MILESTONE 3

AMIT PAL

Data Preprocessing

1. Merging Name Columns:
    - Creates a new column named 'Full Name'.
    - Concatenates the values from original name columns ('First Name of the Provider', 'Middle Initial of the Provider', and 'Last Name/Organization Name of the Provider') using string addition (+).
    - Fills any missing values (NaN) with an empty string ('') in each name column before concatenation using fillna('').
    - Adds spaces (' ') between names to create a properly formatted full name.
2. Cleaning Up the Full Name:
    - Applies the str.strip() method to the 'Full Name' column to remove any leading or trailing whitespace characters, ensuring a clean full name.
3. Dropping Original Name Columns:
    - Uses the drop() method to remove the three original name columns from the DataFrame. This keeps the DataFrame concise and avoids redundancy.

1. Extracting and Repositioning the Full Name Column:
    - Extracts the 'Full Name' column into a separate variable named full_name_column using data.pop('Full Name').
    - Repositions the full_name_column back into the DataFrame at index position 1 (second column) using data.insert(1, 'Full Name', full_name_column). This might be useful if you want the full name to appear earlier in the DataFrame's column order.
2. Displaying Results:
    - Calls the head() method on the modified DataFrame to display the first few rows (default is 5 rows) to verify the changes. This allows you to see how the full names are now structured within the DataFrame.

Converting Object to Numeric Type :

Identify Categorical Features: Determine which features in your dataset are categorical.

Handle Missing Values: Decide how to handle missing values in categorical columns before conversion.

Convert Categorical Features to Numeric: Use techniques like Label Encoding or One-Hot Encoding.

Apply Encoding : Transform the categorical variables into numeric representations.

ISOLATION FOREST:

Import Necessary Libraries: Ensure all required libraries are imported.

Load the Dataset: Load the data into a DataFrame.

Preprocess the Data: Handle missing values, encode categorical variables, and scale features if necessary.

Train-Test Split: Split the data into training and testing sets.

Fit the Isolation Forest Model: Initialize and fit the Isolation Forest model on the training data.

Predict Anomalies: Use the model to predict anomalies and map the predictions to binary values.

Evaluate the Model: Use metrics like the classification report and confusion matrix to evaluate the model.

Visualize the Results: Visualize the anomalies detected by the model using plots.

- **Contamination**: The parameter contamination in Isolation Forest represents the expected proportion of outliers in the dataset. You can set it to a specific value if you have prior knowledge about the expected proportion of anomalies.

- **Feature Engineering**: Depending on your dataset, you might need to perform additional feature engineering steps to improve the performance of the model.

- **Interpretation**: The output of the model should be interpreted carefully, especially in the context of the specific application or domain.

SHAP ANALYSIS OF Isolation-Forest Model:

SHAP (SHapley Additive explanations) analysis of an Isolation Forest model quantifies the impact of each feature on anomaly predictions. It assesses how each feature contributes to identifying anomalies by measuring their influence on the model's decision-making process.

ELLIPTIC ENVELOPE :

Import Necessary Libraries:

- Import required libraries (EllipticEnvelope from sklearn.covariance, pandas, matplotlib, etc.).

Load and Preprocess Data:

- Load your dataset into a pandas DataFrame.
- Preprocess data by handling missing values and encoding categorical variables if needed.

Fit the Elliptic Envelope Model:

- Initialize an EllipticEnvelope object.
- Fit the model to your data using .fit() method.

Predict Outliers:

- Use .predict() method to classify data points as inliers (0) or outliers (-1).

Evaluate and Visualize Results:

- Assess the performance of the model using appropriate metrics.
- Visualize outliers and inliers using plots (scatter plots, histograms, etc.).

ONE CLASS SVM :

Import Necessary Libraries: Import the required libraries (OneClassSVM from sklearn.svm, pandas, matplotlib, etc.).

Load and Preprocess Data: Load your dataset into a pandas DataFrame. Preprocess data by handling missing values and encoding categorical variables if needed.

Normalize or Standardize Data: Scale the features to have zero mean and unit variance using StandardScaler.

Fit the One-Class SVM Model: Initialize an OneClassSVM object with parameters like gamma and nu. Fit the model to your scaled data using .fit() method.

Predict Anomalies: Use .predict() method to classify data points as inliers (1) or outliers (-1).

Visualize Results: Plot a scatter plot to visualize anomalies detected by the One-Class SVM algorithm.