CREDIT CARD FRAUD DETECTION

PHASE 5

Problem Statement:

The problem statement involves building a machine learning model to detect fraudulent transactions. The objective is to create a model that can accurately classify transactions as fraudulent or non-fraudulent. The data is imbalanced, with a significantly lower number of fraudulent transactions compared to non-fraudulent ones. Therefore, data preprocessing and handling class imbalance are essential.

Design Thinking Process:

1. *Understand the Problem*: The problem is about fraud detection, and it requires building a machine learning model. It is important to identify the problem's scope and challenges, including imbalanced data.

2. *Data Collection*: A dataset named "fraudTest.csv" is used to train and test the model. The dataset contains various features related to transactions.

3. *Data Exploration*: Analyze the dataset to understand its structure, check for missing values, and explore the distribution of the target variable ("is_fraud").

4. *Data Preprocessing*:

   a. Handle missing values.

   b. Drop non-numeric columns that cannot be directly used for modeling.

   c. Perform one-hot encoding for categorical columns ("merchant" and "category").

   d. Standardize the "amt" column using StandardScaler.

   e. Address class imbalance using Synthetic Minority Over-sampling Technique (SMOTE).

5. *Model Development*:

   a. Split the data into training and testing sets.

   b. Train a machine learning model using the Random Forest Classifier with hyperparameters like the number of estimators.

   c. Make predictions on the test set.

6. *Model Evaluation*: Evaluate the model's performance using metrics such as accuracy and a classification report to understand precision, recall, F1-score, and other classification performance measures.

Phases of Development:

1. Data Loading and Exploration:

- Loading the dataset: The code starts by loading the "fraudTest.csv" dataset using pandas.

- Exploring the dataset: It prints the first few rows of the dataset, information about the data, the count of missing values, and basic statistics such as mean, standard deviation, and quartiles.

- Examining the target variable: The code displays the count of values in the "is_fraud" column to understand the class distribution.

2. Data Preprocessing:

- Handling Missing Values: The code fills missing values in the "amt" column with the mean of the column.

- Dropping Non-Numeric Columns: Columns like 'first,' 'last,' 'gender,' 'street,' 'city,' 'state,' 'job,' 'dob,' and 'trans_num' are dropped as they are non-numeric and not used for modeling.

- One-Hot Encoding: Categorical columns 'merchant' and 'category' are one-hot encoded.

- Standardization: The 'amt' column is standardized using the StandardScaler.

- Handling Class Imbalance: The Synthetic Minority Over-sampling Technique (SMOTE) is used to address the class imbalance by oversampling the minority class (fraudulent transactions).

3. Model Development:

- Splitting Data: The preprocessed data is split into training and testing sets using `train_test_split`.

- Model Selection: A Random Forest Classifier is selected as the machine learning model.

- Model Training: The Random Forest Classifier is trained on the training data.

4. Model Evaluation:

- Model Testing: The trained model is used to make predictions on the test data.

- Performance Metrics: The code calculates and prints the accuracy of the model and generates a classification report, including precision, recall, F1-score, and other classification performance measures.

Dataset Used:

The dataset "fraudTest.csv" is used for building the fraud detection model. This dataset contains various features related to transactions, and the target variable is "is_fraud," which indicates whether a transaction is fraudulent (1) or not (0). It covers credit cards of 1000 customers doing transactions with a pool of 800 merchants from Kaggle. This dataset contains the sequence of transaction information like

trans_date_trans_time, cc_num, merchant, category, amt, firstname, lastname, gender

street, city, state, zip, latitude, longitude, city_pop, job, trans_num, unix_time, merch_latitude,

merch_longitude is_fraud.

https://www.kaggle.com/datasets/kartik2112/fraud-detection

trans_date_trans_time => it consist of transaction date and time

merchant => it contains merchant name to whom the transaction is being processed.

Category => it consists of a genre of transaction like recharge shopping etc..,

Amt => amount of transaction.

