Detailed Project Report (DPR) for Fraud Transaction Detection

Project Title: Fraud Transaction Detection
Author: Sanika kanade
Email: Kanadesanika2@gmail.com

Objective

Development of a predictive model for monitoring fraudulent insurance claims for private motor products. The model will determine whether a customer is placing a fraudulent insurance claim or not.

Benefits

- Detection of upcoming frauds.
- Provides better insight into the customer base.
- Facilitates easier management of resources.
- Enables manual inspection if fraud is identified.

Technologies

- Machine Learning

Domain

- Banking and Insurance

Project Difficulty Level

- Intermediate

Dataset

The dataset can be obtained through the following link: [Fraud Detection Dataset]

Data Sharing Agreement

- Sample file name: Fraudtrain.csv, Fraudtest.csv
- Length of date stamp: 8 digits
- Length of time stamp: 6 digits
- Number of Columns: 10 (example)
- Column names: Transaction_ID, Trans_date, Trans_time, Customer_ID, Amount, Merchant, Category, Is_Fraud, Gender, Age

- Column data type: Integer, Date, Time, Integer, Float, String, String, Integer, String, Integer

Architecture

1. Data Validation and Data Transformation

In this phase, we ensure that the data is validated against specific criteria before processing further.

- **File Validation:** Validate the integrity and structure of the fraudtrain and fraudtest datasets.
- **Number of Columns:** Ensure both datasets have the correct number of columns as per the defined schema.
- **Column Names:** Verify that column names in both datasets match the expected schema.
- **Data Type Validation:** Validate the data types of columns in both datasets.
- **Missing Values:** Check for missing values in both datasets and handle them appropriately.

2. Data Insertion in Database

Once the data is validated, it is inserted into the database for further processing and analysis.

- **Database Connection:** Connect to the Cassandra database where the fraudtrain and fraudtest datasets will be inserted.
- **Table Creation:** Create tables fraud_train_data and fraud_test_data in Cassandra for storing training and testing data, respectively. If the tables already exist, ensure new data is appended without duplication.
- **Data Insertion:** Insert the validated fraudtrain dataset into the fraud_train_data table and fraudtest dataset into the fraud_test_data table in Cassandra.

3. Model Training

After data insertion, the data is prepared and used to train machine learning models for fraud detection.

- **Data Extraction:** Extract data from the fraud_train_data table in Cassandra for model training.

- **Data Preprocessing:** Perform necessary preprocessing steps including handling missing values, encoding categorical variables, and scaling numerical features.
- **Model Selection:** Train machine learning models such as SVM (Support Vector Machine), Random Forest, and XGBoost on the preprocessed training data. Use techniques like cross-validation to optimize hyperparameters.
- **Model Evaluation:** Evaluate the trained models using metrics such as accuracy, precision, recall, and ROC-AUC score to select the best-performing model for fraud detection.

4. Prediction

Once models are trained, they are deployed to predict fraud transactions on new data (fraudtest dataset).

- **Data Extraction:** Extract data from the fraud_test_data table in Cassandra for model prediction.
- **Data Preprocessing:** Apply the same preprocessing steps used during training on the fraudtest dataset.
- **Model Prediction:** Use the selected model to predict fraud labels for transactions in the fraudtest dataset.
- **Performance Evaluation:** Evaluate the model's performance on the fraudtest dataset using metrics such as accuracy, precision, recall, and ROC-AUC score.

Code

1. Import Libraries

```
import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

from category_encoders import WOEEncoder

from sklearn.linear_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from xgboost import XGBClassifier
```

```
from sklearn.ensemble import RandomForestClassifier

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from sklearn.utils import resample

from sklearn.svm import LinearSVC

from sklearn.naive_bayes import GaussianNB

from    sklearn.metrics    import    accuracy_score,    classification_report,
confusion_matrix

import warnings

# Ignore all warnings

warnings.simplefilter("ignore")
```

2. Load Data

```
from google.colab import drive

drive.mount('/content/drive')

file_path1 = '/content/drive/My Drive/Colab/fraudTrain.csv'

file_path2 = '/content/drive/My Drive/Colab/fraudTest.csv'

train_df = pd.read_csv(file_path1, index_col='Unnamed: 0')

test_df = pd.read_csv(file_path2, index_col='Unnamed: 0')
```

