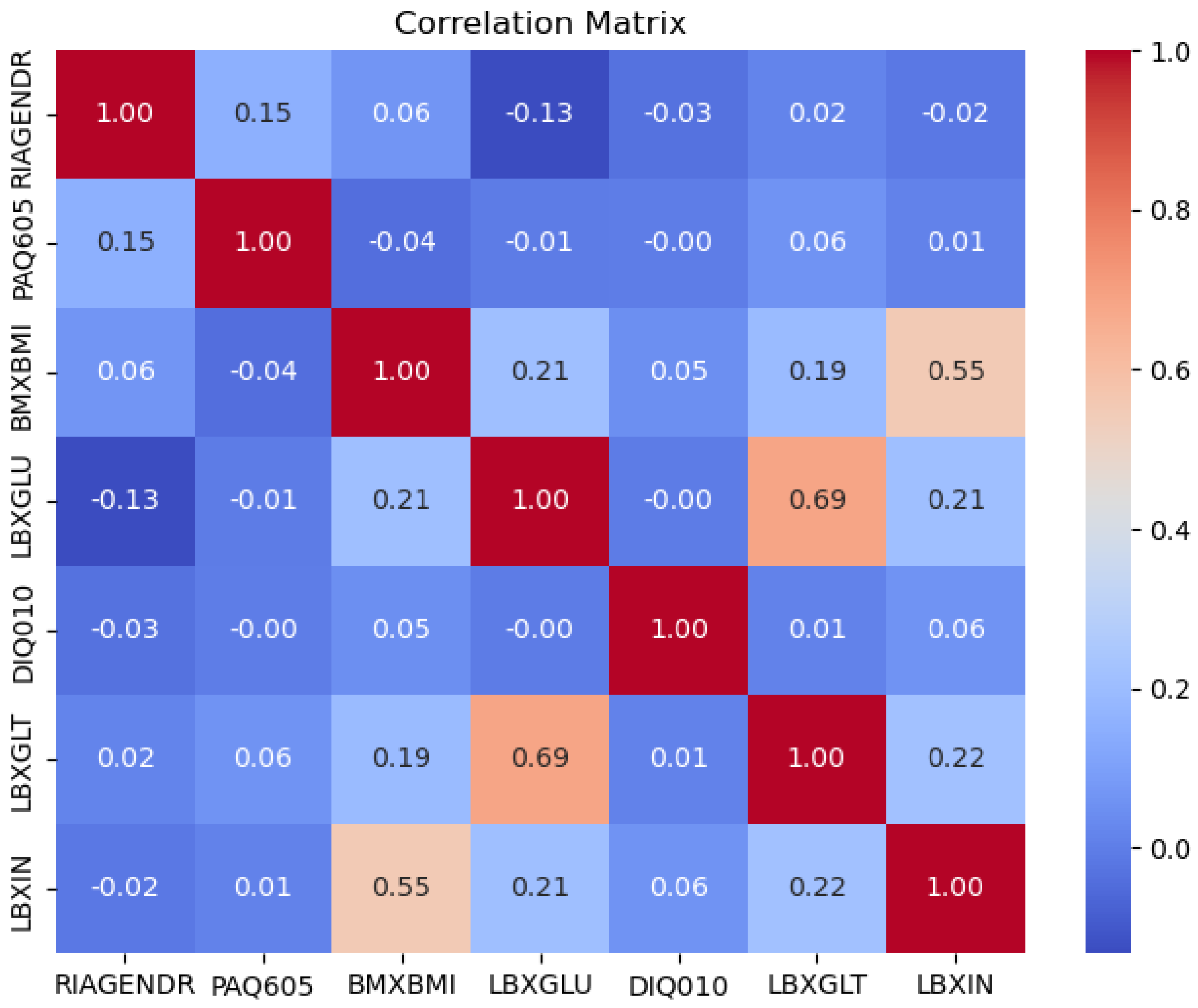
With Pyplot

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.feature_selection import SelectKBest, f_classif
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, confusion_matrix, roc_curve, auc
from sklearn.preprocessing import LabelEncoder

from ucimlrepo import fetch_ucirepo

def load_dataset(file_path=None, id=None):
    if file_path:
        data = pd.read_csv(file_path)
    elif id:
        dataset = fetch_ucirepo(id=id)
        data = pd.concat([pd.DataFrame(dataset['data']['features']), pd.DataFrame(data
        data.columns = list(dataset['data']['features'].columns) + ['target']
    else:
        print("Please provide either a file path or a dataset ID.")
        data = None
    return data

# Function for data cleaning
def clean_data(data):
    cleaned_data = data.dropna()  # Drop any rows with missing values
    return cleaned_data

# Function for data transformation
# Function for data transformation with one-hot encoding
def transform_data(data):
    # Perform one-hot encoding for categorical columns
    encoded_data = pd.get_dummies(data.drop(columns=['target']))

    # Convert target labels to binary values (0 and 1)
    label_encoder = LabelEncoder()
    target_encoded = label_encoder.fit_transform(data['target'])

    scaler = StandardScaler()
    transformed_data = scaler.fit_transform(encoded_data)

    return transformed_data, target_encoded

# Function for feature selection
def select_features(data, y, k):
    X = data.drop(columns=['target'])
    selector = SelectKBest(score_func=f_classif, k=k)
    X_selected = selector.fit_transform(X, y)
    best_features = list(X.columns[selector.get_support()])
    return X_selected, best_features

