In [1]:
```python
# Importing libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import preprocessing
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn import svm
from sklearn import metrics
```

In [30]:
```python
# Loading the dataset
wine = pd.read_csv('winequality-red.csv')
wine
```

Out[30]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8	5
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8	5
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8	6
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5
...	...	...	...	...	...	...	...	...	...	...	...	...
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10.5	5
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	11.2	6
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0	6
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2	5
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	11.0	6

1599 rows × 12 columns

In [31]:
```python
# checking for null values

wine.isnull().sum()
```

Out[31]:
```
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
```

In [32]:
```
wine.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         1599 non-null   float64
 1   volatile acidity      1599 non-null   float64
 2   citric acid           1599 non-null   float64
 3   residual sugar        1599 non-null   float64
 4   chlorides             1599 non-null   float64
 5   free sulfur dioxide   1599 non-null   float64
 6   total sulfur dioxide  1599 non-null   float64
 7   density               1599 non-null   float64
 8   pH                    1599 non-null   float64
 9   sulphates             1599 non-null   float64
 10  alcohol               1599 non-null   float64
 11  quality               1599 non-null   int64  
dtypes: float64(11), int64(1)
memory usage: 150.0 KB
```

In [33]:
```
wine.describe()
```

Out[33]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000
mean	8.319637	0.527821	0.270976	2.538806	0.087467	15.874922	46.467792	0.996747
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.895324	0.001887
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	0.990070
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.000000	0.995600
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.000000	0.996750
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.000000	0.997835
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.000000	1.003690

Data Preprocessing

In [34]:
```
wine[['quality']].value_counts()
```

Out[34]:
```
quality
5          681
6          638
7          199
4           53
8           18
3           10
dtype: int64
```

In [35]:
```
wine['quality'] = wine['quality'].apply(lambda x: 1 if x>=7 else 0)
wine.rename(columns = {'quality' : 'good-quality'}, inplace = True)

# --  I am applying lambda function here to caterorize the wine quality into two parts.
# --  (1 = Good Quality Wine  &  0 = Bad Quality Wine)
lity' to 'good-quality'.
```

In [36]:
```
wine.head()
```

Out[36]:

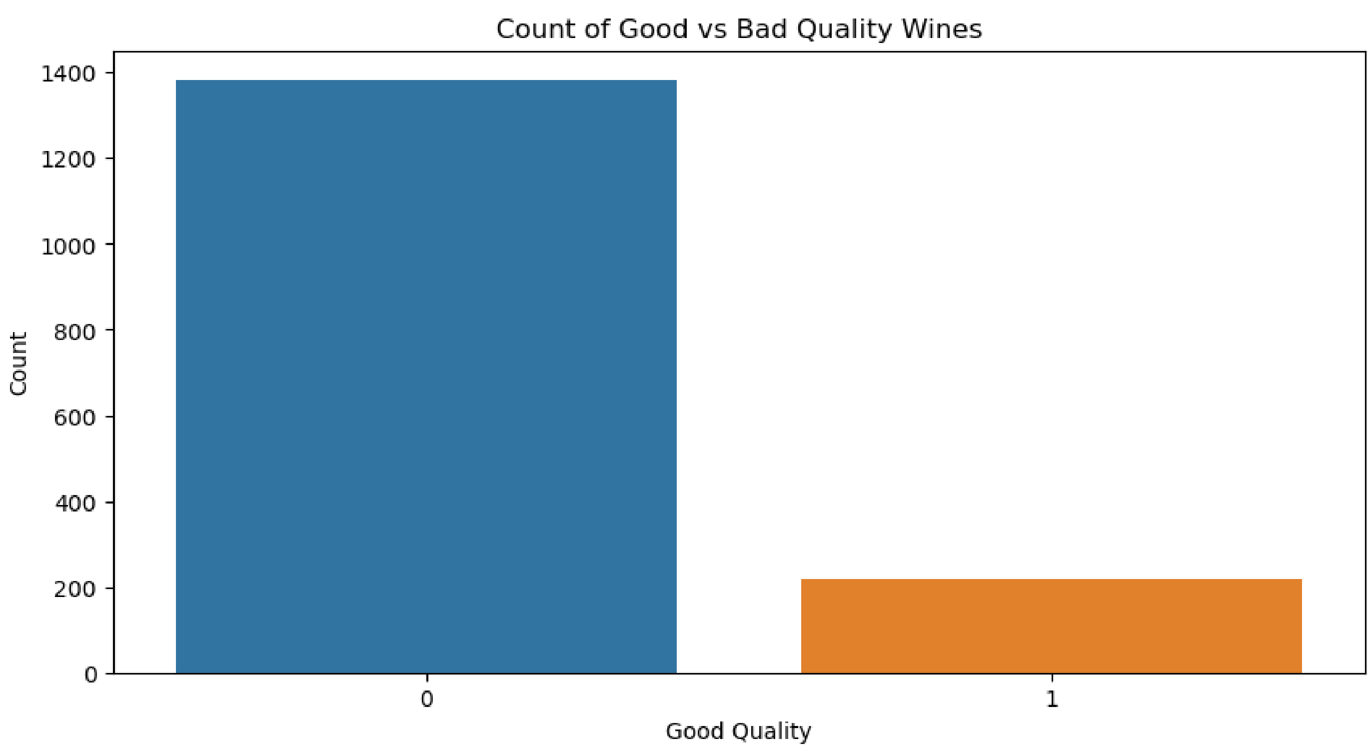
	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	good-quality
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	0
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	0
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0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

In [37]:
```
wine[['good-quality']].value_counts()
```

Out[37]:
```
good-quality
0               1382
1                217
dtype: int64
```

