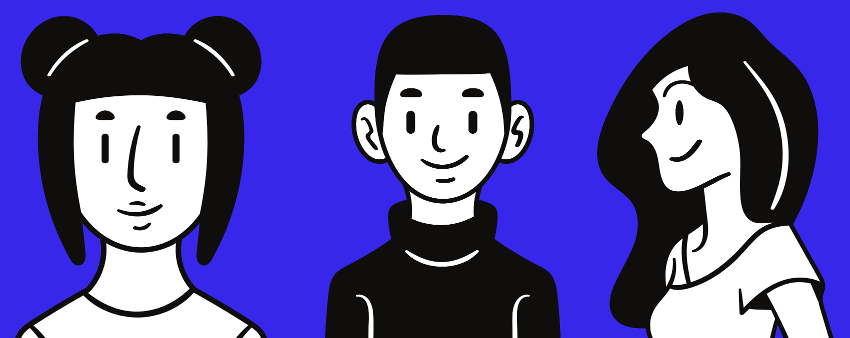
Project Presentation: Data Analysis and Modeling

AUTHORS:
MOLDOBAEVA ADINAI
DUISHENALIEVA MILENA
KUVONDIKOV DILMUROD



PROJECT OVERVIEW:

The goal of this project is to analyze a dataset, extract meaningful insights, and build predictive models to support data-driven decision-making.

Scope of Analysis:

- Dataset Overview
- Exploratory Data Analysis (EDA)
- Data Preprocessing and Feature Engineering
- Machine Learning Model Development
- Hyperparameter Tuning and Optimization
- Model Interpretation and Business Insights

Business Problem:

- Understanding key patterns in the data to improve accuracy in classification and forecasting.
- Identifying high-impact features to enhance predictive performance.
- Exploring customer segmentation to optimize business strategies.

Dataset Overview:

This dataset contains 9,000 entries and 11 features, including both numerical and categorical types. It is intended for a binary classification task, where the target variable (target) takes the value 0 or 1. The objective is to predict this target value using the other features

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9000 entries, 0 to 8999
Data columns (total 11 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   feature_1   9000 non-null   float64
 1   feature_2   9000 non-null   float64
 2   feature_3   8600 non-null   float64
 3   feature_4   9000 non-null   float64
 4   feature_5   9000 non-null   float64
 5   feature_6   8500 non-null   float64
 6   feature_7   9000 non-null   float64
 7   feature_8   9000 non-null   float64
 8   category_1  9000 non-null   object
 9   category_2  9000 non-null   object
 10  target      9000 non-null   int64
dtypes: float64(8), int64(1), object(2)
memory usage: 773.6+ KB
```

ORIGINAL DATASET:

- **Data Characteristics:** 8 numerical features, 2 categorical, target is an integer.
- **Missing Values:** Some missing data — needs to be filled or removed.
- **Class Balance:** Class distribution should be checked before training.
- **Data Preprocessing:** Handle missing values, encode categories, scale features if needed.

```
(9000, 11)
   feature_1  feature_2  feature_3  feature_4  feature_5  feature_6  \
0   0.496714   1.146509  -0.648521   0.833005   0.784920  -2.209437
1  -0.138264  -0.061846        NaN   0.403768   0.704674  -2.498565
2   0.647689   1.395115  -0.764126   1.708266  -0.250029   1.956259
3   1.523030   2.657560  -2.461653   2.649051   0.882201   3.445638
4  -0.234153  -0.499391   0.576097  -0.441656   0.610601   0.211425
5  -0.234137  -0.699415   0.268972  -0.702775   0.702283  -0.332383
6   1.579213   3.117904  -2.885133   3.312708   0.864708   2.045283
7   0.767435   1.730870  -1.445877   1.411070   0.874003   0.674730
8  -0.469474  -0.877919   0.575087  -0.532917  -0.519870        NaN
9   0.542560   1.314738  -0.403383   1.456165  -0.744625   1.987345

   feature_7  feature_8     category_1 category_2  target
0  -1.300105  -2.242241  Above Average   Region C       1
1  -1.339227  -1.942298  Below Average   Region A       0
2   1.190238   1.503559           High   Region C       1
3   2.120913   3.409035           High   Region B       1
4   0.935759  -0.401463  Below Average   Region C       0
5   0.453958  -0.826721  Below Average   Region A       0
6   1.531547   1.771851           High   Region A       1
7   0.812931   1.489838           High   Region A       1
8  -3.002925  -4.779960  Below Average   Region A       0
9   0.431966   3.309386           High   Region C       1
```

CLEANED DATASET

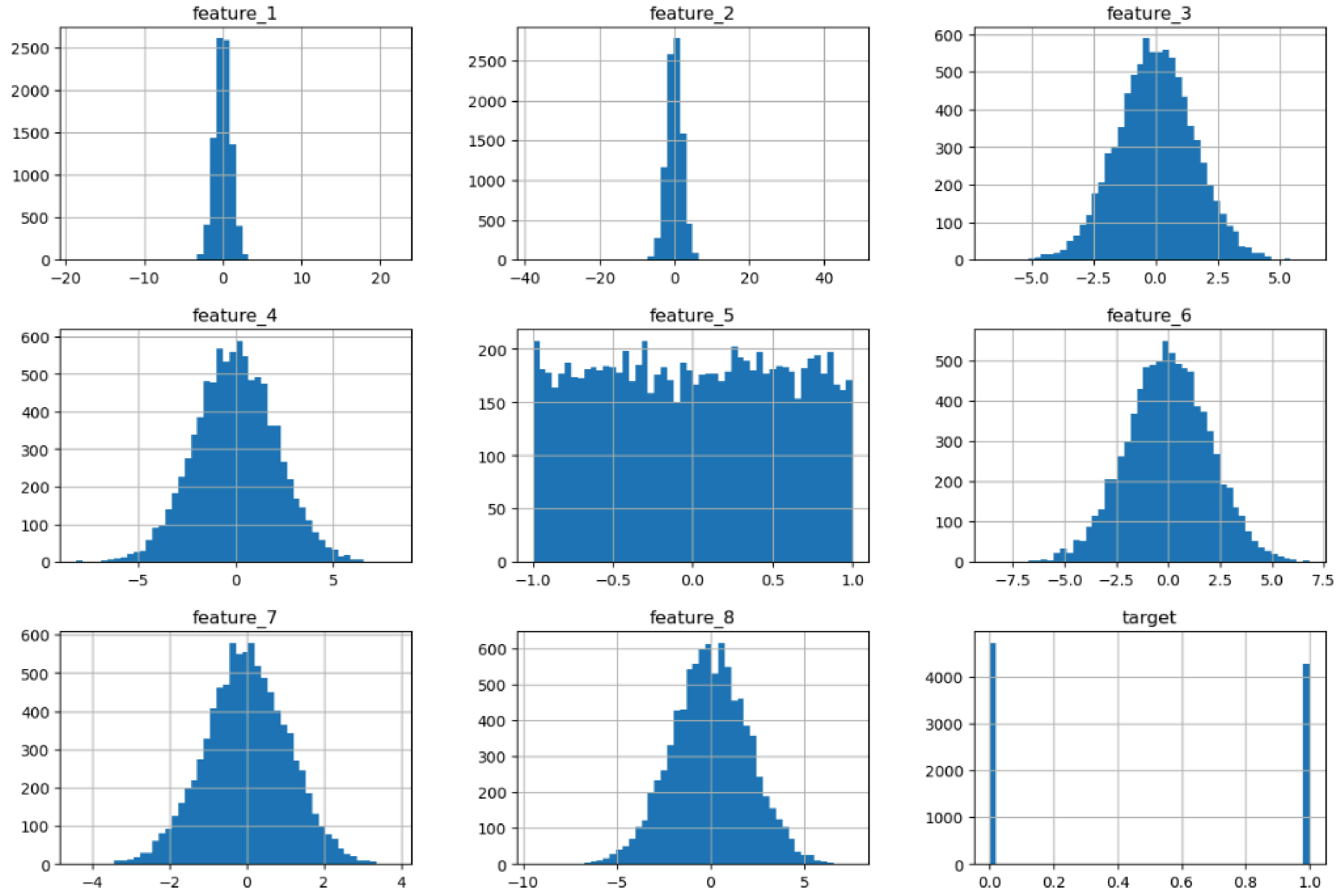
```
   feature_1  feature_2  feature_3  feature_4  feature_5  feature_6  \
0   0.518009   0.593722  -0.455144   0.428411   1.363756  -1.222596
1  -0.144373  -0.033278   0.002307   0.209292   1.224748  -1.381927
2   0.675499   0.722721  -0.536041   0.875218  -0.429068   1.073018
3   1.588618   1.377788  -1.723920   1.355474   1.532275   1.893779
4  -0.244401  -0.260315   0.401806  -0.222283   1.061787   0.111482

