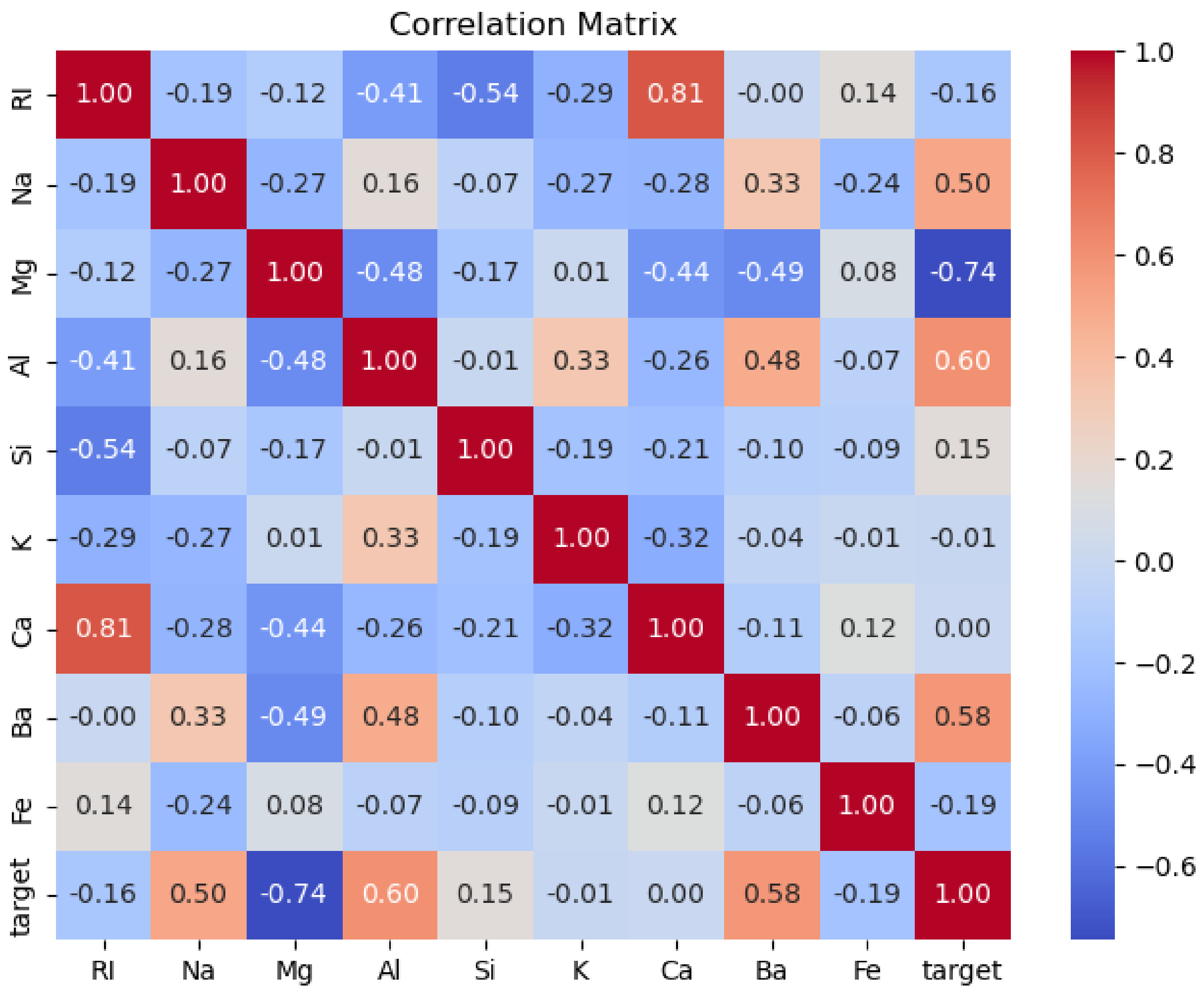
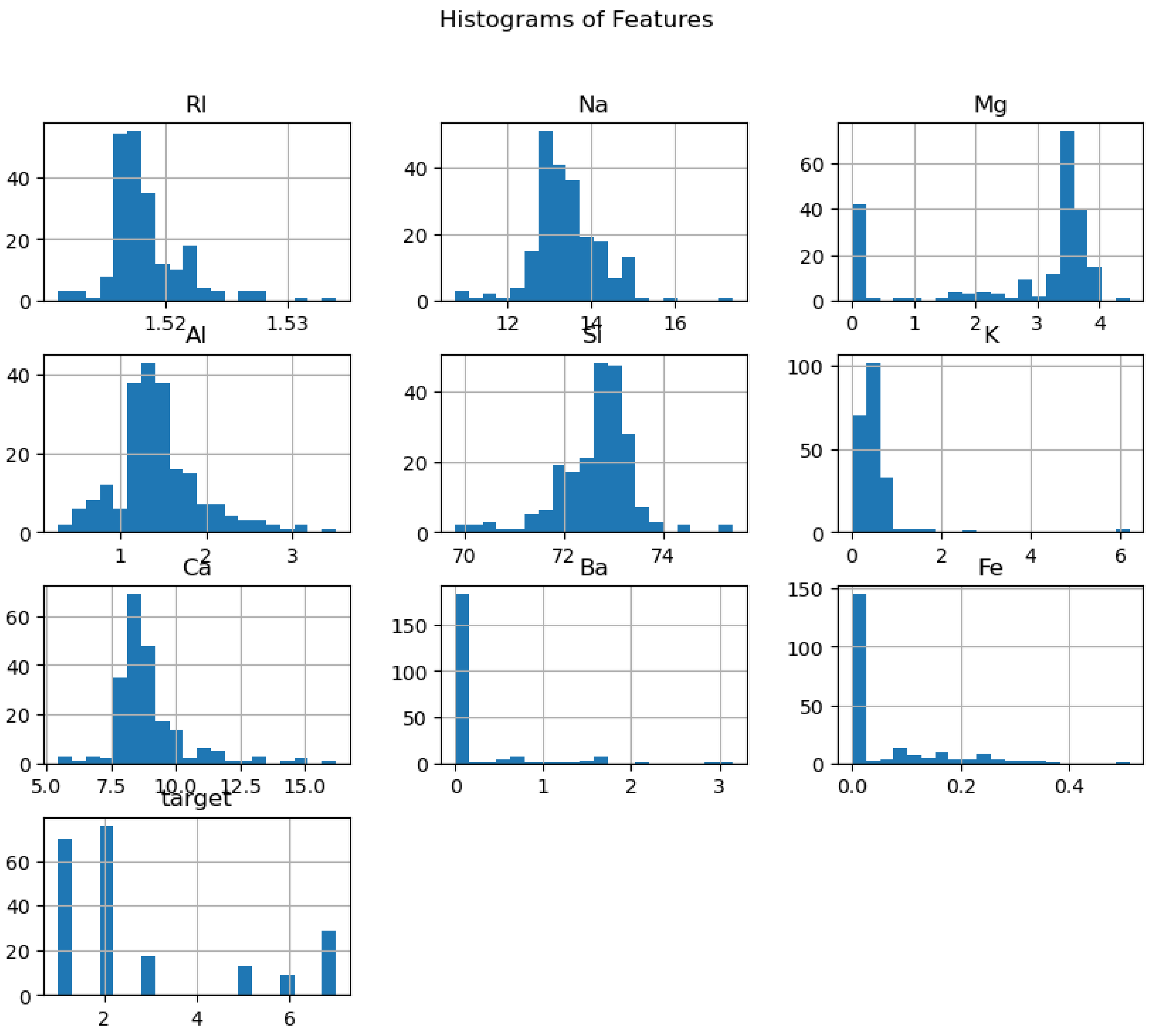
with pyplot

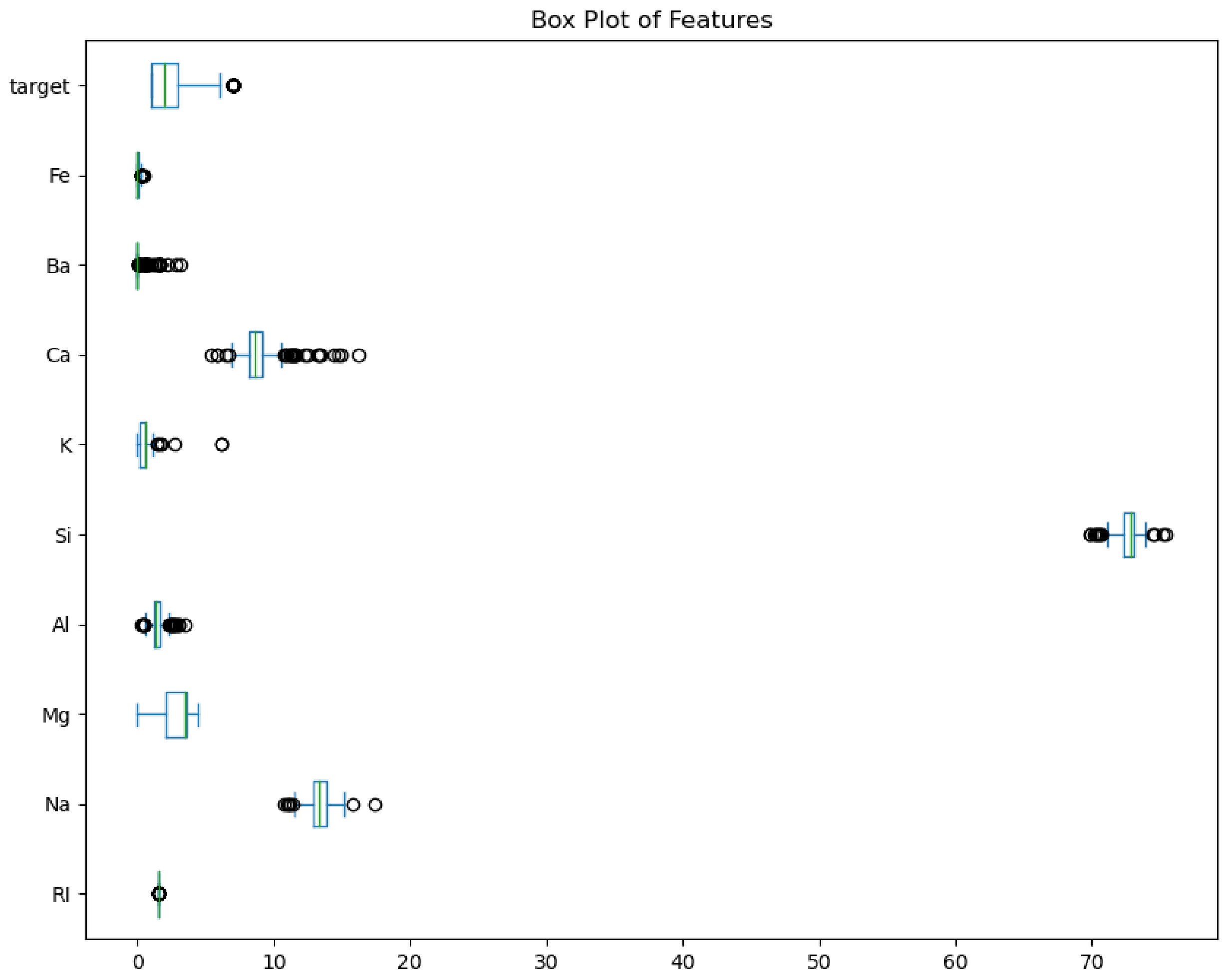
In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.feature_selection import SelectKBest, f_classif
from sklearn.neighbors import KNeighborsClassifier
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
```

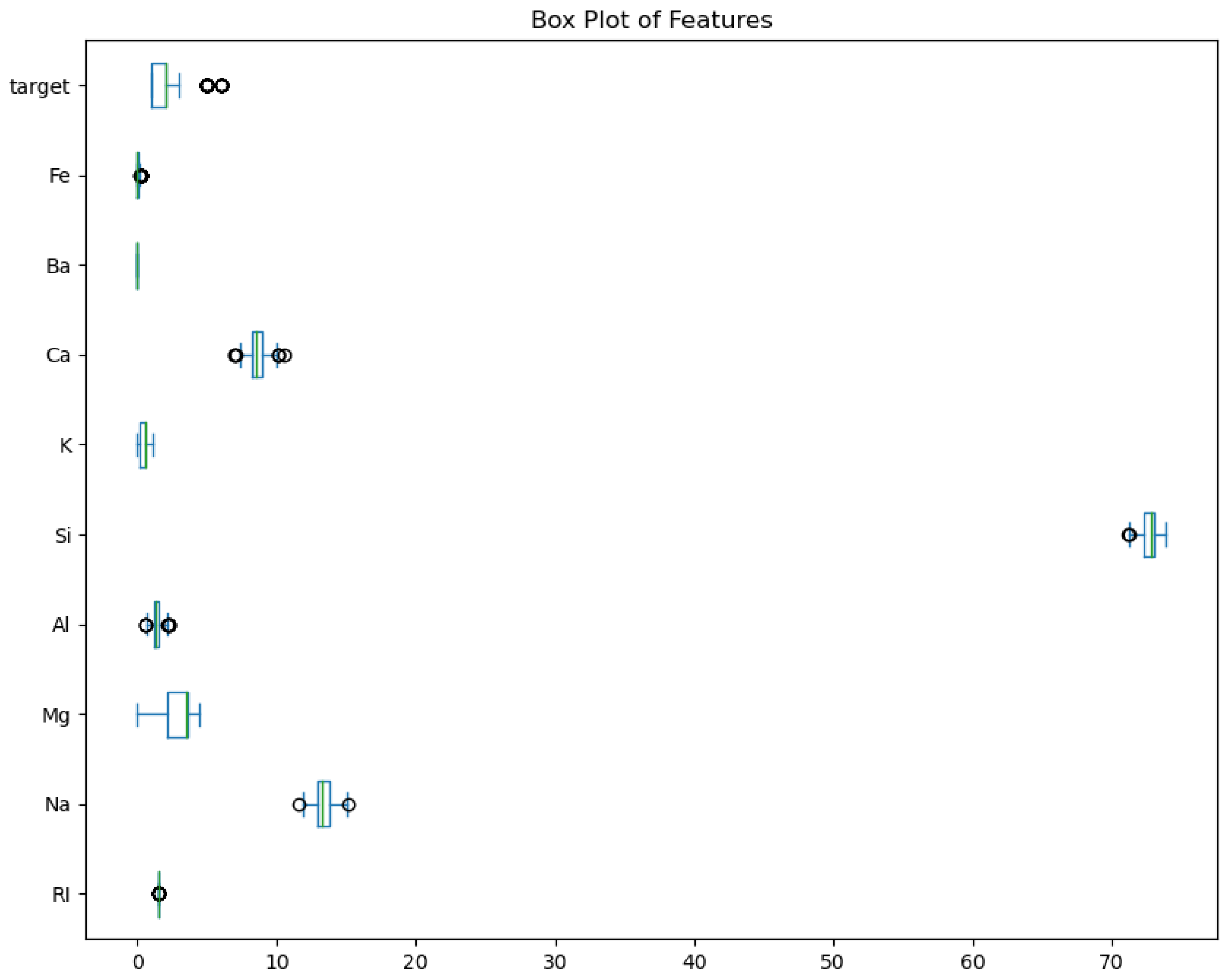
```
def train_model(X_train, y_train):
    knn = KNeighborsClassifier()
    knn.fit(X_train, y_train)
    return knn

