with pyplot

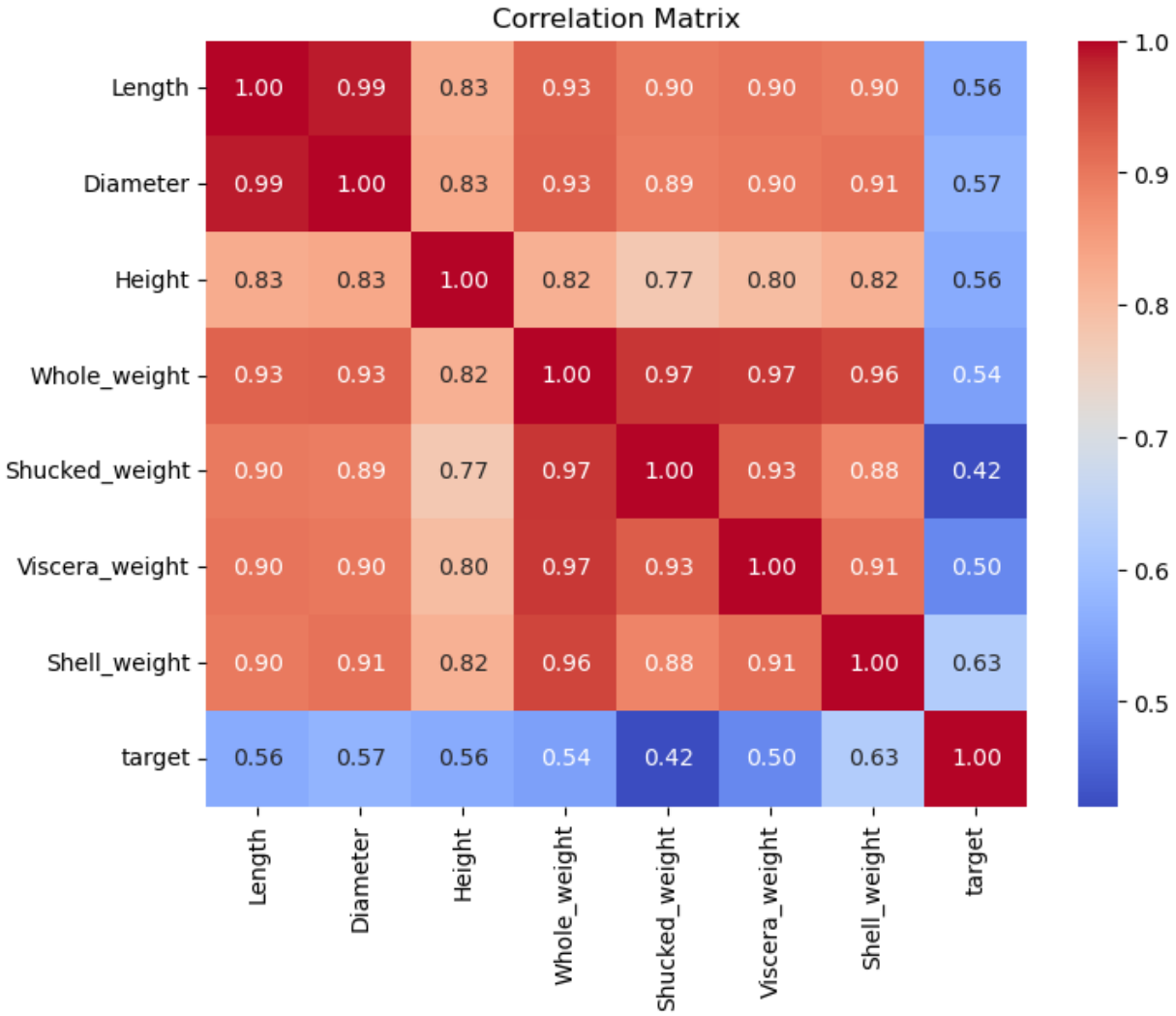
In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.feature_selection import SelectKBest, f_classif
from sklearn.neighbors import KNeighborsClassifier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

