```
In [1]: import numpy as np
        import pandas as pd
        import seaborn as sns
        import matplotlib.pyplot as plt
        import warnings
        from sklearn.model_selection import train_test_split, cross_val_score
        from sklearn.pipeline import make_pipeline
        from sklearn.preprocessing import StandardScaler
        from sklearn.ensemble import RandomForestRegressor
        from sklearn.metrics import make_scorer, mean_absolute_error, mean_squared_error
```

```
In [2]: warnings.filterwarnings('ignore')
```

```
In [3]: # Load the datasets
        df_red = pd.read_csv(r"C:\Users\USER\Downloads\wine\winequality-red.csv", sep=";")
        df_white = pd.read_csv(r"C:\Users\USER\Downloads\wine\winequality-white.csv", sep=";")
```

```
In [4]: # Combine the datasets
        df = pd.concat([df_red, df_white], ignore_index=True)
```

```
In [5]: # Check for null values
        print(df.isnull().sum())
```

```
fixed acidity           0
volatile acidity        0
citric acid             0
residual sugar          0
chlorides               0
free sulfur dioxide     0
total sulfur dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality                 0
dtype: int64
```

```
In [6]: # Check the basic info of the dataframe
        print(df.info())
```

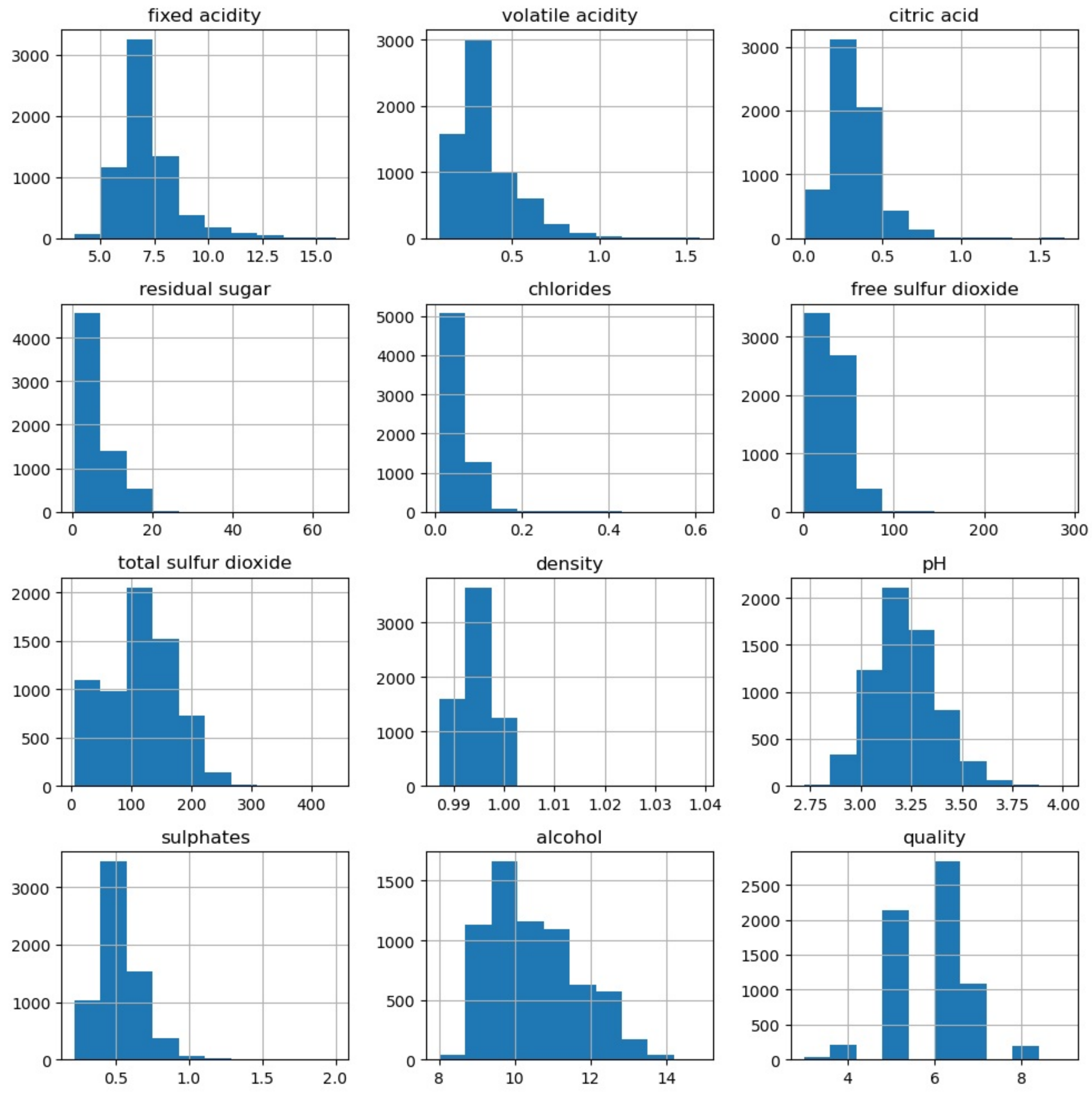
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6497 entries, 0 to 6496
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         6497 non-null   float64
 1   volatile acidity      6497 non-null   float64
 2   citric acid           6497 non-null   float64
 3   residual sugar        6497 non-null   float64
 4   chlorides             6497 non-null   float64
 5   free sulfur dioxide   6497 non-null   float64
 6   total sulfur dioxide  6497 non-null   float64
 7   density               6497 non-null   float64
 8   pH                    6497 non-null   float64
 9   sulphates             6497 non-null   float64
 10  alcohol               6497 non-null   float64
 11  quality               6497 non-null   int64  
dtypes: float64(11), int64(1)
memory usage: 609.2 KB
None
```