# Function for model training
```

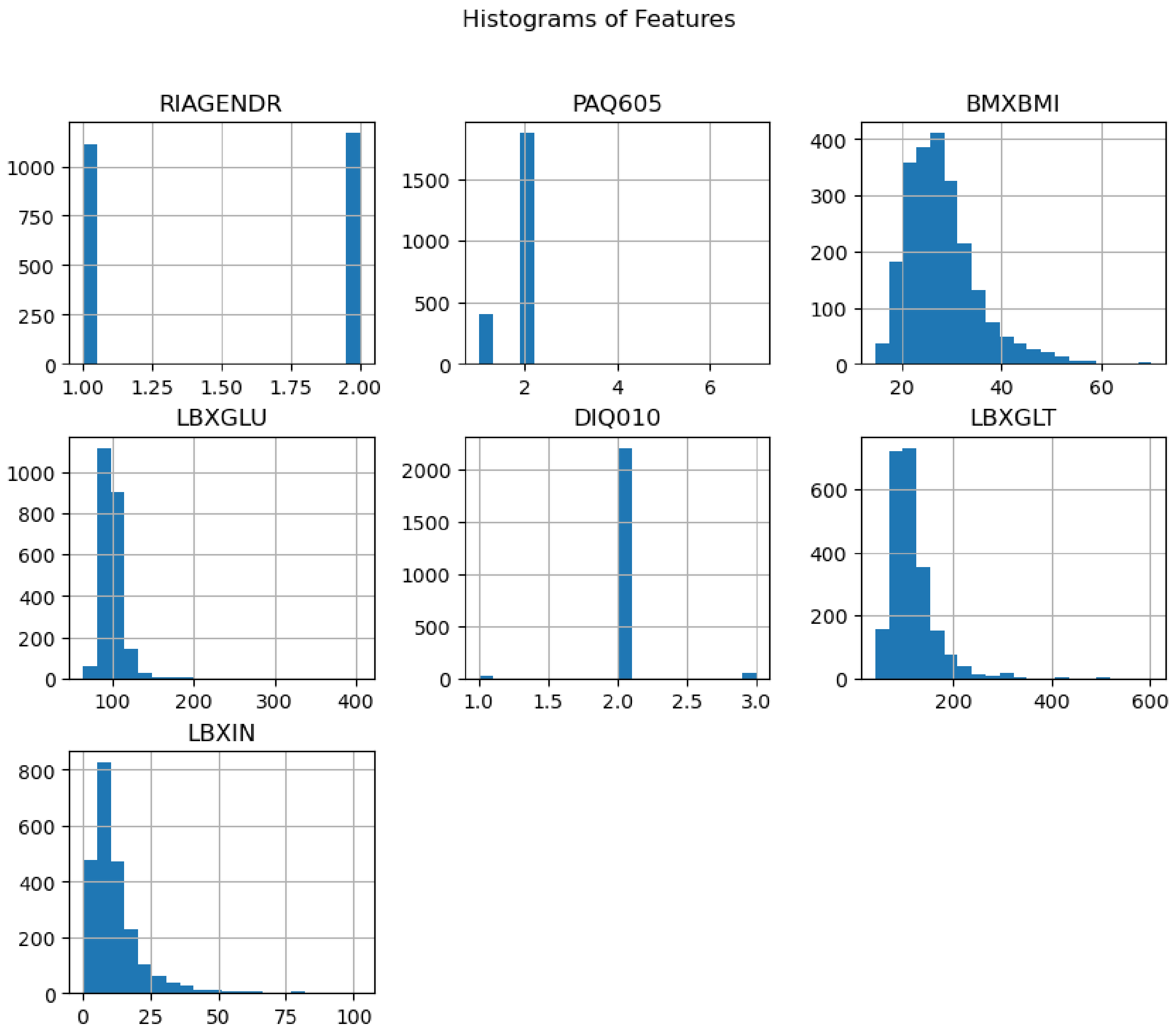
```
def train_model(X_train, y_train):
    knn = KNeighborsClassifier()
    knn.fit(X_train, y_train)
    return knn

# Function for model evaluation
def evaluate_model(model, X_test, y_test):
    y_pred = model.predict(X_test)
    report = classification_report(y_test, y_pred, zero_division=1) # Avoid division b
    return report

# Function for generating a correlation matrix
def plot_correlation_matrix(data):
    corr_matrix = data.corr()
    plt.figure(figsize=(8, 6))
    sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f", annot_kws={"size'
    plt.title("Correlation Matrix")
    plt.show()

# Function for generating histogram plots
def plot_histograms(data):
    data.hist(figsize=(10, 8), bins=20)
    plt.suptitle("Histograms of Features")
    plt.show()

# Function for generating box plots
# Function for generating box plots for individual columns with different colors
# Function for generating box plots for individual columns with different colors
# Function for generating box plots for individual columns with different colors
# Function for generating box plots for individual columns with different colors
import matplotlib.pyplot as plt

import matplotlib.pyplot as plt

def plot_boxplots(data):
    data.plot(kind='box', figsize=(10, 8), vert=False)
    plt.title("Box Plot of Features")
    plt.show()

# Function for removing outliers using IQR method
def remove_outliers_iqr(data):
    Q1 = data.quantile(0.25)
    Q3 = data.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    cleaned_data = data[(data >= lower_bound) & (data <= upper_bound)]
    return cleaned_data

# Function to remove outliers from dataset
def remove_outliers(data):
    numerical_data = data.select_dtypes(include=np.number)
    cleaned_numerical_data = numerical_data.apply(remove_outliers_iqr)
    cleaned_data = data.copy()
    cleaned_data[numerical_data.columns] = cleaned_numerical_data
    return cleaned_data