And other parameters like firstname, lastname, gender

street, city, state, zip, latitude, longitude, city_pop, job, trans_num, unix_time, merch_latitude,

merch_longitude is_fraud,cc_num.

https://www.kaggle.com/datasets/kartik2112/fraud-detection

Data Preprocessing Steps:

1. Handling Missing Values: Filling missing values in the "amt" column with the mean.

2. Dropping Non-Numeric Columns: Removing non-numeric columns like 'first,' 'last,' 'gender,' 'street,' 'city,' 'state,' 'job,' 'dob,' 'trans_num.'

3. One-Hot Encoding: Performing one-hot encoding for categorical columns 'merchant' and 'category.'

4. Standardization: Standardizing the 'amt' column using StandardScaler.

5. Handling Class Imbalance: Using SMOTE to oversample the minority class (fraudulent transactions) to mitigate the class imbalance issue.

Feature Extraction Techniques:

Feature extraction is not explicitly performed in this code. Instead, feature selection is carried out by selecting specific columns from the dataset, including 'cc_num,' 'merchant,' 'category,' and 'amt.' These columns include:

- 'cc_num': Credit card number
- 'merchant': Categorical column representing the merchant involved in the transaction.
- 'category': Categorical column representing the transaction category.
- 'amt': Transaction amount

The code then applies StandardScaler to standardize the 'amt' feature.
Standardization scales the 'amt' feature to have a mean of 0 and a standard deviation of 1, making it more suitable for the machine learning algorithms.

Choice of machine learning algorithm:

1. **Choice of Machine Learning Algorithm**: The Random Forest Classifier is chosen for this task. Random Forest is an ensemble learning method that combines multiple decision trees to make predictions. It is a popular choice for classification tasks like fraud detection because it is robust, handles high-dimensional data well, and is less prone to overfitting. The ensemble nature of Random Forest helps in improving model accuracy and generalization.

2. **Model Training**: The following steps are involved in model training:
    - Data Loading: The dataset is loaded from the "fraudTest.csv" file using pandas.
    - Data Exploration: Some basic exploratory data analysis is performed to understand the dataset, including checking for missing values and checking the distribution of the target variable 'is_fraud.'
    - Data Preprocessing:
        - Feature Selection: The program drops non-numeric columns and keeps only selected numeric features that are relevant for the model.
        - One-Hot Encoding: Categorical features 'merchant' and 'category' are one-hot encoded to convert them into a numeric format.
        - Missing Value Handling: Missing values in the 'amt' column are filled with the mean value.
        - Data Scaling: The 'amt' feature is standardized using StandardScaler to ensure all features have the same scale.
    - Handling Imbalanced Data: The program uses the Synthetic Minority Over-sampling Technique (SMOTE) to oversample the minority class (fraud cases) to address class imbalance. The sampling strategy is set to 0.5, meaning it aims to balance the class distribution by increasing the number of fraud cases to 50% of the non-fraud cases.
    - Data Split: The resampled data is split into training and testing sets.
3. **Model Evaluation**: The following evaluation metrics are used to assess the performance of the Random Forest Classifier:
    - Accuracy: Accuracy is calculated as the ratio of correctly predicted instances to the total number of instances in the test set. It provides a general overview of model performance.
    - Classification Report: The classification report includes metrics such as precision, recall, F1-score, and support for both the 'fraud' and 'non-fraud' classes. These metrics provide a more detailed understanding of the model's performance in classifying each class.

Innovative Techniques or Approaches:

- The program uses SMOTE for handling class imbalance, which is a widely accepted technique for fraud detection. By generating synthetic samples, SMOTE helps improve the model's ability to detect fraudulent cases.

- The use of StandardScaler to standardize the 'amt' feature is a good practice for many machine learning models, ensuring that features are on a similar scale and preventing some features from dominating the model's training.

- Feature selection is performed to include only relevant numeric columns in the model, which can help reduce noise in the data and improve model efficiency.

Conclusion:

The Python program employs a Random Forest Classifier for fraud detection. It makes significant use of data preprocessing, including handling missing values, one-hot encoding, and feature selection to improve data quality and model efficiency. Furthermore, the program addresses class imbalance by applying the SMOTE technique, crucial for detecting fraudulent cases effectively. The choice of evaluation metrics, including precision, recall, and F1-score, provides a comprehensive understanding of the model's performance. Overall, this program offers a well-structured and significant approach to fraud detection, with potential applications in real-world scenarios.