Q. Create a Multi variable Regression model of your choice using suitable dataset.

Regression Models:

1. Multiple Regression
2. Naive Bayes Regression Model
3. Naive Bayes Classification Model
4. Random Forest Classification Model

Perform EDA on the dataset and implement the model and calculate accuracy score, Precision, Recall, F1 score of each linear model

Importing all necessary library

```
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.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.tree import plot_tree
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_curve, auc
```

Load the dataset and displaying head of the dataset.

```
0     p          x           s          n       t    p                f
1     e          x           s          y       t    a                f
2     e          b           s          w       t    l                f
3     p          x           y          w       t    p                f
4     e          x           s          g       f    n                f
```

```
  gill-spacing gill-size gill-color  ... stalk-surface-below-ring  \
0            c         n          k  ...                        s
1            c         b          k  ...                        s
```

```
2           c         b          n  ...                          s
3           c         n          n  ...                          s
4           w         b          k  ...                          s

  stalk-color-above-ring stalk-color-below-ring veil-type veil-color  \
0                      w                      w         p          w
1                      w                      w         p          w
2                      w                      w         p          w
3                      w                      w         p          w
4                      w                      w         p          w

  ring-number ring-type spore-print-color  population habitat
0           o         p                 k           s       u
1           o         p                 n           n       g
2           o         p                 n           n       m
3           o         p                 k           s       u
4           o         e                 n           a       g

[5 rows x 23 columns]
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8124 entries, 0 to 8123
Data columns (total 23 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   class                     8124 non-null   object
 1   cap-shape                 8124 non-null   object
 2   cap-surface               8124 non-null   object
 3   cap-color                 8124 non-null   object
 4   bruises                   8124 non-null   object
 5   odor                      8124 non-null   object
 6   gill-attachment           8124 non-null   object
 7   gill-spacing              8124 non-null   object
 8   gill-size                 8124 non-null   object
 9   gill-color                8124 non-null   object
 10  stalk-shape               8124 non-null   object
 11  stalk-root                8124 non-null   object
 12  stalk-surface-above-ring  8124 non-null   object
 13  stalk-surface-below-ring  8124 non-null   object
 14  stalk-color-above-ring    8124 non-null   object
 15  stalk-color-below-ring    8124 non-null   object
 16  veil-type                 8124 non-null   object
 17  veil-color                8124 non-null   object
 18  ring-number               8124 non-null   object
 19  ring-type                 8124 non-null   object
 20  spore-print-color         8124 non-null   object
 21  population                8124 non-null   object
 22  habitat                   8124 non-null   object
dtypes: object(23)
memory usage: 1.4+ MB
```

Checking null values

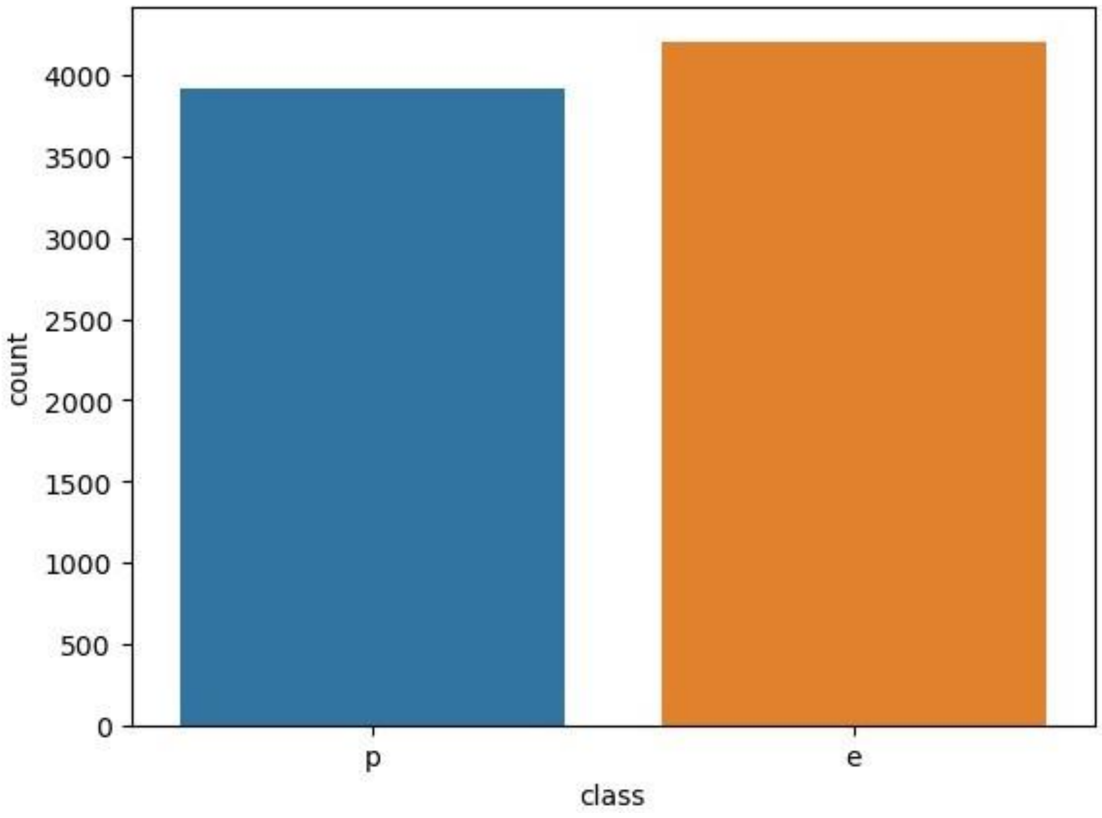
```
df.isnull().sum()
```

```
class                       0
cap-shape                   0
```

```
cap-surface                 0
cap-color                   0
bruises                     0
odor                        0
gill-attachment             0
gill-spacing                0
gill-size                   0
gill-color                  0
stalk-shape                 0
stalk-root                  0
stalk-surface-above-ring    0
stalk-surface-below-ring    0
stalk-color-above-ring      0
stalk-color-below-ring      0
veil-type                   0
veil-color                  0
ring-number                 0
ring-type                   0
spore-print-color           0
population                  0
habitat                     0
dtype: int64
```

Plot the distribution of the target variable 'class'

```
sns.countplot(x='class', data=df)

plt.show()
```



Encoding dataset and print head

```
le = LabelEncoder()
for column in df.columns:
    df[column] = le.fit_transform(df[column])

df.head()
```

```
   class  cap-shape  cap-surface  cap-color  bruises  odor  gill-attachment  \
0      1          5            2          4        1     6                1
1      0          5            2          9        1     0                1
2      0          0            2          8        1     3                1
3      1          5            3          8        1     6                1
4      0          5            2          3        0     5                1

   gill-spacing  gill-size  gill-color  ...  stalk-surface-below-ring  \
0             0          1           4  ...                         2
1             0          0           4  ...                         2
2             0          0           5  ...                         2
3             0          1           5  ...                         2
4             1          0           4  ...                         2

   stalk-color-above-ring  stalk-color-below-ring  veil-type  veil-color  \
0                       7                       7          0           2
1                       7                       7          0           2
2                       7                       7          0           2
3                       7                       7          0           2
4                       7                       7          0           2

   ring-number  ring-type  spore-print-color  population  habitat
0            1          4                  2           3        5
1            1          4                  3           2        1
2            1          4                  3           2        3
3            1          4                  2           3        5
4            1          0                  3           0        1

[5 rows x 23 columns]
```