```
In [7]: # Describe the data
        print(df.describe())
```

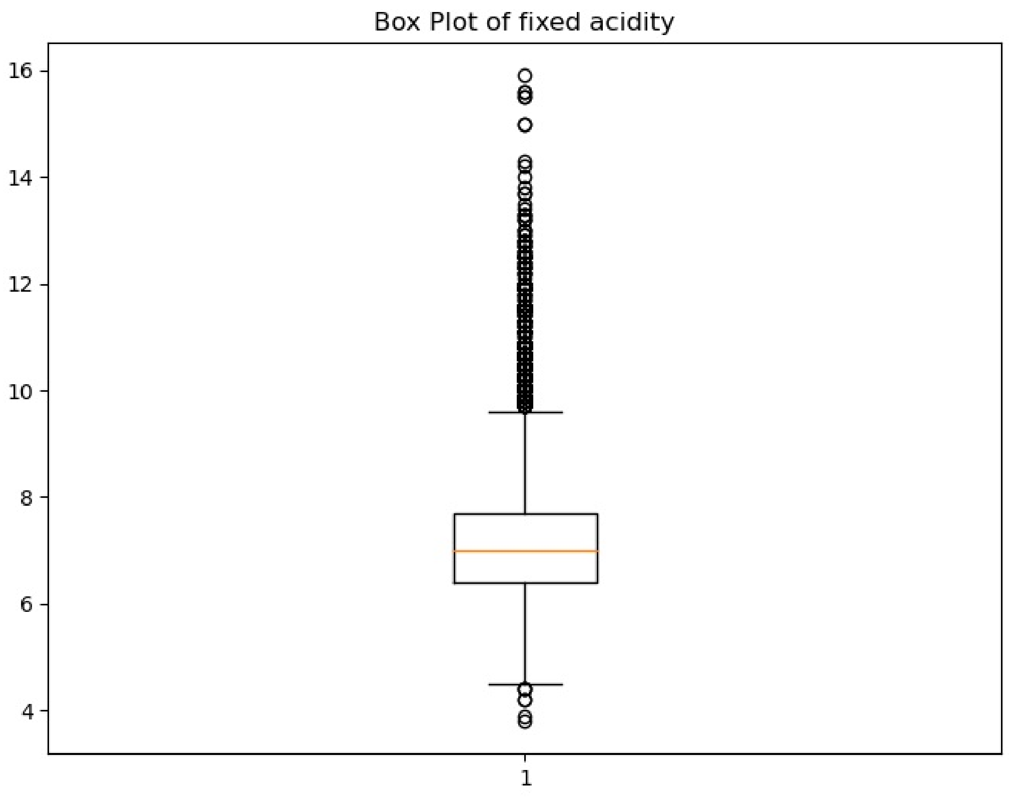
```
       fixed acidity  volatile acidity  citric acid  residual sugar  \
count    6497.000000       6497.000000  6497.000000     6497.000000   
mean        7.215307          0.339666     0.318633        5.443235   
std         1.296434          0.164636     0.145318        4.757804   
min         3.800000          0.080000     0.000000        0.600000   
25%         6.400000          0.230000     0.250000        1.800000   
50%         7.000000          0.290000     0.310000        3.000000   
75%         7.700000          0.400000     0.390000        8.100000   
max        15.900000          1.580000     1.660000       65.800000   

         chlorides  free sulfur dioxide  total sulfur dioxide      density  \
count  6497.000000          6497.000000           6497.000000  6497.000000   
mean      0.056034            30.525319            115.744574     0.994697   
std       0.035034            17.749400             56.521855     0.002999   
min       0.009000             1.000000              6.000000     0.987110   
25%       0.038000            17.000000             77.000000     0.992340   
50%       0.047000            29.000000            118.000000     0.994890   
75%       0.065000            41.000000            156.000000     0.996990   
max       0.611000           289.000000            440.000000     1.038980   