   feature_7  feature_8  target  category_1_Below Average  category_1_High
0  -1.287604  -1.173168     1.0                     False            False
1  -1.326097  -1.017482     0.0                      True            False
2   1.162679   0.771098     1.0                     False             True
3   2.078383   1.760140     1.0                     False             True
4   0.912294  -0.217708     0.0                      True            False

   category_1_Low  category_2_Region B  category_2_Region C
0           False                False                 True
1           False                False                False
2           False                False                 True
3           False                 True                False
4           False                False                 True
```

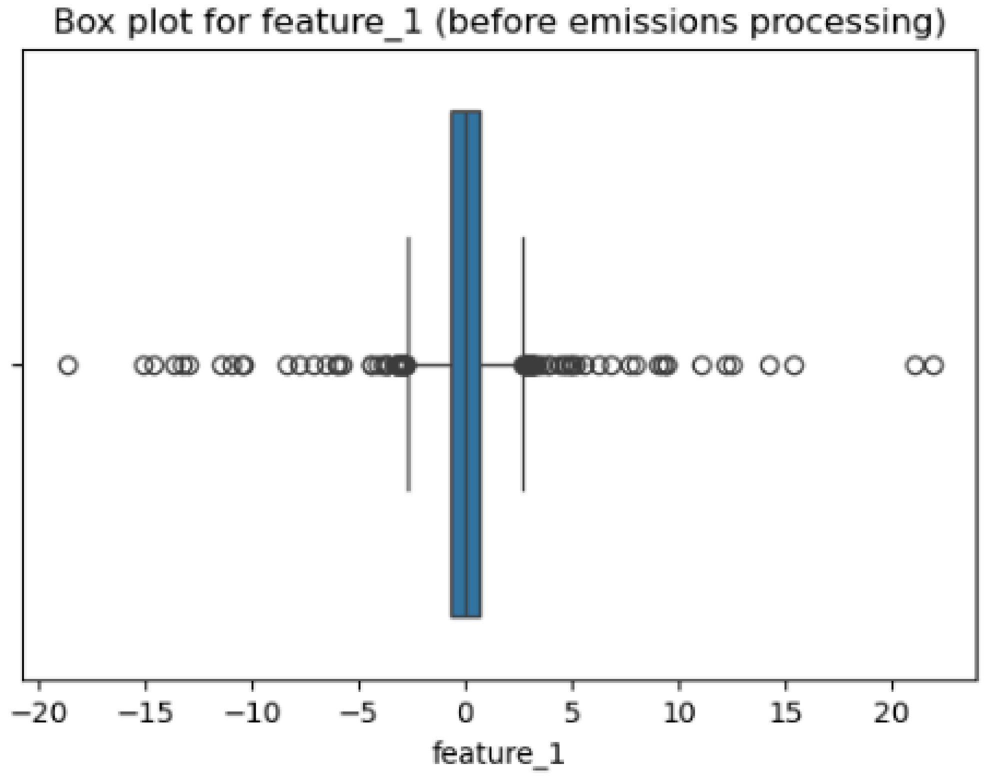
```python
#Remove outliers from given numerical columns
def remove_outliers_iqr(df,columns):
    cleaned_df = df.copy()
    for col in columns:
        Q1 = cleaned_df[col].quantile(0.25) # 1st quartile
        Q3 = cleaned_df[col].quantile(0.75) # 3rd quartile
        IQR = Q3 - Q1  # Interquartile range
        lower = Q1 - 1.5 * IQR
        upper = Q3 + 1.5 * IQR
        # Removing outliers
        cleaned_df = cleaned_df[(cleaned_df[col] >= lower) & (cleaned_df[col] <= upper)]
    return cleaned_df