Exploratory Data Analysis

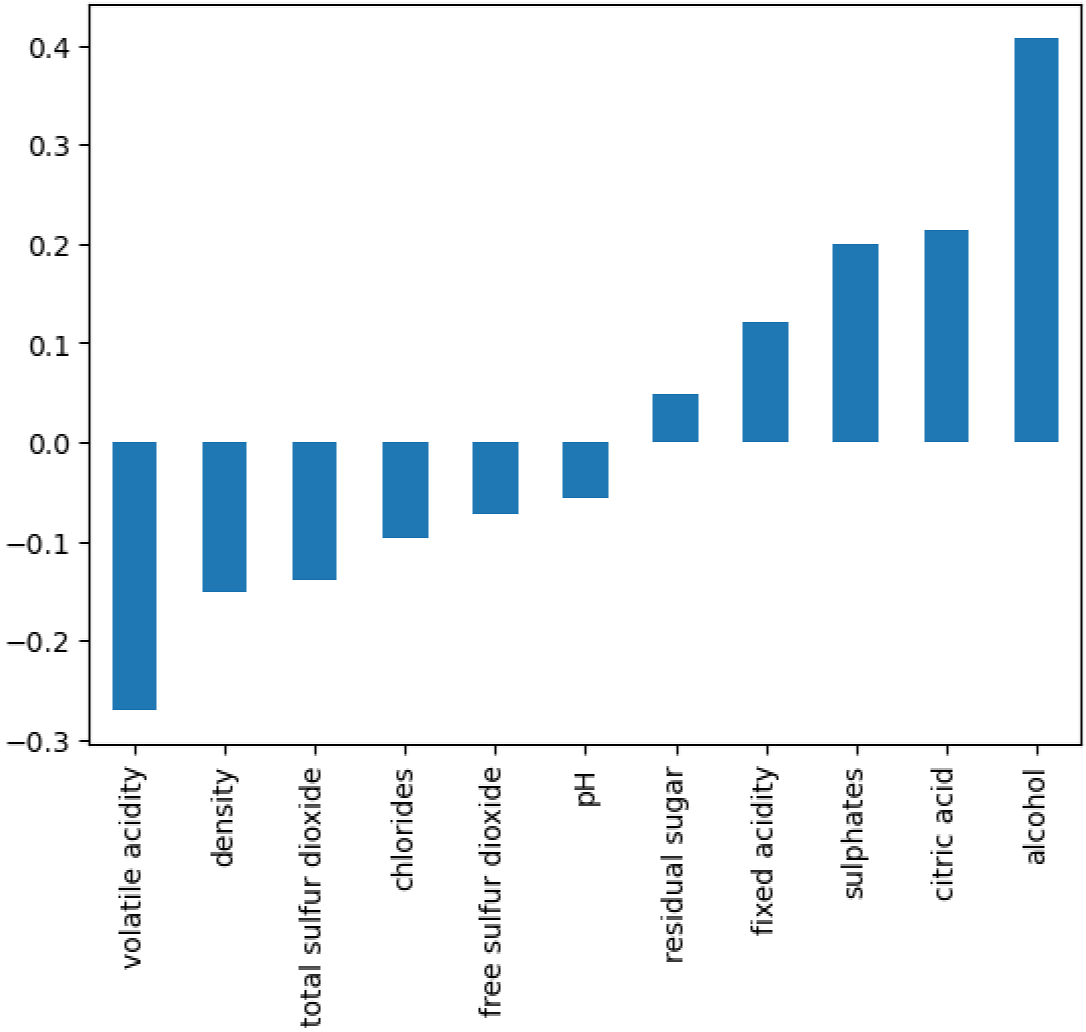
In [25]:
```
plt.figure(figsize=(10,5))
sns.countplot(x='good-quality', data=wine)
plt.xlabel('Good Quality')
plt.ylabel('Count')
plt.title('Count of Good vs Bad Quality Wines')
plt.show()
```



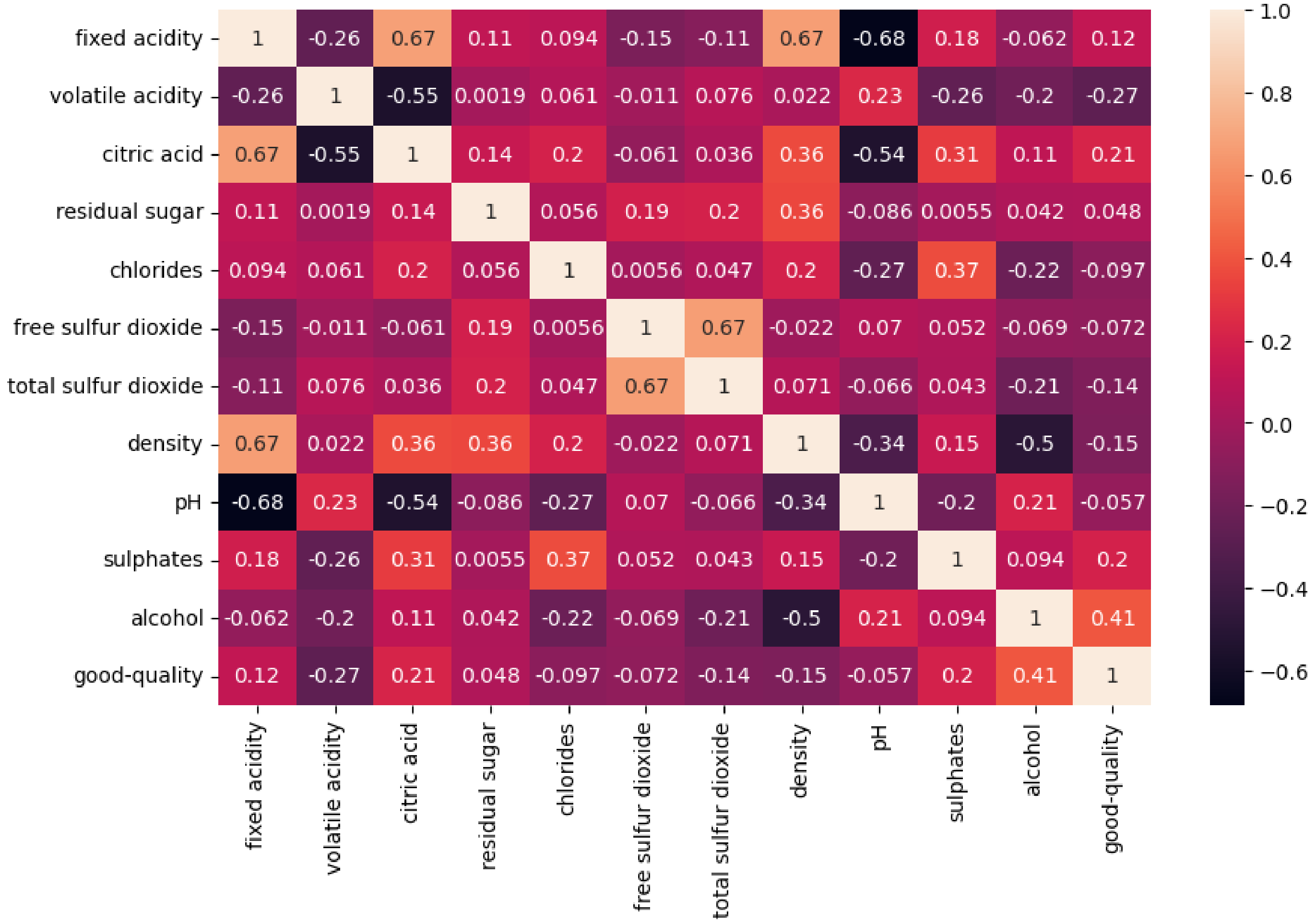
In [48]:
```
wine.corr()['good-quality'][:-1].sort_values().plot(kind = 'bar')
```

Out[48]:
```
<AxesSubplot:>
```

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

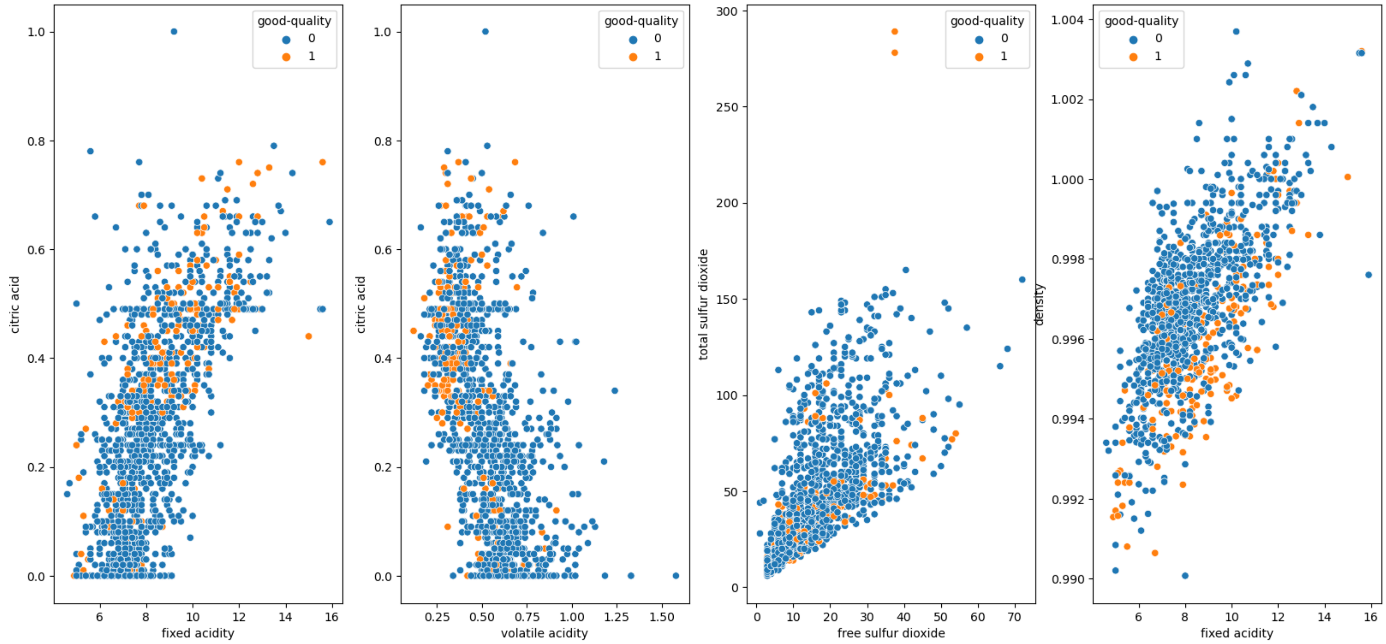


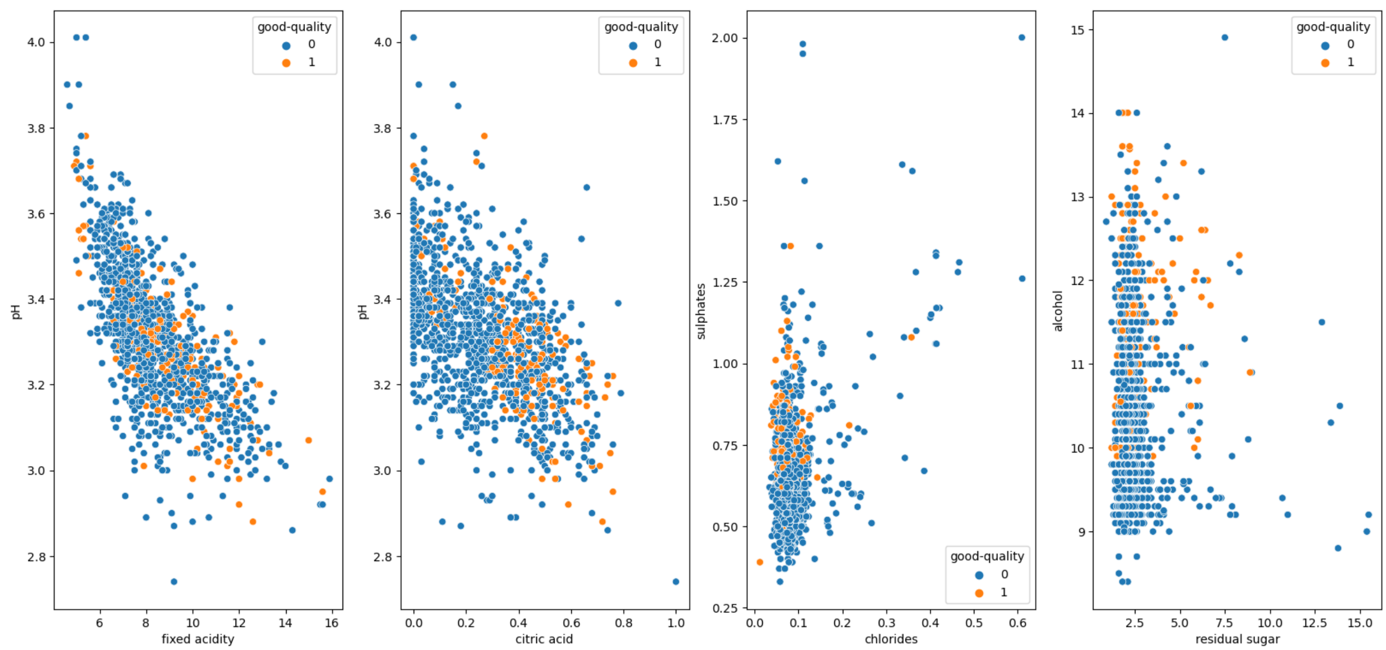
In [52]:
```
plt.figure(figsize = (10,6))
sns.heatmap(wine.corr(), annot = True)
plt.show()
```



In [54]:
```
fig, ax = plt.subplots(2,4,figsize=(20,20))
sns.scatterplot(x = 'fixed acidity', y = 'citric acid', hue = 'good-quality', data = win
sns.scatterplot(x = 'volatile acidity', y = 'citric acid', hue = 'good-quality', data =
sns.scatterplot(x = 'free sulfur dioxide', y = 'total sulfur dioxide', hue = 'good-quali
sns.scatterplot(x = 'fixed acidity', y = 'density', hue = 'good-quality', data = wine, a
sns.scatterplot(x = 'fixed acidity', y = 'pH', hue = 'good-quality', data = wine, ax=ax[
sns.scatterplot(x = 'citric acid', y = 'pH', hue = 'good-quality', data = wine, ax=ax[1,
sns.scatterplot(x = 'chlorides', y = 'sulphates', hue = 'good-quality', data = wine, ax=
sns.scatterplot(x = 'residual sugar', y = 'alcohol', hue = 'good-quality', data = wine,
```

Out[54]:
```
<AxesSubplot:xlabel='residual sugar', ylabel='alcohol'>
```





Train Test Split

In [57]:
```
X_train, X_test, Y_train, Y_test = train_test_split(wine.drop('good-quality', axis=1),
```

In [60]:
```
X_train.shape, X_test.shape, Y_train.shape, Y_test.shape
```

Out[60]:
```
((1279, 11), (320, 11), (1279,), (320,))
```

Model Training

In [61]:
```
from sklearn.linear_model import LogisticRegression
```

In [62]:
```
LR = LogisticRegression(max_iter = 10000)
LR
```

Out[62]:
```
LogisticRegression(max_iter=10000)
```

In [65]:
```
# Train the model

LR.fit(X_train, Y_train)

# Predict the model

pred = LR.predict(X_test)
```

In [67]:
```
pred[:5]
```

Out[67]:
```
array([0, 0, 0, 0, 0], dtype=int64)
```

In [69]:
```
result = pd.DataFrame({'Actual' : Y_test, 'Predicted' : pred})
```

In [70]:
```
pred = LR.predict(X_test)
print("----------------------------------------")
print("Accuracy Score : ",accuracy_score(Y_test, pred))
print("----------------------------------------")
```

```
----------------------------------------
Accuracy Score :  0.859375
----------------------------------------
```

In [93]:
```
import warnings
warnings.filterwarnings("ignore")
```

2. Support Vector Machine (SVM)

In [71]:
```
from sklearn.svm import SVC
```

In [72]:
```
svm = SVC()
svm
```

Out[72]:
```
SVC()
```

In [73]:
```
# Training the Model

svm.fit(X_train, Y_train)

# Testing the Model

pred_svc_train = svm.predict(X_train)
pred_svc_test = svm.predict(X_test)
```

In [76]:
```
print("----------------------------------------")
print("Training Accuracy : ", accuracy_score(Y_train, pred_svc_train))
print("Testing Accuracy : ", accuracy_score(Y_test, pred_svc_test))
print("----------------------------------------")
```

```
---------------------------------------
Training Accuracy :  0.8678655199374511
Testing Accuracy :  0.85625
---------------------------------------
```

3. Decision Tree Classifier

In [77]:
```python
from sklearn.tree import DecisionTreeClassifier
```

In [79]:
```python
DT = DecisionTreeClassifier()
DT
```

Out[79]:
```
DecisionTreeClassifier()
```

In [80]:
```python
# Training the Model

DT.fit(X_train, Y_train)

# Testing the Model

pred_DT_train = DT.predict(X_train)
pred_DT_test = DT.predict(X_test)
```

In [83]:
```python
print("-----------------------------------------")
print("Training Accuracy : ", accuracy_score(Y_train, pred_DT_train))
print("Testing Accuracy : ", accuracy_score(Y_test, pred_DT_test))
print("-----------------------------------------")
```

```
---------------------------------------
Training Accuracy :  1.0
Testing Accuracy :  0.875
---------------------------------------
```

4. K-Nearest Neighbor

In [84]:
```python
from sklearn.neighbors import KNeighborsClassifier
```

In [85]:
```python
knn = KNeighborsClassifier()
knn
```

Out[85]:
```
KNeighborsClassifier()
```

In [94]:
```python
#training the model
knn.fit(X_train, Y_train)

#testing the model
pred_knn_train = knn.predict(X_train)
pred_knn_test = knn.predict(X_test)
```

In [87]:
```python
print("-----------------------------------------")
print("Training Accuracy : ", accuracy_score(Y_train, pred_knn_train))
print("Testing Accuracy : ", accuracy_score(Y_test,pred_knn_test ))
print("-----------------------------------------")
```

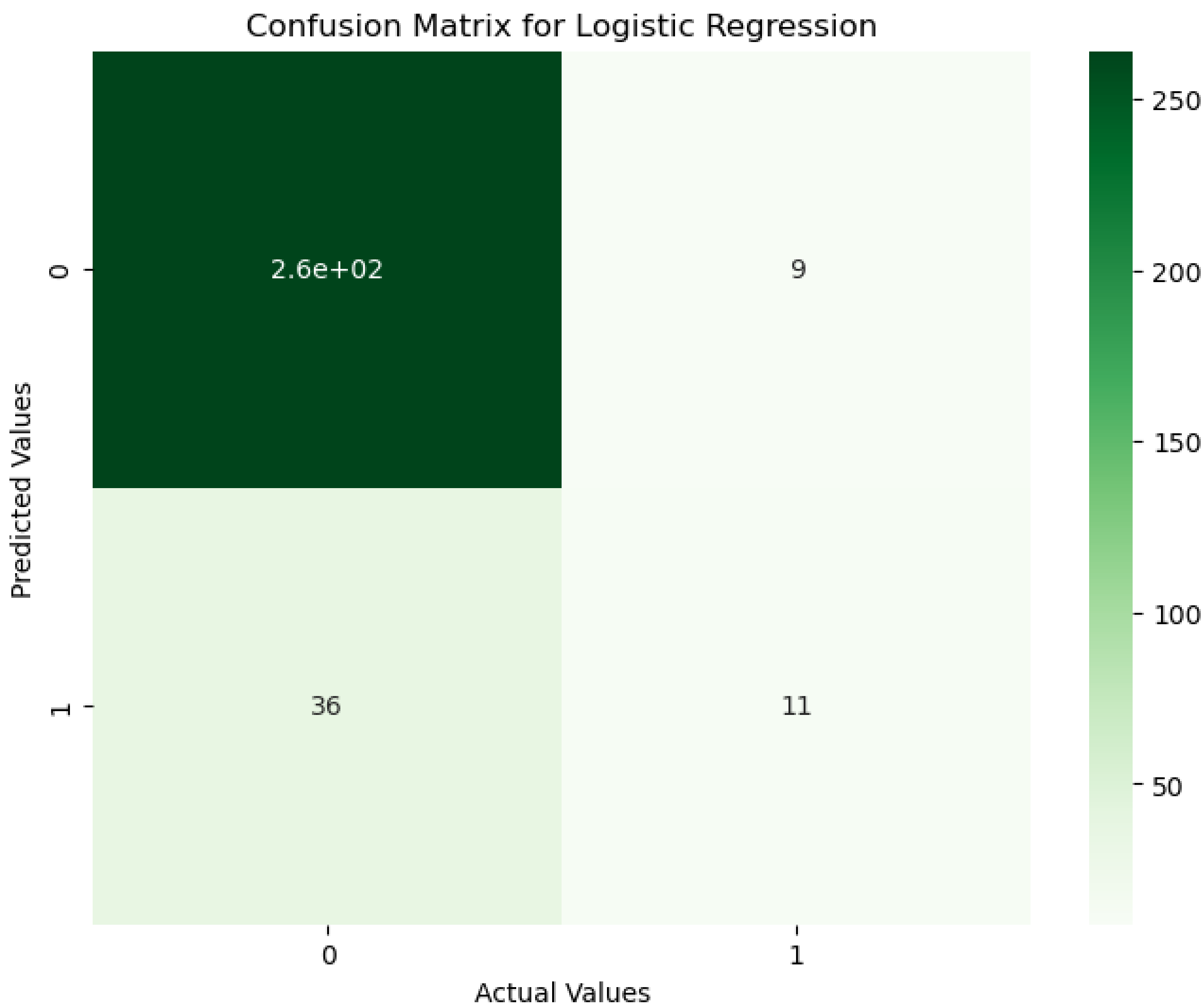
```
---------------------------------------
Training Accuracy :  0.9116497263487099
Testing Accuracy :  0.85625
---------------------------------------
```