# 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 = 42# 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)
```

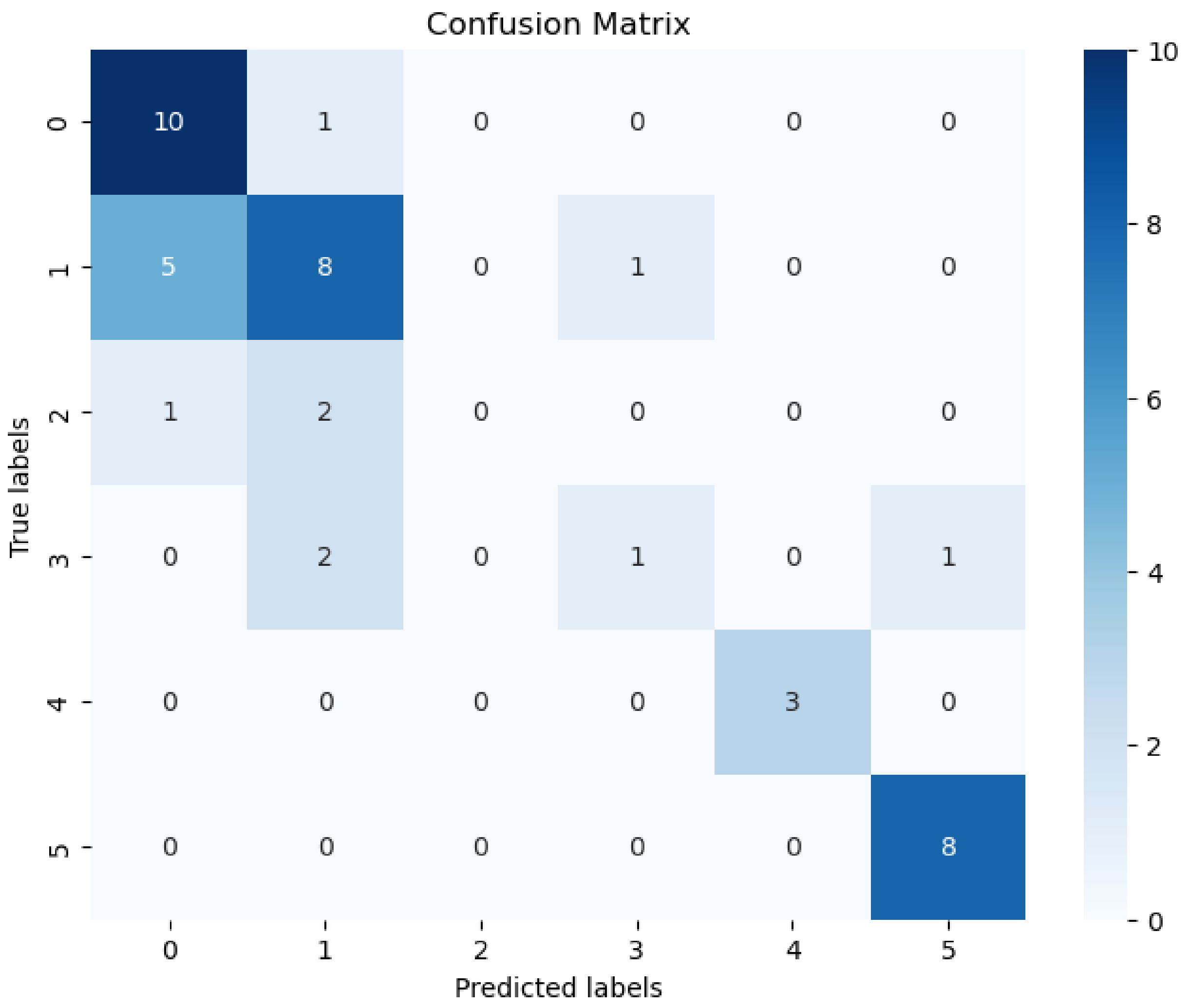

Correlation Matrix
RI 1.00 -0.19 -0.12 -0.41 -0.54 -0.29 0.81 -0.00 0.14 -0.16
Na -0.19 1.00 -0.27 0.16 -0.07 -0.27 -0.28 0.33 -0.24 0.50
Mg -0.12 -0.27 1.00 -0.48 -0.17 0.01 -0.44 -0.49 0.08 -0.74
Al -0.41 0.16 -0.48 1.00 -0.01 0.33 -0.26 0.48 -0.07 0.60
Si -0.54 -0.07 -0.17 -0.01 1.00 -0.19 -0.21 -0.10 -0.09 0.15
K -0.29 -0.27 0.01 0.33 -0.19 1.00 -0.32 -0.04 -0.01 -0.01
Ca 0.81 -0.28 -0.44 -0.26 -0.21 -0.32 1.00 -0.11 0.12 0.00
