transaction_df = assembler.transform(transaction_df)

# Train a logistic regression model for fraud detection
lr = LogisticRegression(featuresCol="features", labelCol="fraud_label")
model = lr.fit(transaction_df)

# Deploy the model: use it to score real-time transactions
real_time_transactions = spark.readStream.format("kafka").option("subscribe",
"transactions").load()
predictions = model.transform(real_time_transactions)
predictions.select("transaction_id",
"prediction").writeStream.outputMode("append").format("console").start()
```

Outcome: By the end of Week 4, participants will have built and deployed a machine learning model for fraud detection using Azure Databricks, scoring real-time transactions.

Week 5: Automating the Fraud Detection System with Azure DevOps

Topics Covered:

- Automating deployment of the fraud detection pipeline with Azure DevOps
- Implementing CI/CD pipelines for deploying and monitoring fraud detection models

Capstone Project Milestone:

- **Objective**: Automate the deployment and monitoring of the fraud detection system using Azure DevOps.

Tasks:

1. **Set Up Azure DevOps**: Configure Azure DevOps pipelines for the continuous integration and deployment of the fraud detection system.
2. **Automate Monitoring**: Set up monitoring and alerting for the deployed fraud detection model, ensuring that fraudulent transactions are flagged in real-time.

Example Code:

```yaml
# Azure DevOps pipeline YAML for deploying the fraud detection model
trigger:
  - main

pool:
  vmImage: 'ubuntu-latest'

steps:
  - task: UsePythonVersion@0
    inputs:
      versionSpec: '3.x'

  - script: |
      pip install -r requirements.txt
      python deploy_fraud_model.py
    displayName: 'Deploy Fraud Detection Model'

  - task: AzureMonitorMetrics@0
    inputs:
      monitorName: 'FraudDetectionAlert'
      alertCriteria: 'Suspicious Transaction Detected'
```

Outcome: By the end of Week 5, participants will have fully automated the deployment of the fraud detection model using Azure DevOps and set up monitoring for real-time fraud detection.

Summary of Outcomes:

1. **Week 1**: Design the Data Warehouse schema for transactional data and initial queries for detecting anomalies.
2. **Week 2**: Preprocess transactional data and engineer features relevant to fraud detection.
3. **Week 3**: Implement real-time anomaly detection using Apache Spark and PySpark.
4. **Week 4**: Build, train, and deploy a machine learning model for fraud detection using Azure Databricks.
5. **Week 5**: Automate the deployment and monitoring of the fraud detection system using Azure DevOps.