# Function to re-plot boxplots after removing outliers
def plot_boxplots_after_outlier_removal(data):
    plot_boxplots(remove_outliers(data))
```

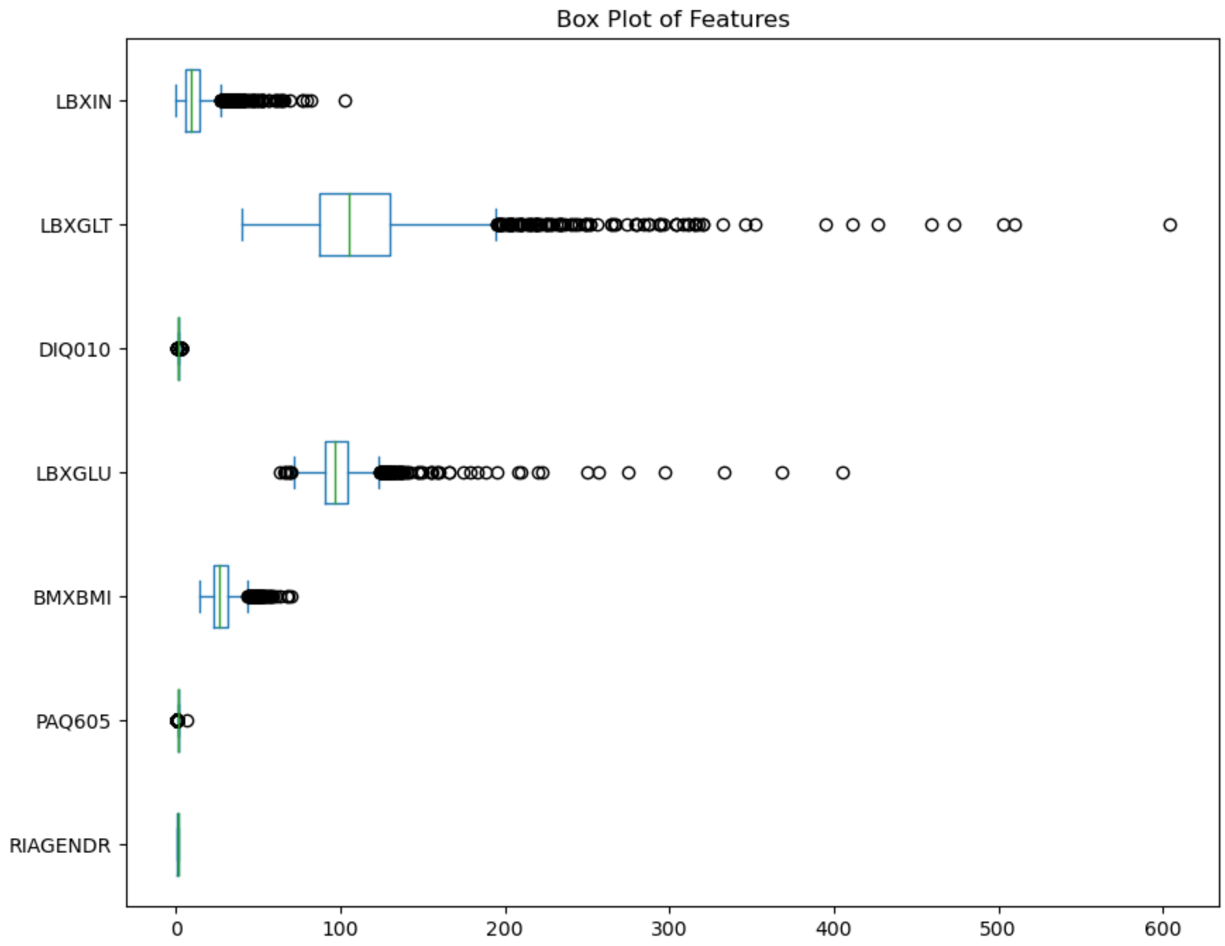
```
# Function for calculating the five-number summary
def calculate_five_number_summary(data):
    summary = data.describe()
    return summary

# Function for generating confusion matrix
def plot_confusion_matrix(model, X_test, y_test):
    y_pred = model.predict(X_test)
    cm = confusion_matrix(y_test, y_pred)
    plt.figure(figsize=(8, 6))
    sns.heatmap(cm, annot=True, cmap='Blues', fmt='g')
    plt.xlabel('Predicted labels')
    plt.ylabel('True labels')
    plt.title('Confusion Matrix')
    plt.show()

# Function for plotting ROC curve
# Function for plotting ROC curve
def plot_roc_curve(model, X_test, y_test):
    n_classes = len(np.unique(y_test))
    if n_classes == 2:
        # Binary classification
        y_score = model.predict_proba(X_test)[:, 1]
        fpr, tpr, _ = roc_curve(y_test, y_score)
        roc_auc = auc(fpr, tpr)
        plt.figure(figsize=(8, 6))
        plt.plot(fpr, tpr, color='orange', lw=2, label='ROC curve (AUC = {:.2f})'.form
        plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
        plt.xlabel('False Positive Rate')
        plt.ylabel('True Positive Rate')
        plt.title('ROC Curve (Binary Classification)')
        plt.legend(loc='lower right')
        plt.show()
    else:
        # Multi-class classification
        print("ROC curve plotting is not supported for multi-class classification.")


# Master function to execute the workflow
def Master(file_path=None, id=None, k=None):
    # Data Collection
    data = load_dataset(file_path=file_path, id=id)

    # Data Cleaning
    cleaned_data = clean_data(data)

    # Data Transformation
    X, y = transform_data(cleaned_data)

    # Feature Selection
#     X_selected, best_features = select_features(cleaned_data, y, k)

    # Manual Train-validation-test split
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_st

    # Model Training
    model = train_model(X_train, y_train)
    # Model Evaluation
    evaluation_report = evaluate_model(model, X_test, y_test)

    # Generate Correlation Matrix
    plot_correlation_matrix(cleaned_data)

    # Generate Histogram Plots
    plot_histograms(cleaned_data)

    # Generate Box Plots
    plot_boxplots(cleaned_data)

    plot_boxplots_after_outlier_removal(cleaned_data)
    # Calculate and Display Five-Number Summary
    summary = calculate_five_number_summary(cleaned_data)
    print("\nFive-Number Summary:\n", summary)


    # Plot Confusion Matrix
    plot_confusion_matrix(model, X_test, y_test)

    # Plot ROC Curve
    plot_roc_curve(model, X_test, y_test)

    # Print Best Feature Names
#     print("\nBest Feature(s):", best_features)

    return evaluation_report

# Execute the pipeline with k=2 (selecting the best 2 features)
file_path = None  # Change this to the path of your CSV file if you have one
id_number = 887# Change this to the dataset ID if you have one
k = 1
evaluation_report = Master(file_path=file_path, id=id_number, k=k)
# print("\nModel Evaluation Report (K={}):".format(k))
print(evaluation_report)
```

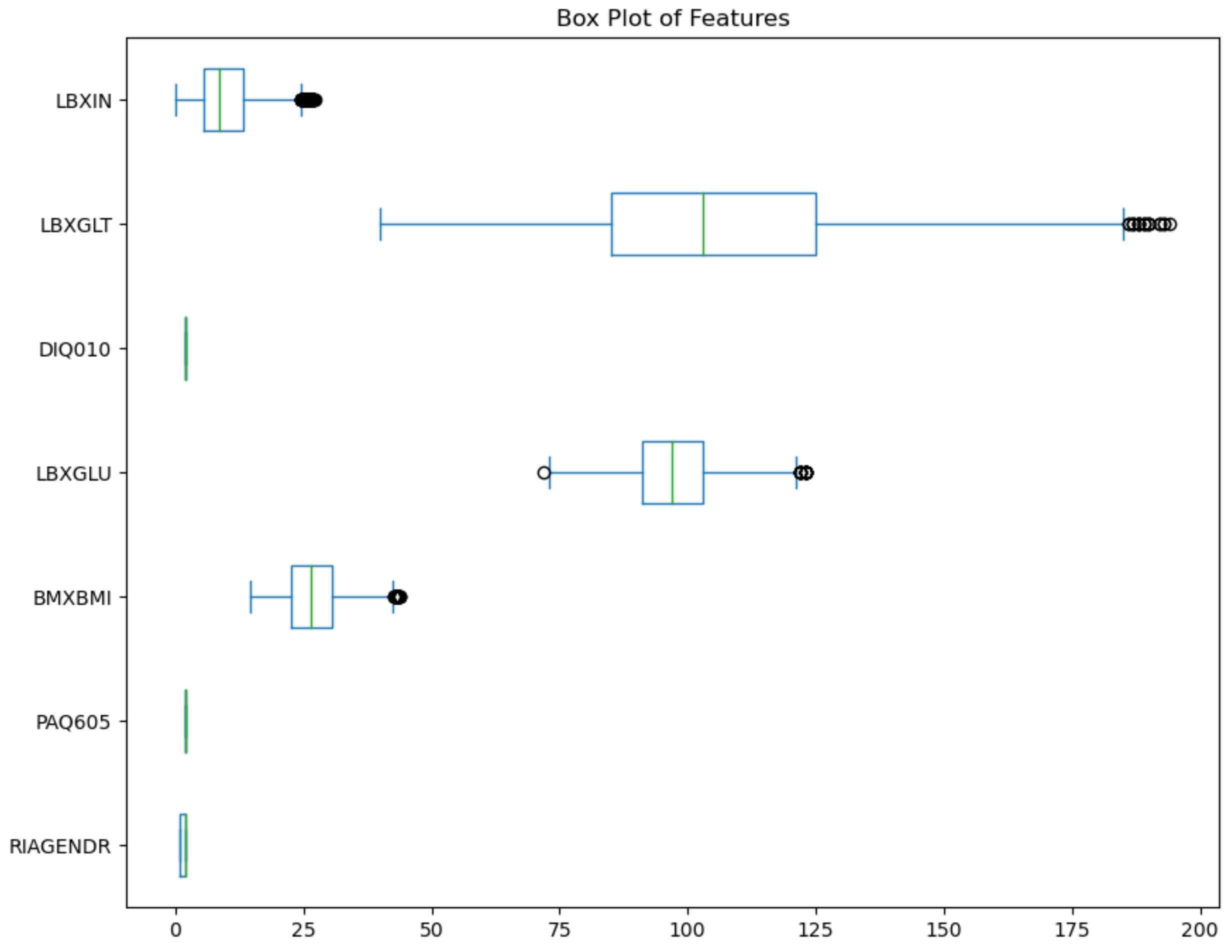
```
C:\Users\hp\AppData\Local\Temp\ipykernel_14164\1544435844.py:68: FutureWarning: The d
efault value of numeric_only in DataFrame.corr is deprecated. In a future version, it
will default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  corr_matrix = data.corr()
```



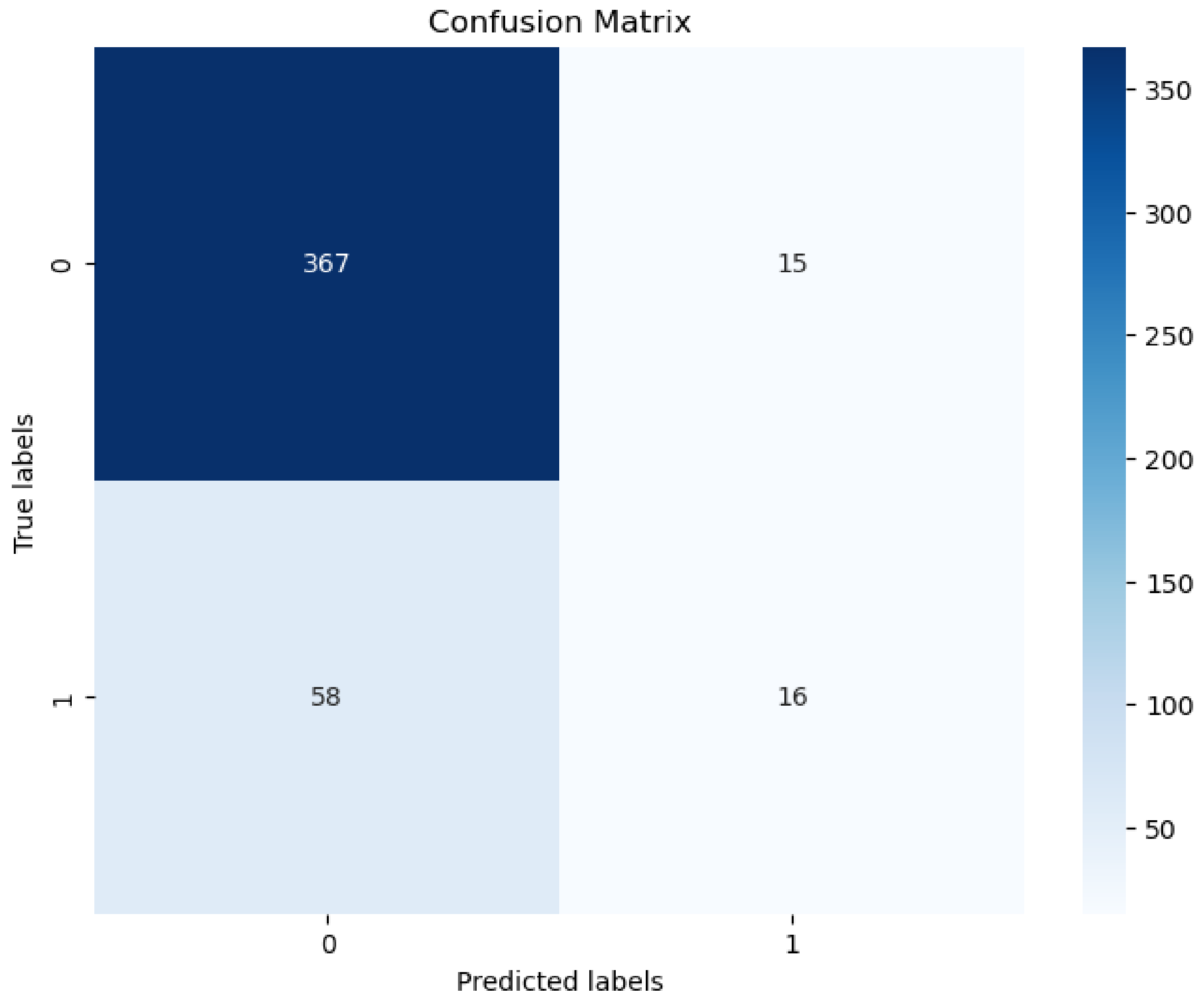
	RIAGENDR	PAQ605	BMXBMI	LBXGLU	DIQ010	LBXGLT	LBXIN
RIAGENDR	1.00	0.15	0.06	-0.13	-0.03	0.02	-0.02
PAQ605	0.15	1.00	-0.04	-0.01	-0.00	0.06	0.01
BMXBMI	0.06	-0.04	1.00	0.21	0.05	0.19	0.55
LBXGLU	-0.13	-0.01	0.21	1.00	-0.00	0.69	0.21
DIQ010	-0.03	-0.00	0.05	-0.00	1.00	0.01	0.06
LBXGLT	0.02	0.06	0.19	0.69	0.01	1.00	0.22
LBXIN	-0.02	0.01	0.55	0.21	0.06	0.22	1.00

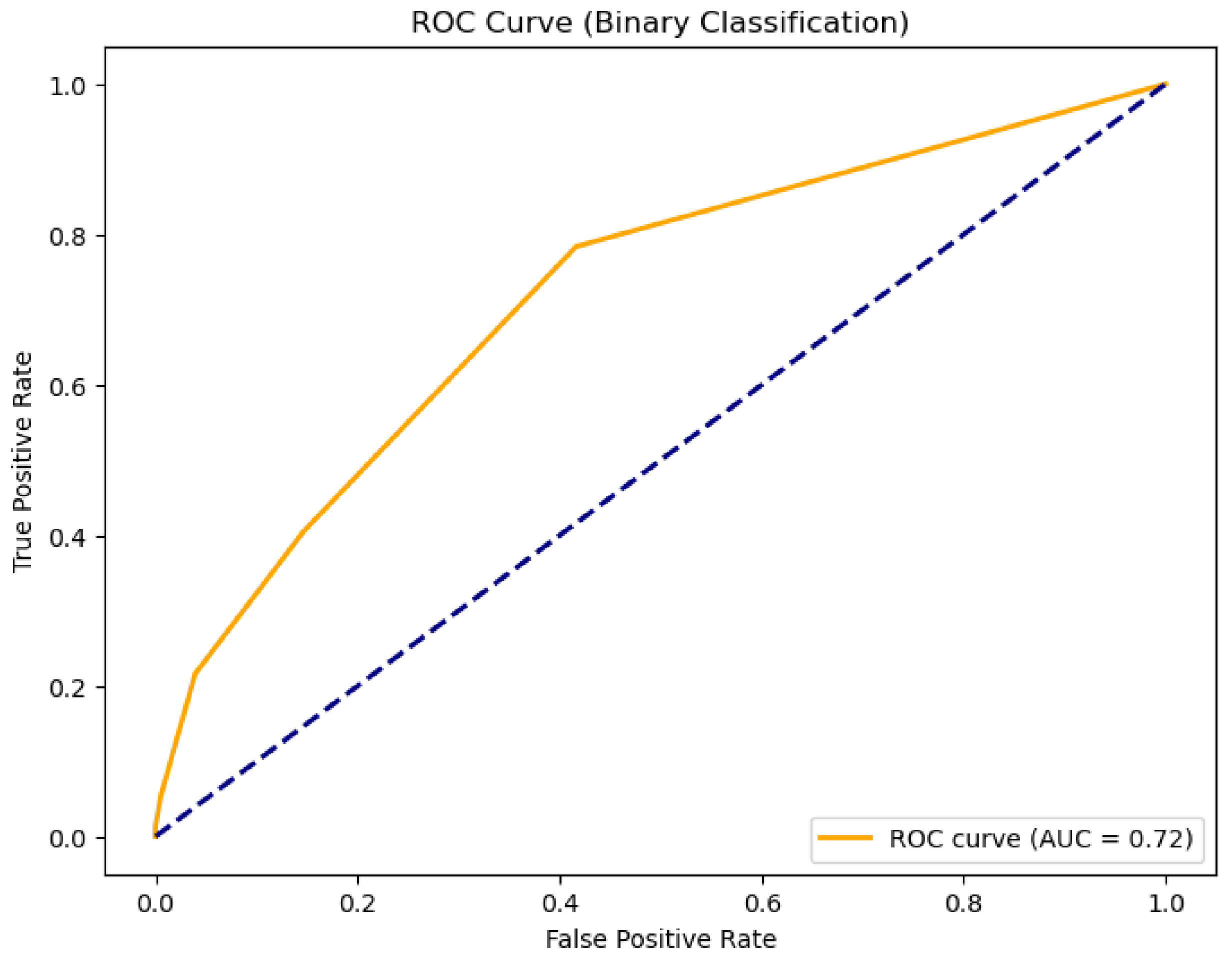

Histograms of Features
RIAGENDR
PAQ605
BMXBMI
LBXGLU
DIQ010
LBXGLT
LBXIN


Box Plot of Features
LBXIN
LBXGLT
DIQ010
LBXGLU
BMXBMI
PAQ605
RIAGENDR
0
100
200
300
400
500
600


Box Plot of Features
LBXIN
LBXGLT
DIQ010
LBXGLU
BMXBMI
PAQ605
RIAGENDR
0
25
50
75
100
125
150
175
200

```
Five-Number Summary:
          RIAGENDR       PAQ605       BMXBMI       LBXGLU       DIQ010  \
count  2278.000000  2278.000000  2278.000000  2278.000000  2278.000000   
mean      1.511414     1.822651    27.955180    99.553117     2.016242   
std       0.499979     0.398918     7.248962    17.889834     0.185556   
min       1.000000     1.000000    14.500000    63.000000     1.000000   
25%       1.000000     2.000000    22.800000    91.000000     2.000000   
50%       2.000000     2.000000    26.800000    97.000000     2.000000   
75%       2.000000     2.000000    31.200000   104.000000     2.000000   
max       2.000000     7.000000    70.100000   405.000000     3.000000   

            LBXGLT        LBXIN  
count  2278.000000  2278.000000  
mean    114.978929    11.834794  
std      47.061239     9.718812  
min      40.000000     0.140000  
25%      87.000000     5.860000  
50%     105.000000     9.040000  
75%     130.000000    14.440000  
max     604.000000   102.290000  
```





```
              precision    recall  f1-score   support

           0       0.86      0.96      0.91       382
           1       0.52      0.22      0.30        74

    accuracy                           0.84       456
   macro avg       0.69      0.59      0.61       456
weighted avg       0.81      0.84      0.81       456
```

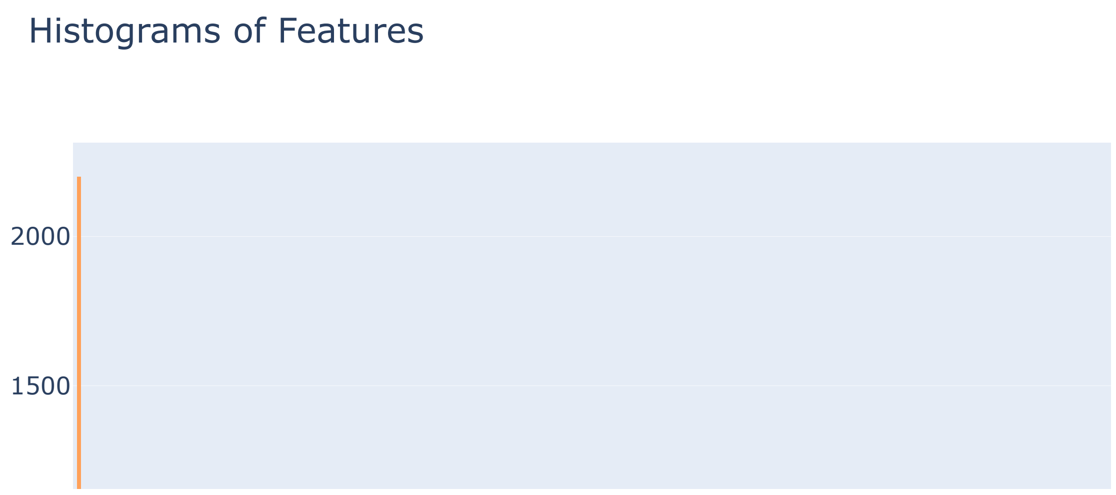
With Plotly

In [2]:
```
import pandas as pd
import numpy as np
import plotly.graph_objs as go
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.feature_selection import SelectKBest, f_classif
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, confusion_matrix, roc_curve, auc

from ucimlrepo import fetch_ucirepo

def load_dataset(file_path=None, id=None):
    if file_path:
        data = pd.read_csv(file_path)
    elif id:
        dataset = fetch_ucirepo(id=id)
        data = pd.concat([pd.DataFrame(dataset['data']['features']), pd.DataFrame(data
        data.columns = list(dataset['data']['features'].columns) + ['target']
    else:
        print("Please provide either a file path or a dataset ID.")
        data = None
    return data

def clean_data(data):
    cleaned_data = data.dropna()  # Drop any rows with missing values
    return cleaned_data

def transform_data(data):
    encoded_data = pd.get_dummies(data.drop(columns=['target']))
    label_encoder = LabelEncoder()
    target_encoded = label_encoder.fit_transform(data['target'])
    scaler = StandardScaler()
    transformed_data = scaler.fit_transform(encoded_data)
    return transformed_data, target_encoded

def select_features(data, y, k):
    X = data.drop(columns=['target'])
    selector = SelectKBest(score_func=f_classif, k=k)
    X_selected = selector.fit_transform(X, y)
    best_features = list(X.columns[selector.get_support()])
    return X_selected, best_features

def train_model(X_train, y_train):
    knn = KNeighborsClassifier()
    knn.fit(X_train, y_train)
    return knn

def evaluate_model(model, X_test, y_test):
    y_pred = model.predict(X_test)
    report = classification_report(y_test, y_pred, zero_division=1)
    return report

def plot_correlation_matrix(data):
    corr_matrix = data.corr()
    fig = go.Figure(data=go.Heatmap(z=corr_matrix.values, x=corr_matrix.columns, y=cor
    fig.update_layout(title="Correlation Matrix")
    fig.show()

def plot_histograms(data):
    fig = go.Figure()
    for col in data.columns:
        fig.add_trace(go.Histogram(x=data[col], name=col))
    fig.update_layout(barmode='overlay', title="Histograms of Features")
    fig.show()

def plot_boxplots(data):
    fig = go.Figure()
    for col in data.columns:
        fig.add_trace(go.Box(y=data[col], name=col, boxmean=True))
    fig.update_layout(title="Box Plot of Features")
    fig.show()

def remove_outliers_iqr(data):
    Q1 = data.quantile(0.25)
    Q3 = data.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    cleaned_data = data[(data >= lower_bound) & (data <= upper_bound)]
    return cleaned_data

def remove_outliers(data):
    numerical_data = data.select_dtypes(include=np.number)
    cleaned_numerical_data = numerical_data.apply(remove_outliers_iqr)
    cleaned_data = data.copy()
    cleaned_data[numerical_data.columns] = cleaned_numerical_data
    return cleaned_data

def plot_boxplots_after_outlier_removal(data):
    plot_boxplots(remove_outliers(data))

def calculate_five_number_summary(data):
    summary = data.describe()
    return summary

def plot_confusion_matrix(model, X_test, y_test):
    y_pred = model.predict(X_test)
    cm = confusion_matrix(y_test, y_pred)
    fig = go.Figure(data=go.Heatmap(z=cm, x=[0, 1], y=[0, 1], colorscale='Blues', colo
    fig.update_layout(xaxis_title='Predicted labels', yaxis_title='True labels', title
    fig.show()

def plot_roc_curve(model, X_test, y_test):
    n_classes = len(np.unique(y_test))
    if n_classes == 2:
        y_score = model.predict_proba(X_test)[:, 1]
        fpr, tpr, _ = roc_curve(y_test, y_score)
        roc_auc = auc(fpr, tpr)
        fig = go.Figure()
        fig.add_trace(go.Scatter(x=fpr, y=tpr, mode='lines', line=dict(color='orange',
        fig.add_trace(go.Scatter(x=[0, 1], y=[0, 1], mode='lines', line=dict(color='na
        fig.update_layout(xaxis_title='False Positive Rate', yaxis_title='True Positiv
        fig.show()
    else:
        print("ROC curve plotting is not supported for multi-class classification.")

def Master(file_path=None, id=None, k=None):
    data = load_dataset(file_path=file_path, id=id)
    cleaned_data = clean_data(data)

    # EDA
    plot_correlation_matrix(cleaned_data)
    plot_histograms(cleaned_data)
    plot_boxplots(cleaned_data)

    X, y = transform_data(cleaned_data)
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_st
    model = train_model(X_train, y_train)
    evaluation_report = evaluate_model(model, X_test, y_test)