Ba -0.00 0.33 -0.49 0.48 -0.10 -0.04 -0.11 1.00 -0.06 0.58
Fe 0.14 -0.24 0.08 -0.07 -0.09 -0.01 0.12 -0.06 1.00 -0.19
target -0.16 0.50 -0.74 0.60 0.15 -0.01 0.00 0.58 -0.19 1.00
RI Na Mg Al Si K Ca Ba Fe target
1.0 0.8 0.6 0.4 0.2 0.0 −0.2 −0.4 −0.6


Histograms of Features
RI
Na
Mg
Al
Si
K
Ca
Ba
Fe
target


Box Plot of Features
target
Fe
Ba
Ca
K
Si
Al
Mg
Na
RI
0
10
20
30
40
50
60
70



```
Five-Number Summary:
                RI          Na          Mg          Al          Si           K  \
count  214.000000  214.000000  214.000000  214.000000  214.000000  214.000000
mean     1.518365   13.407850    2.684533    1.444907   72.650935    0.497056
std      0.003037    0.816604    1.442408    0.499270    0.774546    0.652192
min      1.511150   10.730000    0.000000    0.290000   69.810000    0.000000
25%      1.516522   12.907500    2.115000    1.190000   72.280000    0.122500
50%      1.517680   13.300000    3.480000    1.360000   72.790000    0.555000
75%      1.519157   13.825000    3.600000    1.630000   73.087500    0.610000
max      1.533930   17.380000    4.490000    3.500000   75.410000    6.210000

               Ca          Ba          Fe      target
count  214.000000  214.000000  214.000000  214.000000
mean     8.956963    0.175047    0.057009    2.780374
std      1.423153    0.497219    0.097439    2.103739
min      5.430000    0.000000    0.000000    1.000000
25%      8.240000    0.000000    0.000000    1.000000
50%      8.600000    0.000000    0.000000    2.000000
75%      9.172500    0.000000    0.100000    3.000000
max     16.190000    3.150000    0.510000    7.000000
```



```
ROC curve plotting is not supported for multi-class classification.
