ex5-task13-bwf

August 30, 2024

[1]:
```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler,label_binarize
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, roc_auc_score, roc_curve
import matplotlib.pyplot as plt
```

[2]:
```
df = pd.read_csv('/kaggle/input/wine-quality-dataset/WineQT.csv')
df.head()
```

[2]:

	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

[3]:
```
df.columns
```

[3]:
```
Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
       'pH', 'sulphates', 'alcohol', 'quality', 'Id'],
```

```
      dtype='object')
```

[4]:
```
df.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
```

[5]:
```
df.describe()
```

[5]:

	fixed acidity	volatile acidity	citric acid	residual sugar	\
count	1143.000000	1143.000000	1143.000000	1143.000000	
mean	8.311111	0.531339	0.268364	2.532152	
std	1.747595	0.179633	0.196686	1.355917	
min	4.600000	0.120000	0.000000	0.900000	
25%	7.100000	0.392500	0.090000	1.900000	
50%	7.900000	0.520000	0.250000	2.200000	
75%	9.100000	0.640000	0.420000	2.600000	
max	15.900000	1.580000	1.000000	15.500000	

	chlorides	free sulfur dioxide	total sulfur dioxide	density	\
count	1143.000000	1143.000000	1143.000000	1143.000000	
mean	0.086933	15.615486	45.914698	0.996730	
std	0.047267	10.250486	32.782130	0.001925	
min	0.012000	1.000000	6.000000	0.990070	
25%	0.070000	7.000000	21.000000	0.995570	
50%	0.079000	13.000000	37.000000	0.996680	
75%	0.090000	21.000000	61.000000	0.997845	
max	0.611000	68.000000	289.000000	1.003690	

	pH	sulphates	alcohol	quality	Id
count	1143.000000	1143.000000	1143.000000	1143.000000	1143.000000
mean	3.311015	0.657708	10.442111	5.657043	804.969379
std	0.156664	0.170399	1.082196	0.805824	463.997116
min	2.740000	0.330000	8.400000	3.000000	0.000000
25%	3.205000	0.550000	9.500000	5.000000	411.000000
50%	3.310000	0.620000	10.200000	6.000000	794.000000
75%	3.400000	0.730000	11.100000	6.000000	1209.500000
max	4.010000	2.000000	14.900000	8.000000	1597.000000

[6]:
```
df.columns
```

[6]:
```
Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
       'pH', 'sulphates', 'alcohol', 'quality', 'Id'],
      dtype='object')
```

[7]:
```
# Identify numerical columns
numerical_columns = df.select_dtypes(include=['float64', 'int64']).columns

# Standardize numerical features
scaler = StandardScaler()
df[numerical_columns] = scaler.fit_transform(df[numerical_columns])
```

[8]:
```
df['quality'] = df['quality'].astype(int)  # Convert to integer
```

[9]:
```
# Define the target variable based on quality
df['quality_label'] = df['quality'].apply(lambda x: 1 if x >= 6 else 0)
```

[10]:
```
# Define features and target variable
X = df.drop(['quality', 'quality_label', 'Id'], axis=1)  # Drop unnecessary␣
 ↪columns
y = df['quality_label']
```

[11]:
```
# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=42,stratify=y)
```

[12]:
```
# Initialize and train the Decision Tree model
model = DecisionTreeClassifier()
model.fit(X_train, y_train)
```

[12]:
```
DecisionTreeClassifier()
```

[13]:
```
# Make predictions
y_pred = model.predict(X_test)
print("Classes predicted in test set:", np.unique(y_pred))
```

```
Classes predicted in test set: [0]
```

[14]:
```
# Predict probabilities
y_prob = model.predict_proba(X_test)
print("Shape of y_prob:", y_prob.shape)
```

```
Shape of y_prob: (343, 1)
```

[15]:
```
#y_prob   ## for checking prediction
```

[16]:
```
# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')
```

```
Accuracy: 1.00
```

[ ]:
```

```