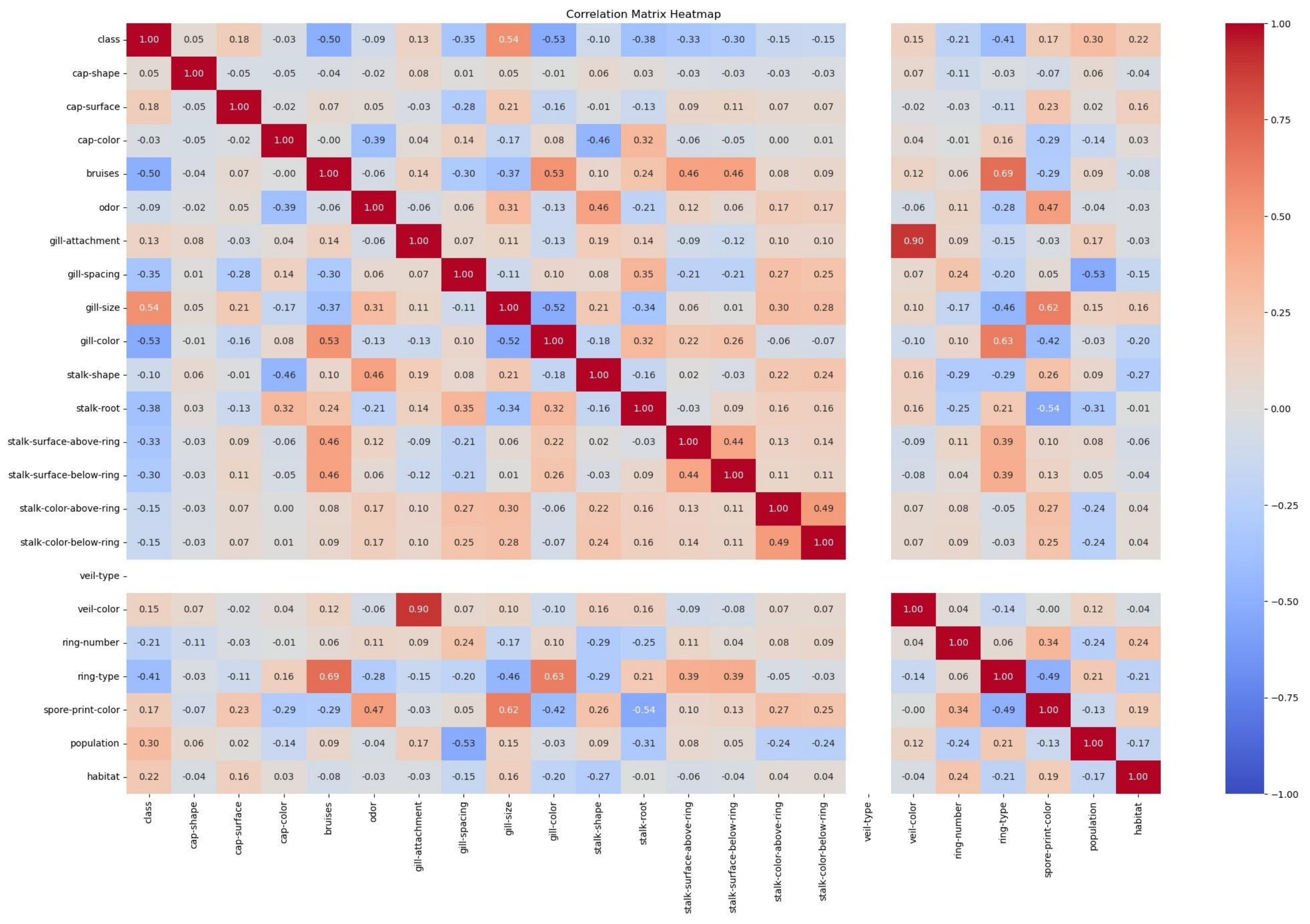
Getting correlation having strongest value

```
correlation_matrix = df.corr()

# Get absolute values of correlation matrix
abs_corr_matrix = correlation_matrix.abs()

# Get pairs of variables with high correlation
strong_corrs = abs_corr_matrix.stack().reset_index()
strong_corrs.columns = ['Feature1', 'Feature2', 'Correlation']
strong_corrs = strong_corrs[strong_corrs['Feature1'] !=
strong_corrs['Feature2']]
strong_corrs = strong_corrs.sort_values(by='Correlation', ascending=False)

top_cor = strong_corrs.head(30)
print(top_cor)
```

```
                Feature1          Feature2  Correlation
358           veil-color   gill-attachment     0.897518
148      gill-attachment        veil-color     0.897518
400            ring-type           bruises     0.692973
106              bruises         ring-type     0.692973
405            ring-type        gill-color     0.629398
216           gill-color         ring-type     0.629398
```

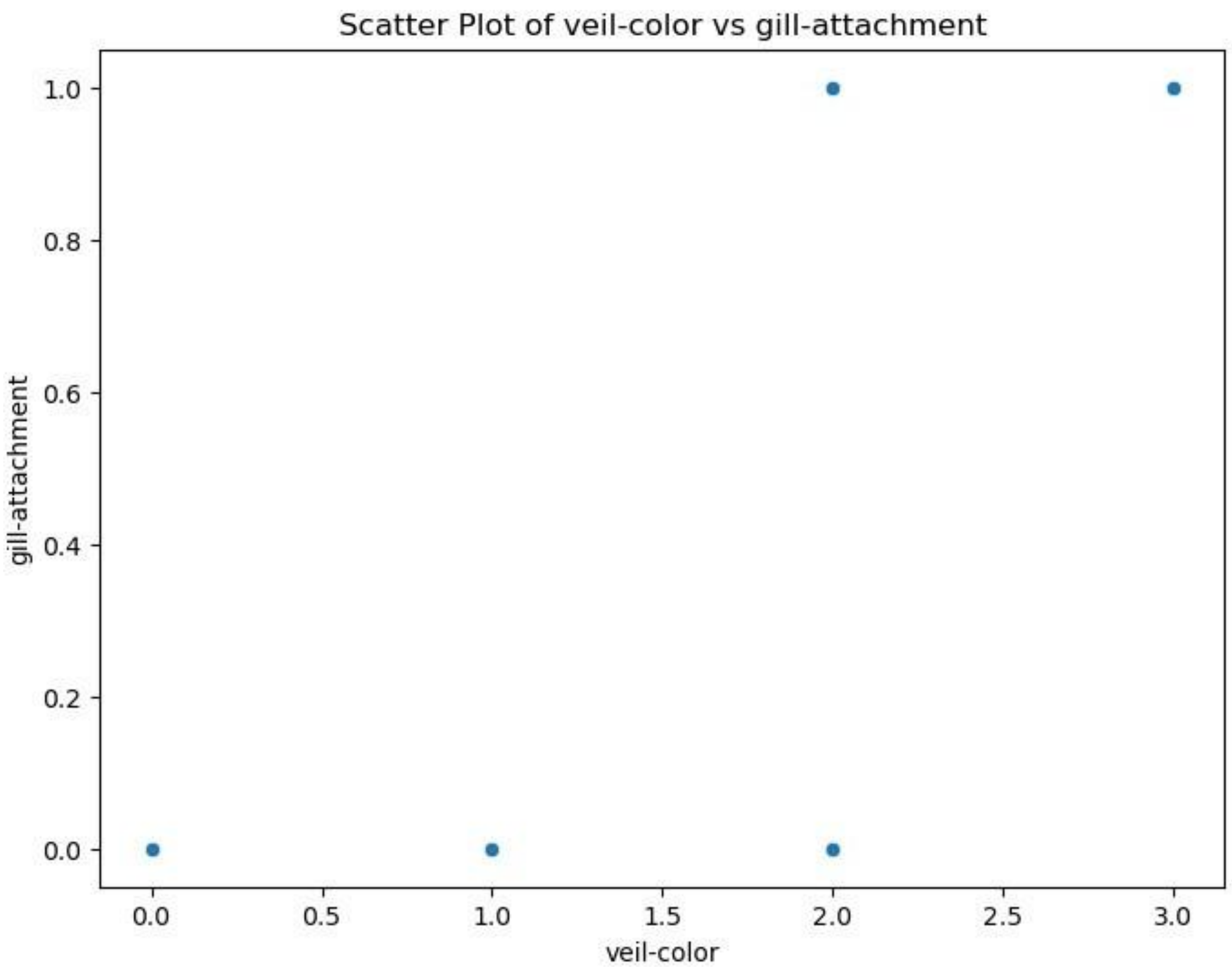
```
195                 gill-size        spore-print-color     0.622991
426         spore-print-color                gill-size     0.622991
176                 gill-size                    class     0.540024
8                       class                gill-size     0.540024
429         spore-print-color               stalk-root     0.536996
261                stalk-root        spore-print-color     0.536996
9                       class               gill-color     0.530566
198                gill-color                    class     0.530566
174              gill-spacing               population     0.529253
447                population             gill-spacing     0.529253
202                gill-color                  bruises     0.527120
97                    bruises               gill-color     0.527120
206                gill-color                gill-size     0.516736
185                 gill-size               gill-color     0.516736
88                    bruises                    class     0.501530
4                       class                  bruises     0.501530
344   stalk-color-below-ring   stalk-color-above-ring     0.491510
323   stalk-color-above-ring   stalk-color-below-ring     0.491510
415                 ring-type        spore-print-color     0.487048
436         spore-print-color                ring-type     0.487048
129                      odor        spore-print-color     0.469055
423         spore-print-color                     odor     0.469055
404                 ring-type                gill-size     0.460872
194                 gill-size                ring-type     0.460872
```

```
plt.figure(figsize=(25, 15))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', vmin=-
1, vmax=1)
plt.title('Correlation Matrix Heatmap')
plt.show()
```



Correlation having highest value

```
top_corr_pair = strong_corrs.iloc[0]
feature1 = top_corr_pair['Feature1']
feature2 = top_corr_pair['Feature2']

plt.figure(figsize=(8, 6))
sns.scatterplot(x=df[feature1], y=df[feature2])
plt.title(f'Scatter Plot of {feature1} vs {feature2}')
plt.xlabel(feature1)
plt.ylabel(feature2)
plt.show()
```



Feature and target for regression model

```
features = ['gill-spacing', 'ring-type', 'gill-size', 'gill-color']
target = 'class'

X = df[features]
y = df[target]

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

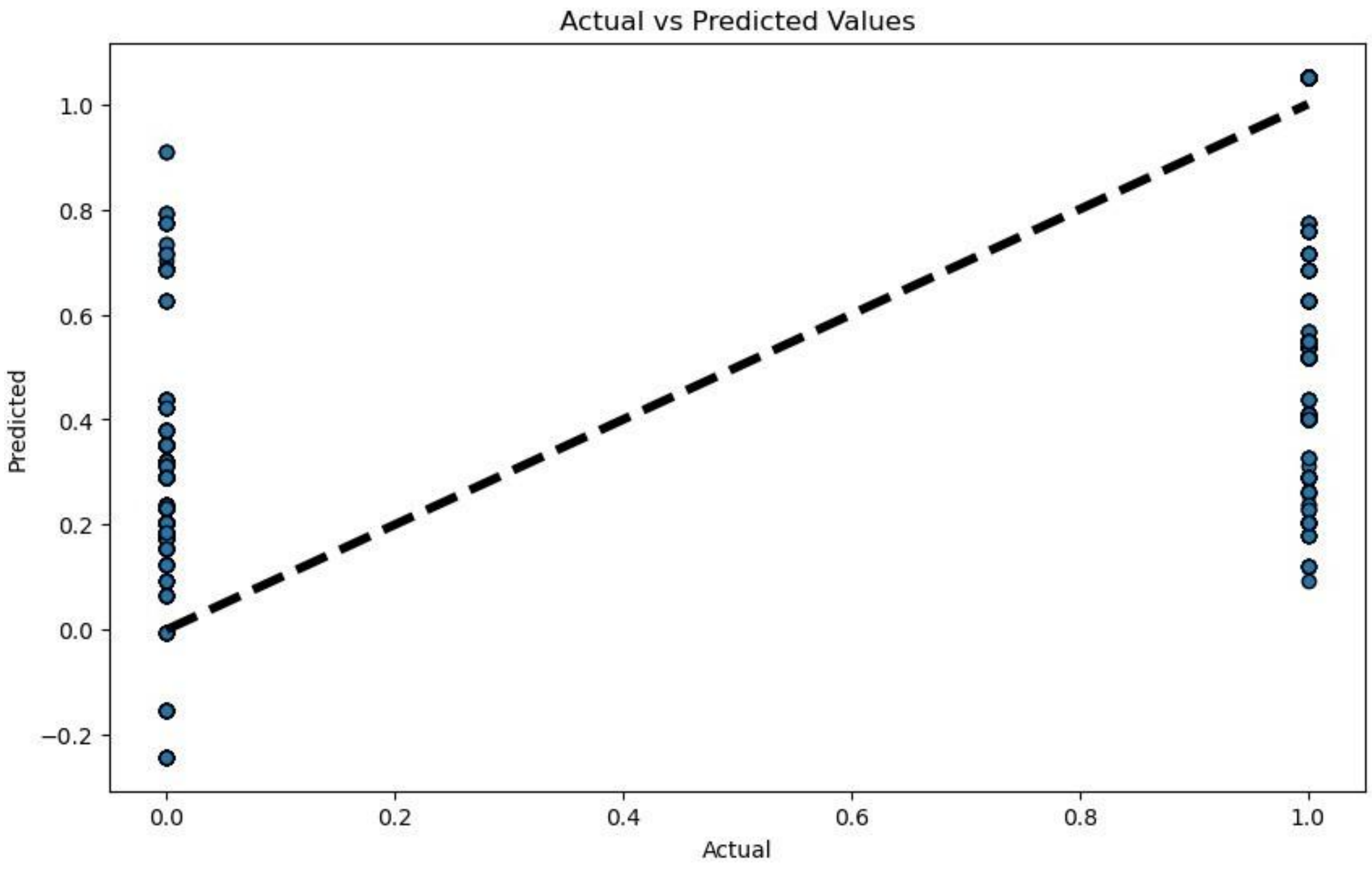
Multiple Regression

```
# Multiple Regression
model_reg = LinearRegression()
model_reg.fit(X_train, y_train)
y_pred_reg = model_reg.predict(X_test)

print("Multiple Regression Evaluation:")
print(f"R-squared: {r2_score(y_test, y_pred_reg):.2f}")
print(f"MAE: {mean_absolute_error(y_test, y_pred_reg):.2f}")
print(f"MSE: {mean_squared_error(y_test, y_pred_reg):.2f}")
print(f"RMSE: {mean_squared_error(y_test, y_pred_reg, squared=False):.2f}")
```

```
Multiple Regression Evaluation:
R-squared: 0.48
MAE: 0.30
MSE: 0.13
RMSE: 0.36
```

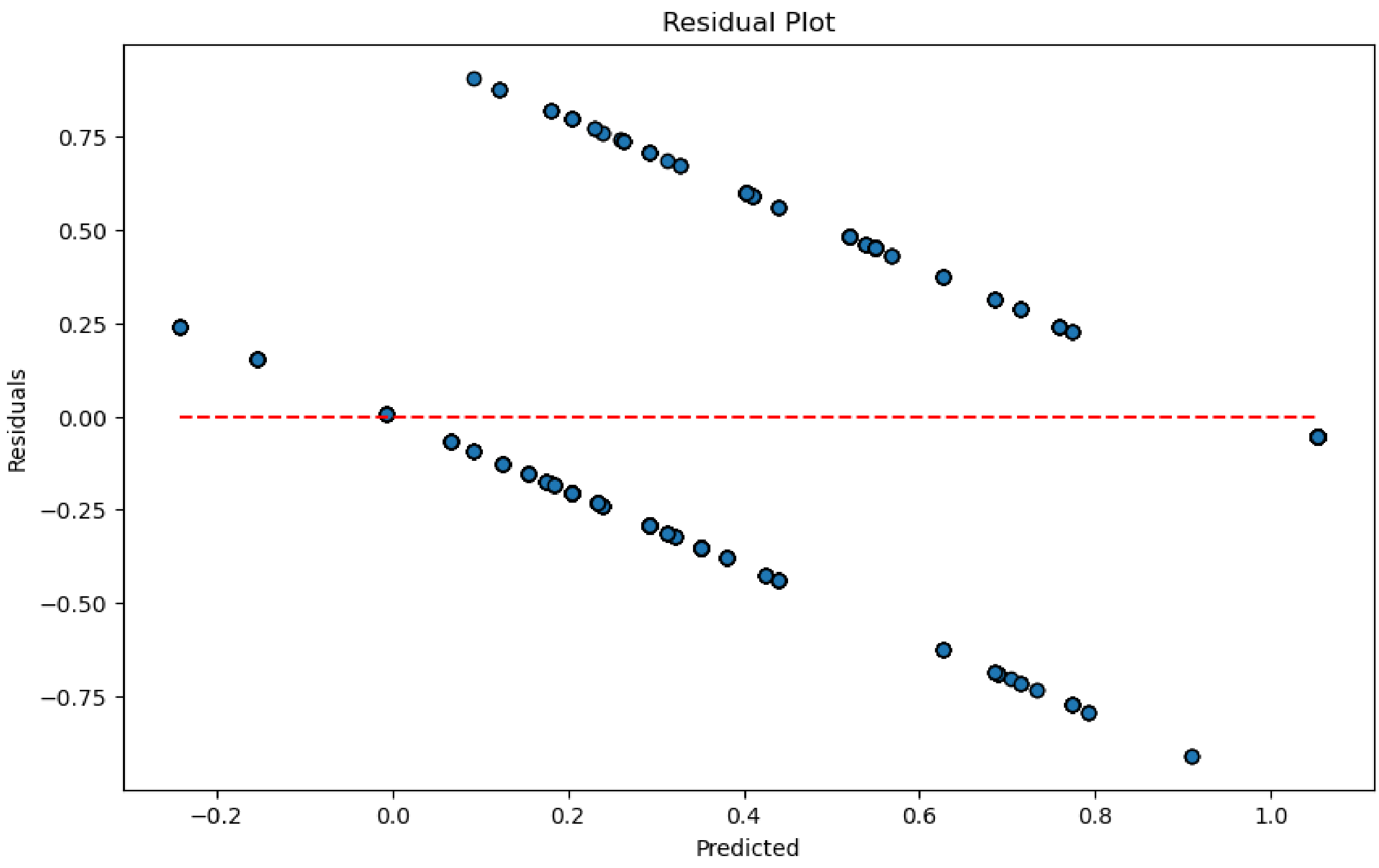
Scatter Plot of Actual vs. Predicted Values

```
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred_reg, edgecolors=(0, 0, 0))
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--',
lw=4)
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.title('Actual vs Predicted Values')
plt.show()
```



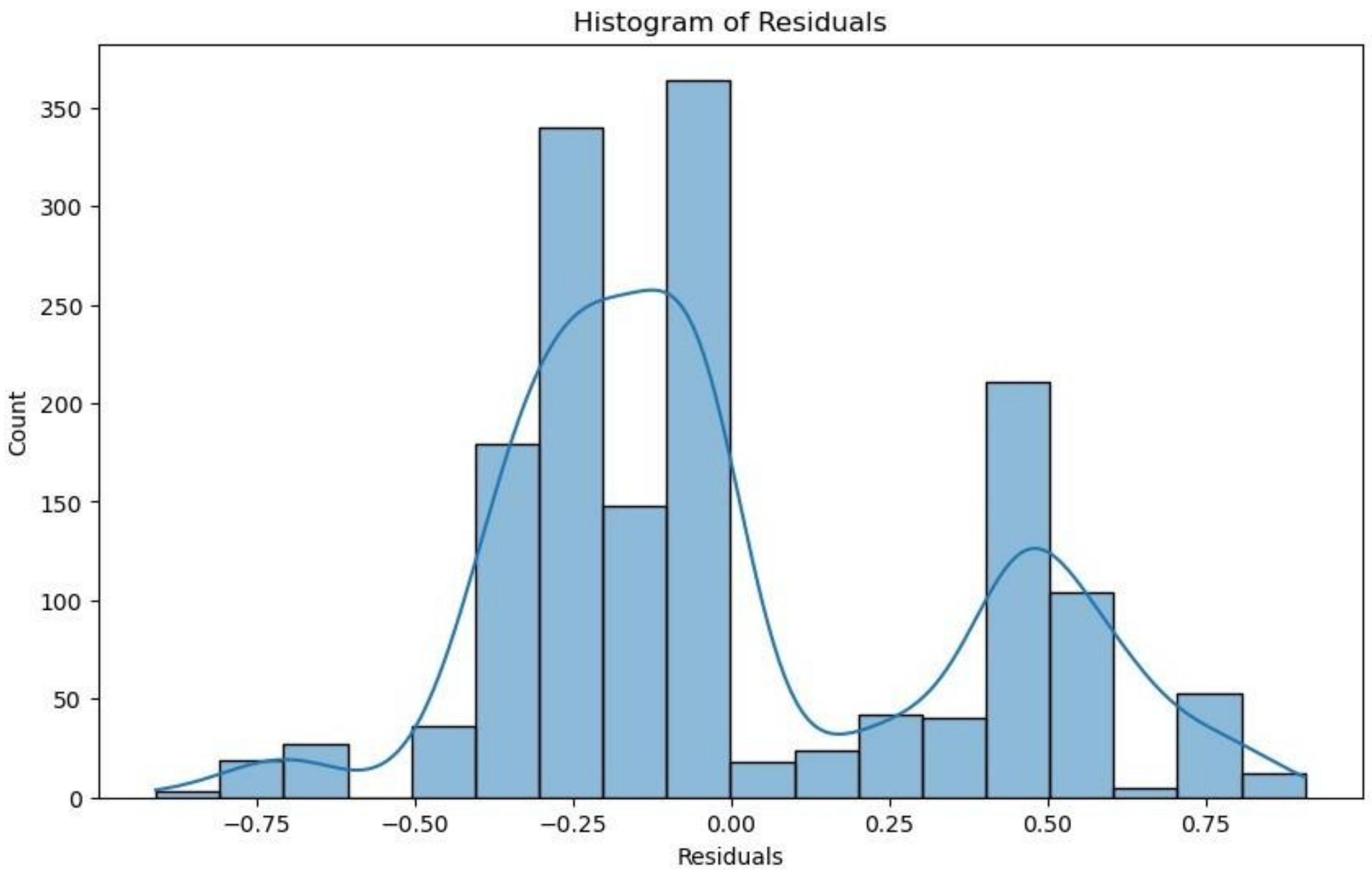
Residual Plot

```
residuals = y_test - y_pred_reg
plt.figure(figsize=(10, 6))
plt.scatter(y_pred_reg, residuals, edgecolors=(0, 0, 0))
plt.hlines(y=0, xmin=y_pred_reg.min(), xmax=y_pred_reg.max(), colors='r',
linestyles='dashed')
plt.xlabel('Predicted')
plt.ylabel('Residuals')
plt.title('Residual Plot')
plt.show()
```



Histogram of Residuals

```
plt.figure(figsize=(10, 6))
sns.histplot(residuals, kde=True)
plt.xlabel('Residuals')
plt.title('Histogram of Residuals')
plt.show()
```



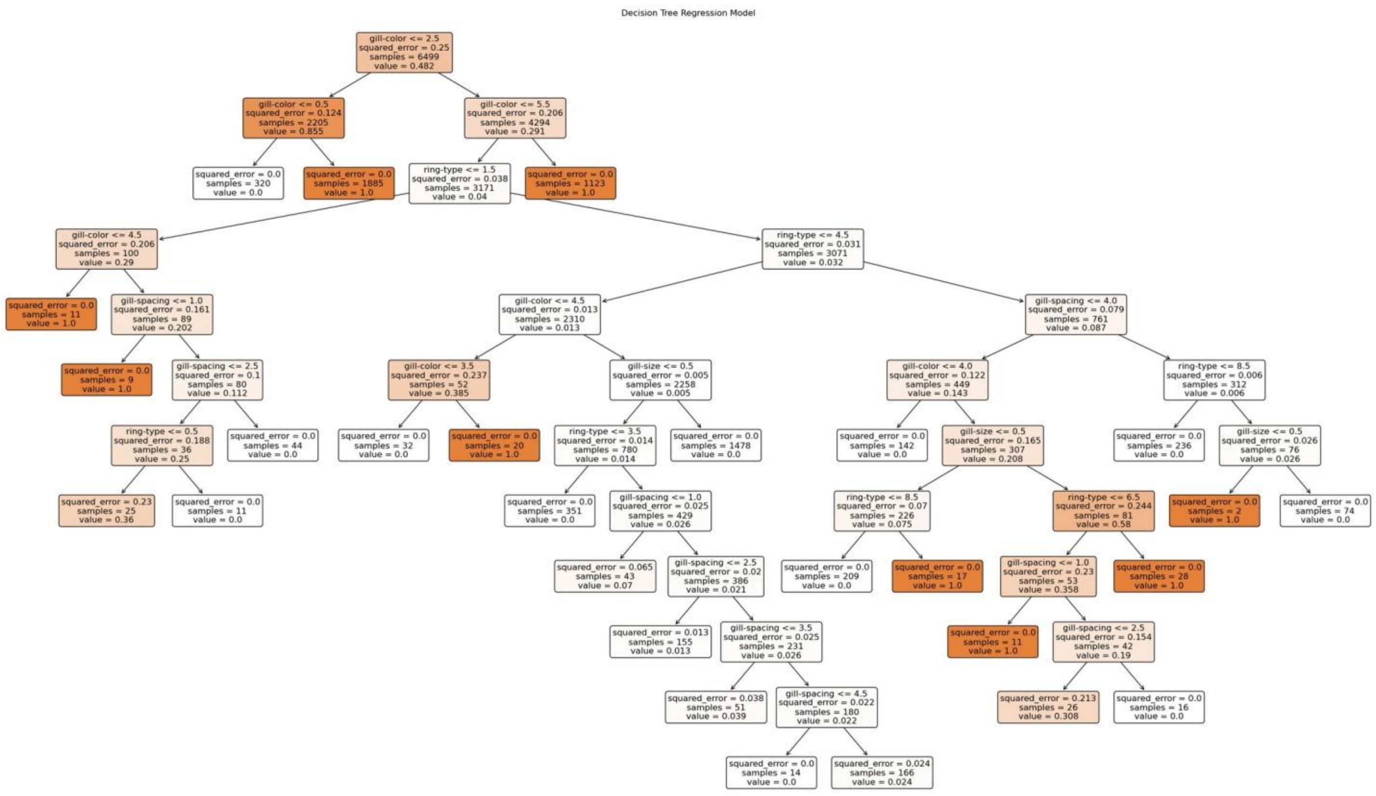
Decision-Tree Regression

```
model_tree = DecisionTreeRegressor(random_state=42)
model_tree.fit(X_train, y_train)
y_pred_tree = model_tree.predict(X_test)

# Evaluation
print("Decision Tree Regression Evaluation:")
print(f"R-squared: {r2_score(y_test, y_pred_tree):.2f}")
print(f"MAE: {mean_absolute_error(y_test, y_pred_tree):.2f}")
print(f"MSE: {mean_squared_error(y_test, y_pred_tree):.2f}")
print(f"RMSE: {mean_squared_error(y_test, y_pred_tree, squared=False):.2f}")
```

```
Decision Tree Regression Evaluation:
R-squared: 0.98
MAE: 0.01
MSE: 0.00
RMSE: 0.07
```

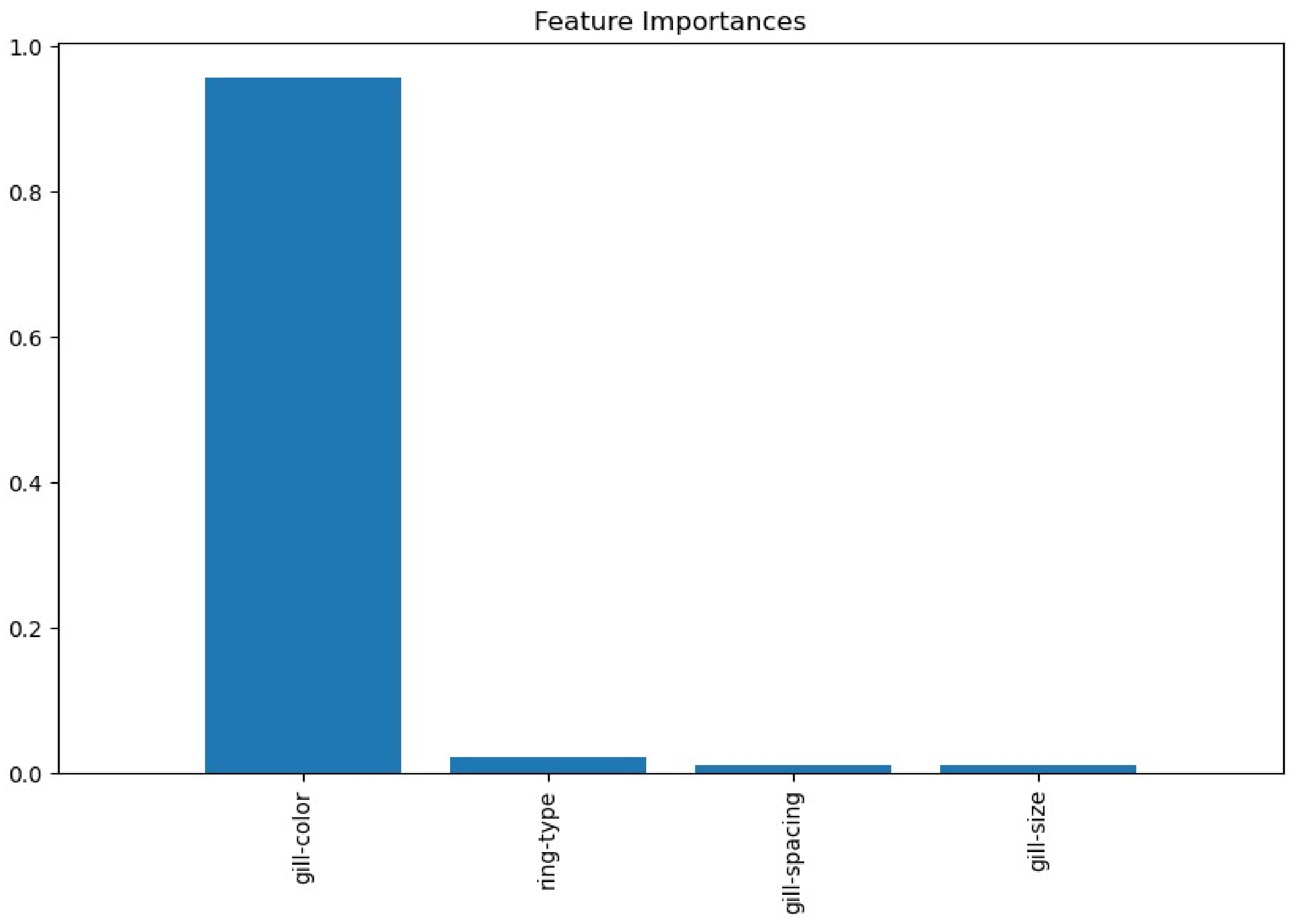
The decision tree plot

```
# Plotting the decision tree
plt.figure(figsize=(35, 20))
plot_tree(model_tree, feature_names=features, filled=True, rounded=True,
fontsize=12)
plt.title('Decision Tree Regression Model')
plt.show()
```


Decision Tree Regression Model
gill-color <= 2.5 squared_error = 0.25 samples = 6499 value = 0.482
gill-color <= 0.5 squared_error = 0.124 samples = 2205 value = 0.855
gill-color <= 5.5 squared_error = 0.206 samples = 4294 value = 0.291
squared_error = 0.0 samples = 320 value = 0.0
squared_error = 0.0 samples = 1885 value = 1.0
ring-type <= 1.5 squared_error = 0.038 samples = 3171 value = 0.04
squared_error = 0.0 samples = 1123 value = 1.0
gill-color <= 4.5 squared_error = 0.206 samples = 100 value = 0.29
ring-type <= 4.5 squared_error = 0.031 samples = 3071 value = 0.032
squared_error = 0.0 samples = 11 value = 1.0
gill-spacing <= 1.0 squared_error = 0.161 samples = 89 value = 0.202
gill-color <= 4.5 squared_error = 0.013 samples = 2310 value = 0.013
gill-spacing <= 4.0 squared_error = 0.079 samples = 761 value = 0.087
squared_error = 0.0 samples = 9 value = 1.0
gill-spacing <= 2.5 squared_error = 0.1 samples = 80 value = 0.112
gill-color <= 3.5 squared_error = 0.237 samples = 52 value = 0.385
gill-size <= 0.5 squared_error = 0.005 samples = 2258 value = 0.005
gill-color <= 4.0 squared_error = 0.122 samples = 449 value = 0.143
ring-type <= 8.5 squared_error = 0.006 samples = 312 value = 0.006
ring-type <= 0.5 squared_error = 0.188 samples = 36 value = 0.25
squared_error = 0.0 samples = 44 value = 0.0
squared_error = 0.0 samples = 32 value = 0.0
squared_error = 0.0 samples = 20 value = 1.0
ring-type <= 3.5 squared_error = 0.014 samples = 780 value = 0.014
squared_error = 0.0 samples = 1478 value = 0.0
squared_error = 0.0 samples = 142 value = 0.0
gill-size <= 0.5 squared_error = 0.165 samples = 307 value = 0.208
squared_error = 0.0 samples = 236 value = 0.0
gill-size <= 0.5 squared_error = 0.026 samples = 76 value = 0.026
squared_error = 0.23 samples = 25 value = 0.36
squared_error = 0.0 samples = 11 value = 0.0
squared_error = 0.0 samples = 351 value = 0.0
gill-spacing <= 1.0 squared_error = 0.025 samples = 429 value = 0.026
ring-type <= 8.5 squared_error = 0.07 samples = 226 value = 0.075
ring-type <= 6.5 squared_error = 0.244 samples = 81 value = 0.58
squared_error = 0.0 samples = 2 value = 1.0
squared_error = 0.0 samples = 74 value = 0.0
squared_error = 0.065 samples = 43 value = 0.07
gill-spacing <= 2.5 squared_error = 0.02 samples = 386 value = 0.021
squared_error = 0.0 samples = 209 value = 0.0
squared_error = 0.0 samples = 17 value = 1.0
gill-spacing <= 1.0 squared_error = 0.23 samples = 53 value = 0.358
squared_error = 0.0 samples = 28 value = 1.0
squared_error = 0.013 samples = 155 value = 0.013
gill-spacing <= 3.5 squared_error = 0.025 samples = 231 value = 0.026
squared_error = 0.0 samples = 11 value = 1.0
gill-spacing <= 2.5 squared_error = 0.154 samples = 42 value = 0.19
squared_error = 0.038 samples = 51 value = 0.039
gill-spacing <= 4.5 squared_error = 0.022 samples = 180 value = 0.022
squared_error = 0.213 samples = 26 value = 0.308
squared_error = 0.0 samples = 16 value = 0.0
squared_error = 0.0 samples = 14 value = 0.0
squared_error = 0.024 samples = 166 value = 0.024

Feature Importances Plot

```
importances = model_tree.feature_importances_
indices = np.argsort(importances)[::-1]

plt.figure(figsize=(10, 6))
plt.title('Feature Importances')
plt.bar(range(X_train.shape[1]), importances[indices], align='center')
plt.xticks(range(X_train.shape[1]), [features[i] for i in indices],
rotation=90)
plt.xlim([-1, X_train.shape[1]])
plt.show()
```



Feature and target for classification model

```
classification_features = ['cap-shape', 'cap-color', 'bruises', 'odor']
classification_target = 'class'

classification_X = df[classification_features]
classification_y = df[classification_target]

X_train, X_test, y_train, y_test = train_test_split(classification_X,
classification_y, test_size=0.2, random_state=42)
```

Naive Bayes Classification

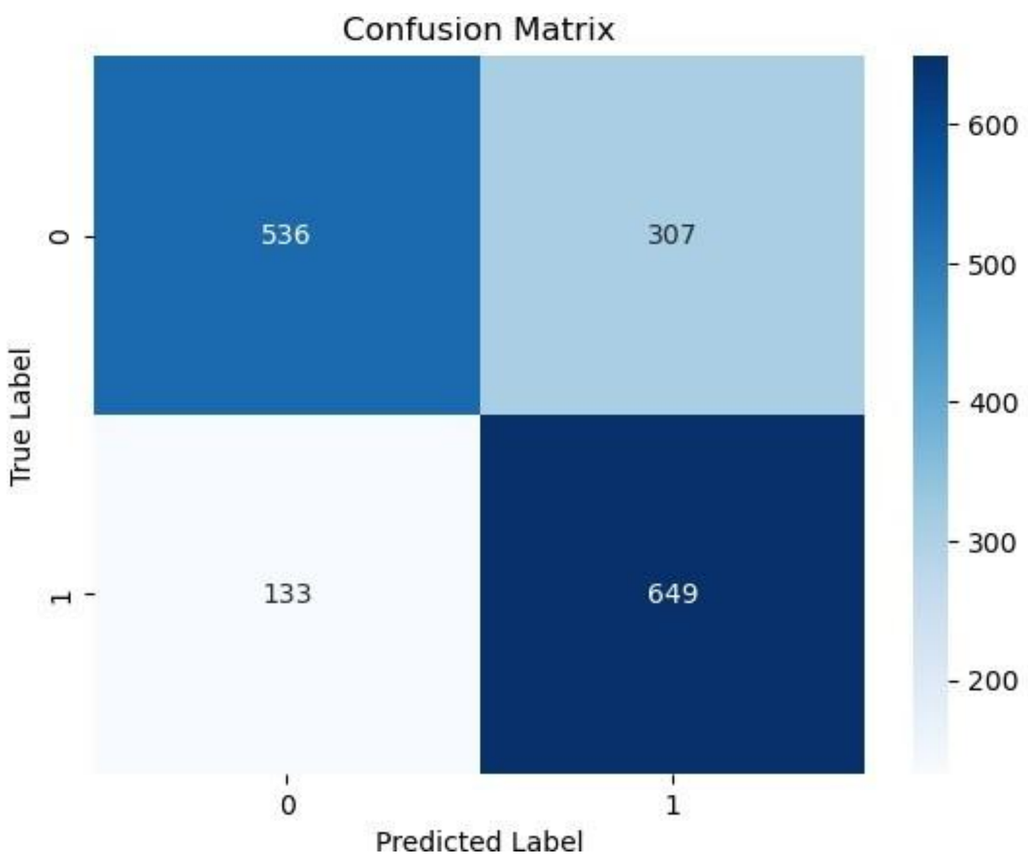
```
# Naive Bayes Classification
model_nb = GaussianNB()
model_nb.fit(X_train, y_train)
y_pred_nb = model_nb.predict(X_test)

print("\nNaive Bayes Classification Evaluation:")
print(f"Accuracy Score:{ accuracy_score(y_test, y_pred_nb):.2f}")
print(f"Precision: {precision_score(y_test, y_pred_nb,
average='weighted'):.2f}")
print(f"Recall: {recall_score(y_test, y_pred_nb, average='weighted'):.2f}")
print(f"F1 Score: {f1_score(y_test, y_pred_nb, average='weighted'):.2f}")
```

```
Naive Bayes Classification Evaluation:
Accuracy Score:0.73
Precision: 0.74
```

```
Recall: 0.73
F1 Score: 0.73
```

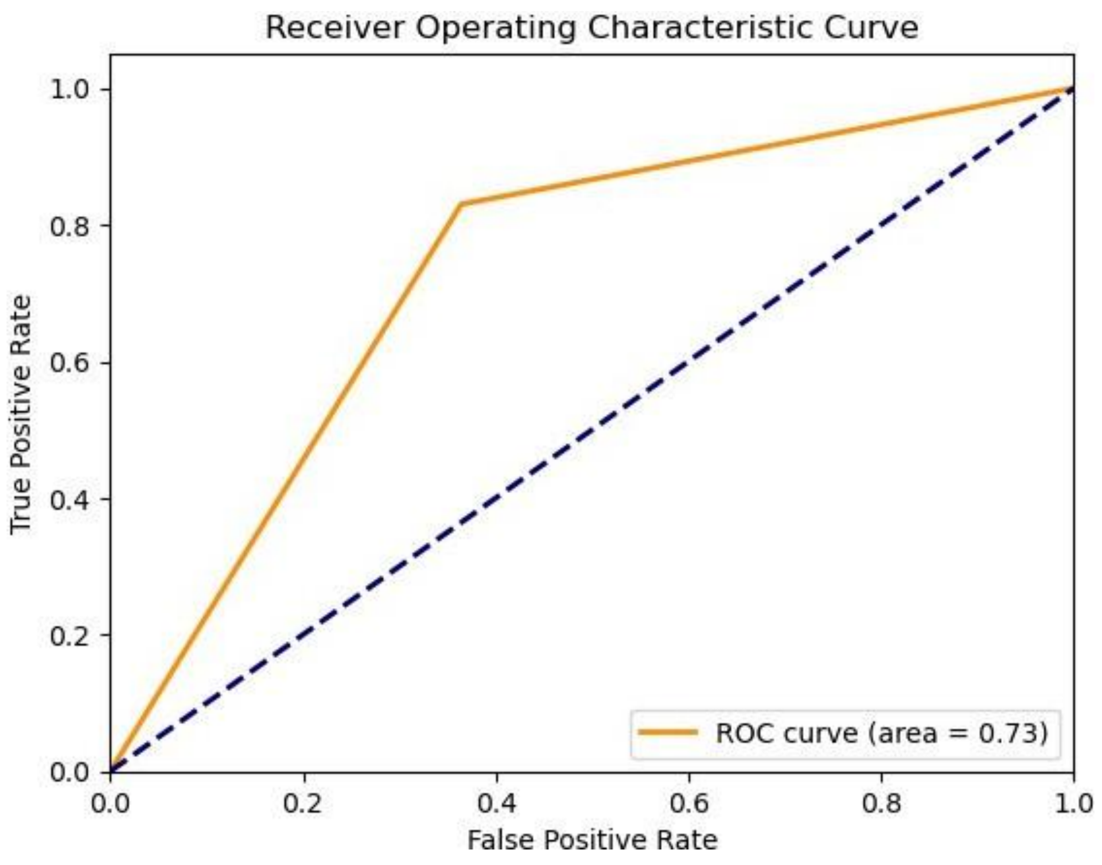
Confusion Matrix for Naive Bayes classification model.

```
conf_matrix = confusion_matrix(y_test, y_pred_nb)  # Replace y_pred_rf with
your predictions

plt.figure()
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix')
plt.show()
```



The ROC curve (Receiver Operating Characteristic curve) for Naive Bayes classification model.

```
fpr, tpr, _ = roc_curve(y_test, y_pred_nb)
roc_auc = auc(fpr, tpr)

plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)'
% roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic Curve')
plt.legend(loc='lower right')
plt.show()
```



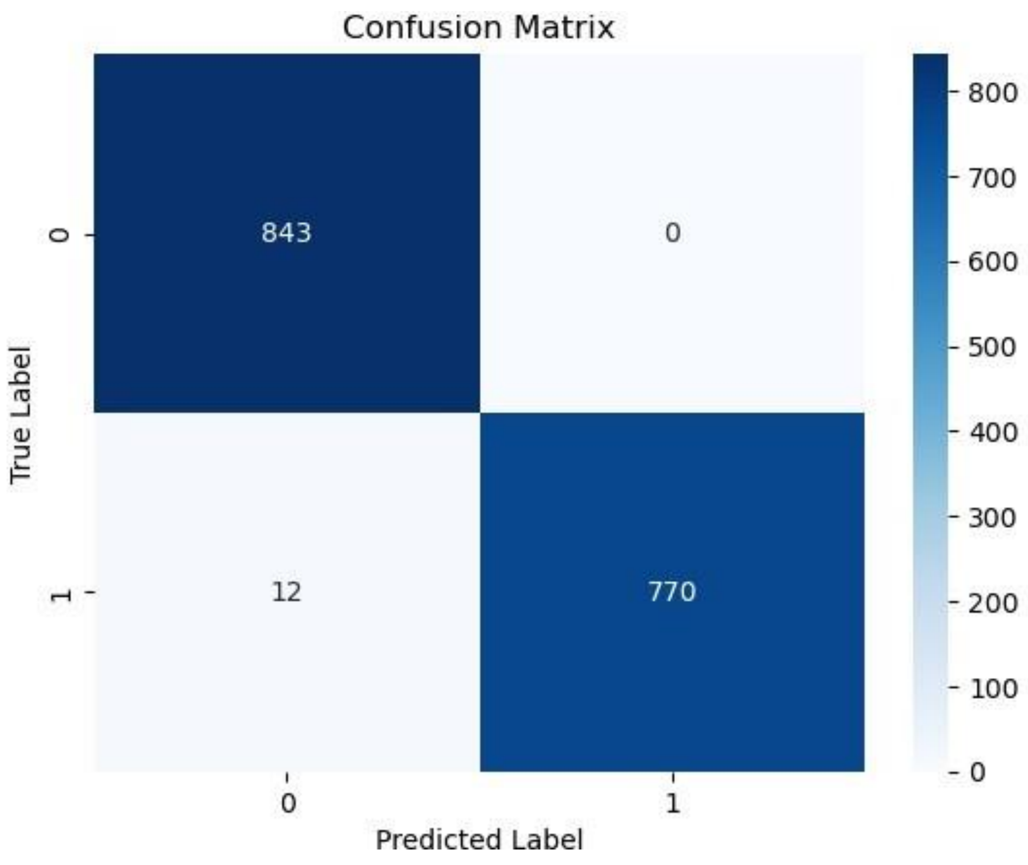
Random Forest Classification

```
# Random Forest Classification
model_rf = RandomForestClassifier(n_estimators=100, random_state=42)
model_rf.fit(X_train, y_train)
y_pred_rf = model_rf.predict(X_test)

print("\nRandom Forest Classification Evaluation:")
print(f"Accuracy Score:{ accuracy_score(y_test, y_pred_rf):.2f}")
print(f"Precision: {precision_score(y_test, y_pred_rf,
average='weighted'):.2f}")
print(f"Recall: {recall_score(y_test, y_pred_rf, average='weighted'):.2f}")
print(f"F1 Score: {f1_score(y_test, y_pred_rf, average='weighted'):.2f}")
```

```
Random Forest Classification Evaluation:
Accuracy Score:0.99
Precision: 0.99
Recall: 0.99
F1 Score: 0.99
```

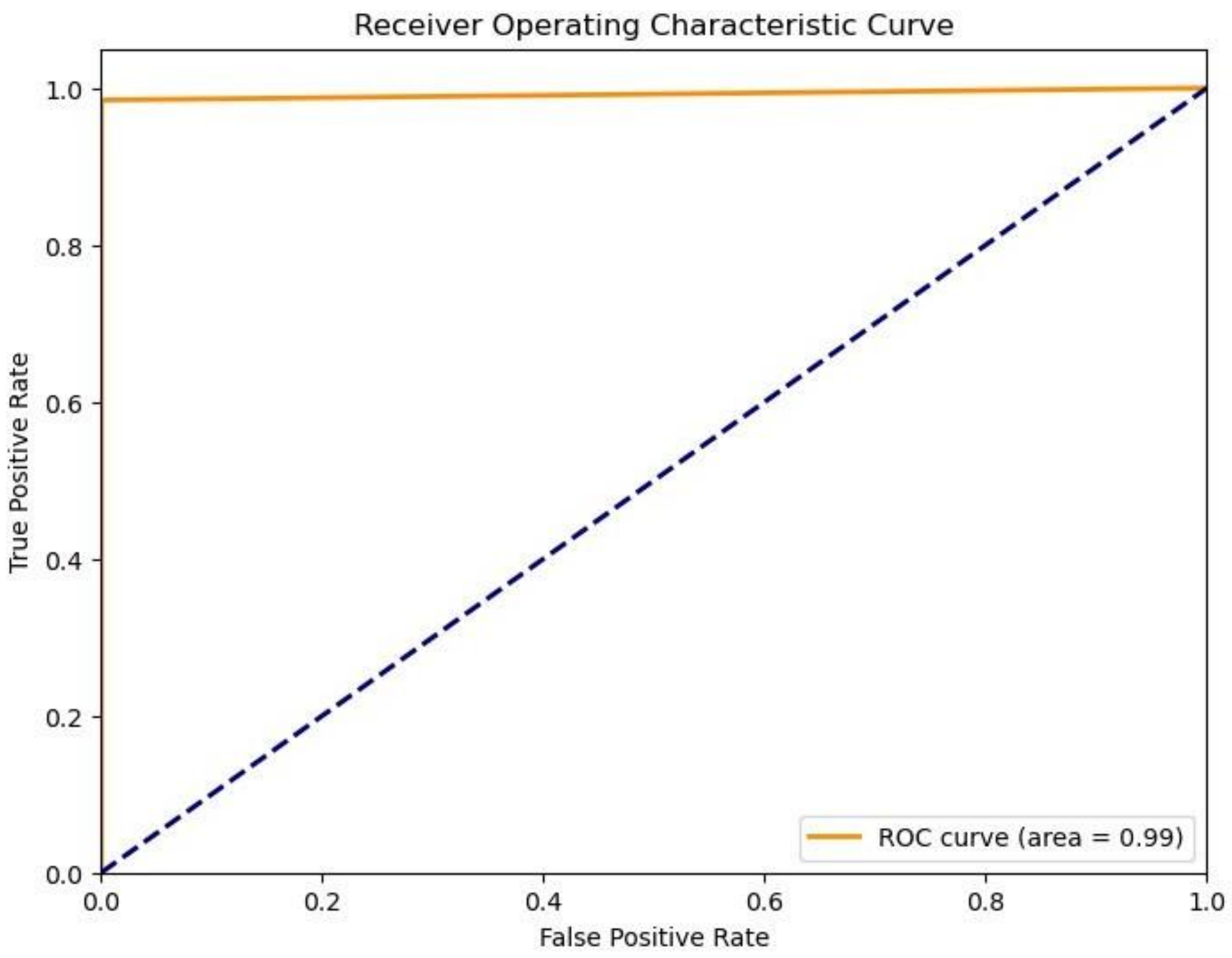
Confusion Matrix for RandomForest classification model.

```
conf_matrix = confusion_matrix(y_test, y_pred_rf)

plt.figure()
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix')
plt.show()
```



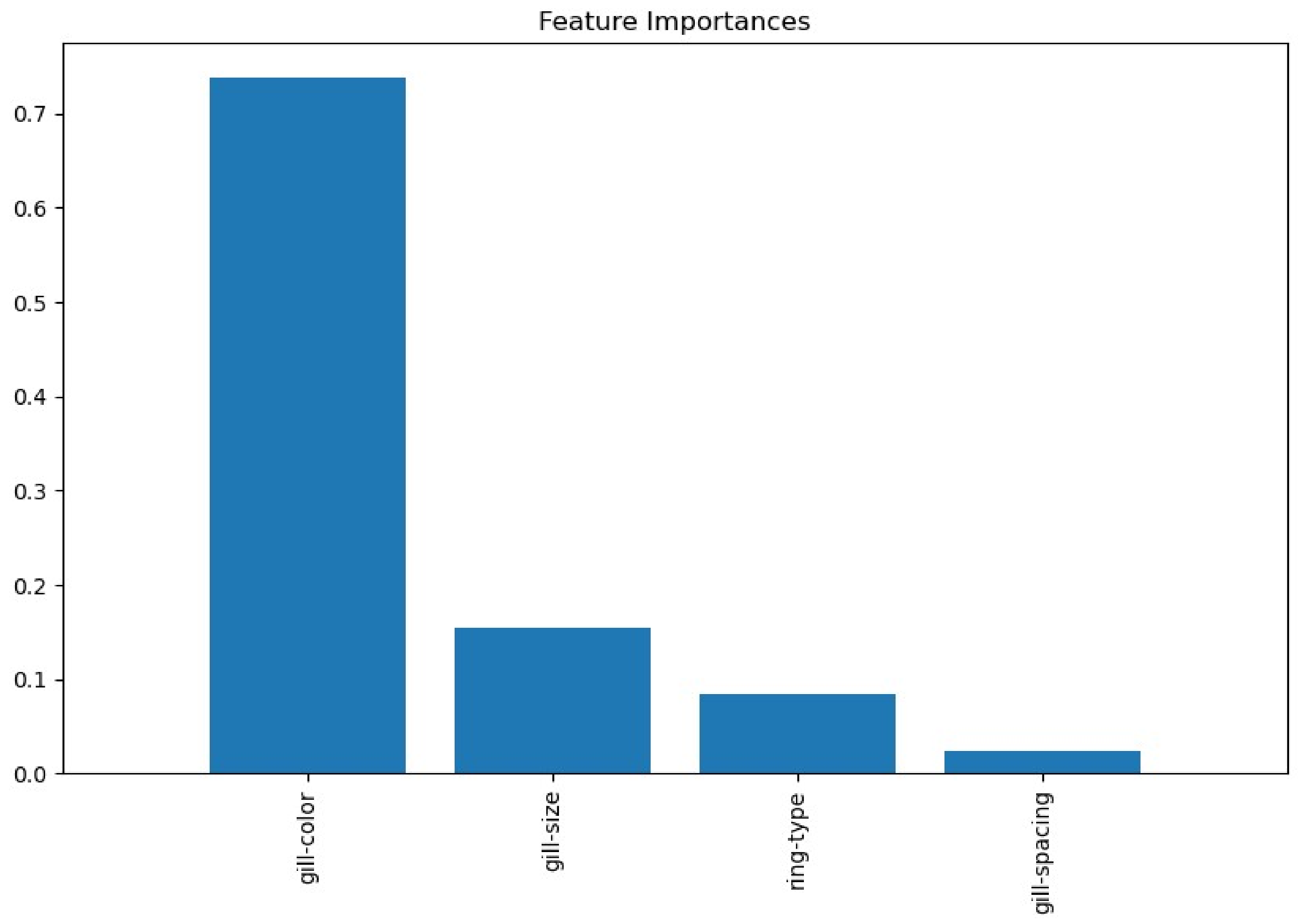
The ROC curve (Receiver Operating Characteristic curve) for RandomForest classification model.

```
fpr, tpr, _ = roc_curve(y_test, y_pred_rf)
roc_auc = auc(fpr, tpr)

plt.figure(figsize=(8,6))
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)'
% roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic Curve')
plt.legend(loc='lower right')
plt.show()
```



Feature Importances Plot (For tree-based classifiers like Random Forest, you can plot the importance of features.)

```
importances = model_rf.feature_importances_
indices = np.argsort(importances)[::-1]

plt.figure(figsize=(10, 6))
plt.title('Feature Importances')
plt.bar(range(X_train.shape[1]), importances[indices], align='center')
plt.xticks(range(X_train.shape[1]), [features[i] for i in indices],
rotation=90)
plt.xlim([-1, X_train.shape[1]])
plt.show()
```


Feature Importances
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
gill-color
gill-size
ring-type
gill-spacing