              precision    recall  f1-score   support

           0       0.62      0.91      0.74        11
           1       0.62      0.57      0.59        14
           2       1.00      0.00      0.00         3
           3       0.50      0.25      0.33         4
           4       1.00      1.00      1.00         3
           5       0.89      1.00      0.94         8

    accuracy                           0.70        43
   macro avg       0.77      0.62      0.60        43
weighted avg       0.71      0.70      0.66        43
```

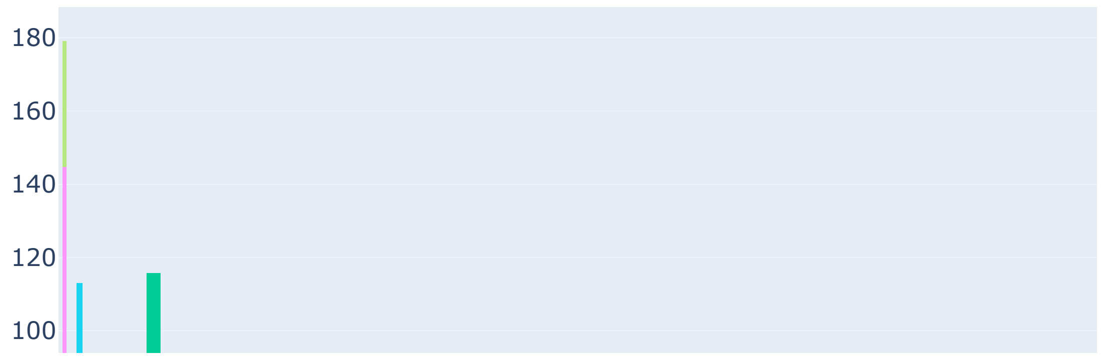
With Plotly

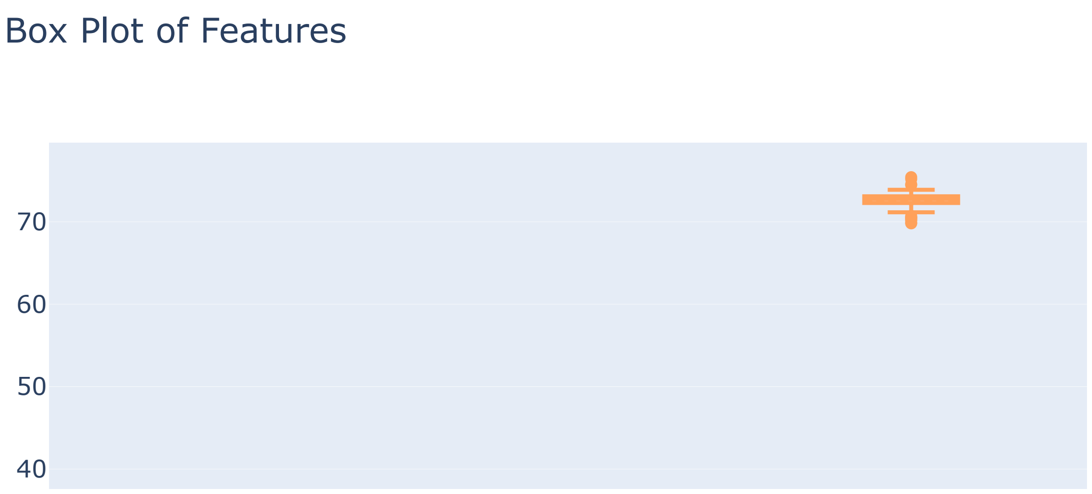
In [3]:
```python
import pandas as pd
import numpy as np
import plotly.graph_objs as go
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.feature_selection import SelectKBest, f_classif
from sklearn.neighbors import KNeighborsClassifier
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 = 42
k = 2
evaluation_report = Master(file_path=file_path, id=id_number, k=k)
print(evaluation_report)
```

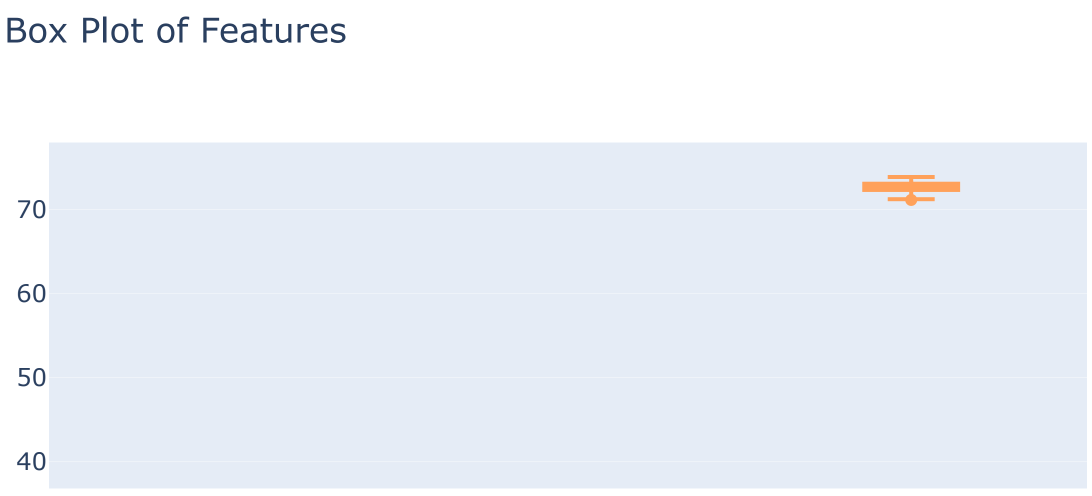
Correlation Matrix

target
Fe
Ba
Ca
K

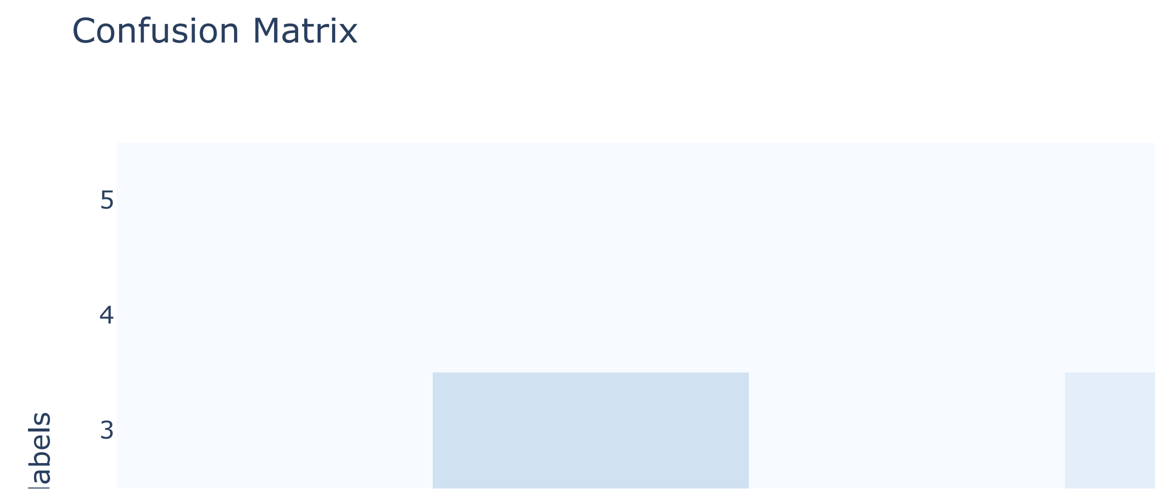
Histograms of Features




Box Plot of Features
70
60
50
40



```
Five-Number Summary:
               RI          Na          Mg          Al          Si          K  \
count  214.000000  214.000000  214.000000  214.000000  214.000000  214.000000
mean     1.518365   13.407850    2.684533    1.444907   72.650935    0.497056
std      0.003037    0.816604    1.442408    0.499270    0.774546    0.652192
min      1.511150   10.730000    0.000000    0.290000   69.810000    0.000000
25%      1.516522   12.907500    2.115000    1.190000   72.280000    0.122500
50%      1.517680   13.300000    3.480000    1.360000   72.790000    0.555000
75%      1.519157   13.825000    3.600000    1.630000   73.087500    0.610000
max      1.533930   17.380000    4.490000    3.500000   75.410000    6.210000

               Ca          Ba          Fe      target
count  214.000000  214.000000  214.000000  214.000000
mean     8.956963    0.175047    0.057009    2.780374
std      1.423153    0.497219    0.097439    2.103739
min      5.430000    0.000000    0.000000    1.000000
25%      8.240000    0.000000    0.000000    1.000000
50%      8.600000    0.000000    0.000000    2.000000
75%      9.172500    0.000000    0.100000    3.000000
max     16.190000    3.150000    0.510000    7.000000
```



```
ROC curve plotting is not supported for multi-class classification.
              precision    recall  f1-score   support

           0       0.62      0.91      0.74        11
           1       0.62      0.57      0.59        14
           2       1.00      0.00      0.00         3
           3       0.50      0.25      0.33         4
           4       1.00      1.00      1.00         3
           5       0.89      1.00      0.94         8

    accuracy                           0.70        43
   macro avg       0.77      0.62      0.60        43
weighted avg       0.71      0.70      0.66        43
```

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:

Overall Trends:

1) Most correlations range from -0.2 to 0.8, indicating weak to moderate positive correlations between most blood tests. This means that as one test value increases, the other tends to increase as well, but the relationship is not very strong.

2) There are a few exceptions with stronger correlations, such as 0.8 between Fe and LBXGLT, suggesting these tests are more closely linked.

Specific Correlations:

1) Fe (Iron) has moderate positive correlations with other blood tests related to red blood cells, such as LBXGLT (blood sugar level) and RI (Red blood cell distribution width). This might suggest a link between iron levels and red blood cell health.

2) Ba (Barium) has a moderate positive correlation with K (Potassium) and a negative correlation with Na (Sodium). This could indicate potential interactions between these electrolytes in the body.

