Low-Level Document (LLD)

Fraud Transaction Detection

Revision Number: 0.1

Deepraj Arya

DOCUMENT CONTROL

Change Record:

VERSION	DATE	AUTHOR	COMMENTS
0.1	14 JAN 2024	Deepraj Arya	Introduction and architecture defined

Reviews:

VERSION	DATE	REVIEWER	COMMENTS
0.1	14 JAN 2024		

Approval Status:

VERSION	REVIEW DATE	REVIEWED BY	APPROVED BY	COMMENTS

Contents

1. Introduction

- **Overview of the project**

  Developed a robust system for detecting fraudulent transactions in credit card transactions to enhance security and minimize financial losses

2. Data Flow Diagram (DFD)

- Visual representation of data flow within the system.
- Components such as Data Ingestion, Data Transformation, Model Training, and Prediction Pipeline.
- Arrows indicating the flow of data between components.

3. Key Features

- Data Description
  - Initially raw data is stored as csv file for the project which was provided by iNeuron
  - Columns:
    - Time: Time elapsed in seconds since the first transaction.
    - V1, V2, ..., V28: Anonymized features resulting from a PCA transformation to protect user identities.
    - Amount: Transaction amount.
    - Class: Binary variable indicating whether the transaction is fraudulent (1) or not (0).

- Data Transformation
  - Implement data transformation techniques to enhance feature representation and prepare data for model training.
  - Handle data normalization, scaling, and feature engineering.

- Technologies Used:
  - Programming Language : Python
  - Machine Learning Libraries: Scikit-learn
  - Version Control: Git and GitHub

4. Component-Level Design

- Data Ingestion Component
  - Class: DataIngestion
  - Methods:
    - __init__: Initialize configuration and logger.
    - initiate_data_ingestion: Ingest raw data, split into train and test sets, and save them.
  - Attributes:
    - ingestion_config: Configuration parameters.

- Data Transformation Component
  - Class: DataTransformation
  - Methods:

        - __init__: Initialize configuration and logger.
        - get_data_transformation: Define preprocessing pipelines.
        - initiate_data_transformation: Apply transformation to train and test data.

    - Attributes:
        - data_transformation_config: Configuration parameters.

- Model Trainer Component
    - Class: ModelTrainer
    - Methods:
        - __init__: Initialize configuration and logger.
        - initiate_model_training: Split data, train models, and save the best model.
    - Attributes:
        - model_trainer_config: Configuration parameters.

- Training Pipeline Component
    - DataIngestion
        - DataIngestion class is initiated to ingest data.
        - Initiate_data_ingestion method is called to perform data ingestion.
        - Train_data_path and test_data_path are returned,representing the paths where the training and testing data are stored.

    - Data Transformation
        - The DataTransformation class is initiated to handle the transformation of data.
        - The ‘initiate_data_transformation’ method is called to perform transformation.
        - ‘train_arr’ and ‘test_arr’ are returned. Representing the transformed training and testing data, respectively. ‘preprocessor_file_path’ is the path where the preprocessor object is stored.
    - Model Training
        - The ModelTrainer class is instantiated to handle the training of machine learning models.
        - The initiate_model_training method is called to split the data and train the models.
        - The trained model is saved, and the path to the saved model is typically configured in ModelTrainerConfig.

- Prediction Pipeline Component
    - PredictionPipeline Class:
        - The PredictionPipeline class is responsible for making predictions using a trained model.
        - It has an `__init__` method, which is empty in this case.
        - The predict method takes a set of features as input, loads the preprocessor and model objects, transforms the features using the preprocessor, and then makes predictions using the trained model.
    - CustomData Class

- The CustomData class represents a container for input features related to time and V1 to V28.
- The __init__ method initializes the class with the provided features
- The get_data_as_dataframe method converts the input features into a dictionary and then creates a DataFrame from the dictionary. This DataFrame represents a single data point.

5. Model Building

The Model Building stage concludes with the availability of a trained model and necessary preprocessing objects for predicting credit card fraud transactions in real-time

6. Model Validation

The primary objective of the Model Validation stage is to assess the performance and generalization capability of the trained machine learning model on unseen data.

7. External Interfaces

- Pandas, NumPy and Scikit-Learn integrated into the project.
- Flask is used for API.

8. Error Handling

- Strategies for logging and reporting errors functionality is used.
- Implemented a robust logging system to capture events and errors throughout the system.
- Incorporate exception handling mechanisms for graceful error recovery.