df_cleaned = remove_outliers_iqr(df,numerical_columns)
```

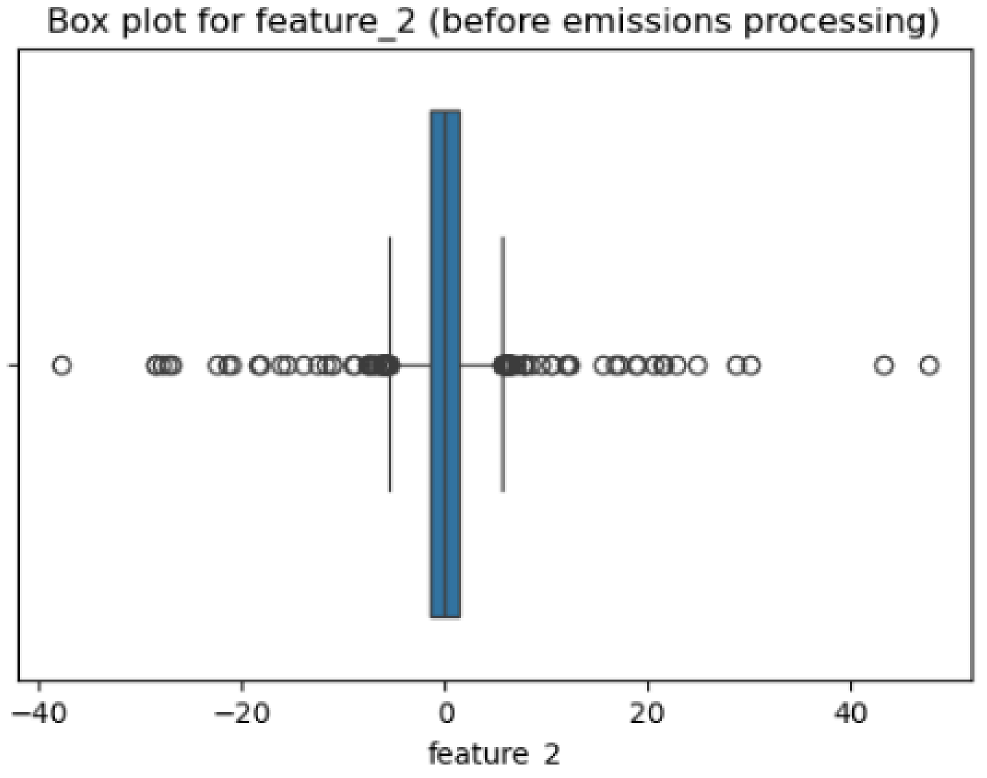
This code removes outliers from specified numerical columns in a DataFrame using the Interquartile Range (IQR) method.

NORMALIZED DATASET

BEFORE:

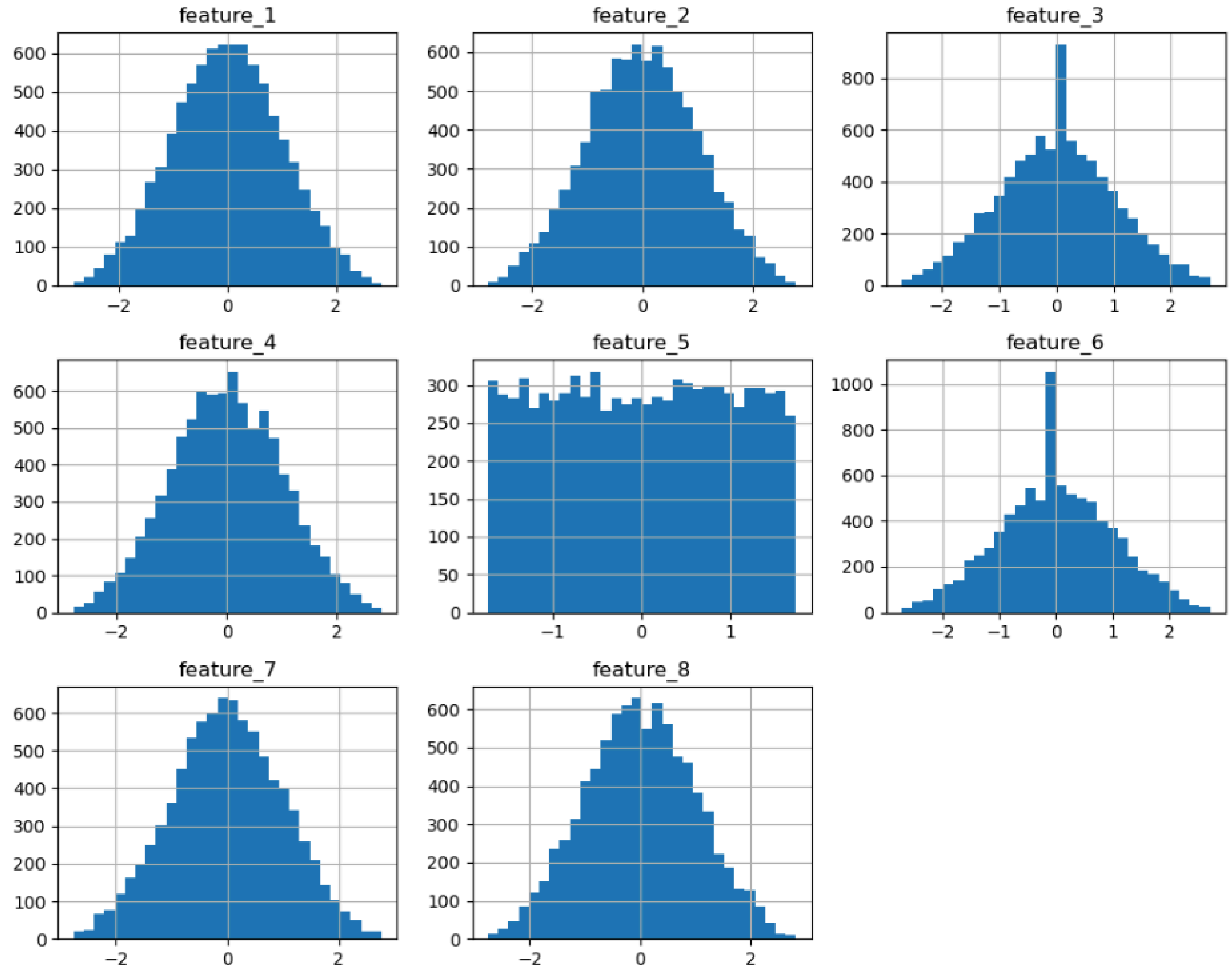


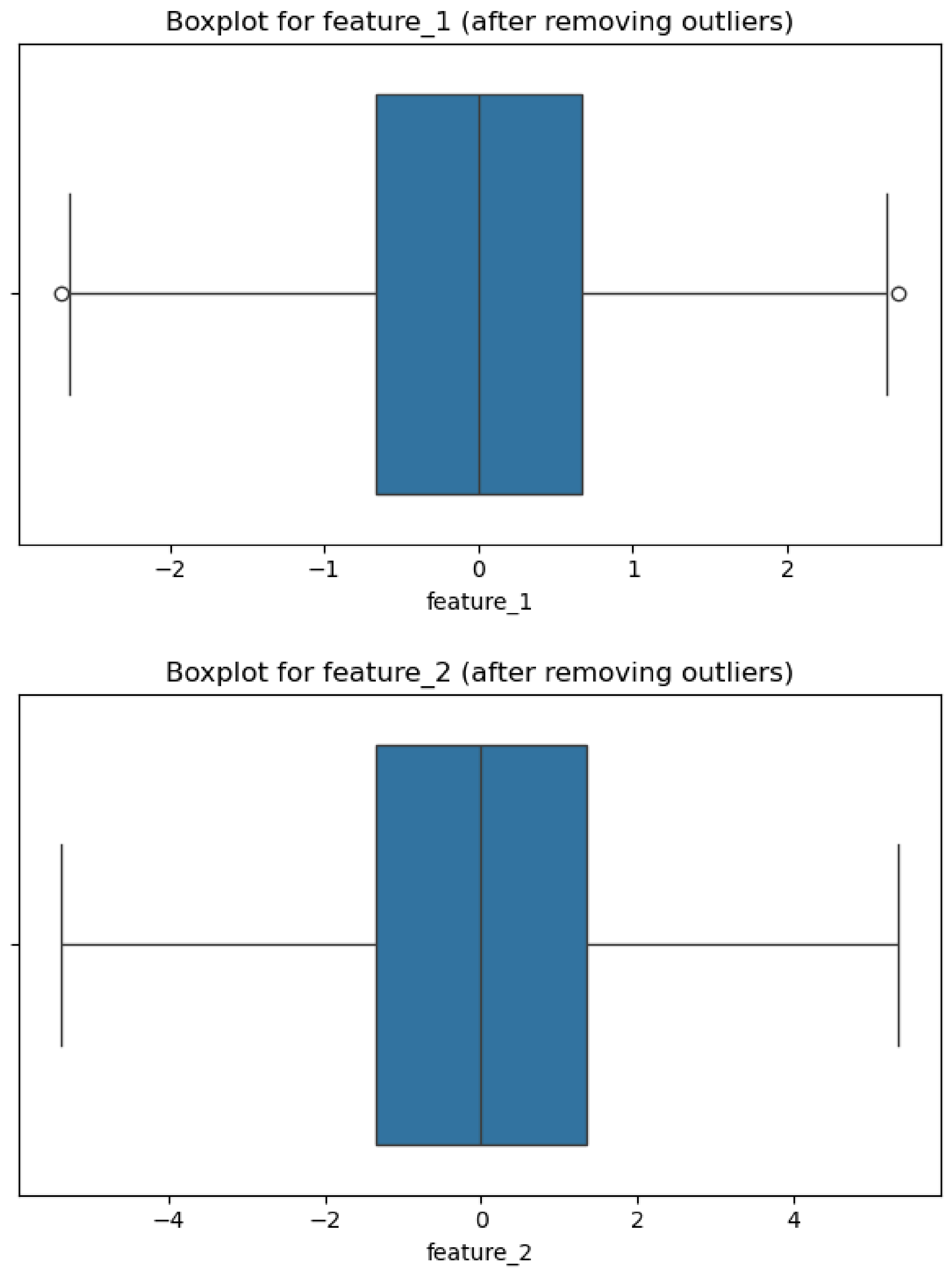




NORMALIZED DATASET

AFTER:



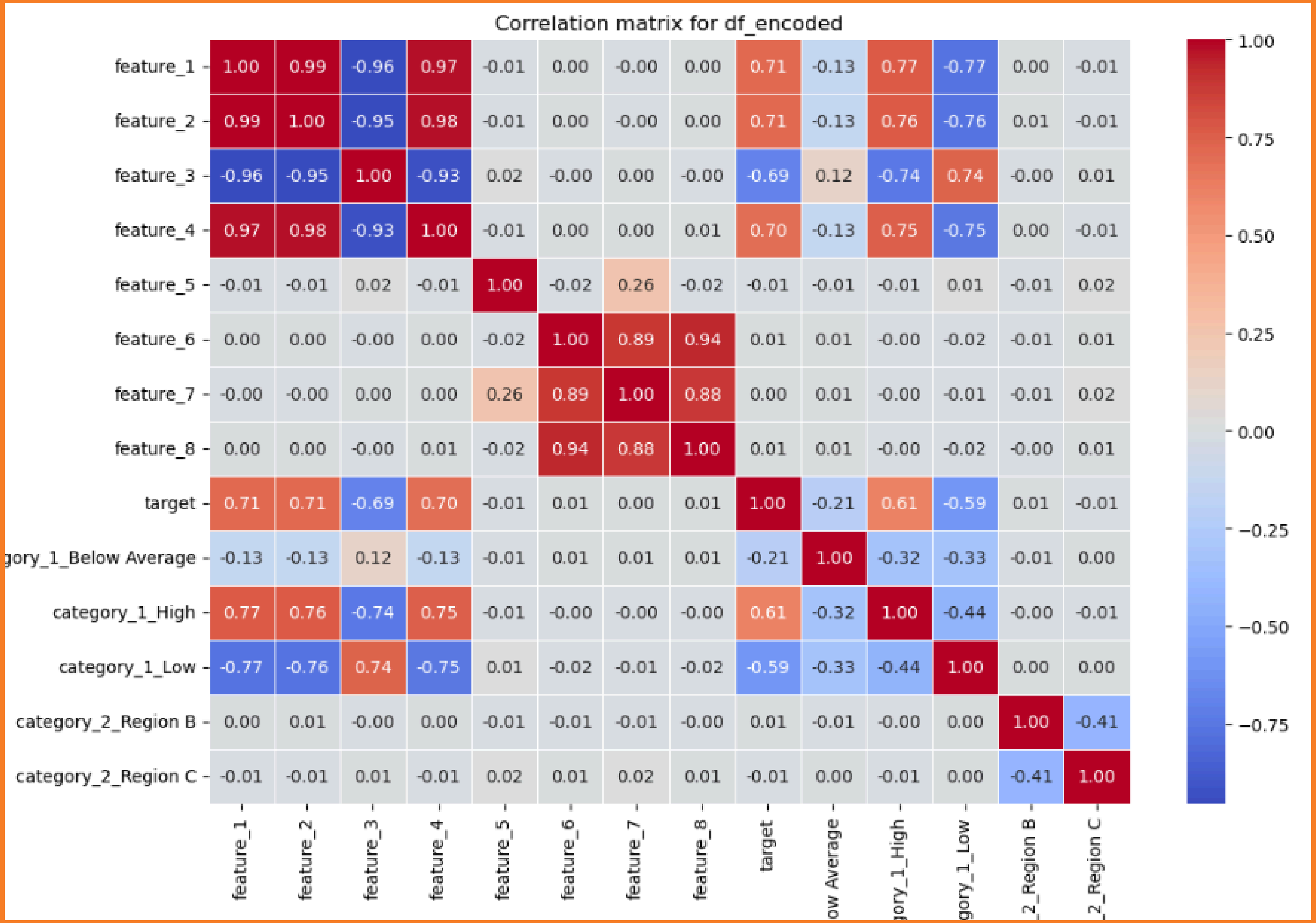


EXPLORATORY DATA ANALYSIS

The correlation matrix shows linear relationships between features.

- feature_1 and feature_2 are strongly positively correlated with the target (0.71).
- feature_3 has a strong negative correlation (-0.69).
- category_1_High and category_1_Low also distinguish the target well (0.61 and -0.59).

These features are highly informative and useful for the model.



T-TEST AND CHI-SQUARE TEST

```python
from scipy.stats import ttest_ind
# Select numeric columns excluding 'target'
numerical_columns = df_encoded.select_dtypes(include=["float64", "int64"]).drop(columns='target').columns
# Perform t-tests and collect results
t_test_results = pd.DataFrame([
    {
        'feature': col,
        't_statistic': round((t_stat := ttest_ind(
            df_encoded[df_encoded['target'] == 0][col],
            df_encoded[df_encoded['target'] == 1][col],
            equal_var=False
        )[0]), 4),
        'p_value': format(ttest_ind(
            df_encoded[df_encoded['target'] == 0][col],
            df_encoded[df_encoded['target'] == 1][col],
            equal_var=False
        )[1], '.4e')
    }
    for col in numerical_columns
])
# Sort and display results by p-value
print(t_test_results.sort_values(by='p_value'))
```

	feature	t_statistic	p_value
0	feature_1	-94.2060	0.0000e+00
1	feature_2	-95.1465	0.0000e+00
2	feature_3	89.6765	0.0000e+00
3	feature_4	-91.6341	0.0000e+00
7	feature_8	-1.0621	2.8822e-01
4	feature_5	0.9378	3.4837e-01
5	feature_6	-0.7573	4.4889e-01
6	feature_7	-0.3025	7.6230e-01

```python
from scipy.stats import chi2_contingency
# Select binary categorical features (one-hot encoded)
binary_columns = [col for col in df_encoded.columns
                  if col.startswith('category_1_') or col.startswith('category_2_')]

# Chi-square tests
chi2_df = pd.DataFrame([
    {
        'Feature': col,
        'Chi2 Statistic': round((stat := chi2_contingency(pd.crosstab(df_encoded[col], df_encoded['target'])))[0], 4),
        'p-value': format(stat[1], '.4e')
    }
    for col in binary_columns
])
print(chi2_df.sort_values(by='p-value'))
```

	Feature	Chi2 Statistic	p-value
1	category_1_High	3225.8919	0.0000e+00
2	category_1_Low	3049.1399	0.0000e+00
4	category_2_Region C	0.4402	5.0702e-01
3	category_2_Region B	0.3173	5.7322e-01
0	category_1_Below Average	380.8968	7.9382e-85

Feature engineering

Removed Features:

- feature_5, feature_6, feature_7 — removed due to low correlation and statistical insignificance.
- The original categorical feature category_2 — removed after one-hot encoding.

Added Features:

- Arithmetic combinations: feature_sum, diff_f1_f2, f1_f4_sum, f1_f4_diff.
- Aggregates: total_features, mean_features, max_feature, min_feature, f1_f4_max, f1_f4_min.
- One-hot encoded features: category_1_High, category_1_Low, category_1_Below Average.

	feature_1	feature_2	feature_3	feature_4	feature_8	target \
0	0.518009	0.593722	-0.455144	0.428411	-1.173168	1.0
1	-0.144373	-0.033278	0.002307	0.209292	-1.017482	0.0
2	0.675499	0.722721	-0.536041	0.875218	0.771098	1.0
3	1.588618	1.377788	-1.723920	1.355474	1.760140	1.0
4	-0.244401	-0.260315	0.401806	-0.222283	-0.217708	0.0

	category_1_Below Average	category_1_High	category_1_Low	feature_sum \
0	False	False	False	1.111731
1	True	False	False	-0.177652
2	False	True	False	1.398220
3	False	True	False	2.966406
4	True	False	False	-0.504716

	total_features	mean_features	diff_f1_f2	max_feature	min_feature \
0	1.084998	0.271250	-0.075713	0.593722	-0.455144
1	0.033948	0.008487	-0.111095	0.209292	-0.144373
2	1.737397	0.434349	-0.047222	0.875218	-0.536041
3	2.597961	0.649490	0.210829	1.588618	-1.723920
4	-0.325194	-0.081298	0.015914	0.401806	-0.260315

	f1_f4_sum	f1_f4_diff	f1_f4_max	f1_f4_min
0	0.946420	0.089598	0.518009	0.428411
1	0.064919	-0.353666	0.209292	-0.144373
2	1.550717	-0.199719	0.875218	0.675499
3	2.944092	0.233144	1.588618	1.355474
4	-0.466684	-0.022117	-0.222283	-0.244401

These modifications aim to improve feature informativeness for modeling.

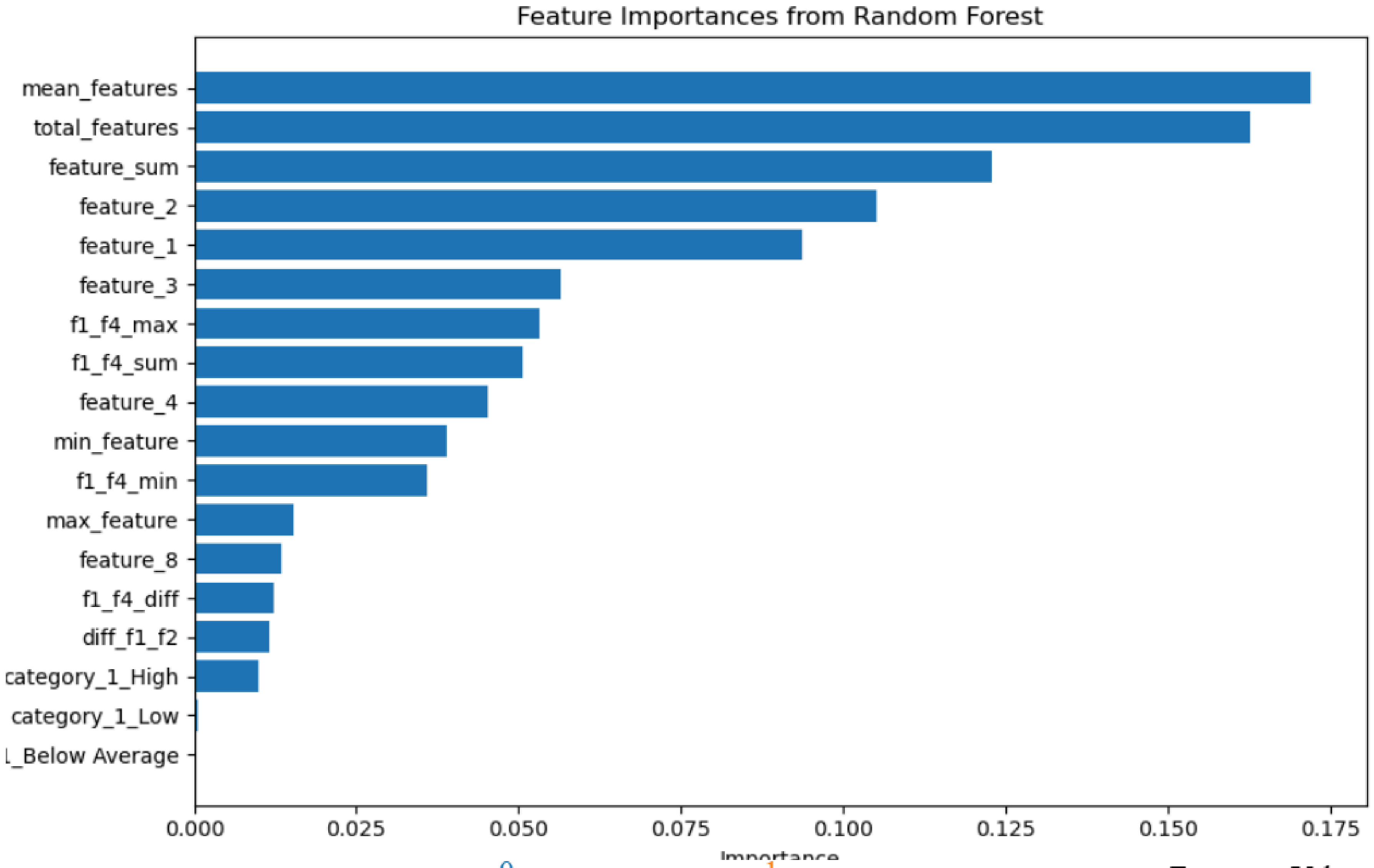
MODDELING (RANDOM FOREST)

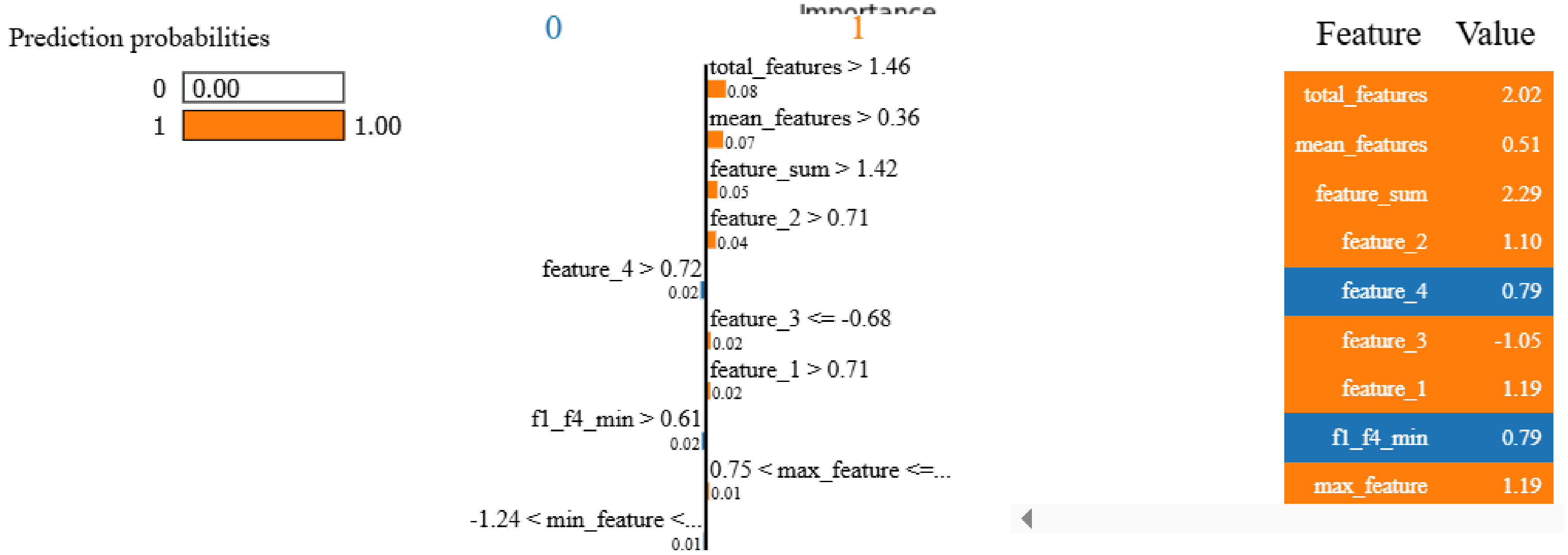
```python
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score, classificati
#Load the data
df = pd.read_csv('FINAL_DATA.csv')

#Separate features and target variable
X = df.drop(columns=['target'])
y = df['target']

#Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.2, random_state=42)

#Train a Random Forest model
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

#Predict on the test set
rf_pred = rf_model.predict(X_test)

#Evaluate accuracy
rf_accuracy = accuracy_score(y_test, rf_pred)
print(f'Random Forest Accuracy: {rf_accuracy:.4f}')

#Classification report
print("\nClassification Report:")
print(classification_report(y_test, rf_pred))

#ROC AUC score
rf_proba = rf_model.predict_proba(X_test)[:, 1]
rf_roc_auc = roc_auc_score(y_test, rf_proba)
print(f"Random Forest ROC AUC Score: {rf_roc_auc:.4f}")
```

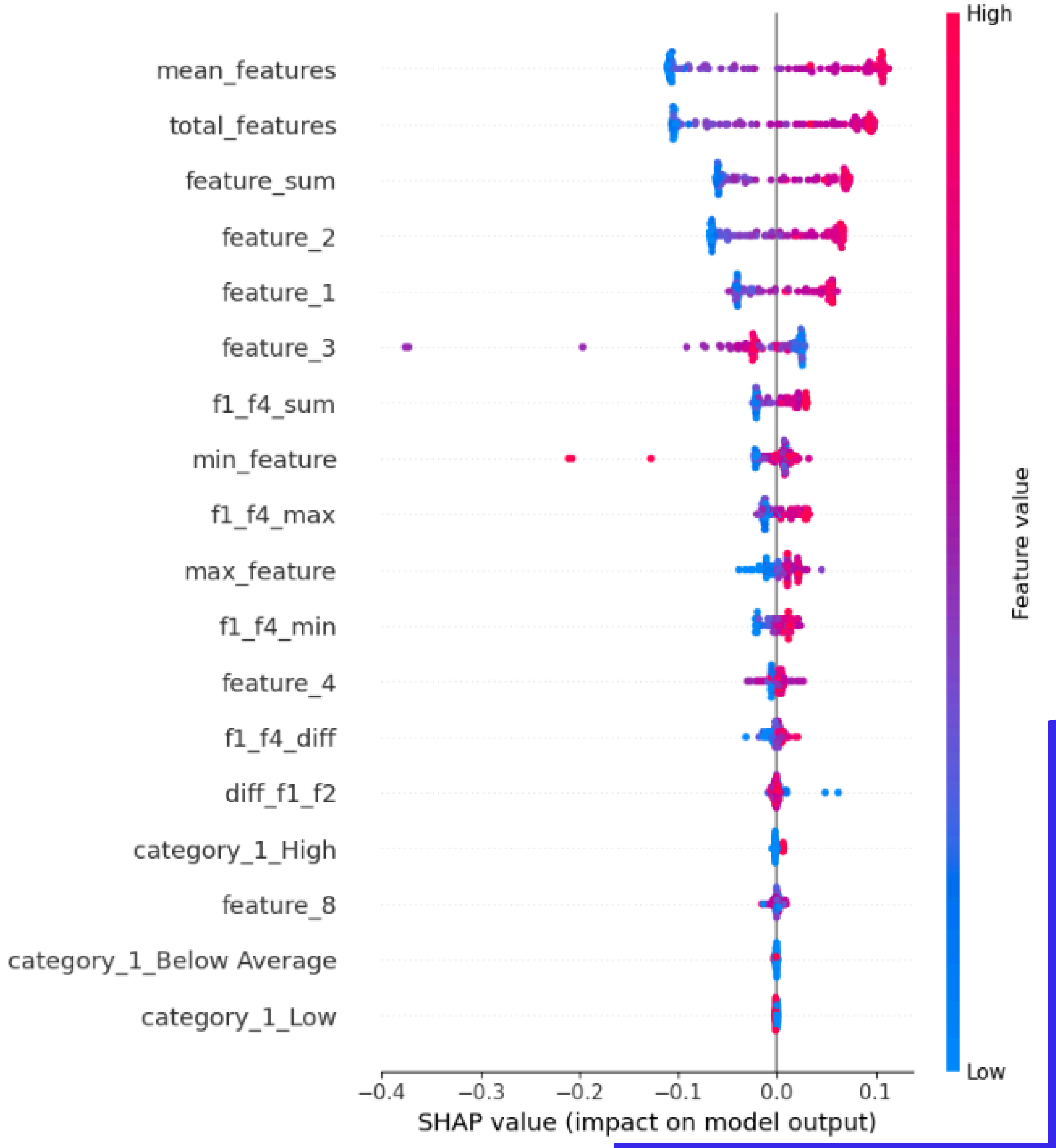
```
Random Forest Accuracy: 0.8708

Classification Report:
              precision    recall  f1-score   support

         0.0       0.86      0.89      0.88       889
         1.0       0.88      0.85      0.86       845

    accuracy                           0.87      1734
   macro avg       0.87      0.87      0.87      1734
weighted avg       0.87      0.87      0.87      1734

Random Forest ROC AUC Score: 0.9455
```

```
Best Hyperparameters: {'max_depth': 10, 'max_features': 'sq
CV Accuracy: 0.8818
Test Accuracy: 0.8743

Classification Report:
              precision    recall  f1-score   support

         0.0       0.86      0.90      0.88       889
         1.0       0.89      0.84      0.87       845

    accuracy                           0.87      1734
   macro avg       0.88      0.87      0.87      1734
weighted avg       0.88      0.87      0.87      1734

