SHARMI DAS N01639206

Step 1: Data Preprocessing

1. **Data Loading**: The code loads the training data from a file named "risk-train.txt" into a Pandas DataFrame.

2. **Handling Missing Values**: It fills missing values in the dataset using forward filling (`method="ffill"`), which fills missing values with the previous non-missing value in the same column.

3. **One-Hot Encoding**: It performs one-hot encoding for categorical variables in the dataset, such as "Z_METHODE," "Z_CARD_ART," "Z_LAST_NAME," and "WEEKDAY_ORDER." One-hot encoding converts categorical variables into binary (0 or 1) columns for each category.

4. **Data Transformation**: It processes the "TIME_ORDER" column by replacing '?' with NaN, converting the column to datetime format, and then calculating the time in minutes past midnight. Missing values are filled with the mean value.

5. **Label Encoding**: It applies label encoding to specified columns, converting categorical variables with 'yes' and 'no' values to 1 and 0, respectively.

6. **Calculating Age**: It calculates the age from the "B_BIRTHDATE" column and creates a new column "AGE" with the calculated age. The original "B_BIRTHDATE" column is dropped.

7. **Replacing '?' with 0:** It replaces '?' with 0 in specific columns where '?' is considered equivalent to 0.

8. **Data Splitting**: The code splits the data into features (X) and the target variable (y) and further divides it into training and validation sets using `train_test_split`. It then scales the features using the `StandardScaler`.

Step 2: Classification Model

1. **Selecting a Classifier**: It selects the Random Forest classifier as the classification algorithm.

Step 3: Model Training and Evaluation

1. **Model Training**: The Random Forest classifier is trained on the preprocessed training data using `model.fit(X_train, y_train)`.

2. **Model Evaluation**: It makes predictions on the validation set using the trained model and evaluates the model's performance using a confusion matrix and a classification report. The confusion matrix provides information about true positives, true negatives, false positives, and false negatives. The classification report includes precision, recall, F1-score, and support for each class.

```
Classification Report:
              precision    recall  f1-score   support

           0       0.95      1.00      0.97      5669
           1       0.40      0.01      0.01       331

    accuracy                           0.94      6000
   macro avg       0.67      0.50      0.49      6000
weighted avg       0.92      0.94      0.92      6000
```

Step 4: Saving the Model

1. **Saving the Mode**l: The trained Random Forest classifier model is saved to a file named "risk.pkl" using the `joblib.dump` function.

Step 5: Making Predictions on Test Data

1. **Data Loading and Preprocessing for Test Data:** Similar preprocessing steps are applied to a test dataset loaded from "risk-test.txt." The "X_test" features are transformed and scaled accordingly.

2. **Imputing Missing Values**: It uses SimpleImputer with the "mean" strategy to fill any missing values in the test data.

3. **Making Predictions**: The trained Random Forest model is used to make predictions on the test data, and the results are stored in the "test_predictions" variable.

Step 6: Saving Predictions

1. **Saving Predictions to a File**: The predictions, along with corresponding order IDs, are saved to a file named "classification_results.txt."

This code provides a complete pipeline for data preprocessing, model training, model evaluation, and generating predictions. It also handles missing values, encodes categorical variables, scales features, and saves the model and predictions for future use.