Note:

To run this file and obtain results for your data, please provide the following two file paths (without quotes):

1. Dataset file path
2. Saved model file path

Also, to run this file you need the following libraries:

1. Pandas
2. Numpy
3. Scikit learn
4. Xgboost
5. Onnxruntime

In [ ]:

```python
import pandas as pd
import numpy as np
import json
import onnxruntime as rt
from sklearn.base import BaseEstimator, TransformerMixin
from sklearn.pipeline import Pipeline
from sklearn.metrics import roc_auc_score, classification_report
from xgboost import DMatrix

# Custom transformer: Select Columns
class SelectColumns(BaseEstimator, TransformerMixin):
    def __init__(self, columns_to_keep):
        self.columns_to_keep = columns_to_keep

    def fit(self, X, y=None):
        return self

    def transform(self, X):
        return X[self.columns_to_keep]


# Custom transformer: Handle outliers in Class 2
class HandleOutliersClass2(BaseEstimator, TransformerMixin):
    def fit(self, X, y=None):
        return self

    def transform(self, X):
        class_2_data = X[X['Class'] == 2]
        for col in X.columns:
            if col != 'Class':
                Q1 = class_2_data[col].quantile(0.25)
                Q3 = class_2_data[col].quantile(0.75)
                IQR = Q3 - Q1
                lower_bound = Q1 - 1.5 * IQR
                upper_bound = Q3 + 1.5 * IQR
                class_2_data[col] = class_2_data[col].clip(lower=lower_bound, upper=upper_bound)
        X.update(class_2_data)
        return X



# Custom transformer: Handle outliers in Class 3
class HandleOutliersClass3(BaseEstimator, TransformerMixin):
    def __init__(self, columns_to_process):
        self.columns_to_process = columns_to_process

    def fit(self, X, y=None):
        return self

    def transform(self, X):
        class_3_data = X[X['Class'] == 3]
        for col in self.columns_to_process:
            Q1 = class_3_data[col].quantile(0.25)
            Q3 = class_3_data[col].quantile(0.75)
            IQR = Q3 - Q1
            # Apply different multipliers based on the column
            if col == 'D':
                lower_bound = Q1 - 1.5 * IQR
                upper_bound = Q3 + 1.5 * IQR
            else:  # For columns B and L
                lower_bound = Q1 - 2 * IQR
                upper_bound = Q3 + 1.5 * IQR

            outliers = (class_3_data[col] < lower_bound) | (class_3_data[col] > upper_bound)
            X.loc[outliers & (X['Class'] == 3), col] = np.nan
        return X


# Custom transformer: Transform Class Column
class TransformClassColumn(BaseEstimator, TransformerMixin):
    def fit(self, X, y=None):
        return self

    def transform(self, X):
        X['Class'] = X['Class'] - X['Class'].min()
        return X



# Function: Apply feature mapping
def apply_feature_mapping(X):
    # Define the custom feature mapping
    feature_mapping = {"B": "f0", "D": "f1", "F": "f2", "I": "f3", "J": "f4", "L": "f5", "M": "f6"}
    return X.rename(columns=feature_mapping)



# Function: Load ONNX model
def load_onnx_model(model_path):
    return rt.InferenceSession(model_path)


# Function: Make predictions
def make_predictions_onnx(model, X):
    input_name = model.get_inputs()[0].name
    pred = model.run(None, {input_name: X.astype(np.float32)})[1]
    return np.array(pred)



# Function: Calculate metrics
def calculate_metrics(y_true, y_pred_prob):
    auc = roc_auc_score(y_true, y_pred_prob, multi_class='ovr', average='macro')
    print(f"AUC Score: {auc:.4f}")
    y_pred_classes = y_pred_prob.argmax(axis=1)
    print("Classification Report:")
    print(classification_report(y_true, y_pred_classes))



# Main function to execute the pipeline
def main():
    # User input for file paths
    dataset_path = input("Enter the path to your dataset (without quotes): ")
    model_path = input("Enter the path to your ONNX model (without quotes): ")

    # Columns to keep and process
    columns_to_keep = ['B', 'D', 'F', 'I', 'J', 'L', 'M', 'Class']
    columns_for_class3 = ['B', 'D', 'L']

    # Load dataset
    df = pd.read_csv(dataset_path)

    # Define the pipeline
    pipeline = Pipeline([
        ('select_columns', SelectColumns(columns_to_keep=columns_to_keep)),
        ('handle_outliers_class2', HandleOutliersClass2()),
        ('handle_outliers_class3', HandleOutliersClass3(columns_to_process=columns_for_class3)),
        ('transform_class_column', TransformClassColumn())
    ])

    # Apply pipeline transformations
    transformed_df = pipeline.fit_transform(df)

    # Split data into X and y
    X = transformed_df.drop(columns=['Class'])
    y = transformed_df['Class']

    # Apply feature mapping
    X_renamed = apply_feature_mapping(X)

    # Load the ONNX model
    onnx_model = load_onnx_model(model_path)

    # Make predictions
    y_pred_prob = make_predictions_onnx(onnx_model, X_renamed.values)

    # Evaluate performance metrics
    calculate_metrics(y, y_pred_prob)

if __name__ == "__main__":
    main()
```

```
C:\Users\Siddhant\AppData\Local\Temp\ipykernel_18472\2601812529.py:36: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  class_2_data[col] = class_2_data[col].clip(lower=lower_bound, upper=upper_bound)
C:\Users\Siddhant\AppData\Local\Temp\ipykernel_18472\2601812529.py:36: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  class_2_data[col] = class_2_data[col].clip(lower=lower_bound, upper=upper_bound)
C:\Users\Siddhant\AppData\Local\Temp\ipykernel_18472\2601812529.py:36: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  class_2_data[col] = class_2_data[col].clip(lower=lower_bound, upper=upper_bound)
C:\Users\Siddhant\AppData\Local\Temp\ipykernel_18472\2601812529.py:36: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  class_2_data[col] = class_2_data[col].clip(lower=lower_bound, upper=upper_bound)
C:\Users\Siddhant\AppData\Local\Temp\ipykernel_18472\2601812529.py:36: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  class_2_data[col] = class_2_data[col].clip(lower=lower_bound, upper=upper_bound)
C:\Users\Siddhant\AppData\Local\Temp\ipykernel_18472\2601812529.py:36: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  class_2_data[col] = class_2_data[col].clip(lower=lower_bound, upper=upper_bound)
C:\Users\Siddhant\AppData\Local\Temp\ipykernel_18472\2601812529.py:36: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  class_2_data[col] = class_2_data[col].clip(lower=lower_bound, upper=upper_bound)
C:\Users\Siddhant\AppData\Local\Temp\ipykernel_18472\2601812529.py:75: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  X['Class'] = X['Class'] - X['Class'].min()
```

```
AUC Score: 0.8922
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.55      0.71     36119
           1       0.75      1.00      0.86     89977
           2       0.84      0.74      0.79    113904

    accuracy                           0.81    240000
   macro avg       0.86      0.76      0.79    240000
weighted avg       0.83      0.81      0.80    240000
```