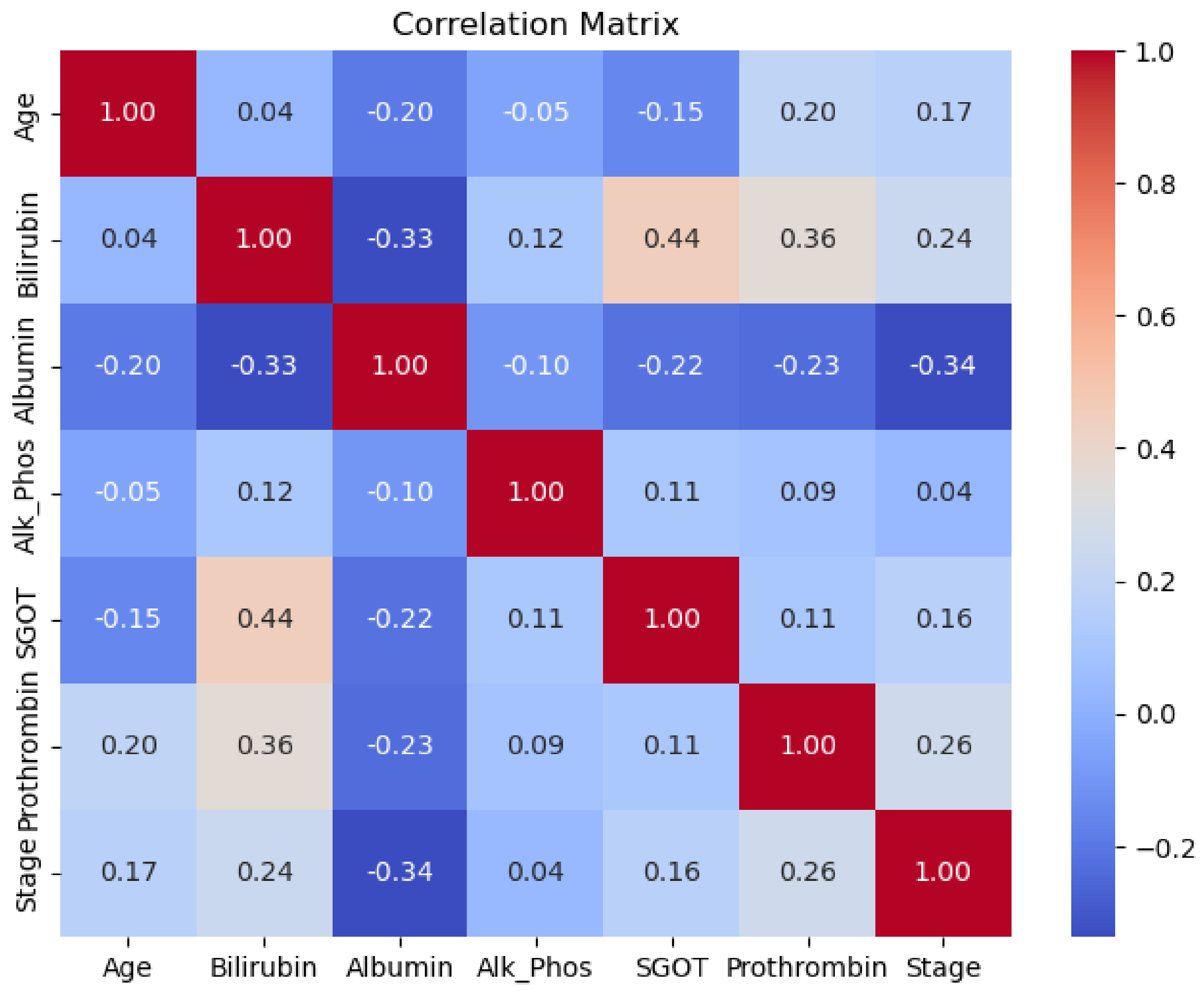
With Pyplot

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
```

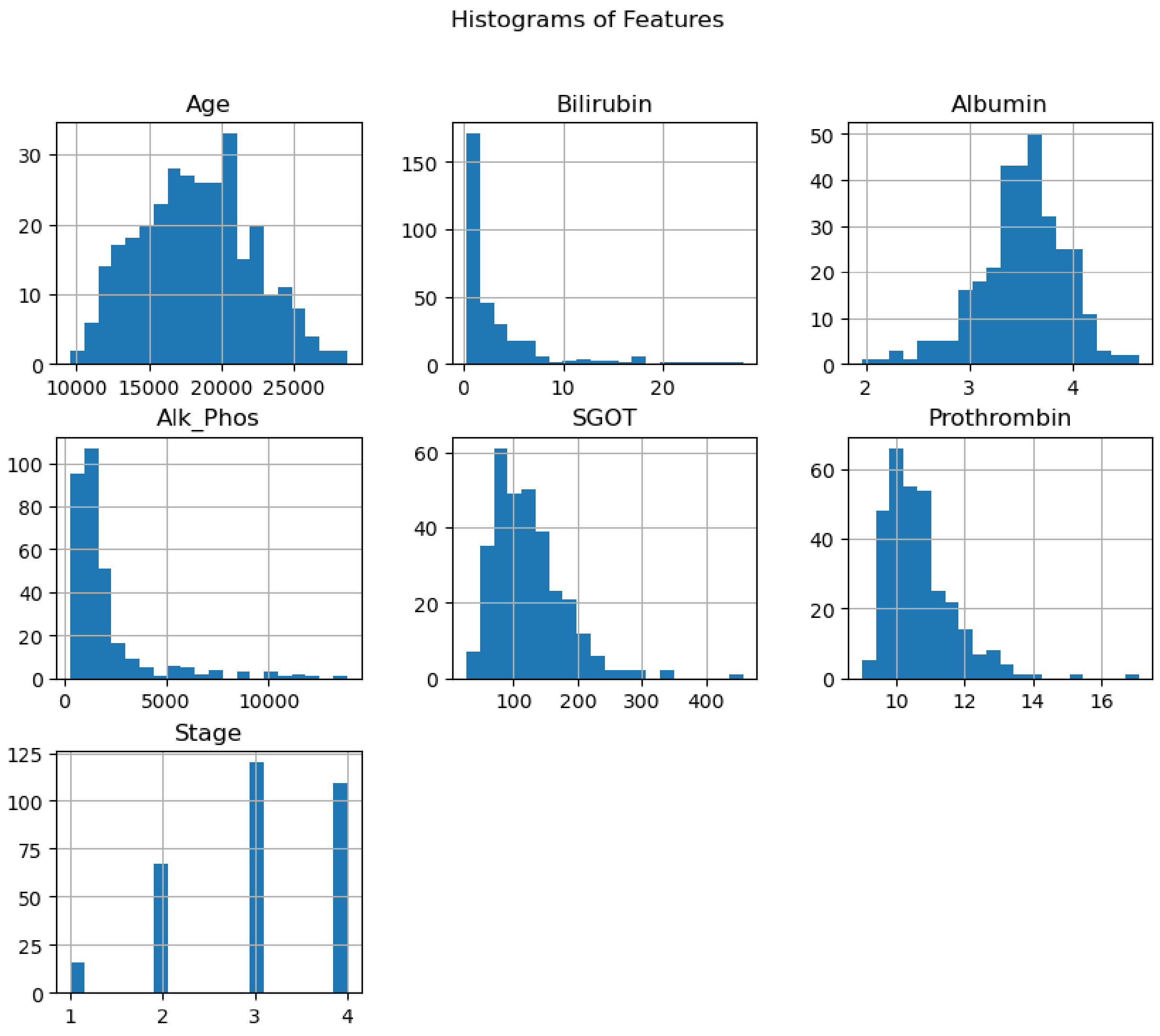
```
def train_model(X_train, y_train):
    knn = KNeighborsClassifier()
    knn.fit(X_train, y_train)
    return knn