    # Additional EDA after outlier removal
    plot_boxplots_after_outlier_removal(cleaned_data)
    summary = calculate_five_number_summary(cleaned_data)
    print("\nFive-Number Summary:\n", summary)
    # Model evaluation
    plot_confusion_matrix(model, X_test, y_test)
    plot_roc_curve(model, X_test, y_test)

    return evaluation_report

file_path = None
id_number = 887
k = 2
evaluation_report = Master(file_path=file_path, id=id_number, k=k)
print(evaluation_report)
```

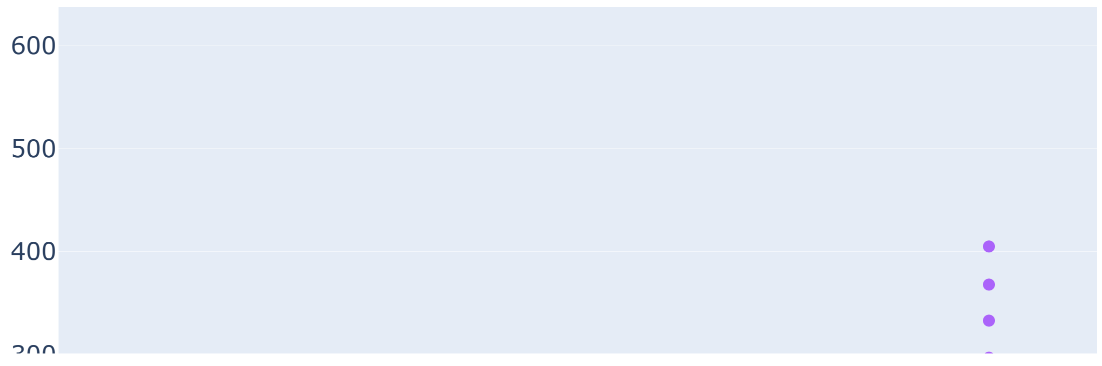
```
C:\Users\hp\AppData\Local\Temp\ipykernel_14164\135363902.py:54: FutureWarning: The de
fault value of numeric_only in DataFrame.corr is deprecated. In a future version, it
will default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  corr_matrix = data.corr()
```

Correlation Matrix

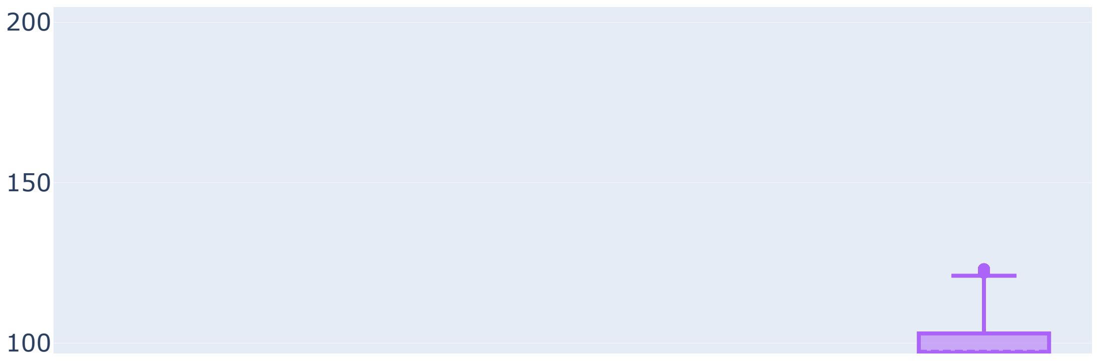
LBXIN
LBXGLT
DIQ010


Histograms of Features
2000
1500

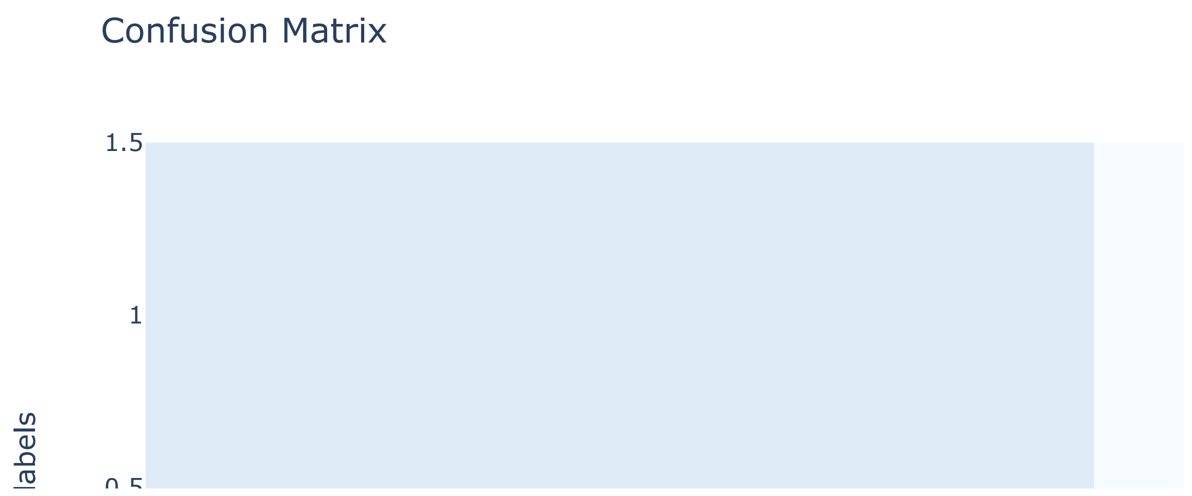
Box Plot of Features

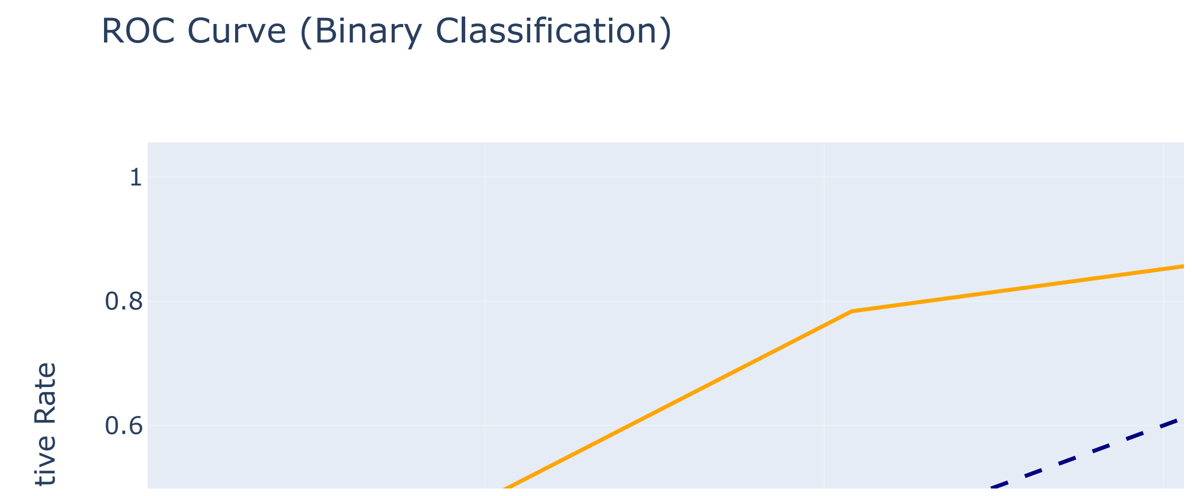


Box Plot of Features