ROC AUC Score for Best Random Forest Model: 0.9509
```

PERFORMANCE (RANDOM FOREST)







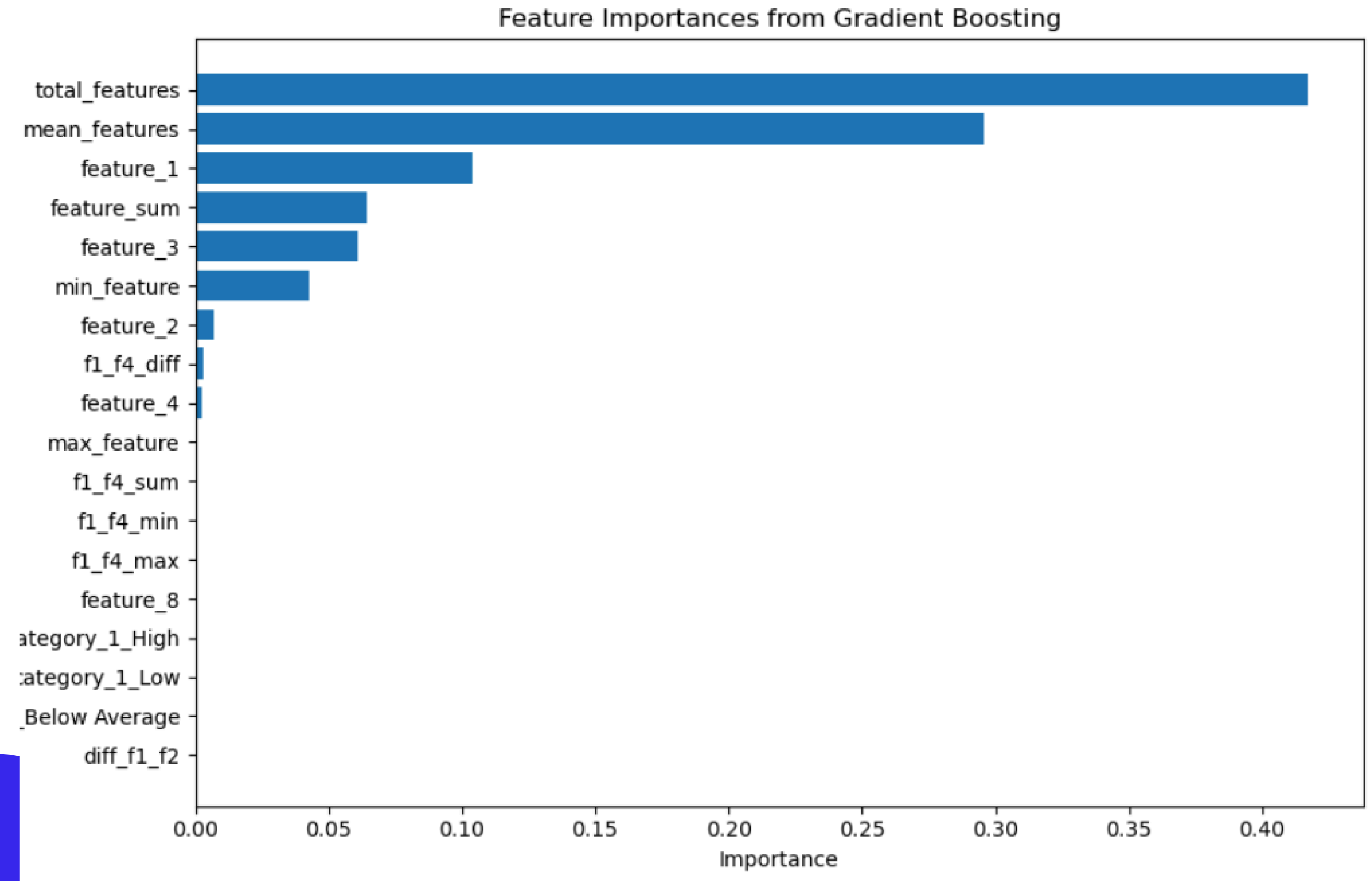
MODDELING (GRADIENT BOOSTING)

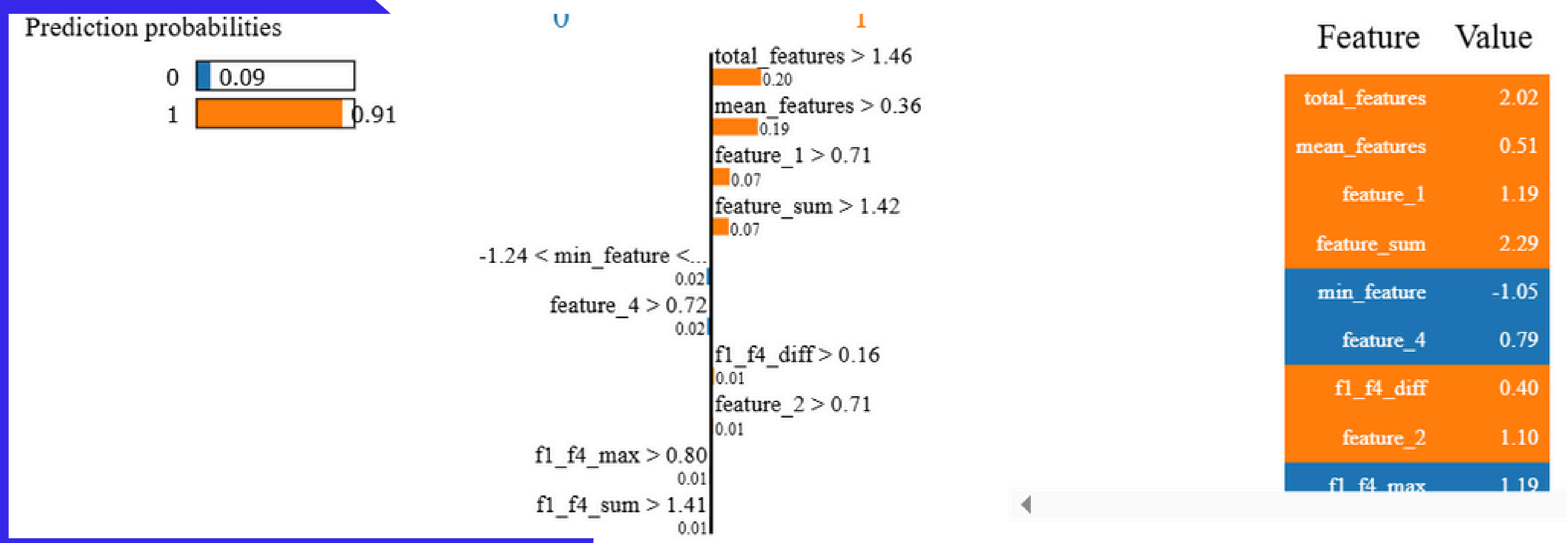
```
from sklearn.ensemble import GradientBoostingClassifier
#Train the Gradient Boosting model
gb_model = GradientBoostingClassifier(n_estimators=100, random_state=42)
gb_model.fit(X_train, y_train)

#Make predictions on the test set
gb_pred = gb_model.predict(X_test)

#Evaluate accuracy
gb_accuracy = accuracy_score(y_test, gb_pred)
print(f'Gradient Boosting Accuracy: {gb_accuracy:.4f}')

#Classification report
print("\nClassification Report for Gradient Boosting:")
print(classification_report(y_test, gb_pred))

#ROC AUC score
gb_proba = gb_model.predict_proba(X_test)[:, 1]
gb_roc_auc = roc_auc_score(y_test, gb_proba)
print(f"Gradient Boosting ROC AUC Score: {gb_roc_auc:.4f}")
```

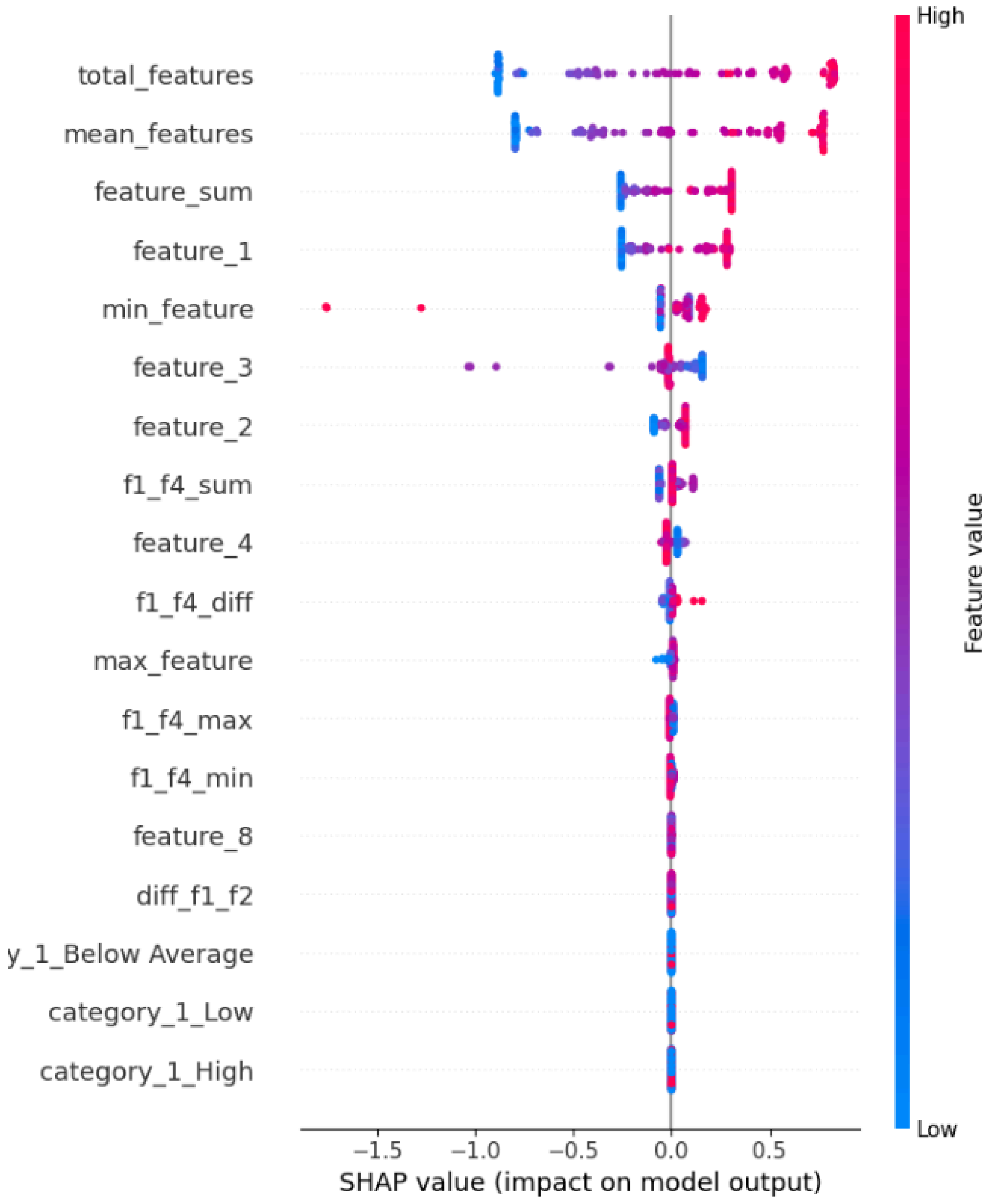
```
Gradient Boosting Accuracy: 0.8737

Classification Report for Gradient Boosting:
              precision    recall  f1-score   support

         0.0       0.86      0.90      0.88       889
         1.0       0.89      0.84      0.87       845

    accuracy                           0.87      1734
   macro avg       0.87      0.87      0.87      1734
weighted avg       0.87      0.87      0.87      1734

Gradient Boosting ROC AUC Score: 0.9518
```

```
Best Hyperparameters: {'learning_rate': 0.01, 'max_depth':
CV Accuracy: 0.8828
Test Accuracy: 0.8789

Classification Report:
              precision    recall  f1-score   support

         0.0       0.86      0.91      0.89       889
         1.0       0.90      0.84      0.87       845

    accuracy                           0.88      1734
   macro avg       0.88      0.88      0.88      1734
weighted avg       0.88      0.88      0.88      1734

ROC AUC Score for Best Gradient Boosting Model: 0.9512
```

PERFORMANCE(GRADIENT BOOSTING)







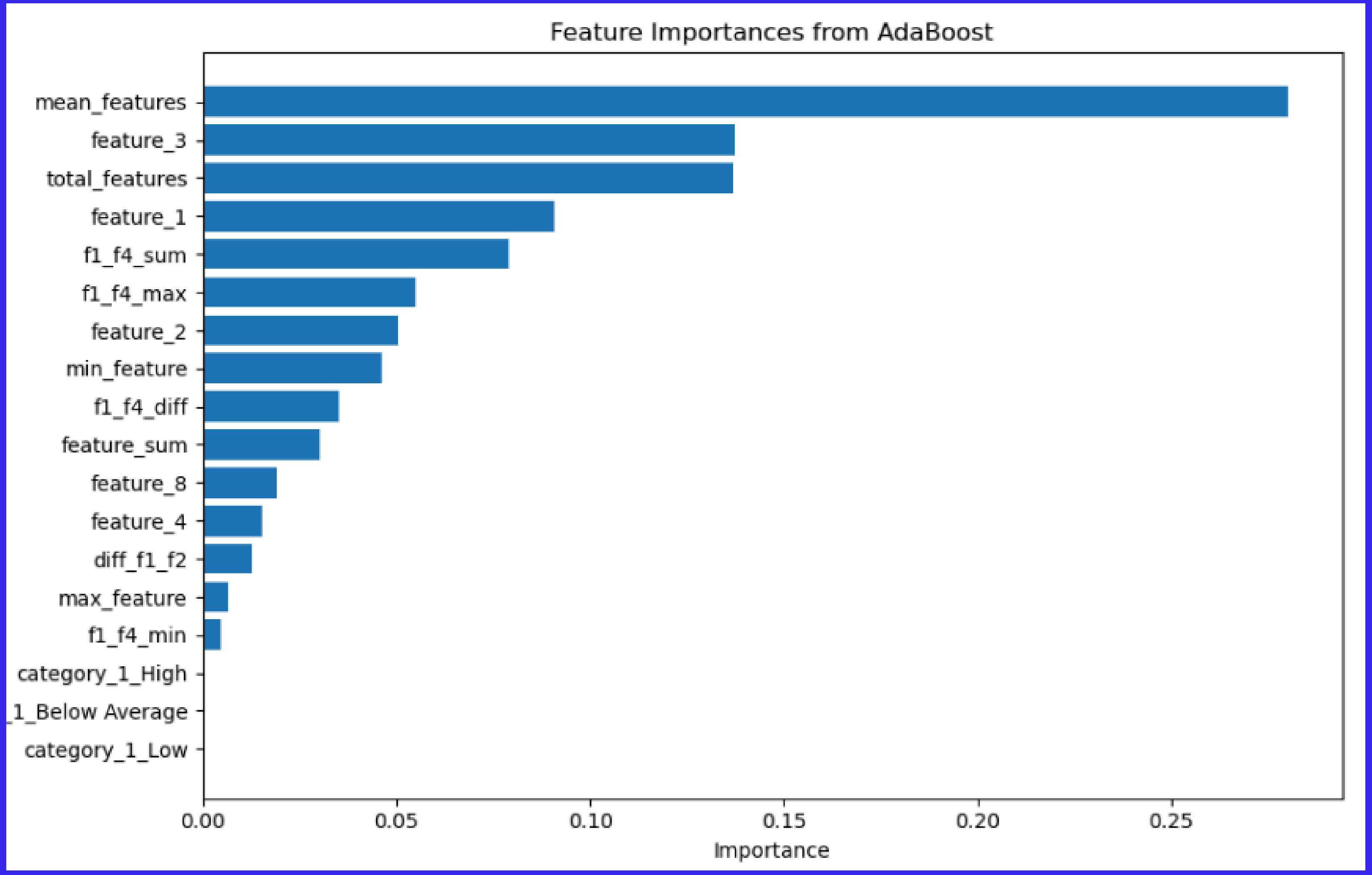
MODDELING (ADABOOST)

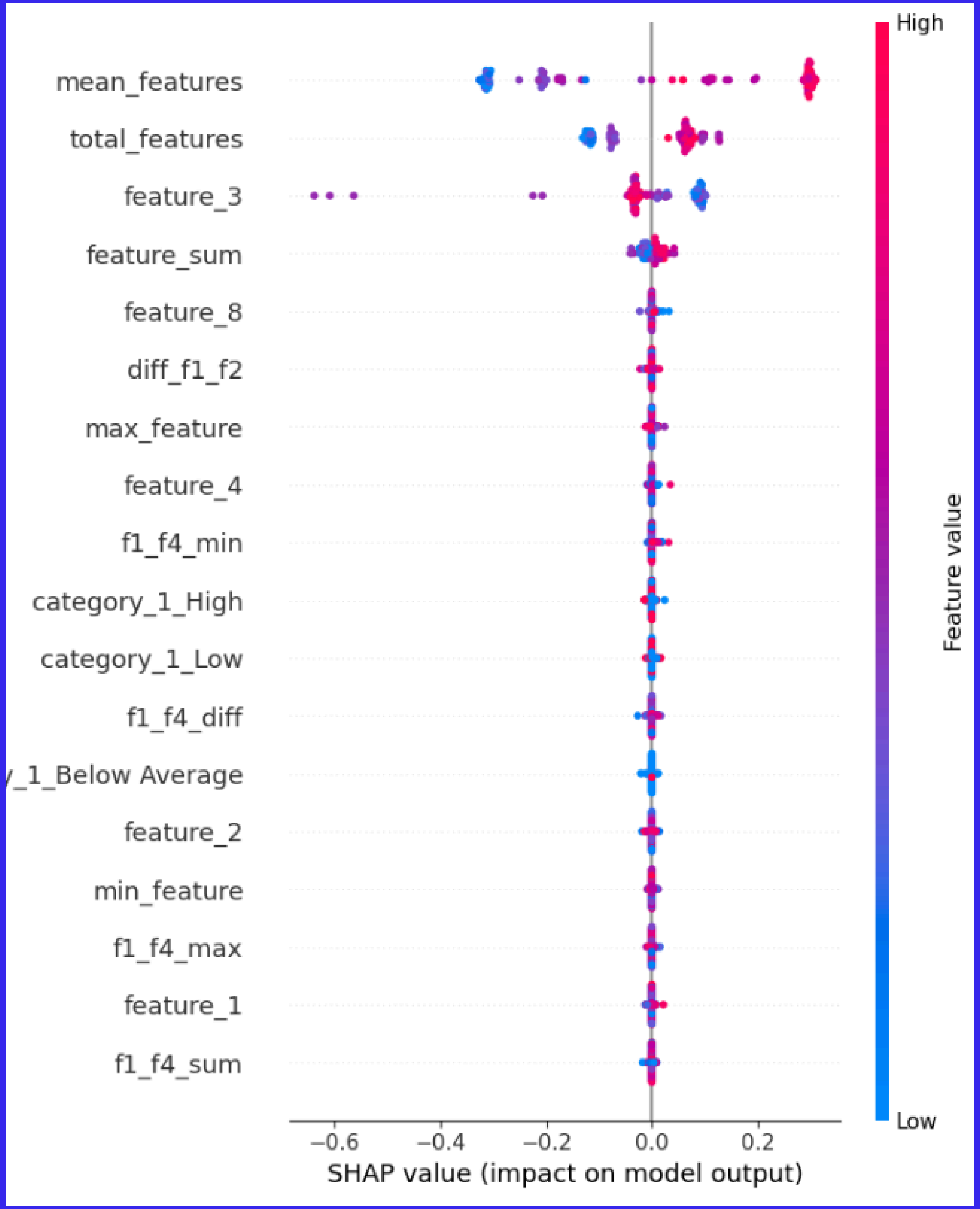
```python
from sklearn.ensemble import AdaBoostClassifier
#Train the AdaBoost model
ab_model = AdaBoostClassifier(n_estimators=100, random_state=42)
ab_model.fit(X_train, y_train)