Model Evaluation

Logistic Regression

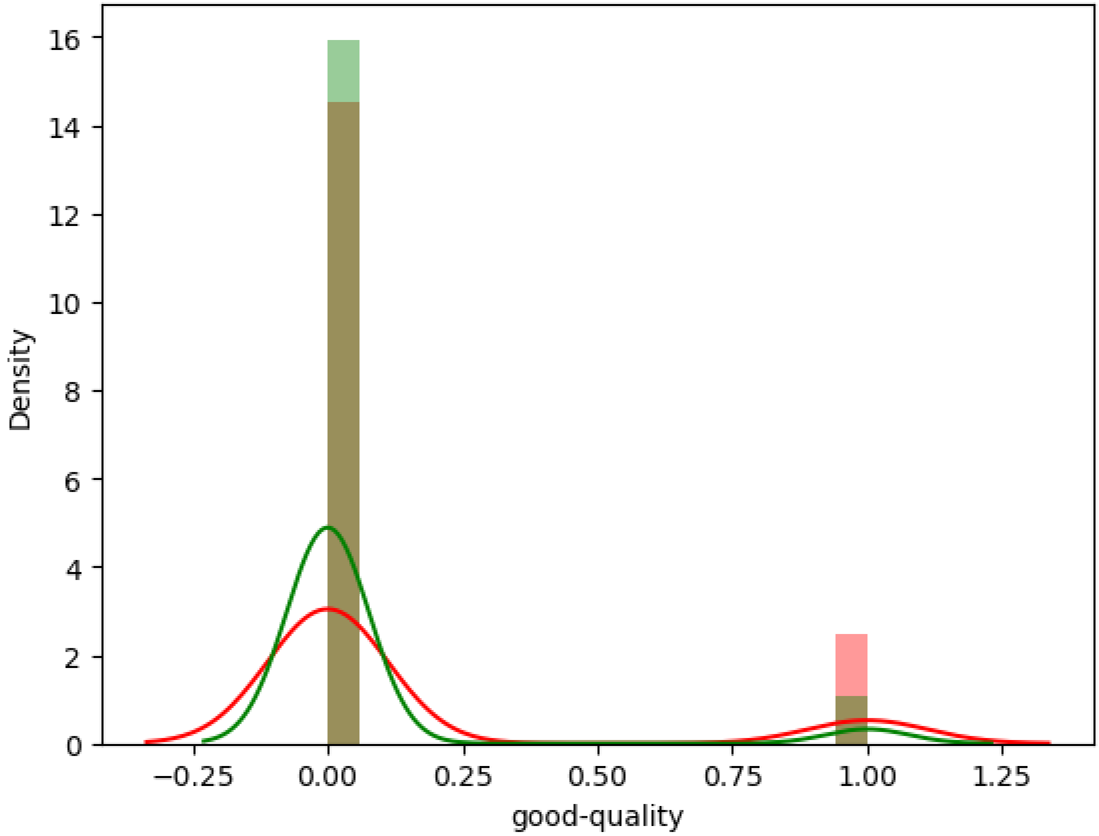
In [105…

```python
# logistic Regression Model Evaluation

plt.figure(figsize=(8,6))
sns.heatmap(confusion_matrix(Y_test, pred), annot=True, cmap='Greens')
plt.ylabel('Predicted Values')
plt.xlabel('Actual Values')
plt.title('Confusion Matrix for Logistic Regression')
plt.show()
```



In [110…

```python
# Distributed plot for the predicted and actual values

ax = sns.distplot(Y_test, hist = True, label = 'Actual', color = 'r')
sns.distplot(pred, hist = True, label = 'Predicted', color = 'g', ax = ax)
plt.show()
```



In [96]:

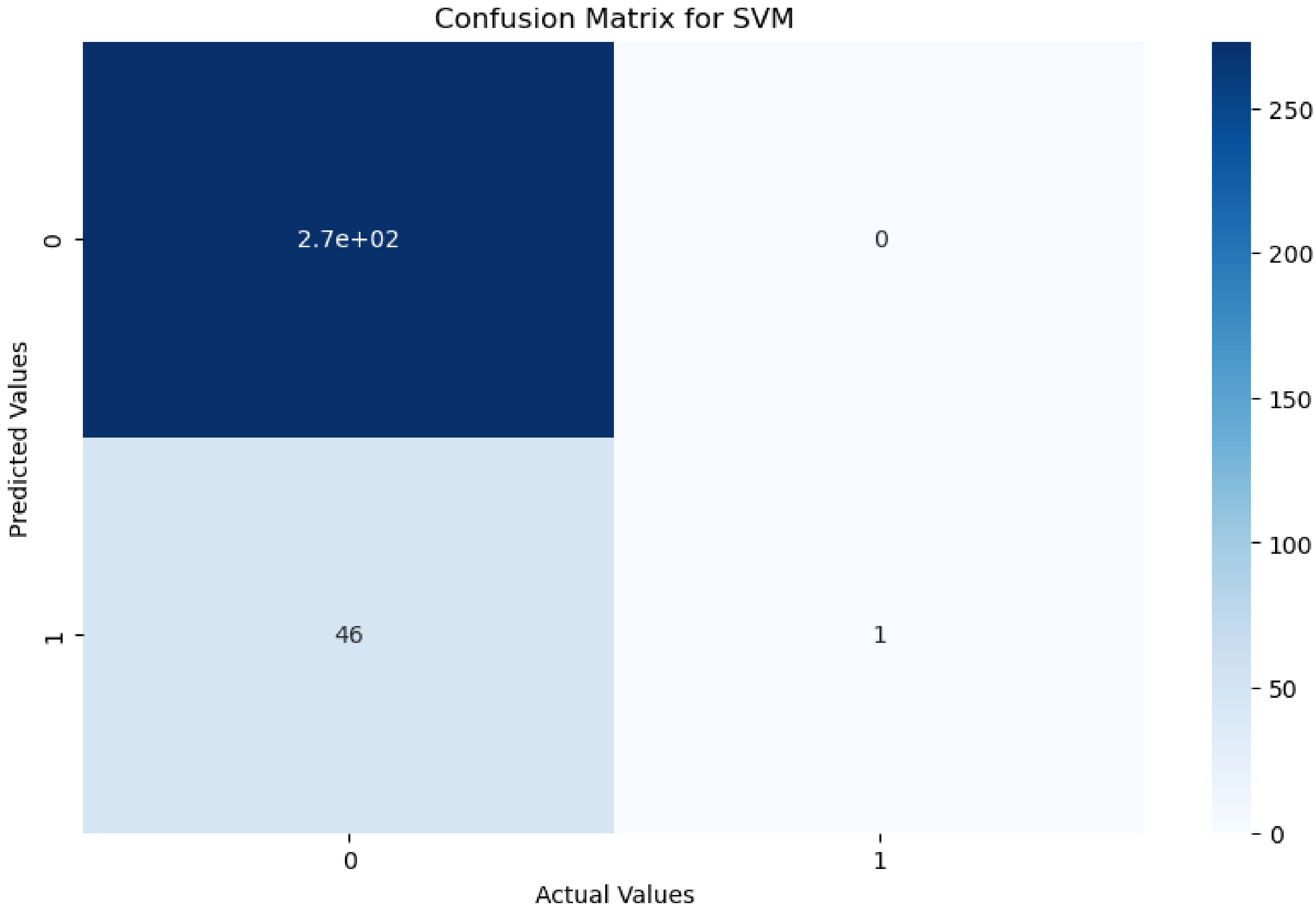
```
print('Logistic Regression Model Accuracy: ', accuracy_score(Y_test, pred))
print('Logistic Regression Model f1 score: ', metrics.f1_score(Y_test, pred))
print('Logistic Regression Model MAE: ', metrics.mean_absolute_error(Y_test, pred))
print('Logistic Regression Model RMSE: ', np.sqrt(metrics.mean_squared_error(Y_test, pre
```

```
Logistic Regression Model Accuracy:  0.859375
Logistic Regression Model f1 score:  0.3283582089552239
Logistic Regression Model MAE:  0.140625
Logistic Regression Model RMSE:  0.375
```

Support Vector Machine (SVM)

In [109…

```
# SVM Model Evaluation

plt.figure(figsize=(10,6))
sns.heatmap(confusion_matrix(Y_test,pred_svc_test), annot = True, cmap = 'Blues')
plt.ylabel('Predicted Values')
plt.xlabel('Actual Values')
plt.title('Confusion Matrix for SVM')
plt.show()
```



In [113…

```
print('SVM Model Accuracy: ', accuracy_score(Y_test, pred_svc_test))
print('SVM Model f1 score: ', metrics.f1_score(Y_test, pred_svc_test))
print('SVM Model MAE: ', metrics.mean_absolute_error(Y_test, pred_svc_test))
print('SVM Model RMSE: ', np.sqrt(metrics.mean_squared_error(Y_test, pred_svc_test)))
```

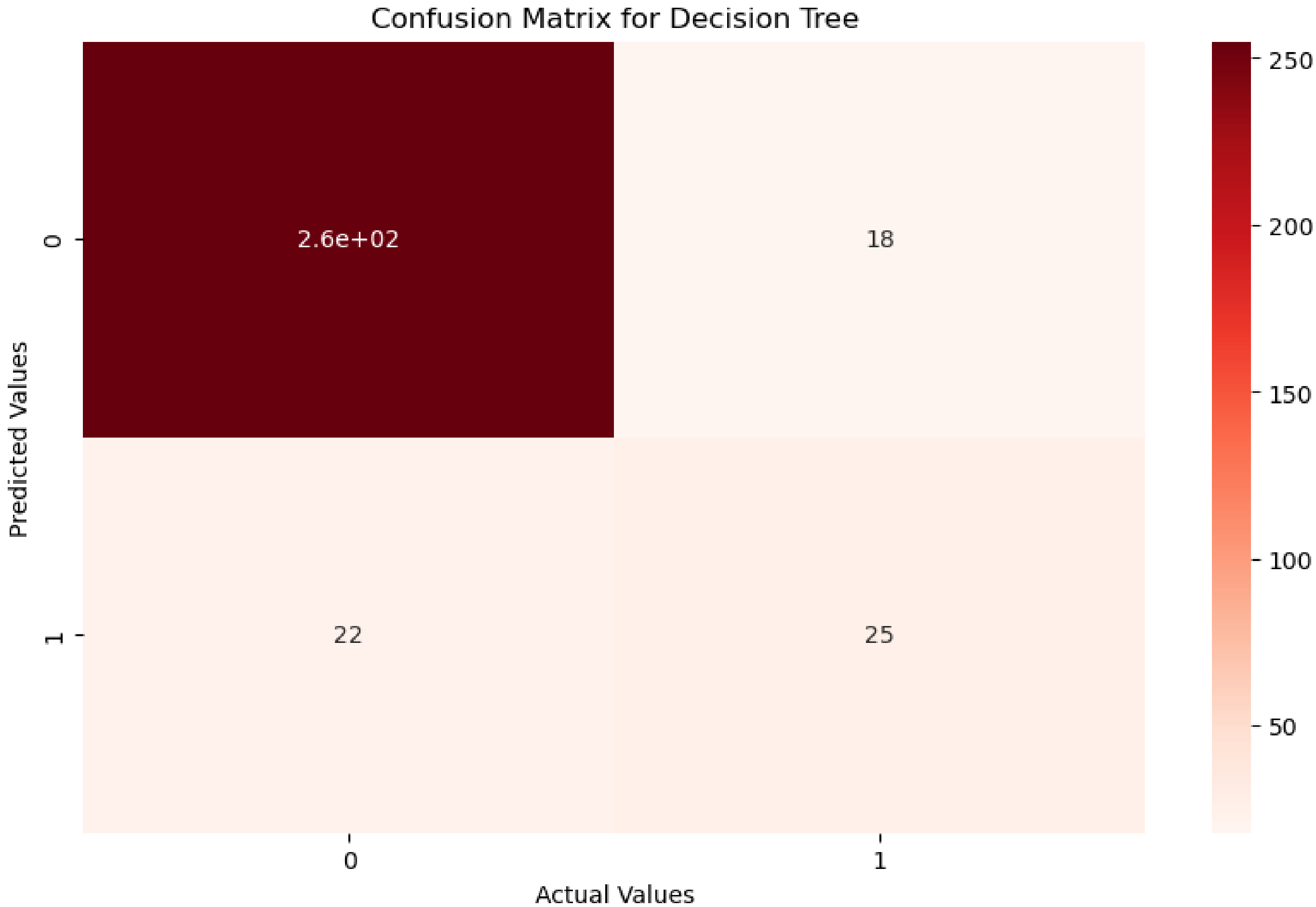
```
SVM Model Accuracy:  0.85625
SVM Model f1 score:  0.04166666666666667
SVM Model MAE:  0.14375
SVM Model RMSE:  0.3791437722025775
```

Decision Tree Classifier

In [116…

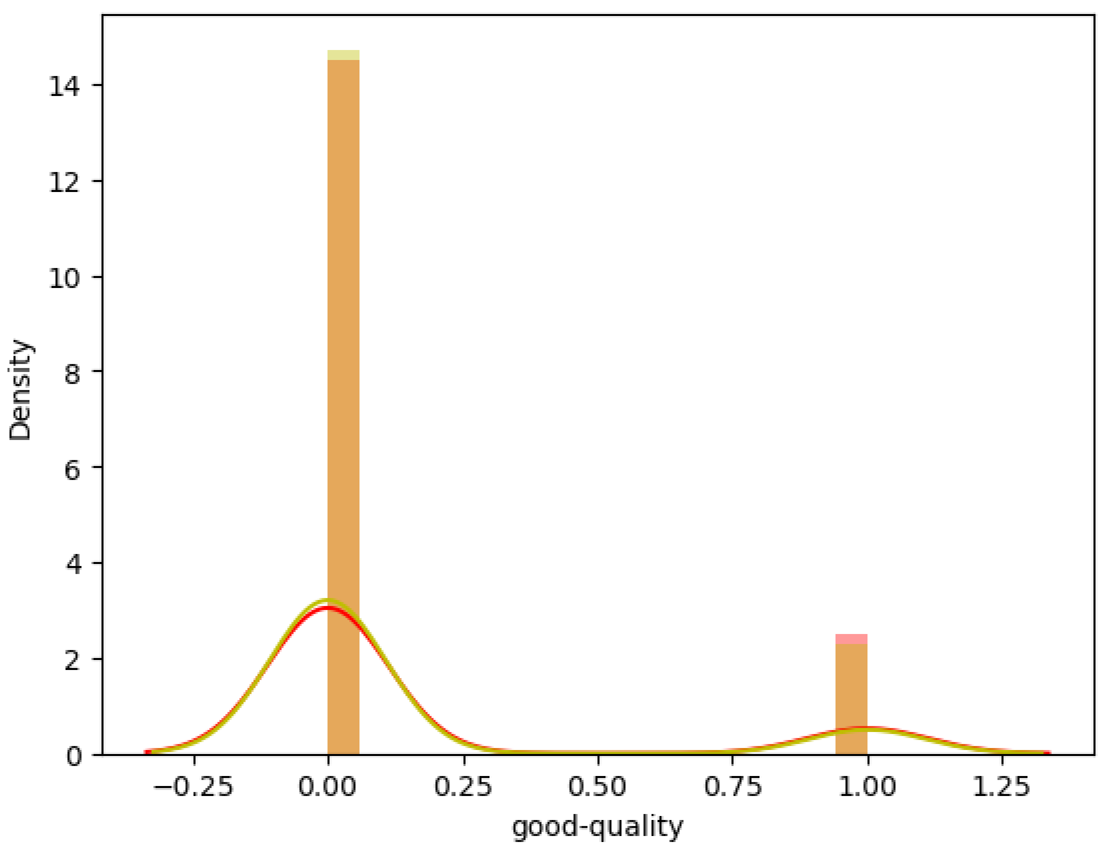
```
# Decision Tree Classifier Model Evaluation

plt.figure(figsize=(10,6))
sns.heatmap(confusion_matrix(Y_test, pred_DT_test), annot=True, cmap='Reds')
plt.ylabel('Predicted Values')
plt.xlabel('Actual Values')
plt.title('Confusion Matrix for Decision Tree')
plt.show()
```

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



In [129…

```
# Distributed plot for the predicted and actual values

ax = sns.distplot(Y_test, hist = True, label = 'Actual', color = 'r')
sns.distplot(pred_DT_test, hist = True, label = 'Predicted', color = 'y', ax = ax)
plt.show()
```



In [118…

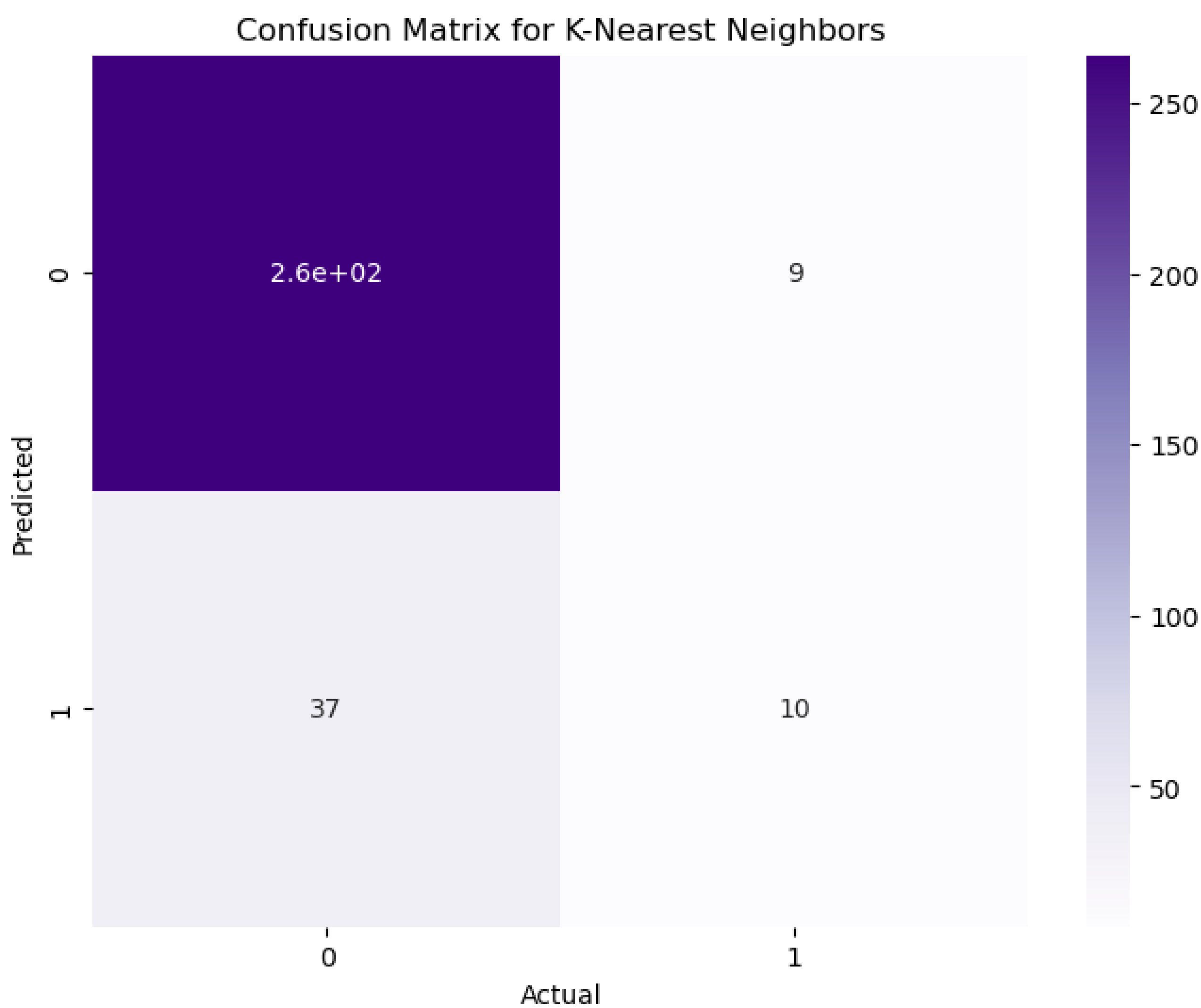
```
print('Decision Tree classifier Model Accuracy: ', accuracy_score(Y_test, pred_DT_test))
print('Decision Tree classifier Model f1 score: ', metrics.f1_score(Y_test, pred_DT_test
print('Decision Tree classifier Model MAE: ', metrics.mean_absolute_error(Y_test, pred_D
print('Decision Tree classifier Model RMSE: ', np.sqrt(metrics.mean_squared_error(Y_test
```

```
Decision Tree classifier Model Accuracy:  0.875
Decision Tree classifier Model f1 score:  0.5555555555555555
Decision Tree classifier Model MAE:  0.125
Decision Tree classifier Model RMSE:  0.3535533905932738
```

K-Nearest Neighbors

In [125…

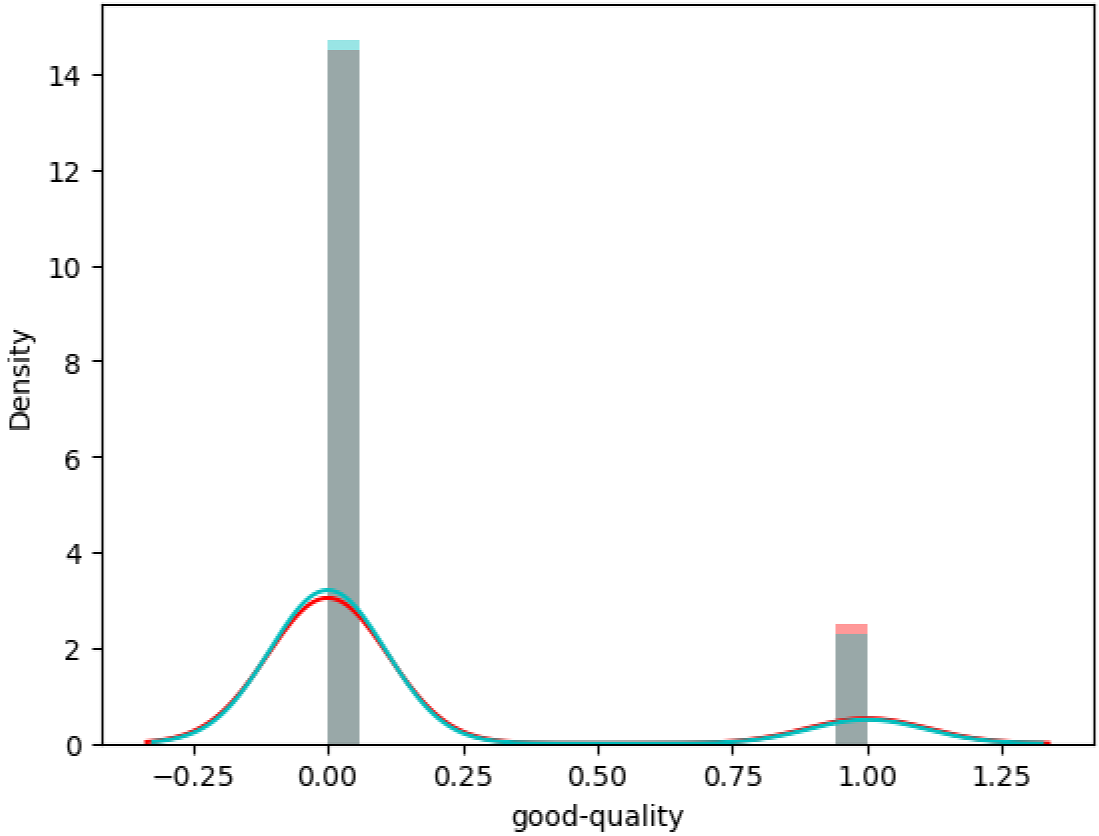
```
# K-Nearest Neighbors Model Evaluation

plt.figure(figsize=(8,6))
sns.heatmap(confusion_matrix(Y_test, pred_knn_test), annot=True, cmap='Purples')
plt.ylabel('Predicted')
plt.xlabel('Actual')
plt.title('Confusion Matrix for K-Nearest Neighbors')
plt.show()
```