3. Exploratory Data Analysis

```
# Display the first few rows of the training dataset

train_df.head(3)

# Display information about the training dataset

train_df.info()

# Display the shape of the training dataset

train_df.shape

# Count the number of fraud and non-fraud transactions

is_fraud = train_df["is_fraud"].value_counts()

print("Yes: ", is_fraud[1])
```

```
print("No: ", is_fraud[0])

# Check for missing values and duplicates

print(train_df.isna().sum().sum())

print(train_df.duplicated().sum())

# Gender Distribution

fig, axb = plt.subplots(ncols=2, nrows=1, figsize=(15, 8))

explode = [0.1, 0.1]

train_df.groupby('gender')['is_fraud'].count().plot.pie(explode=explode,
autopct="%1.1f%%", ax=axb[0])

ax = sns.countplot(x="gender", hue="is_fraud", data=train_df, ax=axb[1])

# Add values on top of each bar

for p in ax.patches:

         ax.annotate(f'{p.get_height()}',  (p.get_x()  +  p.get_width()  /  2.,
p.get_height()),  ha='center',  va='center',  xytext=(0,  10),  textcoords='offset
points')

# Set labels and title

plt.title("Distribution of Gender with Fraud Status")

plt.xlabel("Gender")

plt.ylabel("Count")

# Show the plot

plt.show()

# Fraud count distribution

is_fraud = train_df["is_fraud"].value_counts()

plt.figure(figsize=(10, 6))

plt.subplot(1, 2, 1) # Subplot for the pie chart

plt.pie(is_fraud, labels=["No", "YES"], autopct="%0.0f%%")

plt.title("is_fraud Counts")

plt.tight_layout() # Adjust layout to prevent overlapping

plt.show()
```

4. Feature Engineering

```
# Change date type from object to datetime

train_df['trans_date_trans_time'] =
pd.to_datetime(train_df['trans_date_trans_time'], format='mixed')

test_df['trans_date_trans_time'] =
pd.to_datetime(test_df['trans_date_trans_time'], format='mixed')

# Extract hour and month from transaction date and time

train_df['hour'] = train_df['trans_date_trans_time'].dt.hour

test_df['hour'] = test_df['trans_date_trans_time'].dt.hour

train_df['month'] = train_df['trans_date_trans_time'].dt.month

test_df['month'] = test_df['trans_date_trans_time'].dt.month



# Plot distribution of transactions by hour

f, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 5), sharey=True)

ax1 = sns.histplot(x='hour', data=train_df[train_df["is_fraud"] == 0],
stat="density", bins=24, ax=ax1, color="orange")

ax2 = sns.histplot(x='hour', data=train_df[train_df["is_fraud"] == 1],
stat="density", bins=24, ax=ax2, color="green")

ax1.set_title("Not Fraud")

ax2.set_title("Fraud")

ax1.set_xticks(np.arange(24)) # ticks of the day 0 -> 23

ax2.set_xticks(np.arange(24))
```

5. Data Pre-processing

```
# Count unique transaction numbers

unique_transaction_count = len(train_df['trans_num'].unique())

print("Total count of unique transaction numbers:", unique_transaction_count)

# Remove non-useful columns

columns_to_drop = ['first', 'unix_time', 'dob', 'cc_num', 'zip', 'city',
'street', 'state', 'trans_num', 'trans_date_trans_time']

train_df = train_df.drop(columns_to_drop, axis=1)
```

```
test_df = test_df.drop(columns_to_drop, axis=1)

# Clean merchant column

train_df['merchant']=train_df['merchant'].apply(lambda   x:   x.replace('fraud_',
''))
```

6. Data Encoding

```
# Apply label encoding to gender

train_df['gender'] = train_df['gender'].map({'F': 0, 'M': 1})

# Apply WOE encoding

for col in ['job', 'merchant', 'category', 'lat', 'last']:

                 train_df[col]    =    WOEEncoder().fit_transform(train_df[col],
train_df['is_fraud'])
```

7. Down-Sampling and Scaling

```
# Downsampling the majority class

No_class = train_df[train_df["is_fraud"] == 0]

yes_class = train_df[train_df["is_fraud"] == 1]

No_class   =   resample(No_class,   replace=False,   n_samples=len(yes_class),
random_state=42)

downsampled_df = pd.concat([No_class, yes_class])



# Apply standard scaler

scaler = StandardScaler()

cols = downsampled_df.columns

cols = cols.drop("is_fraud")

downsampled_df[cols] = scaler.fit_transform(downsampled_df[cols])



# Display the balanced data

downsampled_df["is_fraud"].value_counts().plot(kind="bar", color=['r', 'b'])

plt.xticks(ticks=[0, 1], labels=['No Fraud', 'Fraud'])

plt.show()
```

8. Model Building

```
# Initialize machine learning models
log_reg = LogisticRegression()
decision_tree = DecisionTreeClassifier()
random_forest = RandomForestClassifier()
xgboost = XGBClassifier()


# Train the models
models = [log_reg, decision_tree, random_forest, xgboost]
model_names = ['Logistic Regression', 'Decision Tree', 'Random Forest', 'XGBoost']
for model, name in zip(models, model_names):
    model.fit(X_train, y_train)
    predictions = model.predict(X_test)
    print(f"Model: {name}")
    print("Accuracy:", accuracy_score(y_test, predictions))
    print("Classification Report:")
    print(classification_report(y_test, predictions))
    print("Confusion Matrix:")
    print(confusion_matrix(y_test, predictions))
    print("="*60)
```

Conclusion

This project demonstrates the steps taken to develop a fraud transaction detection model. The model was built using various machine learning algorithms, with appropriate data preprocessing and feature engineering to ensure the accuracy and efficiency of the fraud detection system. The results highlight the performance of each model, providing insights into their effectiveness in detecting fraudulent transactions.