#Make predictions on the test set
ab_pred = ab_model.predict(X_test)

#Evaluate accuracy
ab_accuracy = accuracy_score(y_test, ab_pred)
print(f'AdaBoost Accuracy: {ab_accuracy:.4f}')

#Classification report
print("\nClassification Report for AdaBoost:")
print(classification_report(y_test, ab_pred))

#ROC AUC score
ab_proba = ab_model.predict_proba(X_test)[:, 1]
ab_roc_auc = roc_auc_score(y_test, ab_proba)
print(f"AdaBoost ROC AUC Score: {ab_roc_auc:.4f}")
```

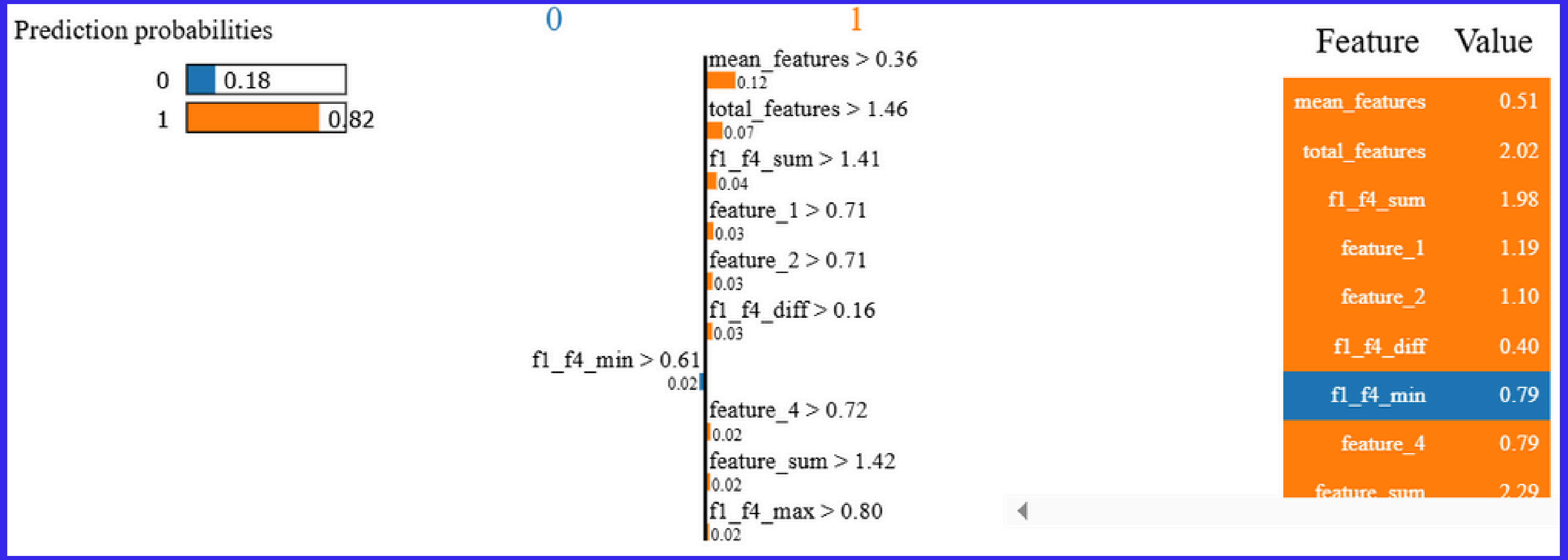
```
AdaBoost Accuracy: 0.8749

Classification Report for AdaBoost:
              precision    recall  f1-score   support

         0.0       0.85      0.92      0.88       889
         1.0       0.91      0.83      0.87       845

    accuracy                           0.87      1734
   macro avg       0.88      0.87      0.87      1734
weighted avg       0.88      0.87      0.87      1734

AdaBoost ROC AUC Score: 0.9480
```

```
Best Hyperparameters: {'estimator__max_depth': 3, 'learn
CV Accuracy: 0.8825
Test Accuracy: 0.8829

Classification Report:
              precision    recall  f1-score   support

         0.0       0.88      0.89      0.89       889
         1.0       0.88      0.87      0.88       845

    accuracy                           0.88      1734
   macro avg       0.88      0.88      0.88      1734
weighted avg       0.88      0.88      0.88      1734

ROC AUC Score for Best AdaBoost Model: 0.9501
```

PERFORMANCE (ADABOOST)







LIME AND SHAP

- The analysis using SHAP and LIME revealed that each model relies on a specific set of features that have the strongest influence on predicting the target variable. By identifying and focusing on these key features, we can improve model accuracy and overall efficiency.
- Less impactful features can be removed or further examined, helping to simplify the model without sacrificing performance. This approach enhances the model's effectiveness and its applicability in real-world business scenarios.

BUSINESS LOGIC

A car service station uses a model to predict the likelihood of a car breaking down based on factors like age, mileage, oil type, and repair history.

- SHAP helps explain how each factor (e.g., age, mileage) influences the prediction.

- LIME explains individual predictions, showing why a specific car is at high risk of breakdown (e.g., due to high mileage or previous issues).

Application in business

- **Personalized Recommendations:** The service station can offer services (e.g., diagnostics or oil changes) based on factors influencing breakdowns.
- **Inventory Optimization:** The service station can prepare necessary parts in advance, knowing which cars are more likely to break down.
- **Marketing and Loyalty:** The service station can offer discounts on services based on breakdown predictions.
- **Decision Support:** SHAP and LIME help managers understand how to improve service by providing explanations for the model's predictions.
- **Regulatory Compliance:** Using SHAP and LIME allows the service station to be transparent in decision-making and ensure customer data protection.

Thanks for your attention!