In [131…

```
# Distributed plot for the predicted and actual values

ax = sns.distplot(Y_test, hist = True, label = 'Actual', color = 'r')
sns.distplot(pred_DT_test, hist = True, label = 'Predicted', color = 'c', ax = ax)
plt.show()
```

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



In [132…

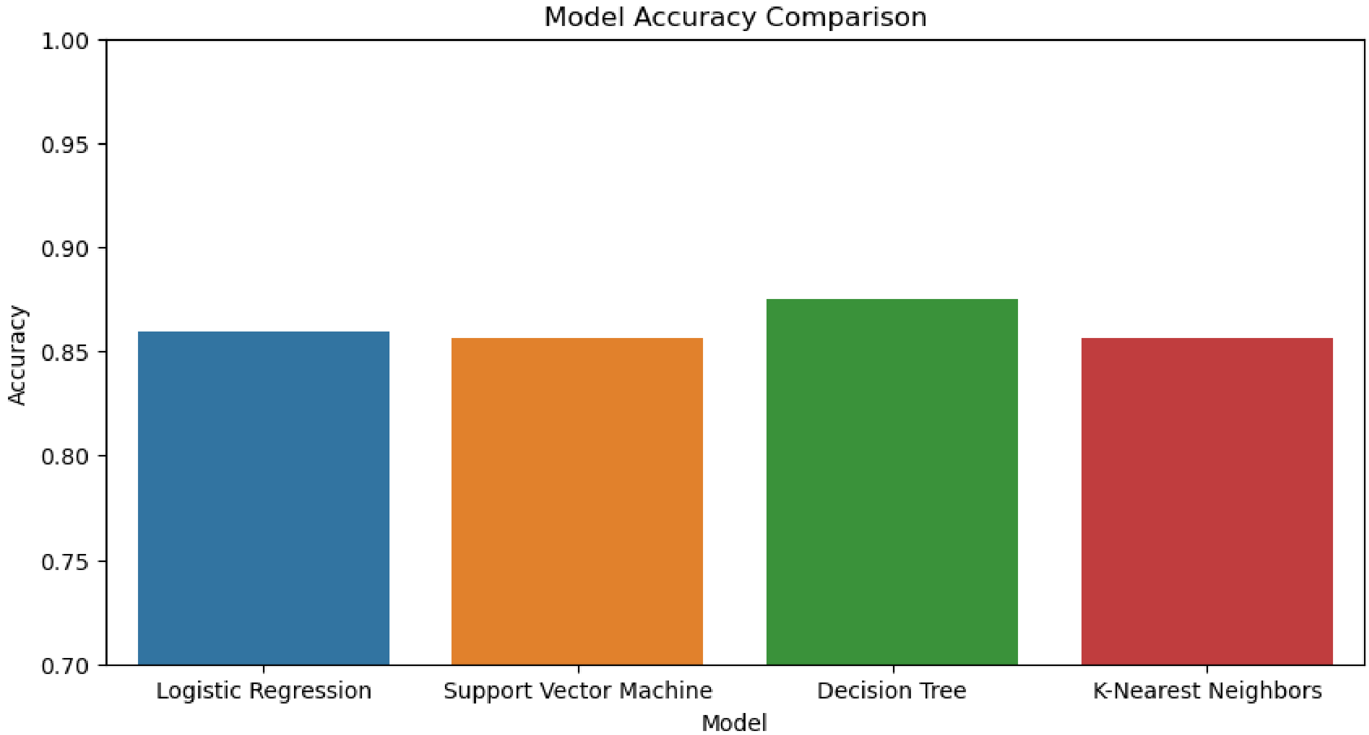
```
print('K-Nearest Neighbors Model Accuracy: ', accuracy_score(Y_test, pred_knn_test))
print('K-Nearest Neighbors Model f1 score: ', metrics.f1_score(Y_test, pred_knn_test))
print('K-Nearest Neighbors Model MAE: ', metrics.mean_absolute_error(Y_test, pred_knn_te
print('K-Nearest Neighbors Model RMSE: ', np.sqrt(metrics.mean_squared_error(Y_test, pre
```

```
K-Nearest Neighbors Model Accuracy:  0.85625
K-Nearest Neighbors Model f1 score:  0.30303030303030304
K-Nearest Neighbors Model MAE:  0.14375
K-Nearest Neighbors Model RMSE:  0.3791437722025775
```

Model Comparison

In [134…

```
models = ['Logistic Regression', 'Support Vector Machine', 'Decision Tree', 'K-Nearest N
accuracy = [accuracy_score(Y_test, pred), accuracy_score(Y_test, pred_svc_test), accurac
plt.figure(figsize=(10,5))
sns.barplot(x=models, y=accuracy)
plt.title('Model Accuracy Comparison')
plt.xlabel('Model')
plt.ylabel('Accuracy')
plt.ylim(0.7, 1.0)
plt.show()
```



Story:

Conclusion

In this project, I evaluated the performance of four machine learning algorithms for the task of red wine quality prediction: logistic regression, support vector machines (SVM), decision tree classifier, and k-nearest neighbors (KNN). We found that the Decision Tree Classifier model performed the best on the test set, with an accuracy of 87.5%.

This means that the Decision Tree Classifier model is able to predict the quality of red wine based on the given features with a high degree of accuracy. This is important because it can help winemakers and consumers to identify high-quality wines.

The other three algorithms also performed well on the test set, with accuracies of 85.6% for SVM, 85.9% for Logistic Regression, and 85.6% for KNN. However, the logistic regression model was the only algorithm to achieve an accuracy of near 86%.

One possible explanation for the superior performance of the logistic regression model is that it is a linear model. This means that it can learn the relationships between the input features and the target variable in a more straightforward way than the other algorithms, which are all non-linear models.

Another possible explanation is that the logistic regression model is less prone to overfitting than the other algorithms. Overfitting occurs when a model learns the training data too well and is unable to generalize to new data. The logistic regression model's regularization parameter can be used to control the amount of overfitting.

Overall, the results of this project suggest that the logistic regression model is a good choice for the task of red wine quality prediction. It is able to achieve a high degree of accuracy while being relatively resistant to overfitting.

One possible direction for future work would be to investigate the use of other machine learning algorithms for red wine quality prediction, such as ensemble methods or deep learning models. Ensemble methods combine the predictions of multiple models to produce a more accurate prediction. Deep learning models are a type of machine learning model that can learn complex relationships between data.

Another possible direction for future work would be to collect more data on red wine quality. This data could be used to train more accurate models and to improve the understanding of the factors that influence red wine quality.