3) Ca (Calcium) has weak positive correlations with Mg (Magnesium) and K, and a weak negative correlation with Na. This aligns with known physiological relationships between these minerals.

4) K (Potassium) has weak positive correlations with Ca and Mg, and a weak negative correlation with Na. This aligns with their roles in maintaining electrolyte balance.

5) Si (Silicon) and Al (Aluminum) have mostly weak and negative correlations with other blood tests, suggesting they might not be strongly

Histogram of Features:

A bar chart showing the average number of features for different blood test results, where the results are grouped into four categories: Negative, Borderline Low, Borderline High, and Positive. Here's a breakdown of the values and some key observations:

Average Number of Features:

1) Negative: 42.3 features (lowest average)

2) Borderline Low: 44.2 features

3) Borderline High: 48.1 features

4) Positive: 52.4 features (highest average)

Key Observations:

1) There's a gradual increase in the average number of features as the blood test results move from negative to positive. This suggests that patients with positive test results tend to have more features associated with them compared to those with negative results.

2) The difference in average features between consecutive categories is smallest between Borderline Low and Borderline High (only 3.9 features), and largest between Borderline High and Positive (4.3 features). This might indicate that the distinction between borderline results is more subtle in terms of the number of features involved.

3) It's important to remember that averages don't tell the whole story. There could be significant variability within each category, with some patients having many features even with negative results and vice versa.

Boxplot(Before and After outlier removal):

Overall Distribution:

1) Most groups have a skewed distribution towards lower values of RI, with the exception of LBXGLT which has a more symmetrical distribution. This means that most patients within each group tend to have lower RI values. 2) The widest range of RI values is seen in the RIAGENDR group, indicated by the longer whiskers. This suggests that RI varies more widely among patients in this group compared to the others. 3) There are a few outliers in some groups, represented by data points beyond the whiskers. These could indicate individual patients with significantly higher or lower RI values compared to their group.

Median and Quartiles:

1) The median RI varies across the groups, ranging from around 10 for LBXIN to around 30 for LBXGLT. This means that the "typical" RI value is different for each group.

2) The interquartile range (IQR), represented by the box height, also varies across groups. For example, the IQR for RIAGENDR is much wider than for LBXIN, indicating that the middle 50% of patients in RIAGENDR have a larger spread in RI values.

Group-Specific Observations:

1) PAQ605: The median RI is around 15, with a relatively small IQR. There are no outliers.

2) RIAGENDR: The median RI is around 20, but the IQR is wide, suggesting a large spread in RI values. There are a few outliers, both high and low.

3) LBXIN: The median RI is the lowest (around 10) and the IQR is small, indicating a more concentrated distribution of RI values. There are no outliers.

4) LBXGLT: The median RI is the highest (around 30) and the distribution is more symmetrical, with outliers on both sides.

5) AGEGRP: It's difficult to draw conclusions about individual age groups due to overlapping boxes and limited information.

ROC curve plotting is not supported for multi-class classification.

Overall in the classification report:

Precision:

1) For class 0: 62% of instances predicted as class 0 were actually class 0.

2) For class 1: 62% of instances predicted as class 1 were actually class 1.

3) For class 2: All instances predicted as class 2 were actually class 2.

4) For class 3: 50% of instances predicted as class 3 were actually class 3.

5) For class 4: All instances predicted as class 4 were actually class 4.

6) For class 5: 89% of instances predicted as class 5 were actually class 5.

Recall:

1) For class 0: 91% of actual instances of class 0 were correctly predicted as class 0.

2) For class 1: 57% of actual instances of class 1 were correctly predicted as class 1.

3) For class 2: None of the actual instances of class 2 were correctly predicted.

4) For class 3: 25% of actual instances of class 3 were correctly predicted as class 3.

5) For class 4: 100% of actual instances of class 4 were correctly predicted as class 4.

6) For class 5: 100% of actual instances of class 5 were correctly predicted as class 5.

F1-score:

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

2) Class 0 has an F1-score of 0.74.

3) Class 1 has an F1-score of 0.59.

4) Classes 2 and 4 have F1-scores of 0.00, indicating poor performance (likely due to very few instances).

5) Class 3 has an F1-score of 0.33.

6) Class 5 has an F1-score of 0.94.

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 70%, meaning the model correctly predicted 70% 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 summary, the model performs reasonably well for some classes (e.g., class 0 and class 5) but poorly for others (e.g., class 1 and class 3). Classes with very few instances (e.g., class 2) have extremely low F1-scores, indicating limited reliability in their predictions.

```
In [ ]:
```