IMPORTING THE ESSENTIAL LIBRARIES

In [14]:
```python
# Importing essential libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Machine Learning libraries
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Choose your machine learning algorithm(s)
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB

# Model evaluation
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.model_selection import cross_val_score

# For deep learning (if needed)
import tensorflow as tf
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Dense

# Ignore warnings (optional)
import warnings
warnings.filterwarnings('ignore')

# Set random seed for reproducibility (optional)
np.random.seed(0)
```

In [ ]:
```python
# pip install numpy pandas matplotlib seaborn scikit-learn tensorflow
```

In [12]:
```python
# Importing essential libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Machine Learning libraries for regression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score

# Data visualization settings (optional)
%matplotlib inline
plt.style.use('ggplot')

# Ignore warnings (optional)
import warnings
warnings.filterwarnings('ignore')

# Set random seed for reproducibility (optional)
np.random.seed(0)
```

SELECTING THE DATA PATH

In [10]:
```python
file_path = r'C:\Users\engin\Downloads\wineQT.csv'
```

DEFINING THE DATA

In [11]:
```python
# Read the CSV file into a DataFrame
wineQT = pd.read_csv(file_path)

# Display the first few rows of the DataFrame to verify the import
wineQT.head()
```

Out[11]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality	Id
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5	0
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5	1
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5	2
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6	3
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5	4

Exploratry Data Analysis

In [15]:
```python
wineQT.shape
```

Out[15]:
```
(1143, 13)
```

In [16]:
```python
wineQT.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1143 entries, 0 to 1142
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         1143 non-null   float64
 1   volatile acidity      1143 non-null   float64
 2   citric acid           1143 non-null   float64
 3   residual sugar        1143 non-null   float64
 4   chlorides             1143 non-null   float64
 5   free sulfur dioxide   1143 non-null   float64
 6   total sulfur dioxide  1143 non-null   float64
 7   density               1143 non-null   float64
 8   pH                    1143 non-null   float64
 9   sulphates             1143 non-null   float64
 10  alcohol               1143 non-null   float64
 11  quality               1143 non-null   int64  
 12  Id                    1143 non-null   int64  
dtypes: float64(11), int64(2)
memory usage: 116.2 KB
```

In [17]:
```python
wineQT.isnull().sum()
```

Out[17]:
```
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
Id                      0
dtype: int64
```

In [18]:
```python
wineQT.describe()
```

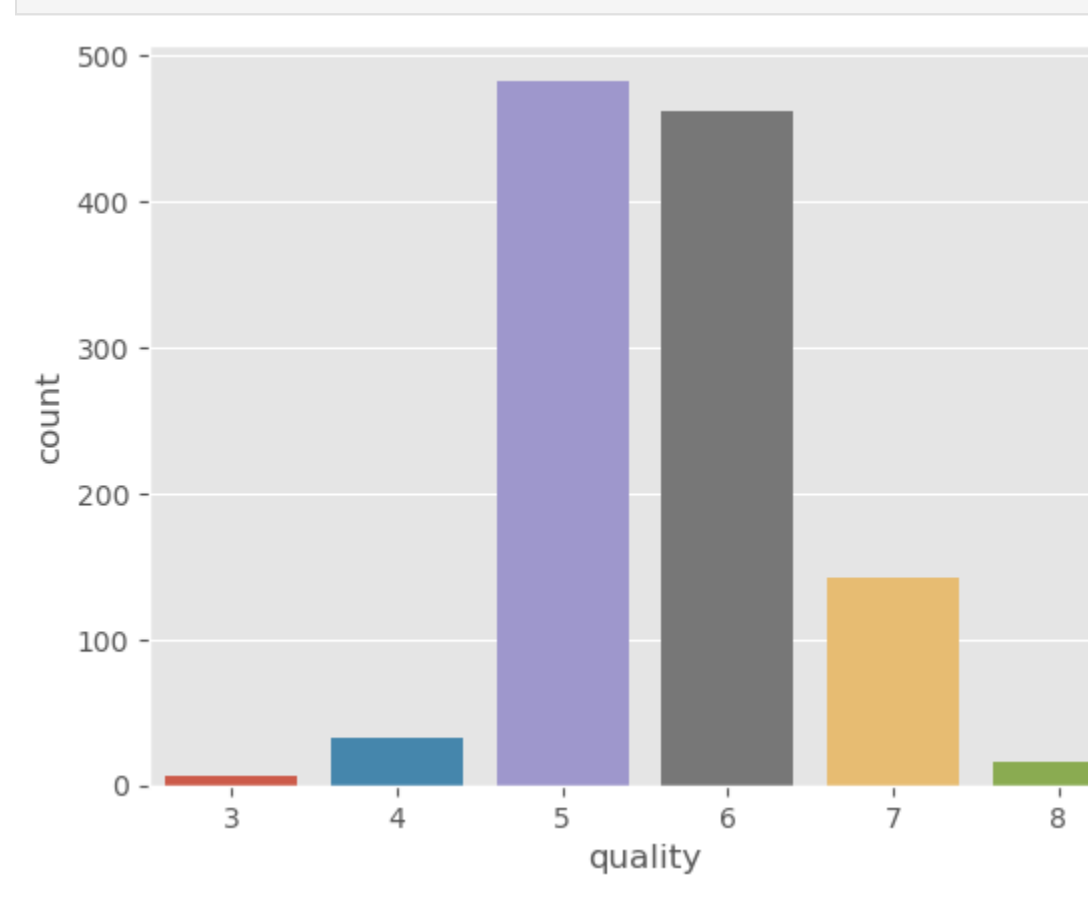
Out[18]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality	Id
count	1143.000000	1143.000000	1143.000000	1143.000000	1143.000000	1143.000000	1143.000000	1143.000000	1143.000000	1143.000000	1143.000000	1143.000000	1143.000000
mean	8.311111	0.531339	0.268364	2.532152	0.086933	15.615486	45.914698	0.996730	3.311015	0.657708	10.442111	5.657043	804.969379
std	1.747595	0.179633	0.196686	1.355917	0.047267	10.250486	32.782130	0.001925	0.156664	0.170399	1.082196	0.805824	463.997116
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	0.990070	2.740000	0.330000	8.400000	3.000000	0.000000
25%	7.100000	0.392500	0.090000	1.900000	0.070000	7.000000	21.000000	0.995570	3.205000	0.550000	9.500000	5.000000	411.000000
50%	7.900000	0.520000	0.250000	2.200000	0.079000	13.000000	37.000000	0.996680	3.310000	0.620000	10.200000	6.000000	794.000000
75%	9.100000	0.640000	0.420000	2.600000	0.090000	21.000000	61.000000	0.997845	3.400000	0.730000	11.100000	6.000000	1209.500000
max	15.900000	1.580000	1.000000	15.500000	0.611000	68.000000	289.000000	1.003690	4.010000	2.000000	14.900000	8.000000	1597.000000

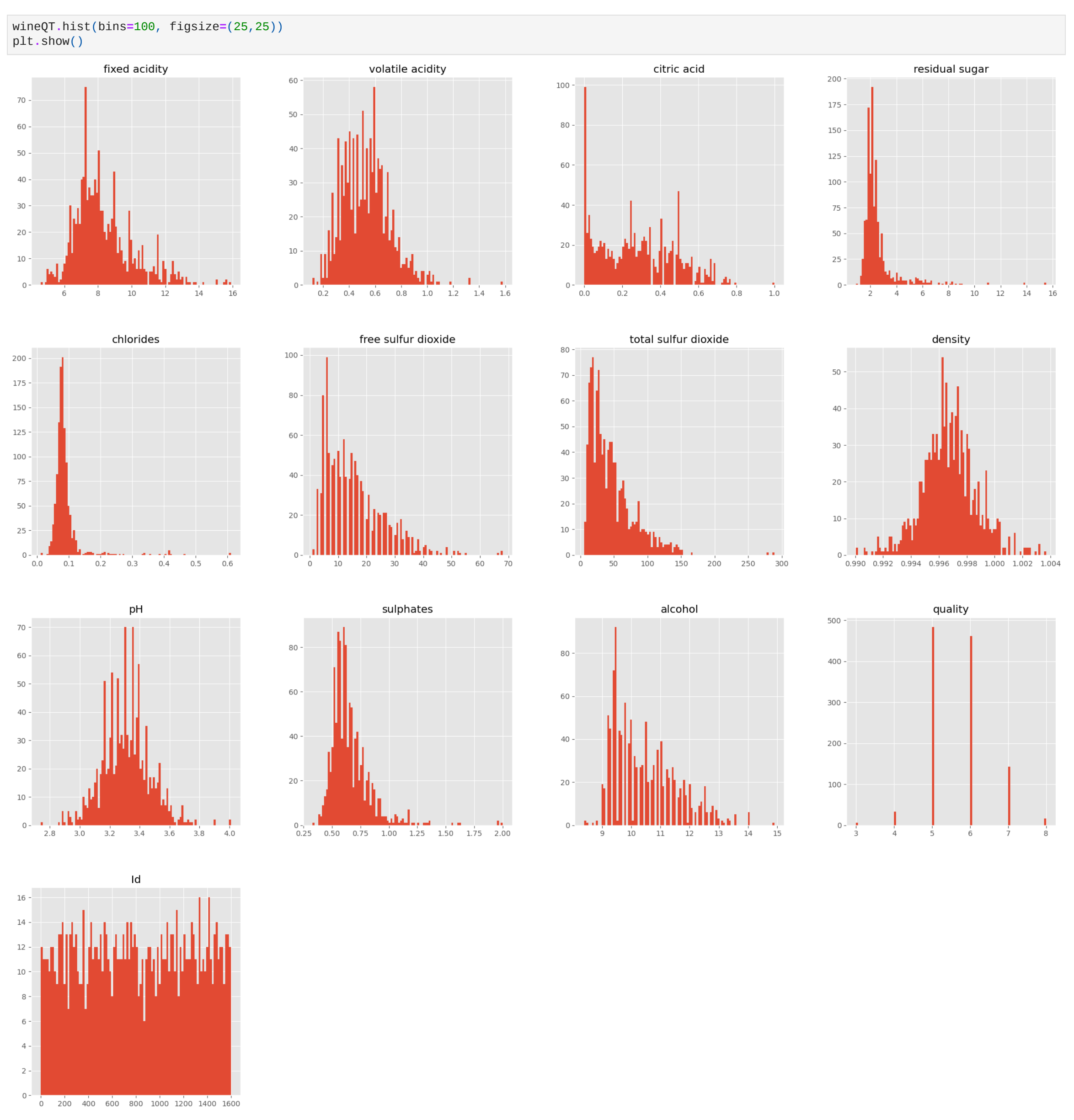
In [21]:
```python
wineQT['quality'].value_counts()
```

Out[21]:
```
quality
5    483
6    462
7    143
4     33
8     16
3      6
Name: count, dtype: int64
```

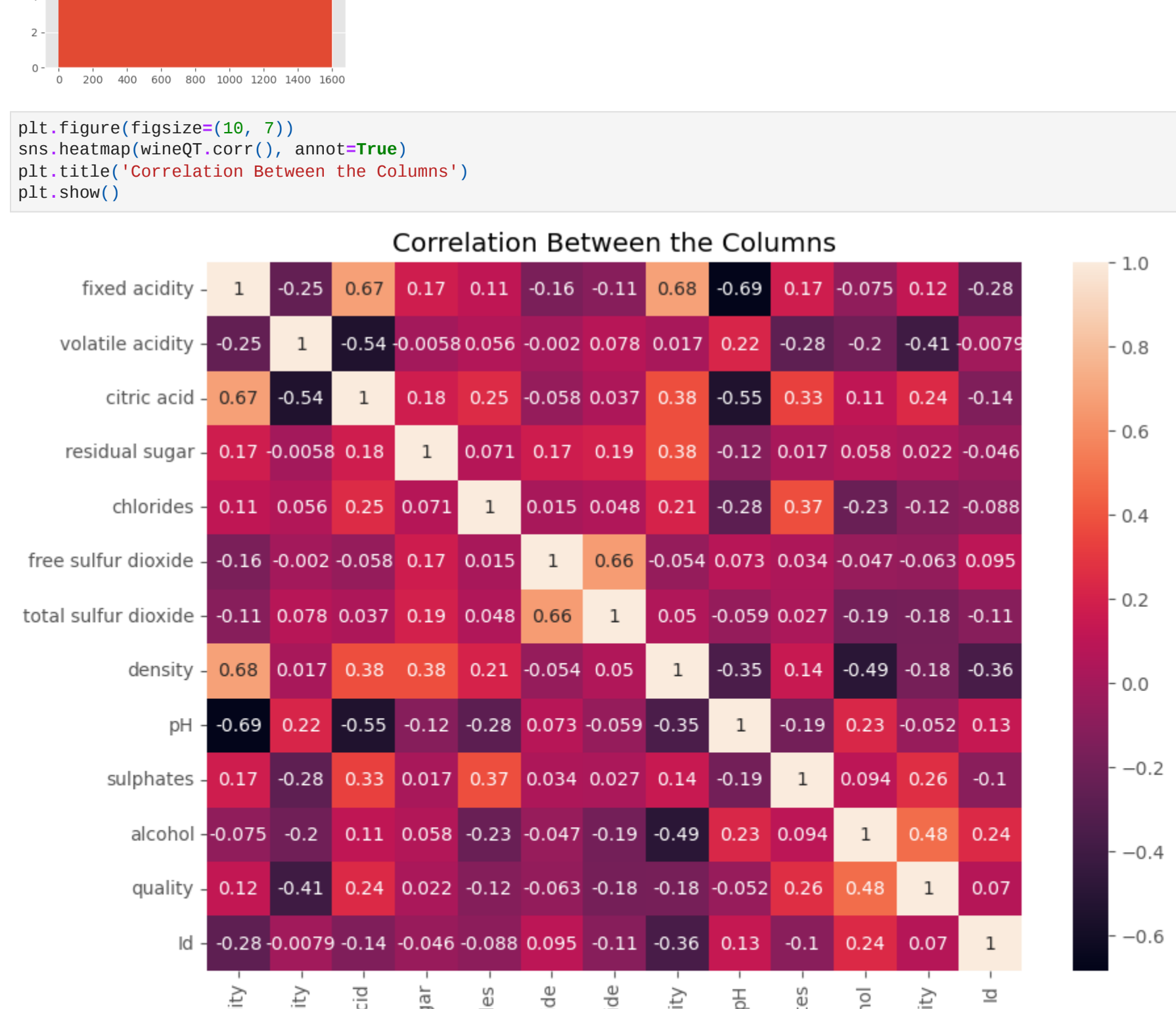
In [35]:
```python
sns.countplot(data=wineQT, x='quality')
plt.show()
```



In [42]:
```python
wineQT.hist(bins=100, figsize=(25,25))
plt.show()
```



In [46]:
```python
plt.figure(figsize=(10, 7))
sns.heatmap(wineQT.corr(), annot=True)
plt.title('Correlation Between the Columns')
plt.show()
```



In [50]:
```python
wineQT.corr()['quality'].sort_values()
```

Out[50]:
```
volatile acidity       -0.407394
total sulfur dioxide   -0.183339
density                -0.175208
chlorides              -0.124085
free sulfur dioxide    -0.063260
pH                     -0.052453
residual sugar          0.022002
Id                      0.069708
fixed acidity           0.121970
citric acid             0.240821
sulphates               0.257710
alcohol                 0.484866
quality                 1.000000
Name: quality, dtype: float64
```

In [52]:
```python
sns.barplot(x=wineQT['quality'], y=wineQT['alcohol'])
plt.title('Bar Plot of Quality vs Alcohol')
plt.show()
```

Data Processing

In [54]:
```python
wineQT['quality'] = wineQT.apply(lambda x: 1 if x['quality'] >= 7 else 0, axis=1)
```

In [55]:
```python
wineQT['quality'].value_counts()
```

Out[55]:
```
quality
0    984
1    159
Name: count, dtype: int64
```

In [56]:
```python
x = wineQT.drop('quality', axis=1)
y = wineQT['quality']
```

In [57]:
```python
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size= 0.3,random_state= 42)
```

In [58]:
```python
print ("x_train", x_train.shape)
print ("y_train", y_train.shape)
print ("x_test", x_test.shape)
print ("y_test", x_test.shape)
```

```
x_train (800, 12)
y_train (800,)
x_test (343, 12)
y_test (343, 12)
```

Model Training

logistic regression model

In [61]:
```python
logreg = LogisticRegression()  # Note the capital 'L' in LogisticRegression
logreg.fit(x_train, y_train)  # Corrected variable names 'x_train' and 'y_train'
logreg_pred = logreg.predict(x_test)
logreg_acc = accuracy_score(y_test, logreg_pred)  # Reversed the order of arguments

print("Test accuracy is: {:.2f}%".format(logreg_acc * 100))  # Fixed the format string
```

```
Test accuracy is: 88.92%
```

In [63]:
```python
print(classification_report(y_test, logreg_pred))
```

```
              precision    recall  f1-score   support

           0       0.91      0.97      0.94       298
           1       0.65      0.33      0.44        45

    accuracy                           0.89       343
   macro avg       0.78      0.65      0.69       343
weighted avg       0.87      0.89      0.87       343
```

In [66]:
```python
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

plt.style.use('classic')
cm = confusion_matrix(y_test, logreg_pred, labels=logreg.classes_)  # Fixed 'labels' and 'logreg.classes_'
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=logreg.classes_)  # Fixed 'display_labels'

print("TN", cm[0, 0])  # Fixed indices
print("FN", cm[1, 0])  # Fixed indices
print("TP", cm[1, 1])  # Fixed indices
print("FP", cm[0, 1])  # Fixed indices
```

```
TN 290
FN 30
TP 15
FP 8
```

In [67]:
```python
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

plt.style.use('classic')
cm = confusion_matrix(y_test, logreg_pred, labels=logreg.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=logreg.classes_)

# Plot the confusion matrix as a heatmap
disp.plot(cmap='Blues', values_format='d')

plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```

Decision Tree Model

In [68]:
```python
# Create a Decision Tree classifier
dt_classifier = DecisionTreeClassifier(random_state=0)
```

In [69]:
```python
# Train the model on the training data
dt_classifier.fit(x_train, y_train)
```

Out[69]:
```
DecisionTreeClassifier
DecisionTreeClassifier(random_state=0)
```

In [70]:
```python
# Make predictions on the test data
y_pred = dt_classifier.predict(x_test)
```

In [71]:
```python
# Calculate and print accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.8629737609329446
```

In [72]:
```python
# Generate and print the classification report
class_report = classification_report(y_test, y_pred)
print("Classification Report:\n", class_report)
```

```
Classification Report:
               precision    recall  f1-score   support

           0       0.94      0.90      0.92       298
           1       0.48      0.60      0.53        45

    accuracy                           0.86       343
   macro avg       0.71      0.75      0.73       343
weighted avg       0.88      0.86      0.87       343
```

In [73]:
```python
# Generate and print the confusion matrix
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", cm)
```

```
Confusion Matrix:
 [[269  29]
 [ 18  27]]
```

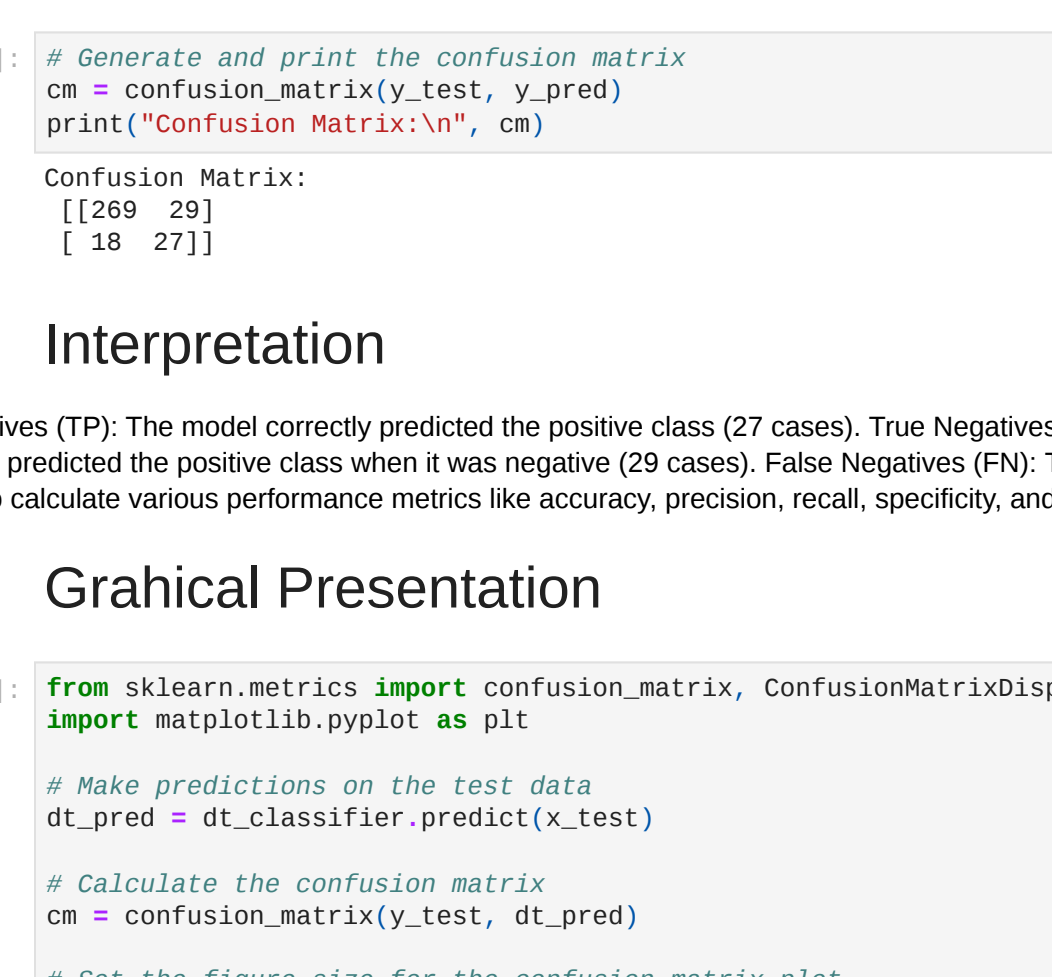
Interpretation

True Positives (TP): The model correctly predicted the positive class (27 cases). True Negatives (TN): The model correctly predicted the negative class (269 cases). False Positives (FP): The model incorrectly predicted the positive class when it was negative (29 cases). False Negatives (FN): The model incorrectly predicted the negative class when it was positive (18 cases). These numbers can be used to calculate various performance metrics like accuracy, precision, recall, specificity, and the F1-score to evaluate your decision tree model's performance in a binary classification context.

Grahical Presentation

In [84]:
```python
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt

# Make predictions on the test data
dt_pred = dt_classifier.predict(x_test)

# Calculate the confusion matrix
cm = confusion_matrix(y_test, dt_pred)

# Set the figure size for the confusion matrix plot
plt.figure(figsize=(8, 6))

# Create a ConfusionMatrixDisplay
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=dt_classifier.classes_)

# Plot the confusion matrix as a heatmap
disp.plot(cmap='Blues', values_format='d')

# Set the title and axis labels
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')

# Show the plot
plt.show()
```

```
<Figure size 640x480 with 0 Axes>
```



LINEAR REGRESSION MODEL

In [89]:
```python
# Import necessary libraries
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

In [90]:
```python
# Load your dataset (replace 'your_data.csv' with your dataset file)
# Read the CSV file into a DataFrame
wineQT = pd.read_csv(file_path)

# Display the first few rows of the DataFrame to verify the import
wineQT.head()
```

Out[90]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality	Id
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5	0
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5	1
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5	2
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6	3
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5	4

In [95]:
```python
# Select your features (X) and target variable (y)
X = wineQT[['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar', 'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'densi
y = wineQT['quality']
```

In [96]:
```python
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [97]:
```python
# Create a Linear Regression model
model = LinearRegression()
```

In [98]:
```python
# Train the model on the training data
model.fit(X_train, y_train)
```

Out[98]:
```
LinearRegression
LinearRegression()
```

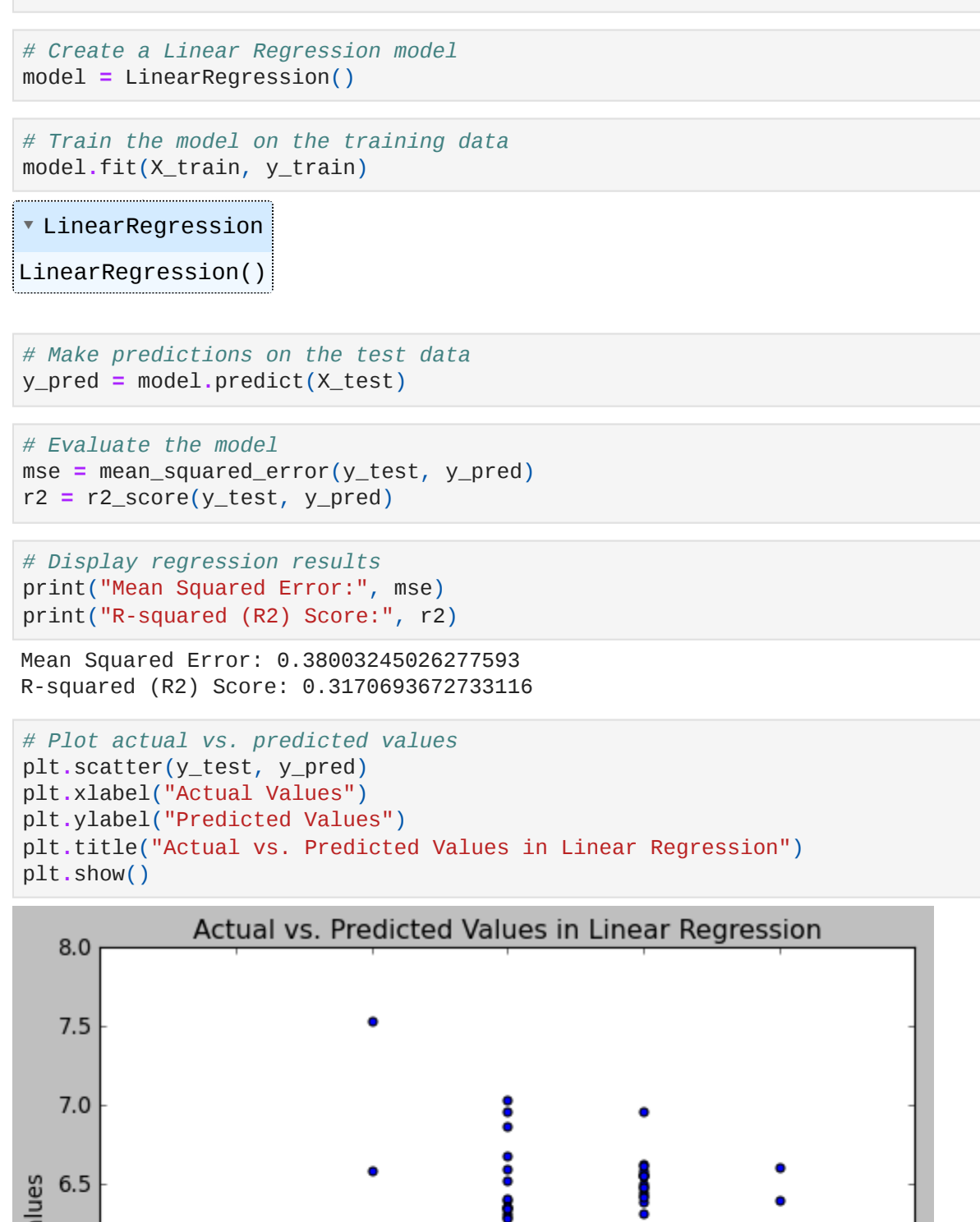
In [99]:
```python
# Make predictions on the test data
y_pred = model.predict(X_test)
```

In [100…
```python
# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

In [101…
```python
# Display regression results
print("Mean Squared Error:", mse)
print("R-squared (R2) Score:", r2)
```

```
Mean Squared Error: 0.38003245026277593
R-squared (R2) Score: 0.3170693672733116
```

In [102…
```python
# Plot actual vs. predicted values
plt.scatter(y_test, y_pred)
plt.xlabel("Actual Values")
plt.ylabel("Predicted Values")
plt.title("Actual vs. Predicted Values in Linear Regression")
plt.show()
```



Interpreting the results of the linear regression analysis:

The Mean Squared Error (MSE) is approximately 0.380, indicating the model's predictions are off by about 0.38 units squared, on average. Lower MSE values are generally preferred, but the assessment of whether this is good or bad depends on the specific context of the data and the problem.

The R-squared (R2) score is about 0.317, implying that roughly 31.7% of the variance in the target variable is explained by the model. An R2 score closer to 1 suggests a better model fit, while an R2 of 0 indicates that the model doesn't explain any of the variance.

In summary, the model seems to have some predictive power, but it only explains a portion of the variance in the target variable. There may be other unaccounted factors contributing to the variance. Further feature engineering, different model selection, or additional data collection could potentially enhance the model's performance.

In [ ]: