Project Documentation: Online Payment Fraud Detection

***Project Overview***

Fraudulent transactions in online payments can lead to severe financial losses and undermine trust in financial systems. This project implements a machine learning pipeline to detect fraudulent transactions using supervised learning algorithms. The primary goal is to evaluate different classification models for accuracy and select the most effective one for fraud detection.

***Dataset Description***

The dataset used in this project contains online payment transaction records with attributes that include:

- Transaction details: step, type, amount, oldbalanceOrg, newbalanceOrig, oldbalanceDest, newbalanceDest

- Entities involved: nameOrig, nameDest

- Fraud labels: isFraud (target variable)

***Key Dataset Characteristics:***

- Binary Classification: The target variable, isFraud, indicates whether a transaction is fraudulent (1) or not (0).

- Multiclass Categorical Data: The type column contains transaction types such as CASH\_OUT, PAYMENT, etc.

***Project Workflow***

1. Data Loading and Exploration

- The dataset is loaded from the file using pandas.

- Summary statistics, data types, and dimensions are analysed.

- Missing values are handled to ensure clean data.

2. Exploratory Data Analysis (EDA)

- Distribution of transaction types (type) and fraud labels (isFraud) is visualised using bar plots.

- Outliers in numerical features are visualised using box plots.

3. Data Preprocessing

- Label Encoding: The type column is mapped to numeric values for compatibility with machine learning models.

- Feature Engineering: Irrelevant features (nameOrig, nameDest) and features with low correlation are dropped.

- Feature Scaling: Numerical features are standardised using StandardScaler.

4. Data Splitting

- The dataset is split into training and testing subsets (70% train, 30% test) using train\_test\_split.

5. Model Training and Evaluation

Three machine learning models were trained and evaluated:

1. Logistic Regression

2. Decision Tree Classifier

3. Naive Bayes Classifier

 Evaluation Metrics:

- Accuracy: Overall correctness of predictions.

- Precision: Proportion of predicted frauds that are actually fraudulent.

- Recall: Proportion of actual frauds correctly identified.

- F1 Score: Harmonic mean of precision and recall.

- ROC AUC: Measures model's capability to distinguish between classes.

Confusion Matrix:

A confusion matrix is generated for each model to visualize:

- True Positives (TP)

- True Negatives (TN)

- False Positives (FP)

- False Negatives (FN)

6. Model Comparison

- A summary table (DataFrame) compares the accuracy rates of the three models.

- A pie chart visualizes the contribution of evaluation metrics for each model.

***Key Functions***

1. Main Functionality

- load\_data(file\_path): Reads the dataset into a DataFrame.

- explore\_data(df): Displays dataset dimensions, summary statistics, and data types.

- handle\_missing\_data(df): Checks and handles missing values.

- encode\_labels(df, column, mapping): Encodes categorical labels.

- prepare\_features\_target(df, target\_column): Splits data into features and target.

- scale\_data(train\_data, test\_data): Standardizes numerical features.

- evaluate\_model(name, model, x\_test, y\_test): Evaluates and visualizes model performance.

2. Plotting Functions

- visualize\_outliers(df): Displays a boxplot for identifying outliers.

- plot\_category\_counts(df, column): Plots the count distribution of a categorical feature.

- plot\_scores(model\_name, scores, score\_names): Visualizes evaluation metrics as a pie chart.

***Results***

Model Performance Summary

Classification Model   Accuracy Rate (%)

Logistic Regression   99.87

Decision Tree            99.95

Naive Bayes              99.61

***Observations***

1. Decision Tree: Achieved the highest accuracy, indicating its capability to capture non-linear relationships.

2. Logistic Regression: Provided competitive accuracy but may struggle with complex patterns.

3. Naive Bayes: Showed reasonable performance but is sensitive to feature independence assumptions.

***Strengths and Limitations***

- Strengths: All models demonstrated high accuracy rates, with distinct trade-offs in complexity and interpretability.

- Limitations: Imbalanced data may lead to inflated accuracy scores. Additional techniques like oversampling or undersampling could improve model performance on minority classes.

***Future Enhancements***

1. Hyperparameter Tuning: Optimize model parameters using Grid Search or Random Search.

2. Additional Features: Engineer new features to improve model performance.

3. Ensemble Methods: Implement advanced techniques like Random Forests or Gradient Boosting for better accuracy.

4. Deployment: Deploy the model as an API for real-time fraud detection.

***Execution Instructions***

1. Pre-requisites:

   - Python 3.6+

   - Required libraries: numpy, pandas, seaborn, matplotlib, sklearn

2. Run the Script:

   - Update the file\_path in the script with the dataset's location.

   - Execute the script to load the data, train models, and evaluate performance.

3. Output:

   - Confusion matrices, evaluation metrics, and a summary DataFrame comparing model performances.

***Conclusion***

This project provides an end-to-end solution for online payment fraud detection using machine learning. By comparing three models, it identifies the best approach for accurate and reliable fraud prediction. With further enhancements, this pipeline can be adapted for real-world deployment.