                pH    sulphates      alcohol      quality  
count  6497.000000  6497.000000  6497.000000  6497.000000  
mean      3.218501     0.531268    10.491801     5.818378  
std       0.160787     0.148806     1.192712     0.873255  
min       2.720000     0.220000     8.000000     3.000000  
25%       3.110000     0.430000     9.500000     5.000000  
50%       3.210000     0.510000    10.300000     6.000000  
75%       3.320000     0.600000    11.300000     6.000000  
max       4.010000     2.000000    14.900000     9.000000  
```

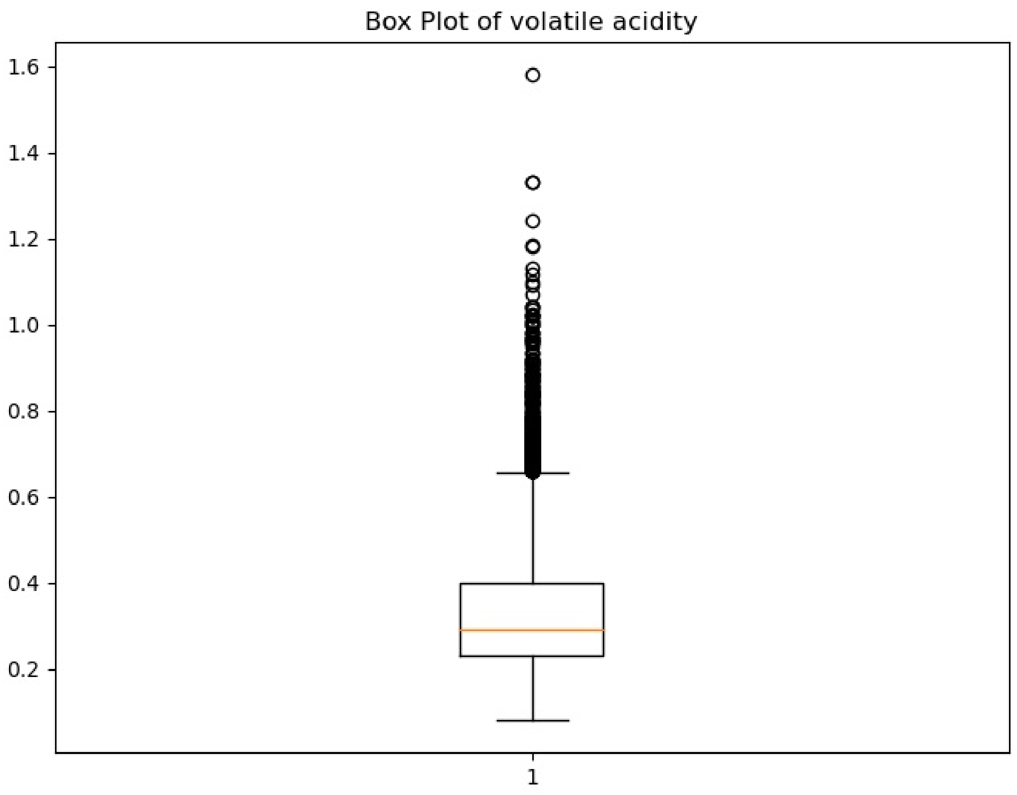
In [8]:

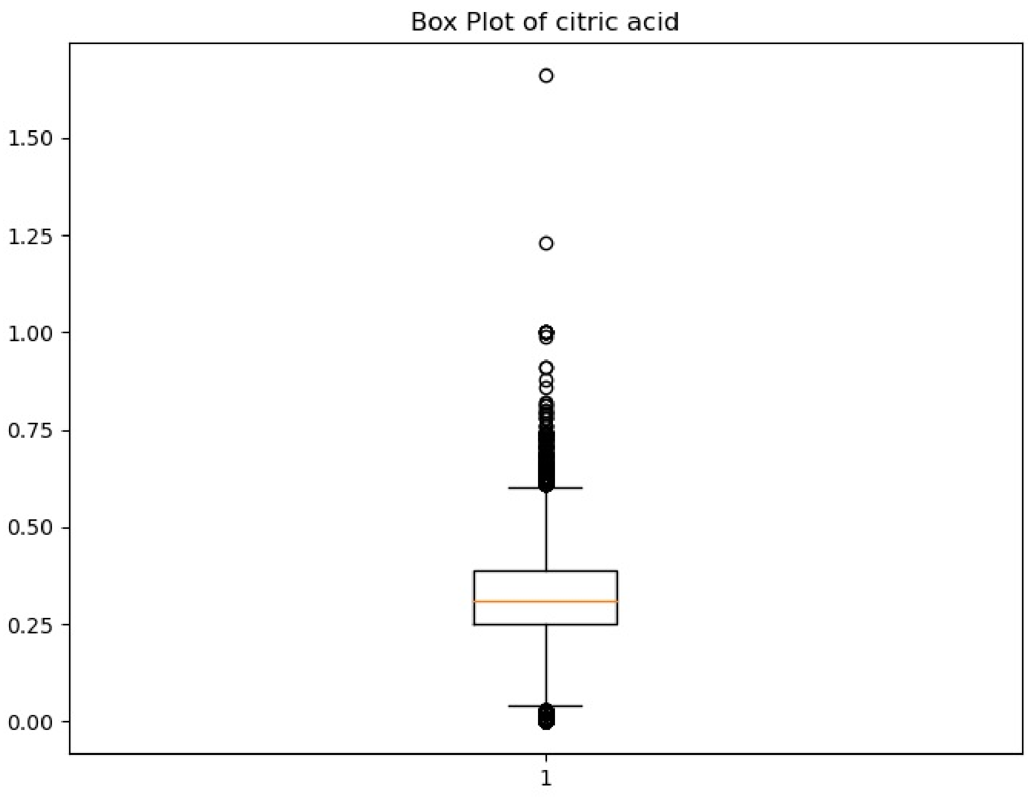
```python
# Plot histograms for all features
df.hist(figsize=(10, 10))
plt.tight_layout()
plt.show()
```

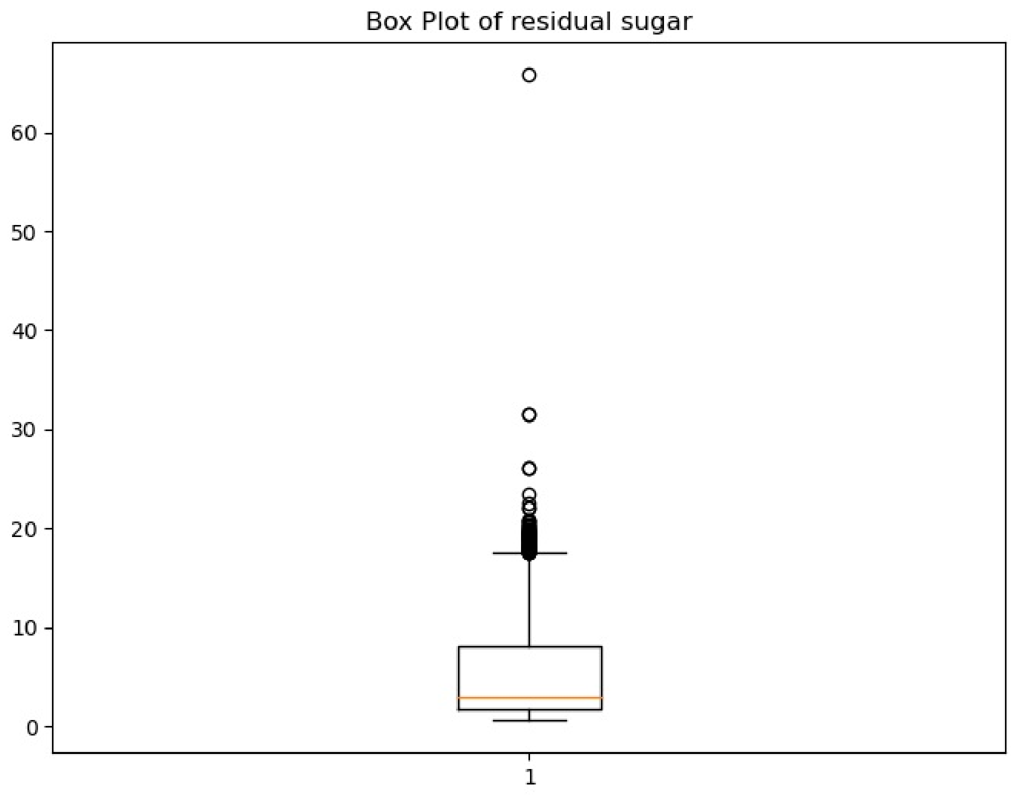


In [9]:
```python
# Boxplots for each feature
for column in df.columns:
    plt.figure(figsize=(8, 6))
    plt.boxplot(df[column])
    plt.title(f'Box Plot of {column}')
    plt.show()
```

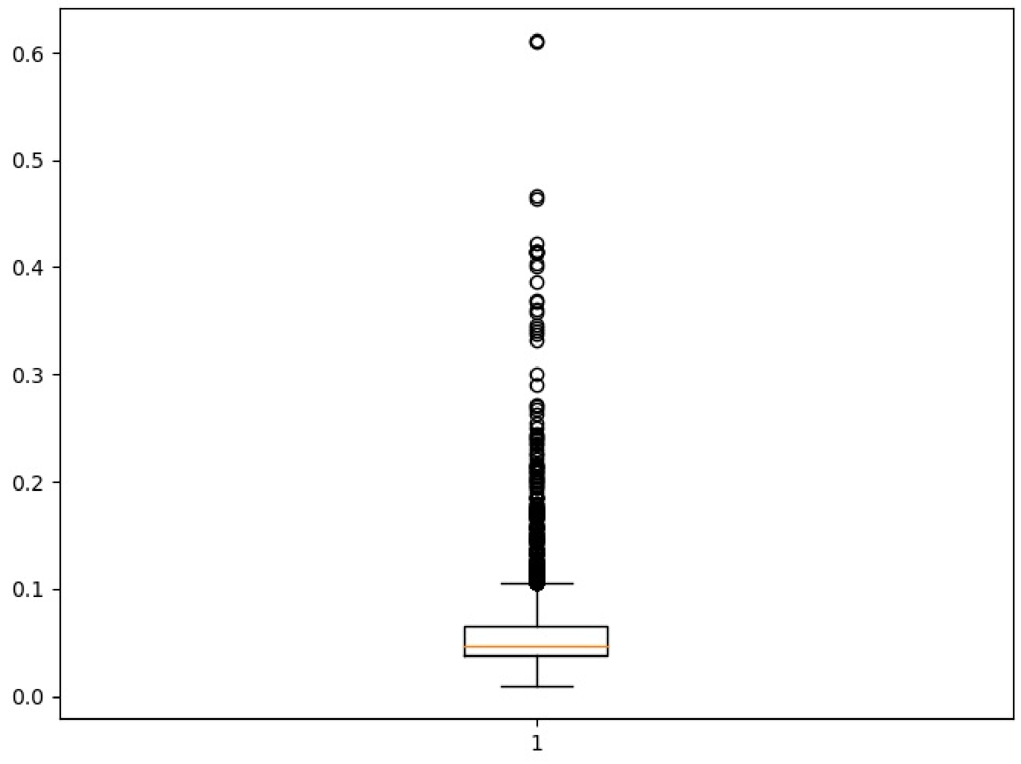

Box Plot of fixed acidity
16
14
12
10
8
6
4
1


Box Plot of volatile acidity
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
1

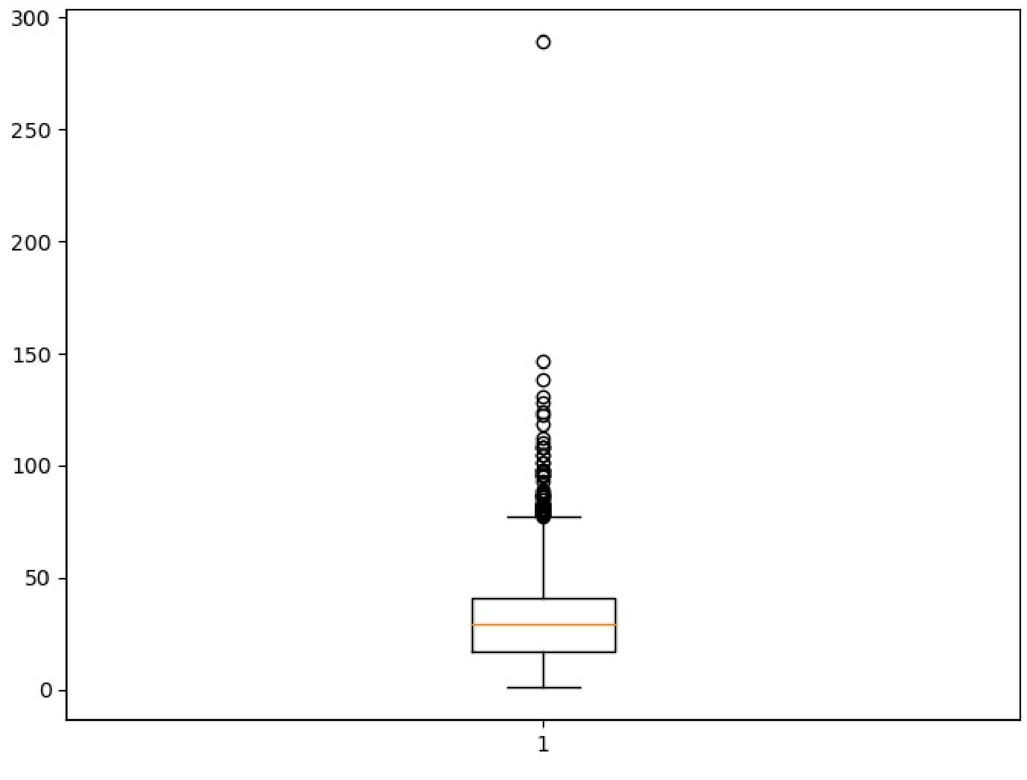

Box Plot of citric acid
1.50
1.25
1.00
0.75
0.50
0.25
0.00
1

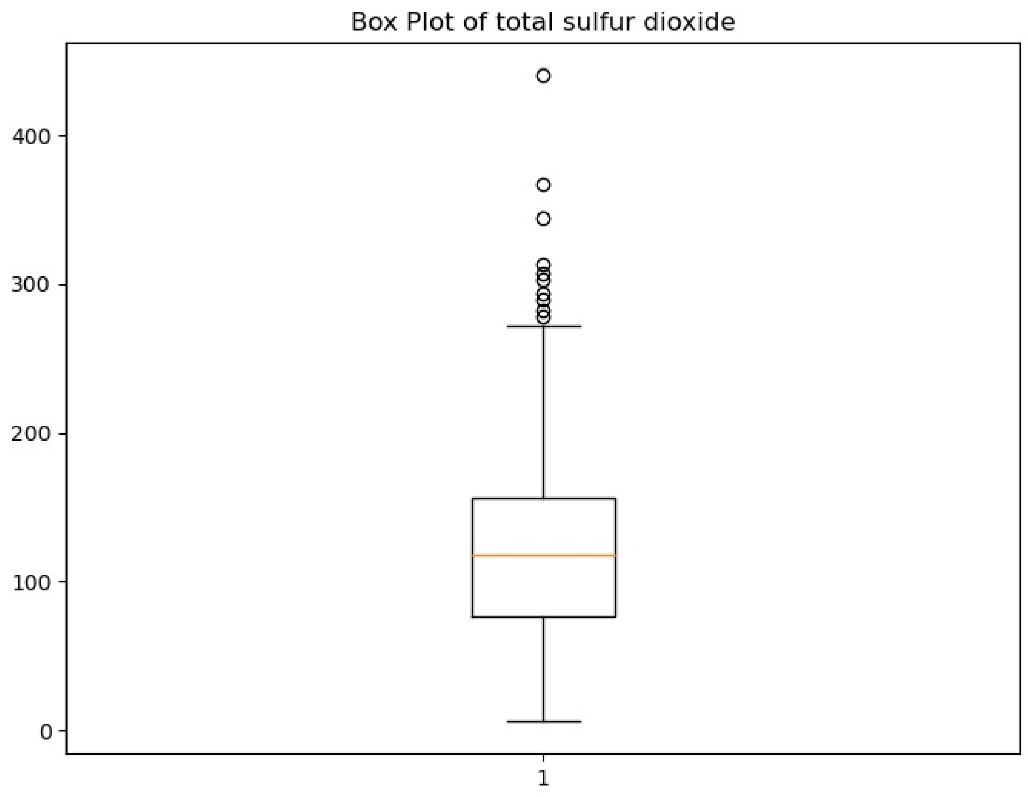

Box Plot of residual sugar
60
50
40
30
20
10
0
1

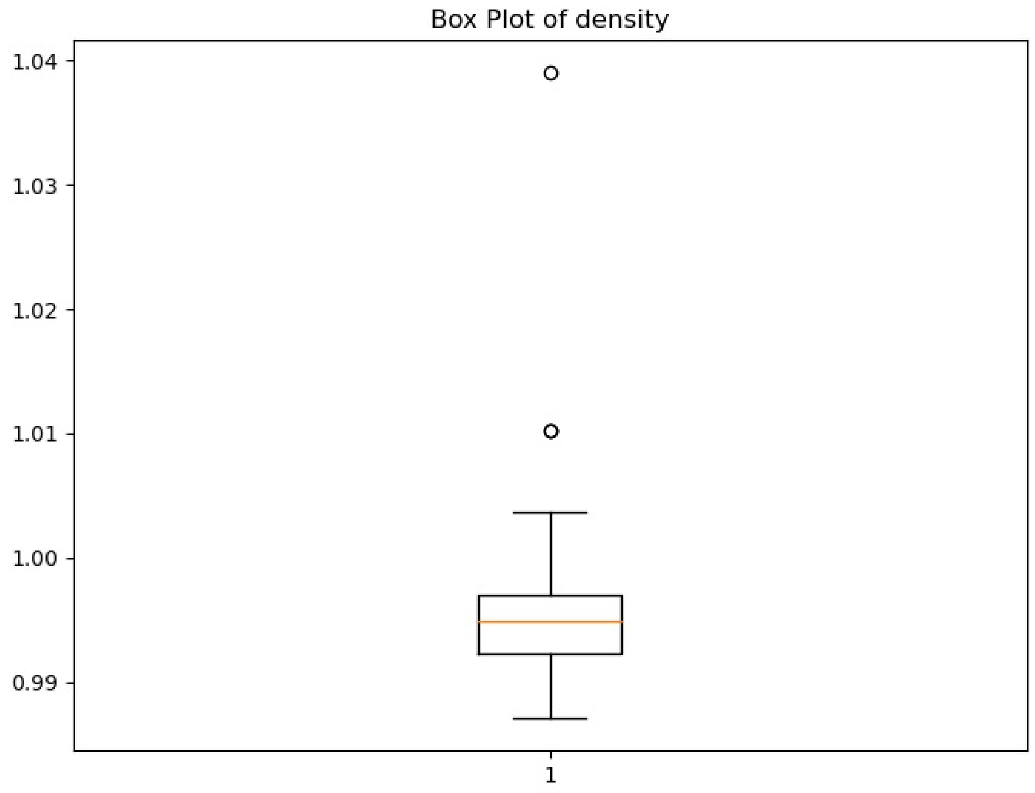
Box Plot of chlorides

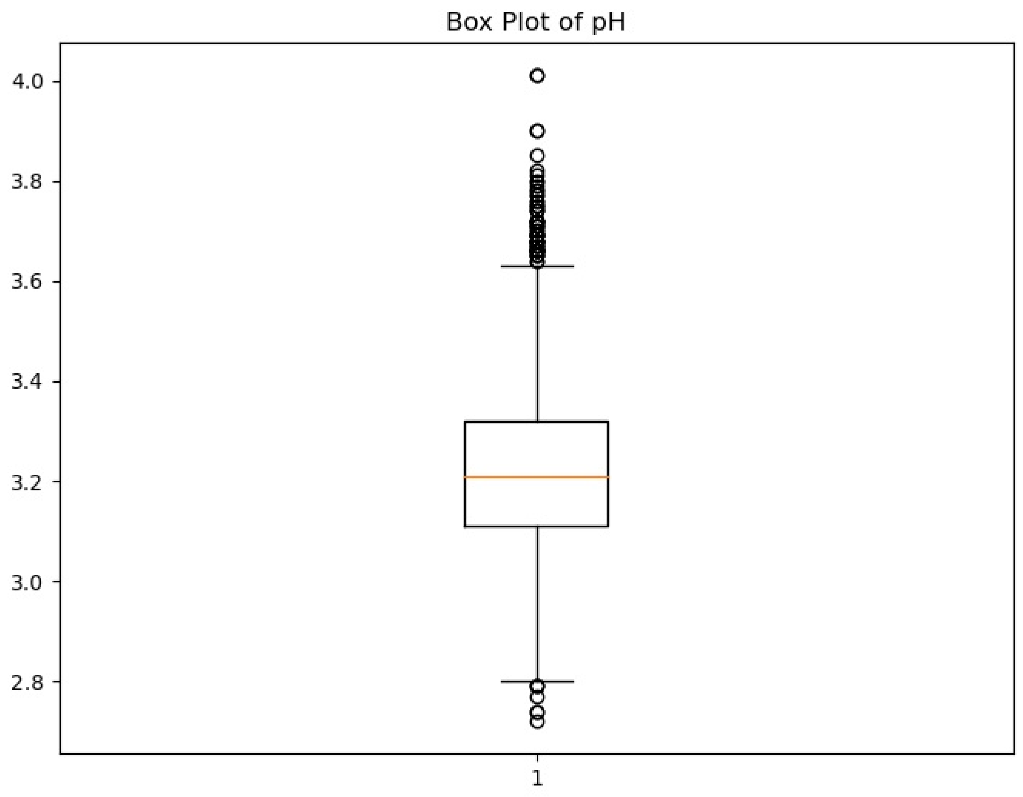


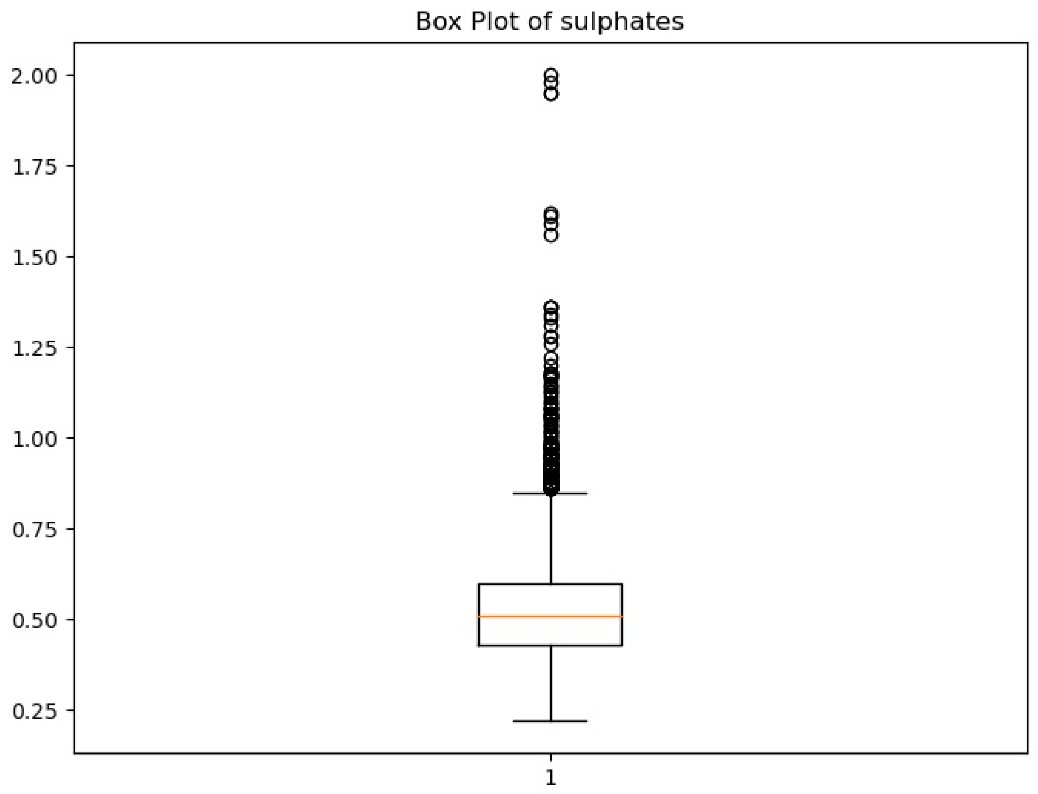
Box Plot of free sulfur dioxide

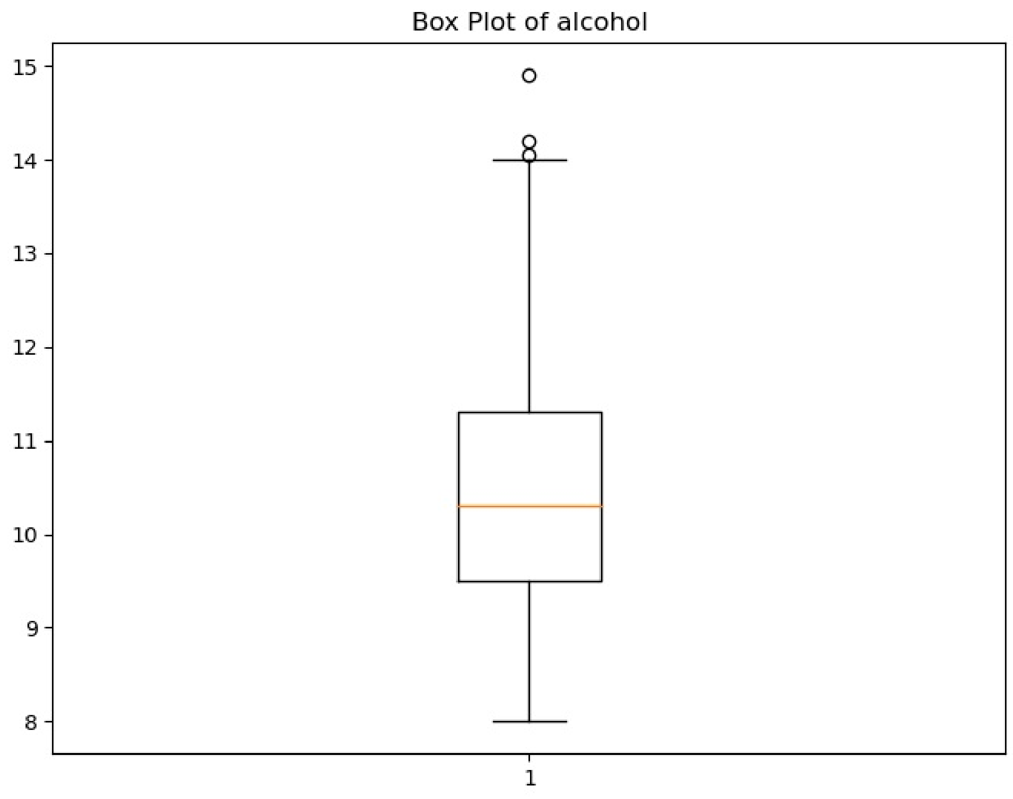


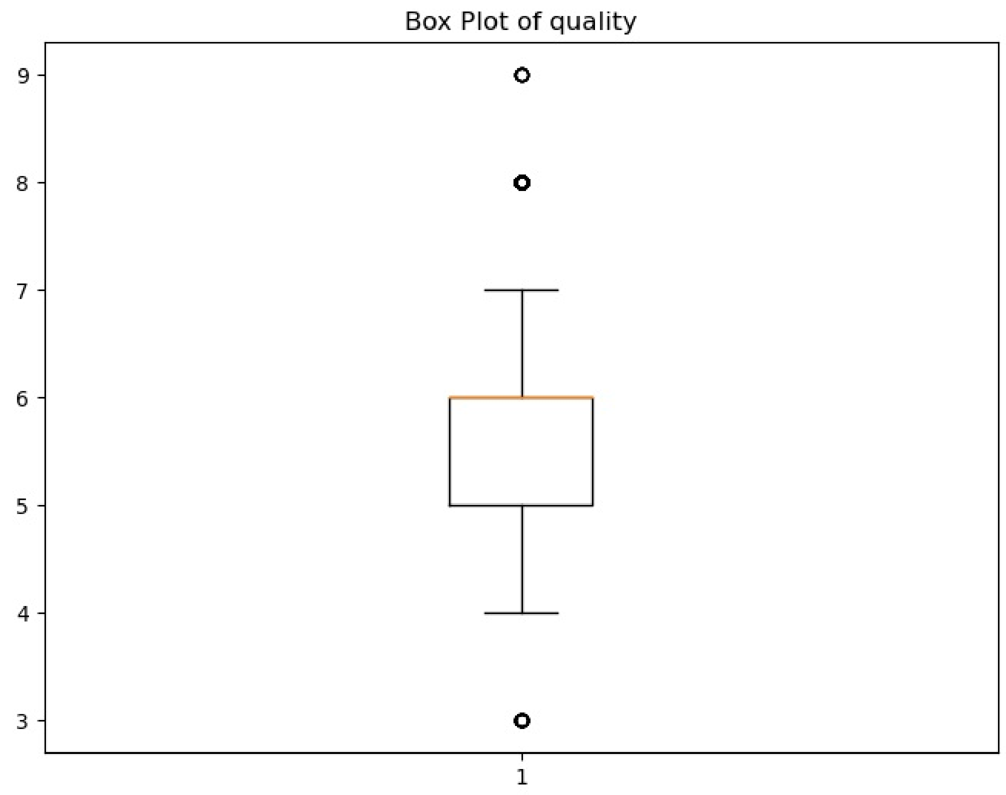

Box Plot of total sulfur dioxide
400
300
200
100
0
1


Box Plot of density
1.04
1.03
1.02
1.01
1.00
0.99
1

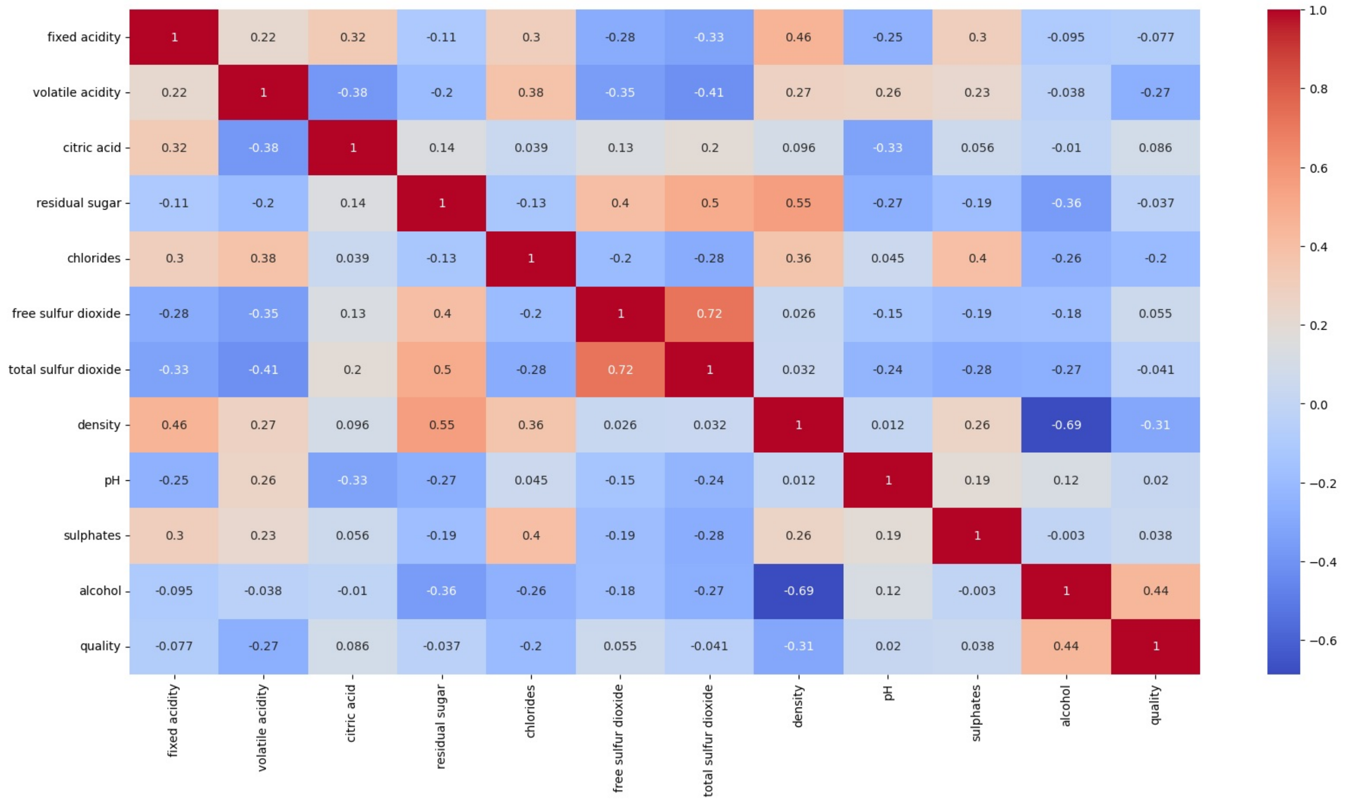

Box Plot of pH
4.0
3.8
3.6
3.4
3.2
3.0
2.8
1


Box Plot of sulphates
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
1





In [10]:
```
# Correlation heatmap
plt.figure(figsize=(20, 10))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.show()
```



In [11]:
```
# Prepare data for machine learning
X = df.drop('quality', axis=1)
y = df['quality']
```

In [12]:
```
# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=89)
```

In [13]:
```
# Print the shape of the training and testing data
print(X_train.shape, X_test.shape)
```

```
(5197, 11) (1300, 11)
```

In [14]:
```
# Create the pipeline
pipeline = make_pipeline(StandardScaler(), RandomForestRegressor(random_state=42))
```

In [15]:
```
# Using R-squared for cross-validation
r2_scores = cross_val_score(pipeline, X, y, scoring='r2', cv=5)
mean_r2 = r2_scores.mean()
print(f'Mean R-squared: {mean_r2:.2f}')
```

```
Mean R-squared: 0.27
```

Interpretation of R-squared:

0 to 1 Range: R^2 ranges from 0 to 1. An R^2 value closer to 1 indicates a better fit. Low R^2 (0.27): This suggests that the model explains only a small portion of the variance, implying a potentially weak model for predicting wine quality based on the given features.

In [16]:
```
from sklearn.linear_model import LogisticRegression
```

In [17]:
```
# Create a new pipeline with LogisticRegression
lr_pipeline = make_pipeline(StandardScaler(), LogisticRegression(random_state=42, max_iter=10000))
```

In [18]:
```
lr_scores = cross_val_score(lr_pipeline, X, y, scoring='accuracy', cv=5, n_jobs=-1)
mean_lr_accuracy = lr_scores.mean()
print(f'Mean Accuracy: {mean_lr_accuracy:.2f}')
```

```
Mean Accuracy: 0.50
```

In [21]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
```

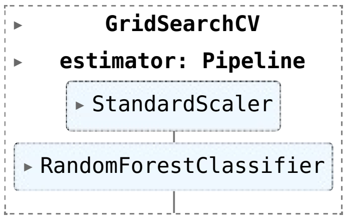
In [22]:
```
rf_pipeline = make_pipeline(StandardScaler(), RandomForestClassifier(random_state=42))
```

In [23]:
```
param_grid = {
    'randomforestclassifier__n_estimators': [50, 100, 200],
    'randomforestclassifier__max_depth': [None, 10, 20, 30],
    'randomforestclassifier__min_samples_split': [2, 5, 10],
    'randomforestclassifier__min_samples_leaf': [1, 2, 4]
}
```

In [24]:
```
grid_search = GridSearchCV(rf_pipeline, param_grid, cv=5, scoring='accuracy', n_jobs=-1)
```

```
grid_search.fit(X_train, y_train)
```

Out[24]:



In [26]:

```
# Evaluate best model from Grid Search
best_rf_model = grid_search.best_estimator_
```

In [27]:

```
# Cross-validation score
rf_scores = cross_val_score(best_rf_model, X, y, scoring='accuracy', cv=5, n_jobs=-1)
mean_rf_accuracy = rf_scores.mean()
print(f'Mean Accuracy with Random Forest: {mean_rf_accuracy:.2f}')
```

```
Mean Accuracy with Random Forest: 0.48
```

Okay, here we try 3 models but, its not working properly any of the model, we may need to do more some feature engineering, we will try second version of this code again

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js