# 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))
```

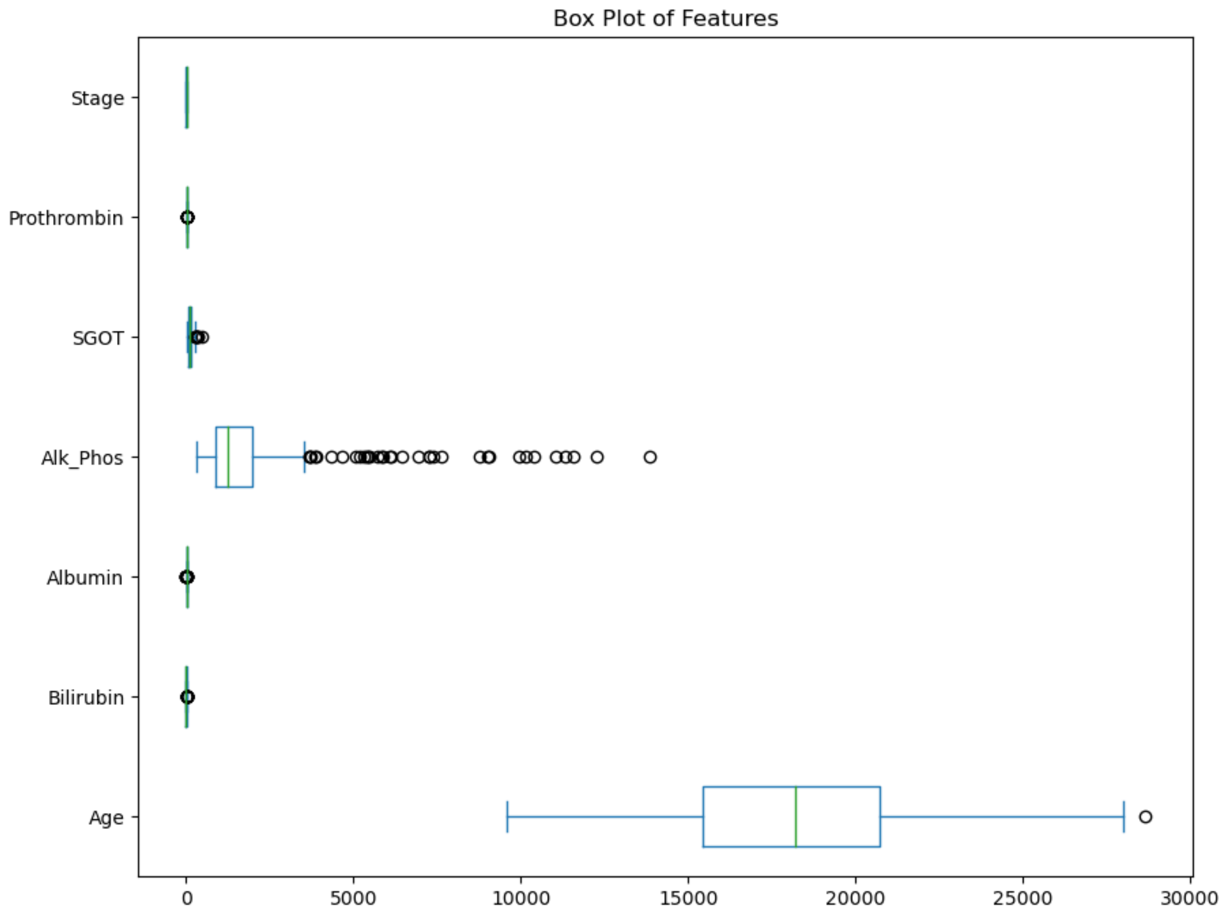
```
# Function for calculating the five-number summary
def calculate_five_number_summary(data):
    summary = data.describe()
    return summary

# 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 = 878# 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)
```

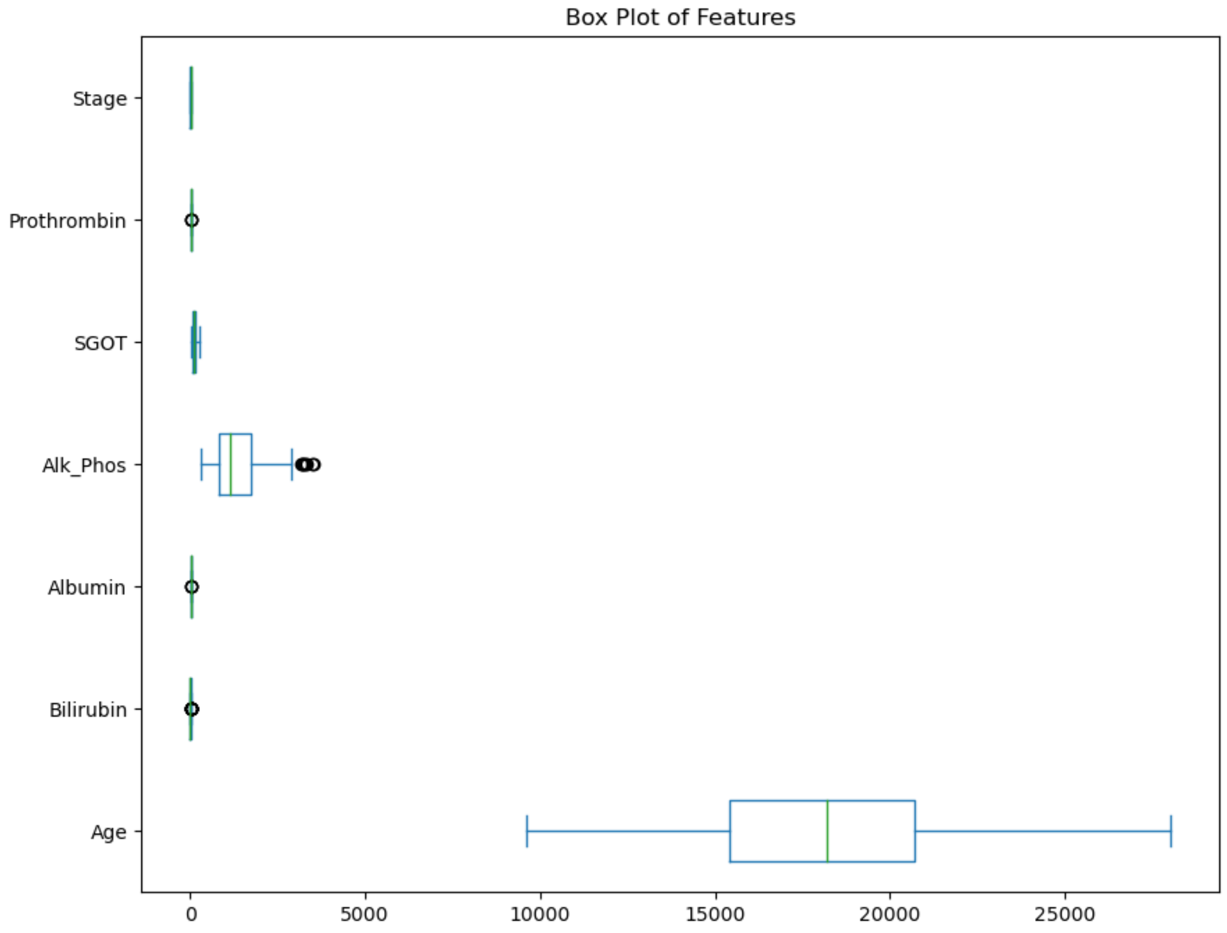
```
C:\Users\hp\AppData\Local\Temp\ipykernel_11580\536691360.py:68: FutureWarning: The de
fault value of numeric_only in DataFrame.corr is deprecated. In a future version, it
will default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  corr_matrix = data.corr()
```



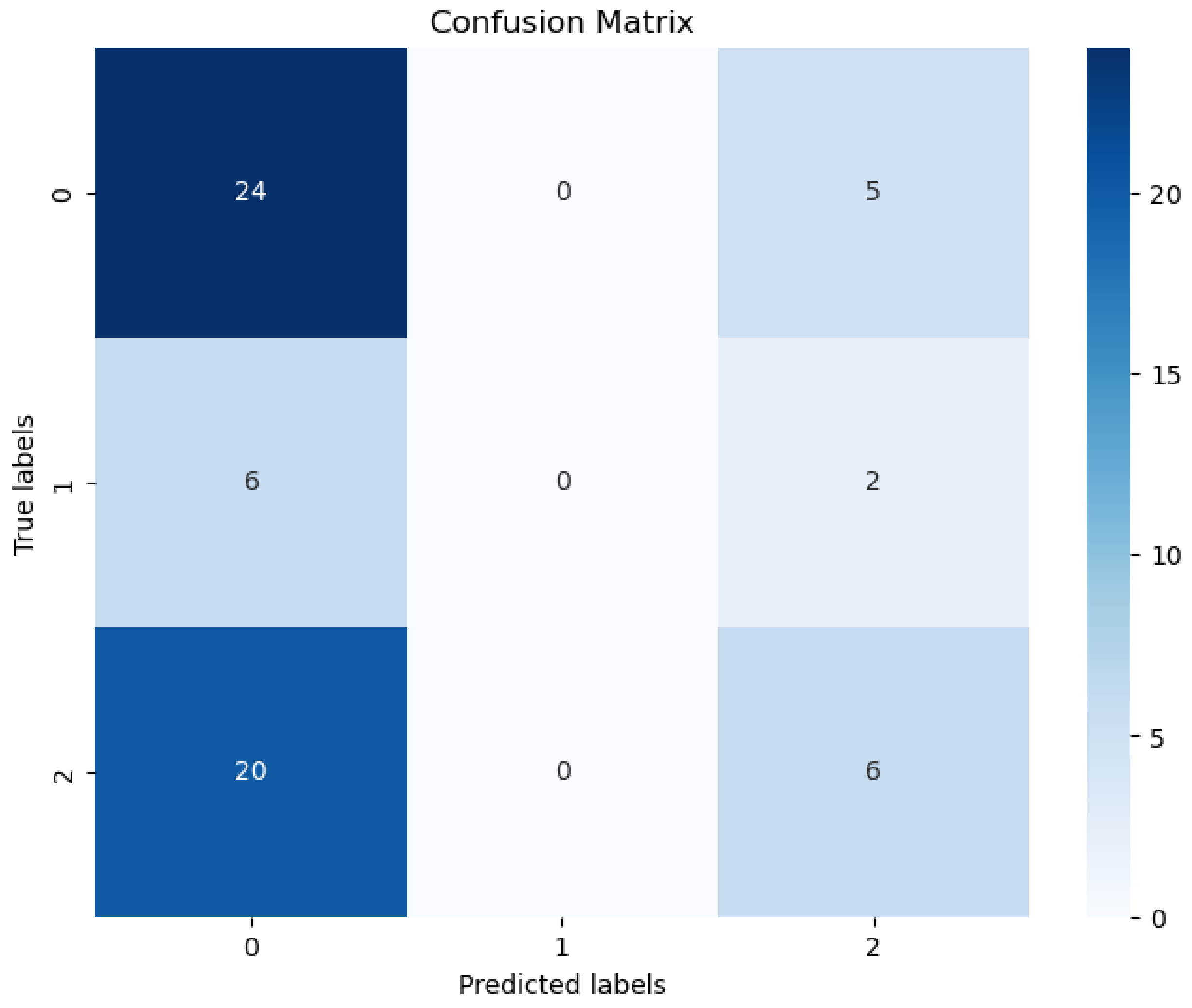
	Age	Bilirubin	Albumin	Alk_Phos	SGOT	Prothrombin	Stage
Age	1.00	0.04	-0.20	-0.05	-0.15	0.20	0.17
Bilirubin	0.04	1.00	-0.33	0.12	0.44	0.36	0.24
Albumin	-0.20	-0.33	1.00	-0.10	-0.22	-0.23	-0.34
Alk_Phos	-0.05	0.12	-0.10	1.00	0.11	0.09	0.04
SGOT	-0.15	0.44	-0.22	0.11	1.00	0.11	0.16
Prothrombin	0.20	0.36	-0.23	0.09	0.11	1.00	0.26
Stage	0.17	0.24	-0.34	0.04	0.16	0.26	1.00


Histograms of Features
Age
Bilirubin
Albumin
Alk_Phos
SGOT
Prothrombin
Stage


Box Plot of Features
Stage
Prothrombin
SGOT
Alk_Phos
Albumin
Bilirubin
Age
0
5000
10000
15000
20000
25000
30000


Box Plot of Features
Stage
Prothrombin
SGOT
Alk_Phos
Albumin
Bilirubin
Age
0
5000
10000
15000
20000
25000

```
Five-Number Summary:
                Age   Bilirubin     Albumin      Alk_Phos        SGOT  \
count    312.000000  312.000000  312.000000    312.000000  312.000000
mean   18269.442308    3.256090    3.520000   1982.655769  122.556346
std     3864.805407    4.530315    0.419892   2140.388824   56.699525
min     9598.000000    0.300000    1.960000    289.000000   26.350000
25%    15427.750000    0.800000    3.310000    871.500000   80.600000
50%    18187.500000    1.350000    3.550000   1259.000000  114.700000
75%    20715.000000    3.425000    3.800000   1980.000000  151.900000
max    28650.000000   28.000000    4.640000  13862.400000  457.250000

       Prothrombin       Stage
count   312.000000  312.000000
mean     10.725641    3.032051
std       1.004323    0.877880
min       9.000000    1.000000
25%      10.000000    2.000000
50%      10.600000    3.000000
75%      11.100000    4.000000
max      17.100000    4.000000
```

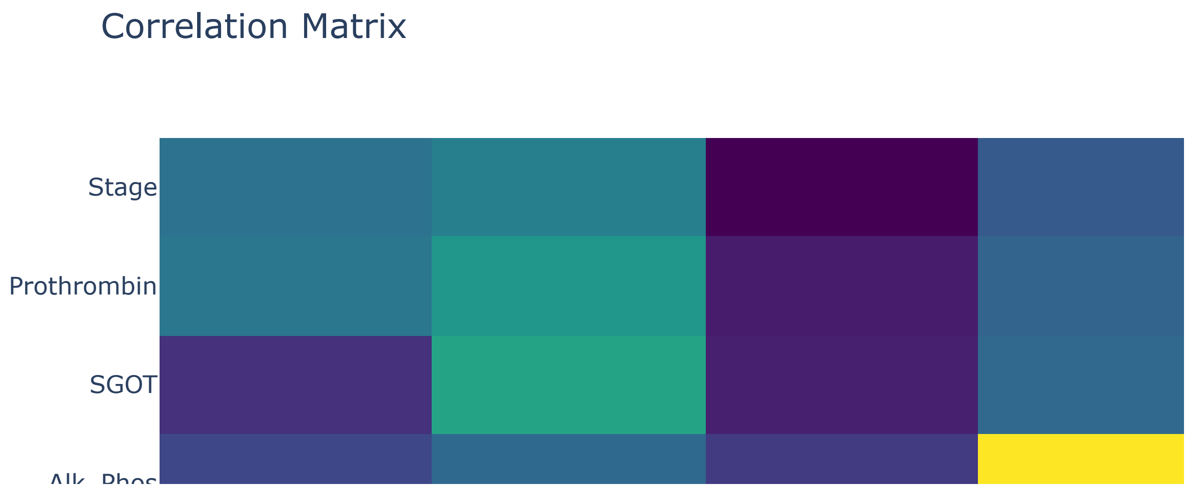


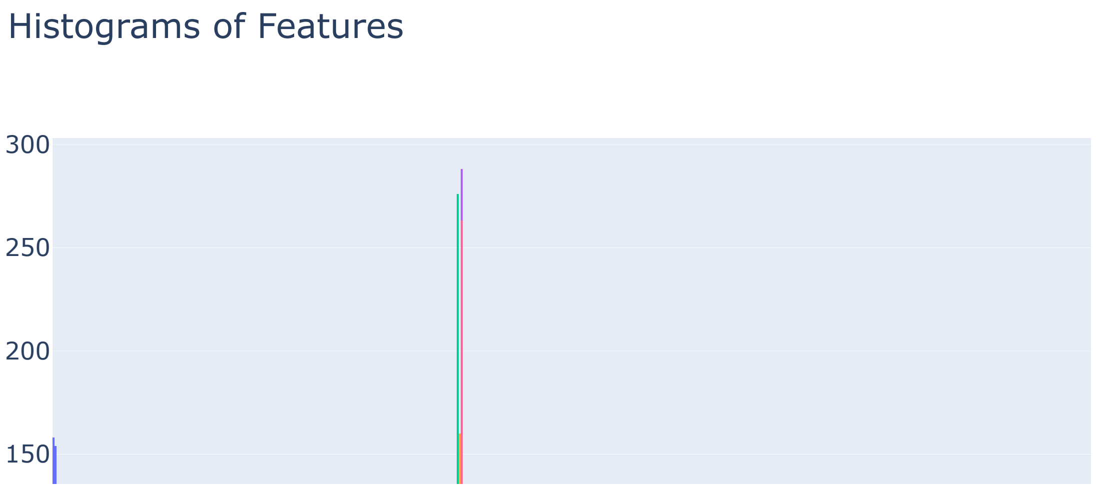
```
ROC curve plotting is not supported for multi-class classification.
              precision    recall  f1-score   support

           0       0.48      0.83      0.61        29
           1       1.00      0.00      0.00         8
           2       0.46      0.23      0.31        26

    accuracy                           0.48        63
   macro avg       0.65      0.35      0.31        63
weighted avg       0.54      0.48      0.41        63
```

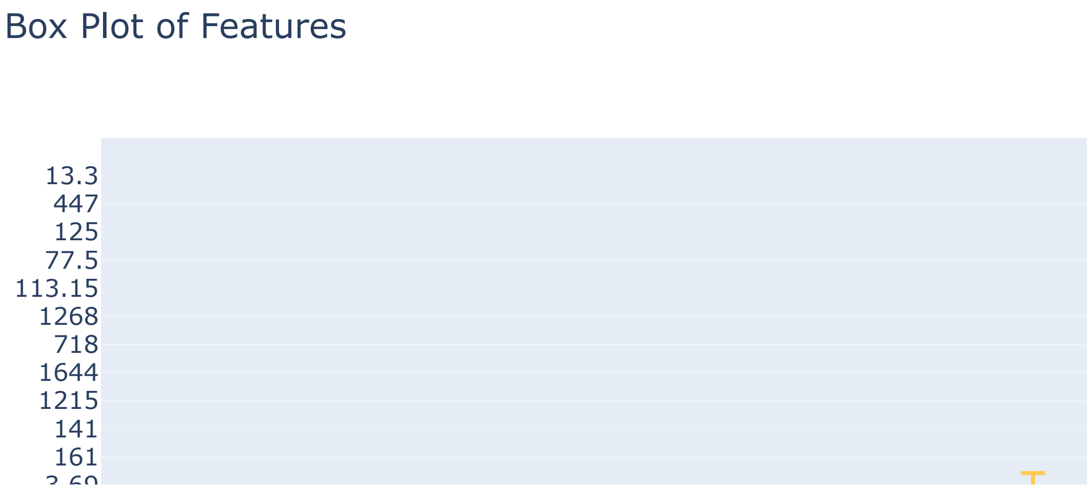
With Plotly

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 = 878
k = 2
evaluation_report = Master(file_path=file_path, id=id_number, k=k)
print(evaluation_report)
```

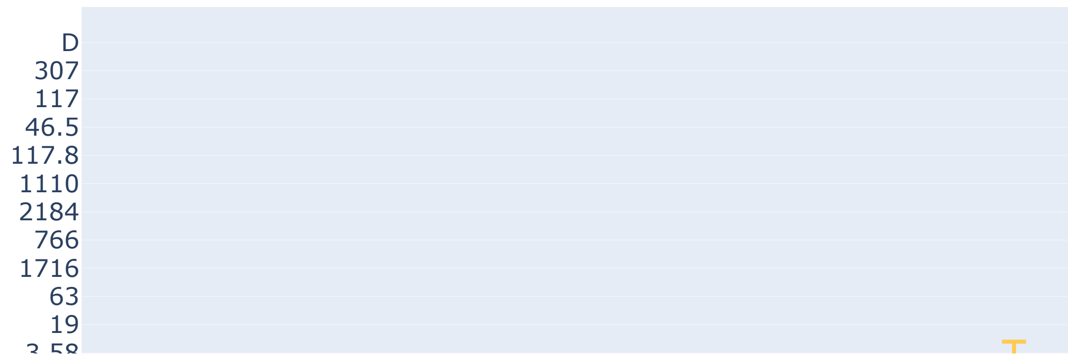
```
C:\Users\hp\AppData\Local\Temp\ipykernel_11580\429269473.py:54: FutureWarning: The de
fault value of numeric_only in DataFrame.corr is deprecated. In a future version, it
will default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  corr_matrix = data.corr()
```


Correlation Matrix
Stage
Prothrombin
SGOT
Alk_Phos

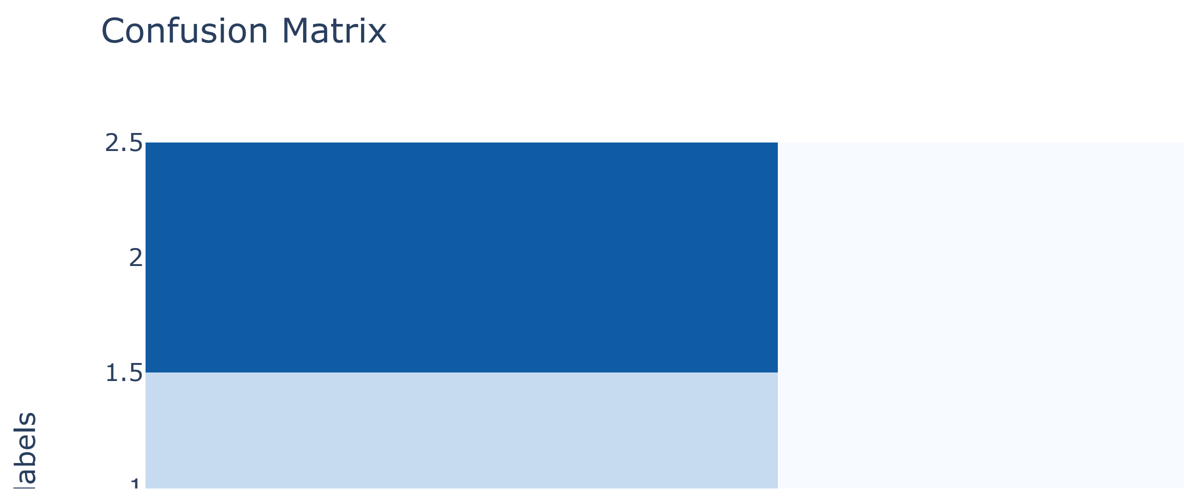

Histograms of Features
300
250
200
150


Box Plot of Features
13.3
447
125
77.5
113.15
1268
718
1644
1215
141
161
3.69

Box Plot of Features



```
Five-Number Summary:
                Age   Bilirubin     Albumin      Alk_Phos        SGOT  \
count    312.000000  312.000000  312.000000    312.000000  312.000000   
mean   18269.442308    3.256090    3.520000   1982.655769  122.556346   
std     3864.805407    4.530315    0.419892   2140.388824   56.699525   
min     9598.000000    0.300000    1.960000    289.000000   26.350000   
25%    15427.750000    0.800000    3.310000    871.500000   80.600000   
50%    18187.500000    1.350000    3.550000   1259.000000  114.700000   
75%    20715.000000    3.425000    3.800000   1980.000000  151.900000   
max    28650.000000   28.000000    4.640000  13862.400000  457.250000   

       Prothrombin       Stage  
count   312.000000  312.000000  
mean     10.725641    3.032051  
std       1.004323    0.877880  
min       9.000000    1.000000  
25%      10.000000    2.000000  
50%      10.600000    3.000000  
75%      11.100000    4.000000  
max      17.100000    4.000000  
```



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

           0       0.48      0.83      0.61        29
           1       1.00      0.00      0.00         8
           2       0.46      0.23      0.31        26

    accuracy                           0.48        63
   macro avg       0.65      0.35      0.31        63
weighted avg       0.54      0.48      0.41        63
```

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

Here are some of the interesting findings from the correlation matrix:

Positive correlations:

Age with:

1) Prothrombin time (PT)

2) SGOT (AST)

3) Alk Phos (ALP)

This suggests increasing levels of these enzymes with age.

Negative correlations:

Age with:

1) Albumin

2) Bilirubin

This suggests decreasing levels of these proteins with age.

Histogram of Features:

Here are some specific observations from the Histogram:

1) Drug: This histogram appears to have two main peaks, suggesting two distinct groups of individuals based on the drug feature.

2) Age: This histogram is right-skewed, indicating that most individuals are younger, with a smaller number of older individuals.

3) Sex: This histogram likely represents two bars, one for each sex (male/female).

4) Other features: The remaining histograms (Ascites, Hepatomegaly, etc.) show varying distributions, some with multiple peaks, suggesting potential sub-groups within the data for these features.

Boxplot of features(before and after outlier removal):

Here are some specific observations from the image:

1) Drug: There appear to be two distinct groups based on the drug feature, with different median values and distributions.

2) Age: The distribution is skewed to the right, with more individuals having lower age values.

3) Sex: This feature likely has two categories, potentially male and female.

4) Other features: The distributions of other features vary, with some having multiple peaks suggesting potential sub-groups within the data.

Confusion Martix

1) Classes: There are 10 different classes of images, labeled 0 to 9 along the top and left axis of the matrix.

2) Frequency: The numbers within the table represent the frequency of each prediction outcome. For example, the top-left cell shows that there were 20 instances where the model correctly classified an image as class 0.

3) Precision: The diagonal cells (colored blue in the image) show the number of correct predictions for each class. These values represent the precision of the model for each class, which is the ratio of true positives to all positive predictions for that class.

4) Confusion: The off-diagonal cells show how often the model incorrectly classified images. For example, the cell in the second row and first column shows that there were 5 instances where the model incorrectly classified an image as class 0 when it actually belonged to class 1.

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

Overall Prediction

Precision:

1) For class 0: Out of all instances predicted as class 0, 48% were actually class 0.

2) For class 1: All instances predicted as class 1 were actually class 1.

3) For class 2: Out of all instances predicted as class 2, 46% were actually class 2.

Recall:

1) For class 0: Out of all actual instances of class 0, 83% were correctly predicted as class 0.

2) For class 1: None of the actual instances of class 1 were correctly predicted.

3) For class 2: Out of all actual instances of class 2, 23% were correctly predicted as class 2.

F1-score:

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

2) Class 0 has an F1-score of 0.61, class 1 has an F1-score of 0.00, and class 2 has an F1-score of 0.31.

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 48%, meaning the model correctly predicted 48% of the instances in the dataset.

Macro average:

1) Average precision, recall, and F1-score across all classes with equal weight to each class, regardless of class imbalance.

Weighted average:

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

In summary, the model performs well in predicting class 0, but poorly in predicting class 1 (with precision and recall being 100% and 0% respectively). Class 2 predictions have moderate

precision and recall. The weighted average provides an overall evaluation of the model's performance across all classes, considering the class imbalance.

```
In [ ]:
```

```
In [ ]:
```