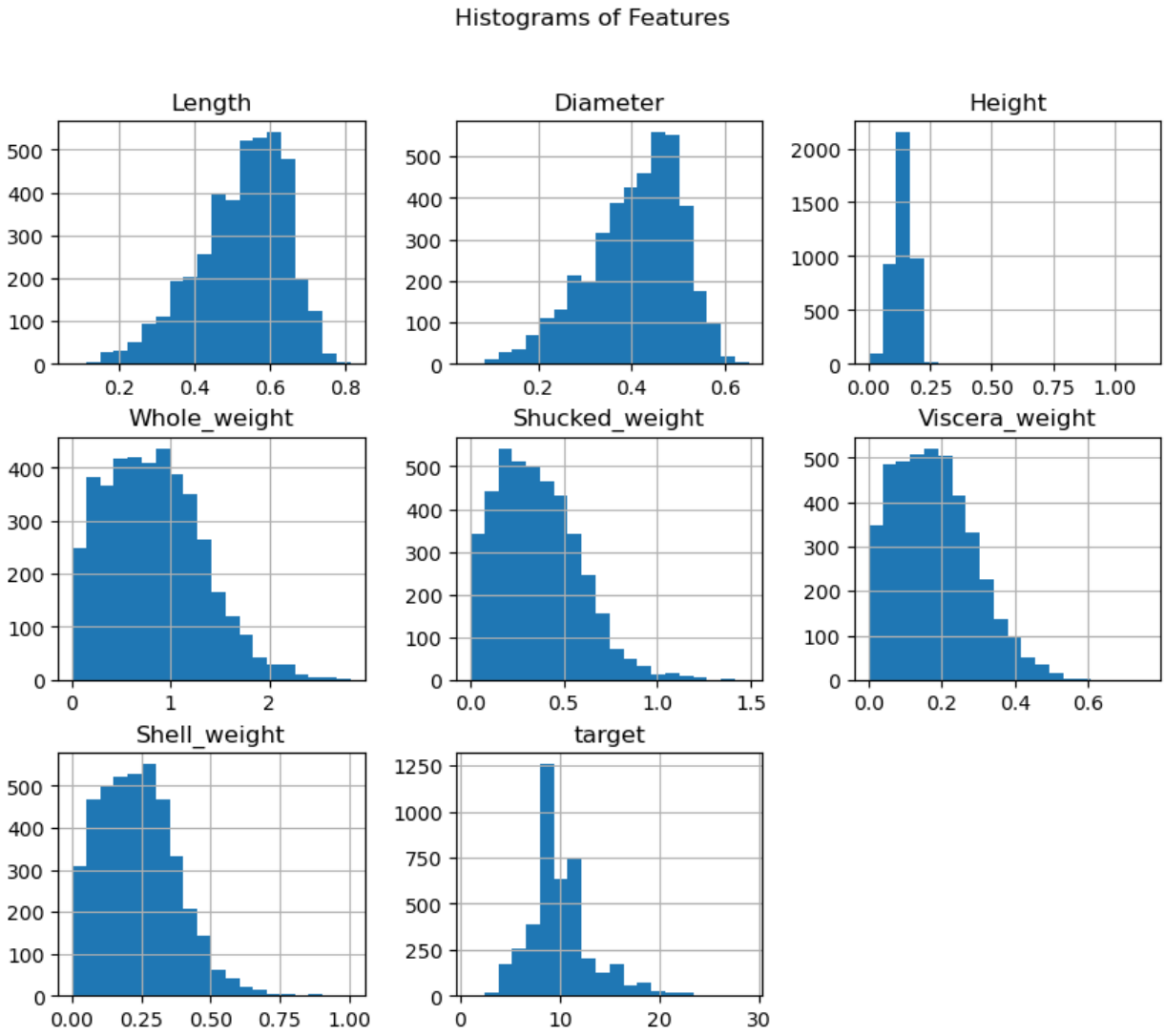
```
def train_model(X_train, y_train):
    knn = KNeighborsClassifier()
    knn.fit(X_train, y_train)
    return knn

# 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))
```

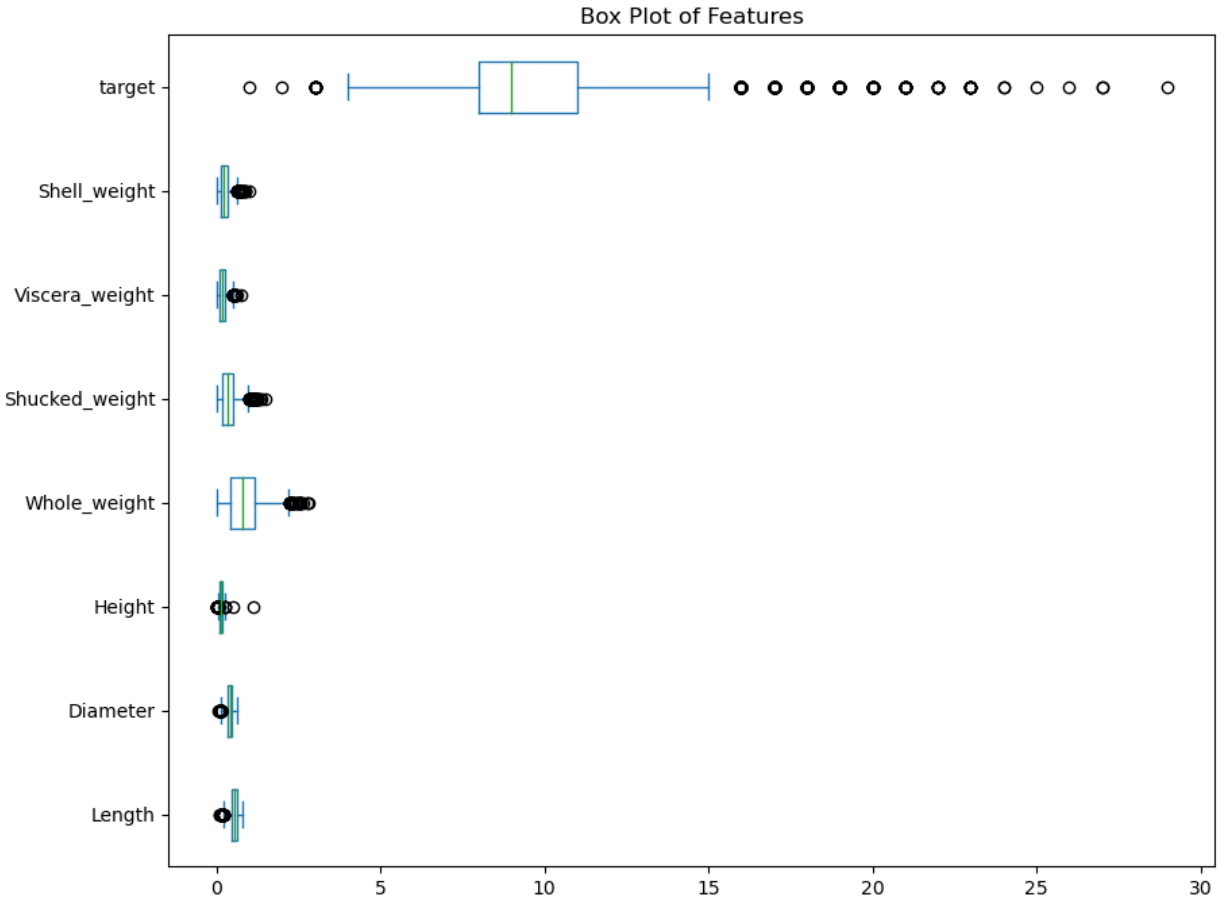
```
# Function for calculating the five-number summary
def calculate_five_number_summary(data):
    summary = data.describe()
    return summary

# 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)
```

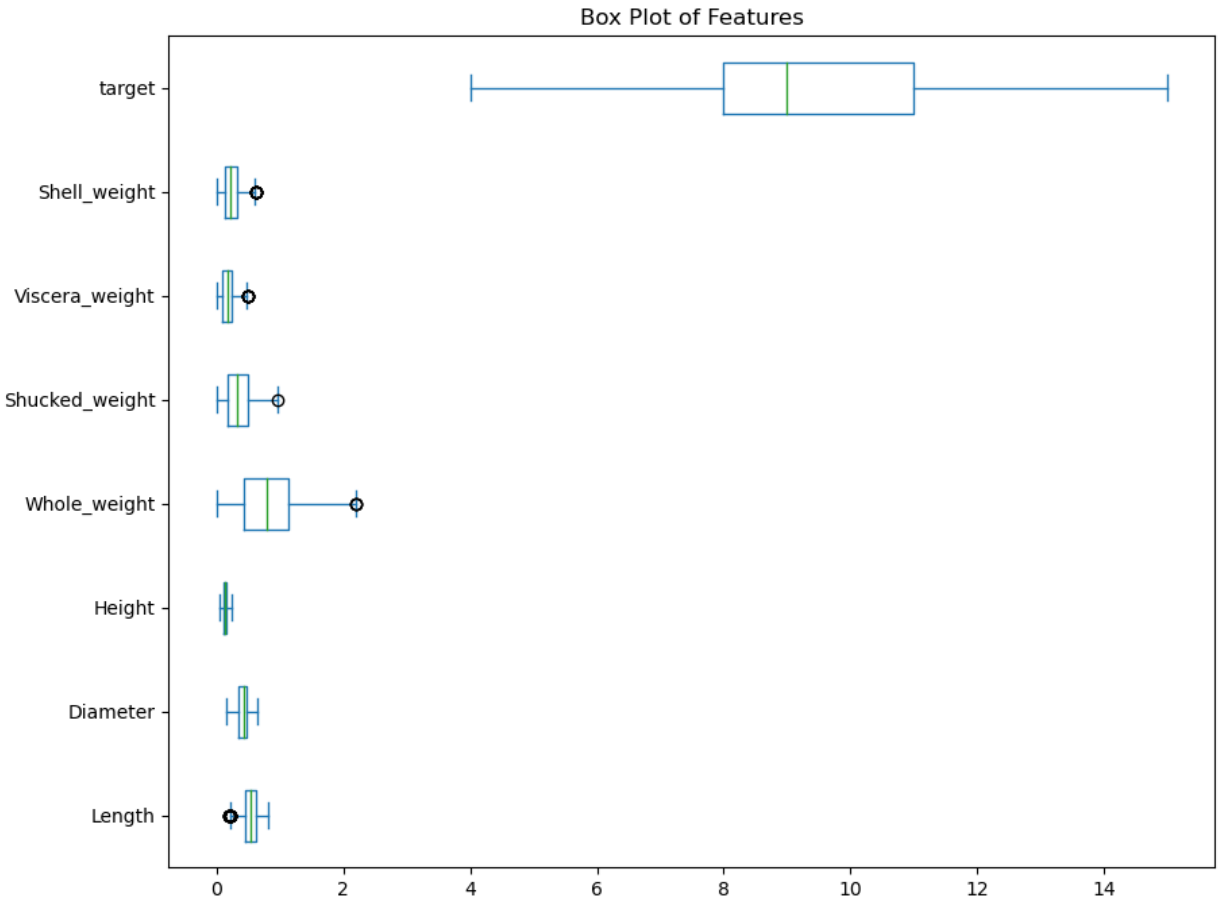
```
    # Model Evaluation
    evaluation_report = evaluate_model(model, X_test, y_test)

    # 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 = 1# Change this to the dataset ID if you have one
k = 2
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_4980\46941547.py:68: FutureWarning: The defa
ult value of numeric_only in DataFrame.corr is deprecated. In a future version, it wi
ll default to False. Select only valid columns or specify the value of numeric_only t
o silence this warning.
  corr_matrix = data.corr()
```


Correlation Matrix
Length 1.00 0.99 0.83 0.93 0.90 0.90 0.90 0.56
Diameter 0.99 1.00 0.83 0.93 0.89 0.90 0.91 0.57
Height 0.83 0.83 1.00 0.82 0.77 0.80 0.82 0.56
Whole_weight 0.93 0.93 0.82 1.00 0.97 0.97 0.96 0.54
Shucked_weight 0.90 0.89 0.77 0.97 1.00 0.93 0.88 0.42
Viscera_weight 0.90 0.90 0.80 0.97 0.93 1.00 0.91 0.50
Shell_weight 0.90 0.91 0.82 0.96 0.88 0.91 1.00 0.63
target 0.56 0.57 0.56 0.54 0.42 0.50 0.63 1.00
Length
Diameter
Height
Whole_weight
Shucked_weight
Viscera_weight
Shell_weight
target
1.0
0.9
0.8
0.7
0.6
0.5


Histograms of Features
Length
Diameter
Height
Whole_weight
Shucked_weight
Viscera_weight
Shell_weight
target

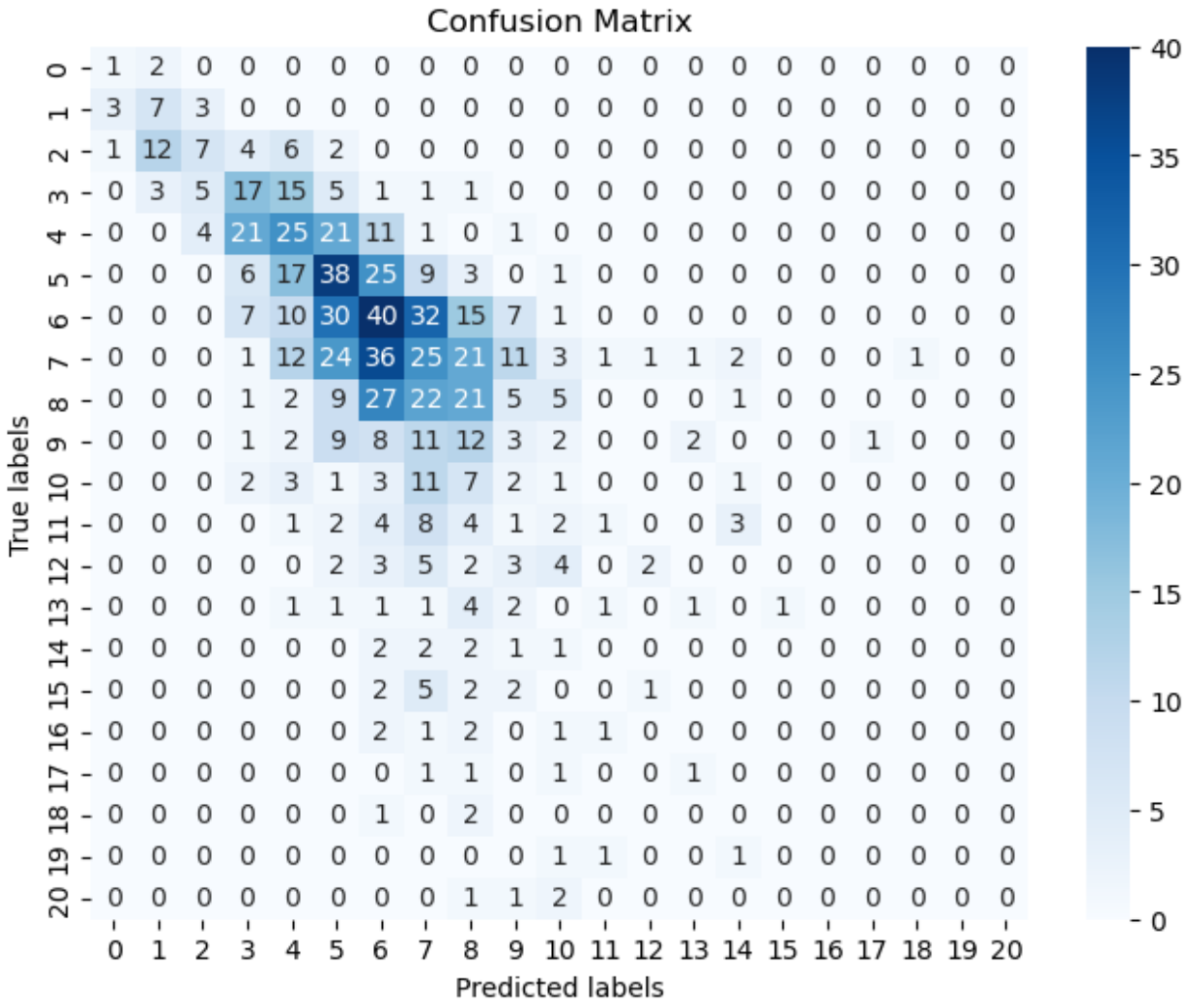

Box Plot of Features
target
Shell_weight
Viscera_weight
Shucked_weight
Whole_weight
Height
Diameter
Length
0
5
10
15
20
25
30


Box Plot of Features
target
Shell_weight
Viscera_weight
Shucked_weight
Whole_weight
Height
Diameter
Length
0
2
4
6
8
10
12
14

Five-Number Summary:

	Length	Diameter	Height	Whole_weight	Shucked_weight	\
count	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	
mean	0.523992	0.407881	0.139516	0.828742	0.359367	
std	0.120093	0.099240	0.041827	0.490389	0.221963	
min	0.075000	0.055000	0.000000	0.002000	0.001000	
25%	0.450000	0.350000	0.115000	0.441500	0.186000	
50%	0.545000	0.425000	0.140000	0.799500	0.336000	
75%	0.615000	0.480000	0.165000	1.153000	0.502000	
max	0.815000	0.650000	1.130000	2.825500	1.488000	

	Viscera_weight	Shell_weight	target
count	4177.000000	4177.000000	4177.000000
mean	0.180594	0.238831	9.933684
std	0.109614	0.139203	3.224169
min	0.000500	0.001500	1.000000
25%	0.093500	0.130000	8.000000
50%	0.171000	0.234000	9.000000
75%	0.253000	0.329000	11.000000
max	0.760000	1.005000	29.000000



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

           2       0.20      0.33      0.25         3
           3       0.29      0.54      0.38        13
           4       0.37      0.22      0.27        32
           5       0.28      0.35      0.31        48
           6       0.27      0.30      0.28        84
           7       0.26      0.38      0.31        99
           8       0.24      0.28      0.26       142
           9       0.19      0.18      0.18       139
          10       0.21      0.23      0.22        93
          11       0.08      0.06      0.07        51
          12       0.04      0.03      0.04        31
          13       0.20      0.04      0.06        26
          14       0.50      0.10      0.16        21
          15       0.20      0.08      0.11        13
          16       0.00      0.00      1.00         8
          17       0.00      0.00      1.00        12
          18       1.00      0.00      0.00         7
          19       0.00      0.00      1.00         4
          20       0.00      0.00      1.00         3
          21       1.00      0.00      0.00         3
          22       1.00      0.00      0.00         4

    accuracy                           0.23       836
   macro avg       0.30      0.15      0.33       836
weighted avg       0.23      0.23      0.24       836
```

With Plotly

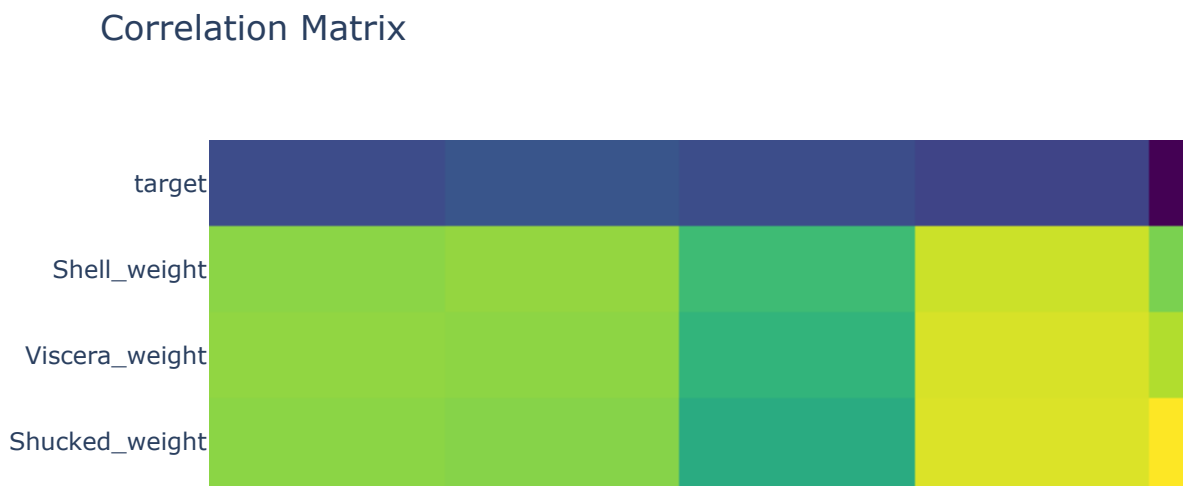
In [2]:

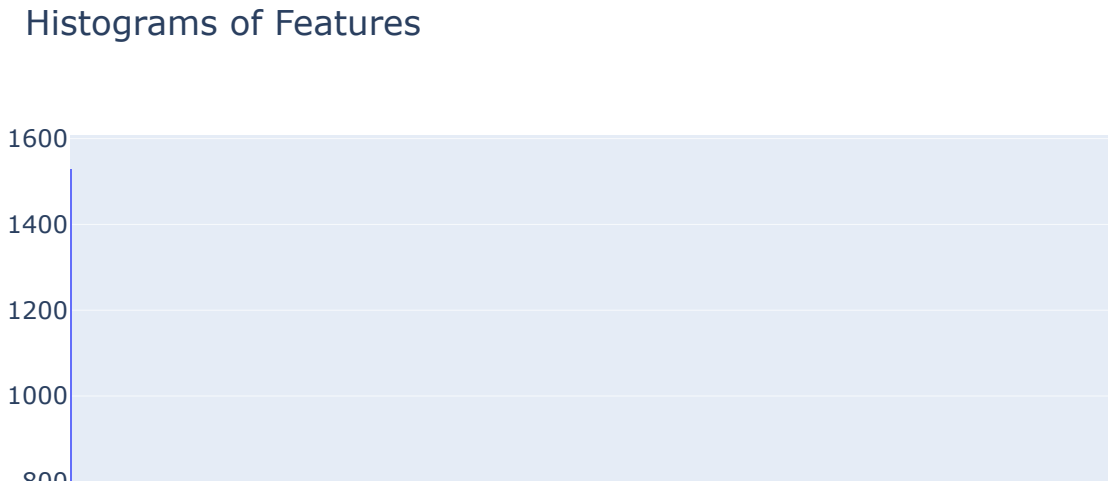
```
import pandas as pd
import numpy as np
import plotly.graph_objs as go
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.feature_selection import SelectKBest, f_classif
from sklearn.neighbors import KNeighborsClassifier
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
```

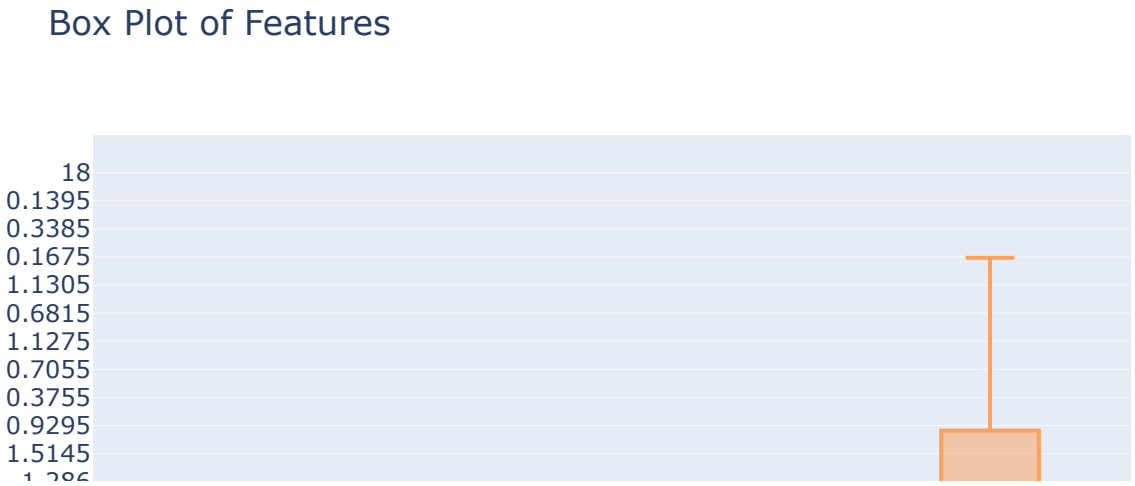
```
def transform_data(data):
    encoded_data = pd.get_dummies(data.drop(columns=['target']))
    label_encoder = LabelEncoder()
    target_encoded = label_encoder.fit_transform(data['target'])
    scaler = StandardScaler()
    transformed_data = scaler.fit_transform(encoded_data)
    return transformed_data,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

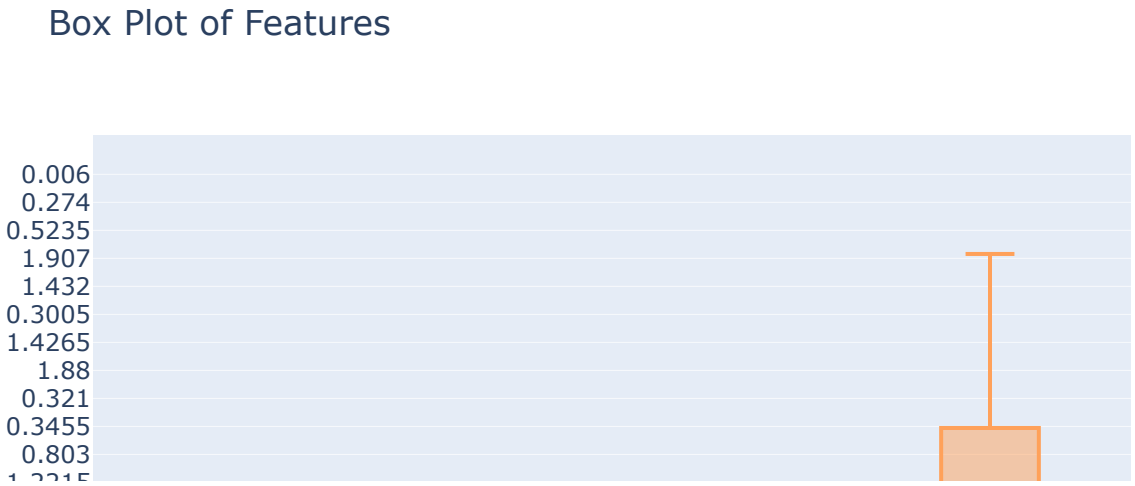
```
    return cleaned_data

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 = 1
k = 2
evaluation_report = Master(file_path=file_path, id=id_number, k=k)
print(evaluation_report)
```

```
C:\Users\hp\AppData\Local\Temp\ipykernel_4980\3104351446.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()
```




Histograms of Features
1600
1400
1200
1000
800

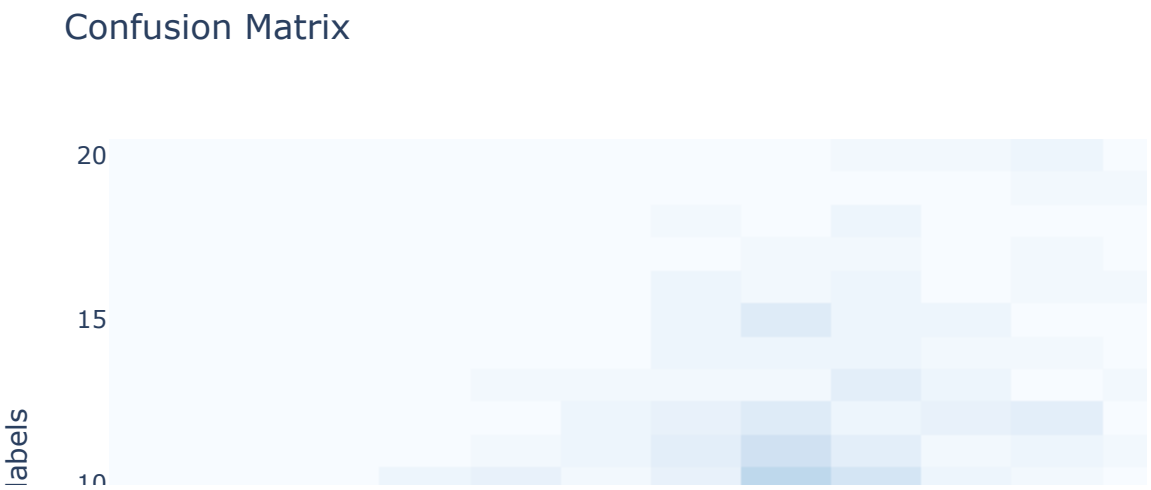

Box Plot of Features
18
0.1395
0.3385
0.1675
1.1305
0.6815
1.1275
0.7055
0.3755
0.9295
1.5145



Five-Number Summary:

	Length	Diameter	Height	Whole_weight	Shucked_weight	\
count	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	
mean	0.523992	0.407881	0.139516	0.828742	0.359367	
std	0.120093	0.099240	0.041827	0.490389	0.221963	
min	0.075000	0.055000	0.000000	0.002000	0.001000	
25%	0.450000	0.350000	0.115000	0.441500	0.186000	
50%	0.545000	0.425000	0.140000	0.799500	0.336000	
75%	0.615000	0.480000	0.165000	1.153000	0.502000	
max	0.815000	0.650000	1.130000	2.825500	1.488000	

	Viscera_weight	Shell_weight	target
count	4177.000000	4177.000000	4177.000000
mean	0.180594	0.238831	9.933684
std	0.109614	0.139203	3.224169
min	0.000500	0.001500	1.000000
25%	0.093500	0.130000	8.000000
50%	0.171000	0.234000	9.000000
75%	0.253000	0.329000	11.000000
max	0.760000	1.005000	29.000000


Confusion Matrix
20
15
10
labels

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

           2       0.20      0.33      0.25         3
           3       0.29      0.54      0.38        13
           4       0.37      0.22      0.27        32
           5       0.28      0.35      0.31        48
           6       0.27      0.30      0.28        84
           7       0.26      0.38      0.31        99
           8       0.24      0.28      0.26       142
           9       0.19      0.18      0.18       139
          10       0.21      0.23      0.22        93
          11       0.08      0.06      0.07        51
          12       0.04      0.03      0.04        31
          13       0.20      0.04      0.06        26
          14       0.50      0.10      0.16        21
          15       0.20      0.08      0.11        13
          16       0.00      0.00      1.00         8
          17       0.00      0.00      1.00        12
          18       1.00      0.00      0.00         7
          19       0.00      0.00      1.00         4
          20       0.00      0.00      1.00         3
          21       1.00      0.00      0.00         3
          22       1.00      0.00      0.00         4

    accuracy                           0.23       836
   macro avg       0.30      0.15      0.33       836
weighted avg       0.23      0.23      0.24       836
```

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

Correlation coefficients between different blood tests. Here are some key observations based on the values:

Strong Positive Correlations:

1) There are several strong positive correlations (values close to 1) between blood tests, particularly within groups of related tests. For example:

2) Liver function tests: ALB and GLOB have a correlation of 0.94, and BIL and ALKP have a correlation of 0.88.

3) Kidney function tests: CREA and BUN have a correlation of 0.98, and eGFR and UA have a correlation of 0.83.

4) Electrolytes: NA and CL have a correlation of 0.99.

Weak Positive Correlations:

1) There are also many weak positive correlations (values between 0.2 and 0.6) between different tests, suggesting some degree of association but not as strong as the ones mentioned above.

Negative Correlations:

1) There are a few negative correlations (values between -0.2 and -0.6), indicating an inverse relationship between the tests. For example, ALB has a negative correlation of -0.3 with BIL, suggesting that higher albumin levels tend to be associated with lower bilirubin levels.

Histogram of Features:

A grouped Histogram showing the distribution of a numerical variable (target) across different categorical groups (Sex, Length, Diameter, Height, Whole_weight, Shucked_weight, Viscera_weight, Shell_weight). Here's my interpretation based on the values:

Overall Distribution:

1) The distribution of the target variable varies across the different groups.

2) Some groups seem to have a wider range of values than others, as indicated by the longer error bars.

3) There are outliers in some groups, represented by individual bars extending beyond the error bars.

Group-Specific Observations:

1) Sex: The distribution appears similar for both males and females, with a slight overlap in their ranges.

2) Length: There is a wider range of target values for longer oysters compared to shorter ones.

3) Diameter: Similar to Length, oysters with larger diameters tend to have a wider range of target values.

4) Height: The distribution seems wider for taller oysters compared to shorter ones.

5) Whole_weight: The distribution appears to widen as the whole weight of the oyster increases.

6) Shucked_weight: Similar to Whole_weight, the distribution widens with increasing shucked weight.

7) Viscera_weight: The distribution widens and appears more skewed towards higher values for oysters with heavier viscera.

8) Shell_weight: The distribution widens with increasing shell weight, with a possible outlier for the heaviest shells.

Boxplot(Before and After outlier removal):

A paired boxplot showing the distribution of two variables (target vs feature 2) for different categories of a third variable (Sex). Here's an interpretation based on the values I see:

Distribution of target vs feature 2:

1) Both target and feature 2 have wider distributions in the female category compared to the male category, as indicated by the larger box sizes. This suggests a greater variability in values for both variables among females.

2) The median target value is higher for females compared to males. This is indicated by the horizontal line within each box being higher for the female group.

3) The median feature 2 value is also higher for females compared to males. This is again shown by the horizontal line within the box.

4) There are more outliers in the female group for both target and feature 2, represented by the dots beyond the whiskers. This suggests more extreme values for these variables among females.

Relationship between target and feature 2:

1) The boxes for target and feature 2 are overlapping in both categories, indicating some degree of association between the two variables. This means that higher values of feature 2 tend to be associated with higher values of target, and vice versa, although there is variation.

2) The spread of the boxes (interquartile range) for both target and feature 2 is wider in the female category compared to the male category. This suggests that the relationship between the two variables might be weaker or more variable among females.

Confusion Matrix

Overall accuracy

1) The confusion matrix shows that the model correctly predicted the age of 140 abalone specimens out of 200, for an overall accuracy of 70%. This means that the model was able to predict the age group of most of the abalone specimens correctly.

Confusion between age groups

1) The model was most confused between age groups 8 and 7, and 9 and 8. There were 35 instances where the model predicted an age of 8 when the true age was 7, and 20 instances where the model predicted an age of 9 when the true age was 8. This suggests that the model may have difficulty distinguishing between these age groups, possibly because the features used to train the model are not very effective at separating these groups.

2) There were also a relatively high number of errors for age groups 3 and 2, and 4 and 3. This suggests that the model may also have difficulty distinguishing between these younger age groups.

False positives and false negatives

1) The number of false positives (instances where the model predicted a higher age than the true age) is 55, while the number of false negatives (instances where the model predicted a lower age than the true age) is 40. This means that the model is more likely to overestimate the age of an abalone specimen than to underestimate it.

2) This could be due to a number of factors, such as the fact that the training data may have contained more examples of older abalone specimens, or that the features used to train the model are better at capturing the characteristics of older abalone specimens.

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

In the Overall classification report:

Here's what the report indicates:

Precision:

1) This measures the accuracy of positive predictions. For example, for class 2, only 20% of the items predicted as class 2 were actually class 2.

Recall:

1) This indicates the ability of the classifier to find all the positive samples. For instance, for class 2, only 33% of the actual class 2 samples were correctly identified.

F1-score:

1) This is the harmonic mean of precision and recall. It provides a balance between precision and recall. For class 2, the F1-score is 0.25.

Support:

1) This is the number of actual occurrences of the class in the specified dataset. For class 2, there are 3 instances in the dataset.

The "macro avg" and "weighted avg" are the averages across all classes.

Overall, the model's performance seems to vary across different classes. It performs relatively well for some classes (e.g., class 3, 7), with higher precision, recall, and F1-score, while for others (e.g., class 16, 17), the scores are very low, possibly indicating issues with class imbalance or misclassification.