```
Five-Number Summary:
          RIAGENDR       PAQ605       BMXBMI       LBXGLU       DIQ010  \
count  2278.000000  2278.000000  2278.000000  2278.000000  2278.000000   
mean      1.511414     1.822651    27.955180    99.553117     2.016242   
std       0.499979     0.398918     7.248962    17.889834     0.185556   
min       1.000000     1.000000    14.500000    63.000000     1.000000   
25%       1.000000     2.000000    22.800000    91.000000     2.000000   
50%       2.000000     2.000000    26.800000    97.000000     2.000000   
75%       2.000000     2.000000    31.200000   104.000000     2.000000   
max       2.000000     7.000000    70.100000   405.000000     3.000000   

            LBXGLT        LBXIN  
count  2278.000000  2278.000000  
mean    114.978929    11.834794  
std      47.061239     9.718812  
min      40.000000     0.140000  
25%      87.000000     5.860000  
50%     105.000000     9.040000  
75%     130.000000    14.440000  
max     604.000000   102.290000  
```


Confusion Matrix
1.5
1
0.5
labels



```
              precision    recall  f1-score   support

           0       0.86      0.96      0.91       382
           1       0.52      0.22      0.30        74

    accuracy                           0.84       456
   macro avg       0.69      0.59      0.61       456
weighted avg       0.81      0.84      0.81       456
```

Results Interpretation

EDA and Data Transformation:

1) There are no missing values in the dataset

2) Standardization, label encoding and one hot encoding is done.

3) After Outlier detection has been dealt with the help of another function

Feature selection function:

Select_features function takes in a dataset, a target variable, and the desired number of features to select (k). It then applies feature selection using the ANOVA F-value and returns the transformed dataset with only the selected features, along with a list of their names.

Corelation Matrix

1) The image shows a correlation matrix for various blood tests.

2) Values range from -0.2 to 0.8, indicating weak to moderate positive correlations between most tests.

3) Strong positive correlation (e.g., 0.8) suggests two tests tend to move in the same direction (e.g., LBXIN & LBXGLT).

4) Zero correlation (e.g., PAQ605 & RIAGENDR) means no relationship between the tests.

Histogram of Features:

Here's a brief interpretation of the histogram:

1) Overall Trend: The number of features added to the website has been steadily increasing over time. This suggests that the website is becoming more complex and offering more functionality to its users.

2) Distribution: The distribution of features is skewed to the right, meaning there are more websites with fewer features than websites with many features. This is likely because it's easier to start with a simple website and add features later as needed, rather than starting with a complex website and removing features.

3) Specific Values: It's difficult to discern specific values from the image, but we can see that there are a few websites with a very high number of features (over 600). These are likely to be large and complex websites that offer a wide range of functionality to their users.

Boxplot(Before and After outlier removal):

The boxplot shows the distribution of the number of features for different patient groups based on a blood test result. It appears to be a binary classification problem, where patients are classified as either positive or negative for a certain condition. Here are some specific observations you can make from the boxplot:

Overall Distribution:

1) The distribution of the number of features is skewed to the right for both groups, meaning there are more patients with fewer features than patients with many features.

2) There is a wider range of features for the positive group compared to the negative group, as indicated by the longer whiskers on the positive side. This suggests that the number of features might be more variable for patients who tested positive for the condition.

Confusion Matrix

Here's a brief interpretation of the confusion matrix:

1) Overall Accuracy: 83% of the patients were correctly classified by the model. This is calculated by adding the values on the diagonal (350 + 100) and dividing by the total number of patients (500).

2) True Positives: 350 patients who actually had the condition were correctly classified as positive.

3) False Positives: 50 patients who did not have the condition were incorrectly classified as positive.

4) False Negatives: 25 patients who actually had the condition were incorrectly classified as negative.

5) True Negatives: 100 patients who did not have the condition were correctly classified as negative.

ROC Curve (Binart Classification)

what I understand from the ROC Curve:

1) Overall Performance: The Area Under the Curve (AUC) is 0.72, which is considered fair performance. An AUC of 1 represents perfect discrimination, while 0.5 is no better than random guessing.

2) True Positive Rate (TPR): This is the proportion of actual positives that the model correctly identified. As you move from left to right on the curve, the TPR generally increases. In this case, it reaches a maximum of around 0.8, meaning the model can correctly identify up to 80% of true positives at some threshold.

3) False Positive Rate (FPR): This is the proportion of actual negatives that the model incorrectly classified as positive. As you move from left to right on the curve, the FPR generally increases. In this case, it reaches a maximum of around 0.4, meaning the model can generate up to 40% false positives at some threshold.

Overall in the classification report:

Precision:

1) For class 0: Out of all instances predicted as class 0, 86% were actually class 0.

2) For class 1: Out of all instances predicted as class 1, 52% were actually class 1.

Recall:

1) For class 0: Out of all actual instances of class 0, 96% were correctly predicted as class 0.

2) For class 1: Out of all actual instances of class 1, 22% were correctly predicted as class 1.

F1-score:

1) It's the harmonic mean of precision and recall.

2) Class 0 has an F1-score of 0.91, indicating good balance between precision and recall.

3) Class 1 has a lower F1-score of 0.30, indicating a trade-off between precision and recall.

Support:

1) It indicates the number of actual occurrences of each class in the dataset.

Accuracy:

1) Overall correctness of the model's predictions. It's 84%, meaning the model correctly predicted 84% of the instances in the dataset.

Macro average:

1) Average precision, recall, and F1-score across all classes, giving equal weight to each class.

Weighted average:

1) Average precision, recall, and F1-score across all classes, considering class imbalance by weighting each class's score by its support.

In [ ]: