```
import numpy as np
import pandas as pd
import scipy.stats
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
from statsmodels.formula.api import ols
from statsmodels.stats.multicomp import pairwise_tukeyhsd
from scipy.stats import chi2_contingency
import statsmodels.formula.api as smf
```

```
/usr/local/lib/python3.7/dist-packages/statsmodels/tools/_testing.py:19: FutureWarnir
  import pandas.util.testing as tm
```

```
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-2-21dc3c638f66> in <module>()
      1 from google.colab import files
----> 2 uploaded = files.upload()

/usr/local/lib/python3.7/dist-packages/google/colab/files.py in upload()
     67   local_filenames = dict()
     68
---> 69   while result['action'] != 'complete':
     70     result = _output.eval_js(
     71         'google.colab._files._uploadFilesContinue("{output_id}")'.format(

TypeError: 'NoneType' object is not subscriptable
```

SEARCH STACK OVERFLOW

```
df= pd.read_csv("EDA_RedWine.csv")
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Unnamed: 0            1599 non-null   int64
 1   fixed_acidity         1599 non-null   float64
 2   volatile_acidity      1599 non-null   float64
 3   citric_acid           1599 non-null   float64
 4   residual_sugar        1599 non-null   float64
 5   chlorides             1599 non-null   float64
```

```
 6   free_sulfur_dioxide   1599 non-null   float64
 7   total_sulfur_dioxide  1599 non-null   float64
 8   density               1599 non-null   float64
 9   pH                    1599 non-null   float64
 10  sulphates             1599 non-null   float64
 11  alcohol               1599 non-null   float64
 12  quality               1599 non-null   int64
dtypes: float64(11), int64(2)
memory usage: 162.5 KB
```

```
df= df.drop(['Unnamed: 0'], axis=1)
```

```
df.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
```

```
df.isna().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
```

```
df.describe()
```

	fixed_acidity	volatile_acidity	citric_acid	residual_sugar	chlorides	fre
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	
mean	8.290901	0.526429	0.270922	2.322358	0.081194	
std	1.655860	0.174045	0.194614	0.609493	0.017822	
min	4.600000	0.120000	0.000000	0.900000	0.040000	
25%	7.100000	0.390000	0.090000	1.900000	0.070000	
50%	7.900000	0.520000	0.260000	2.200000	0.079000	
75%	9.200000	0.640000	0.420000	2.600000	0.090000	

```
df.shape
```

```
(1599, 12)
```

```
X = df.loc[:, df.columns != 'quality']
y = df.loc[:, df.columns == 'quality']


from sklearn.model_selection import train_test_split


X_train, X_test, y_train, y_test= train_test_split(X,y, test_size=0.3, random_state= 14)


#Logistic Regression
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split


"""#Logistic Regression

#### Log_r
"""

Log_r= LogisticRegression(max_iter= 10000000000000000000000000000000)



Log_r = Log_r.fit(X_train, y_train)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:985: DataConversic
  y = column_or_1d(y, warn=True)
```

```
y_pred_train = Log_r.predict(X_train)
y_pred_train
```

```
array([4, 4, 4, ..., 4, 3, 4])
```

```
y_pred_test = Log_r.predict(X_test)
```

```
y_pred_test
```

```
array([3, 3, 4, 3, 4, 4, 3, 4, 3, 4, 3, 3, 5, 3, 3, 3, 4, 3, 3, 4, 4, 3,
       4, 3, 4, 4, 4, 3, 3, 4, 3, 3, 4, 4, 4, 5, 4, 4, 3, 4, 4, 3, 3, 4,
       3, 4, 3, 4, 4, 4, 4, 3, 4, 3, 3, 4, 3, 3, 3, 3, 4, 3, 4, 3, 4, 3,
       3, 4, 3, 4, 3, 3, 4, 4, 4, 3, 4, 4, 4, 4, 3, 4, 4, 3, 3, 4, 3, 3,
       3, 4, 4, 3, 5, 4, 4, 3, 4, 4, 3, 4, 4, 4, 4, 3, 3, 4, 3, 4, 4, 3,
       3, 3, 4, 4, 3, 3, 5, 3, 3, 3, 3, 3, 4, 3, 3, 3, 3, 3, 3, 3, 4, 4,
       4, 4, 4, 4, 3, 3, 4, 3, 4, 4, 4, 3, 3, 3, 3, 4, 3, 4, 3, 4, 4, 3,
       4, 3, 3, 3, 3, 3, 4, 3, 4, 4, 3, 3, 3, 4, 4, 3, 3, 3, 3, 3, 3, 4,
       3, 4, 4, 3, 4, 3, 3, 3, 4, 4, 3, 4, 3, 3, 4, 3, 3, 3, 3, 5, 3, 3,
       4, 3, 4, 3, 3, 3, 4, 3, 4, 4, 4, 4, 3, 4, 3, 4, 4, 4, 4, 4, 3, 4,
       3, 4, 4, 3, 4, 5, 3, 3, 3, 3, 3, 4, 4, 4, 4, 3, 3, 4, 3, 3, 4, 3,
       4, 3, 3, 4, 3, 4, 4, 4, 3, 4, 3, 3, 4, 4, 4, 3, 3, 3, 3, 3, 3, 3,
       3, 4, 3, 4, 4, 3, 4, 4, 3, 4, 3, 3, 3, 3, 3, 4, 3, 3, 4, 3, 3, 4,
       3, 4, 3, 4, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 3, 4, 4, 4, 4, 3, 3, 4,
       4, 4, 4, 4, 3, 3, 3, 3, 4, 3, 4, 4, 4, 4, 4, 4, 3, 3, 3, 4, 4, 3,
       3, 3, 4, 3, 4, 3, 4, 3, 4, 4, 3, 3, 3, 3, 3, 4, 3, 3, 4, 4, 3, 4,
       3, 3, 3, 3, 3, 4, 3, 4, 4, 4, 3, 3, 3, 4, 4, 4, 4, 4, 4, 3, 4, 3,
       4, 4, 4, 3, 3, 3, 5, 5, 4, 4, 3, 3, 3, 5, 4, 4, 3, 4, 4, 3, 5, 4,
       3, 4, 5, 4, 4, 4, 4, 4, 4, 3, 4, 4, 3, 4, 3, 3, 4, 3, 4, 3, 3, 3,
       4, 3, 4, 4, 3, 3, 3, 4, 4, 3, 3, 3, 3, 3, 4, 3, 3, 4, 4, 4, 4, 4,
       4, 4, 3, 4, 3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 3, 3, 4, 3, 4, 3, 4, 4,
       4, 3, 3, 4, 3, 4, 4, 3, 4, 3, 3, 4, 5, 4, 3, 3, 3, 3])
```

```
from sklearn import metrics
from sklearn.metrics import accuracy_score
```

```
print(metrics.accuracy_score(y_pred_train,y_train))
```

```
0.610366398570152
```

```
print(metrics.accuracy_score(y_pred_test,y_test))
```

```
0.5541666666666667
```

```
"""#### Log_r1"""
```

```
#Multiclass = ovr
```

```
Log_r1 = LogisticRegression(multi_class='ovr')
```

```
Log_r1 = Log_r1.fit(X_train, y_train)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:985: DataConversic
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:818: Converg
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:818: Converg
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:818: Converg
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:818: Converg
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:818: Converg
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:818: Converg
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
```

```
y_pred_train1 = Log_r1.predict(X_train)
y_pred_train1
```

```
array([4, 4, 4, ..., 4, 3, 4])
```

```
y_pred_test1 = Log_r1.predict(X_test)
y_pred_test1
```

```
array([4, 3, 4, 3, 4, 4, 3, 3, 3, 4, 3, 3, 4, 3, 3, 3, 4, 3, 3, 4, 4, 3,
       4, 3, 4, 4, 4, 3, 3, 3, 3, 3, 4, 4, 4, 5, 4, 4, 4, 4, 4, 3, 3, 4,
       3, 4, 3, 4, 4, 4, 4, 3, 4, 3, 3, 4, 3, 3, 3, 3, 4, 3, 4, 3, 4, 4,
       3, 4, 3, 4, 3, 3, 4, 3, 4, 3, 4, 4, 4, 4, 3, 4, 4, 3, 3, 4, 3, 3,
```

```
       3, 4, 4, 3, 4, 4, 4, 4, 4, 4, 3, 4, 4, 4, 4, 3, 3, 4, 3, 3, 4, 3,
       3, 3, 4, 4, 3, 3, 4, 3, 3, 3, 3, 3, 4, 3, 3, 3, 3, 3, 3, 3, 4, 4,
       4, 4, 4, 4, 3, 3, 4, 3, 4, 4, 4, 3, 3, 3, 3, 4, 3, 4, 3, 4, 4, 3,
       4, 3, 3, 3, 3, 3, 3, 3, 4, 4, 3, 3, 3, 4, 4, 3, 3, 3, 3, 3, 3, 4,
       3, 4, 4, 3, 4, 3, 3, 3, 4, 4, 3, 4, 3, 3, 4, 3, 3, 3, 3, 4, 3, 3,
       4, 3, 4, 3, 3, 3, 4, 3, 4, 4, 4, 4, 3, 3, 3, 4, 4, 4, 4, 3, 3, 4,
       3, 4, 4, 3, 4, 4, 3, 3, 3, 3, 3, 4, 4, 4, 4, 3, 3, 4, 3, 3, 4, 3,
       4, 3, 3, 4, 3, 4, 4, 4, 3, 4, 3, 3, 4, 4, 4, 3, 3, 3, 3, 3, 3, 3,
       3, 4, 3, 3, 4, 3, 4, 4, 3, 4, 3, 3, 3, 3, 3, 4, 3, 3, 4, 3, 3, 4,
       3, 4, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 4, 3, 4, 3, 4, 4, 3, 4, 3, 3, 4,
       4, 4, 4, 4, 3, 3, 3, 3, 4, 3, 3, 4, 4, 4, 4, 4, 4, 3, 3, 4, 4, 3,
       3, 3, 4, 3, 4, 3, 4, 3, 3, 3, 3, 3, 3, 3, 3, 3, 4, 3, 3, 4, 4, 3, 4,
       3, 3, 3, 3, 3, 4, 3, 4, 4, 4, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 3, 3,
       4, 4, 4, 3, 3, 3, 4, 4, 4, 4, 3, 3, 3, 4, 4, 4, 3, 4, 4, 3, 4, 4,
       3, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 4, 3, 4, 3, 3, 4, 3, 4, 3, 3, 3,
       4, 3, 4, 4, 3, 3, 3, 4, 4, 3, 3, 3, 3, 3, 4, 3, 3, 4, 4, 4, 4, 4,
       3, 4, 3, 4, 3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 3, 3, 3, 3, 4, 3, 4, 4,
       4, 3, 3, 4, 3, 4, 4, 3, 4, 3, 3, 4, 4, 4, 3, 3, 3, 3])
```

```
print(metrics.accuracy_score(y_pred_train1,y_train))
```

```
0.5960679177837355
```

```
print(metrics.accuracy_score(y_pred_test1,y_test))
```

```
0.5416666666666666
```

```
"""#### Log_r2"""

#________multiclass= multinomial, solver = newton_cg

Log_r2 = LogisticRegression(multi_class='multinomial',solver='newton-cg')

Log_r2 = Log_r2.fit(X_train, y_train)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:985: DataConversic
  y = column_or_1d(y, warn=True)
```

```
y_pred_train2 = Log_r2.predict(X_train)
y_pred_train2
```

```
array([4, 4, 4, ..., 4, 3, 4])
```

```
y_pred_test2 = Log_r2.predict(X_test)
y_pred_test2
```

```
array([3, 3, 4, 3, 4, 4, 3, 4, 3, 4, 3, 3, 5, 3, 3, 3, 4, 3, 3, 4, 4, 3,
       4, 3, 4, 4, 4, 3, 3, 4, 3, 3, 4, 4, 4, 5, 4, 4, 3, 4, 4, 3, 3, 4,
```

```
       3, 4, 3, 4, 4, 4, 4, 3, 4, 3, 3, 4, 3, 3, 3, 3, 4, 3, 4, 3, 4, 3,
       3, 4, 3, 4, 3, 3, 4, 4, 4, 3, 4, 4, 4, 4, 3, 4, 4, 3, 3, 4, 3, 3,
       3, 4, 4, 3, 5, 4, 4, 3, 4, 4, 3, 4, 4, 4, 4, 3, 3, 4, 3, 4, 4, 3,
       3, 3, 4, 4, 3, 3, 5, 3, 3, 3, 3, 3, 4, 3, 3, 3, 3, 3, 3, 3, 4, 4,
       4, 4, 4, 4, 3, 3, 4, 3, 4, 4, 4, 3, 3, 3, 3, 4, 3, 4, 3, 4, 4, 3,
       4, 3, 3, 3, 3, 3, 4, 3, 4, 4, 3, 3, 3, 4, 4, 3, 3, 3, 3, 3, 3, 4,
       3, 4, 4, 3, 4, 3, 3, 3, 4, 4, 3, 4, 3, 3, 4, 3, 3, 3, 3, 5, 3, 3,
       4, 3, 4, 3, 3, 3, 4, 3, 4, 4, 4, 4, 3, 4, 3, 4, 4, 4, 4, 4, 3, 4,
       3, 4, 4, 3, 4, 5, 3, 3, 3, 3, 3, 4, 4, 4, 4, 3, 3, 4, 3, 3, 4, 3,
       4, 3, 3, 4, 3, 4, 4, 4, 3, 4, 3, 3, 4, 4, 4, 3, 3, 3, 3, 3, 3, 3,
       3, 4, 3, 4, 4, 3, 4, 4, 3, 4, 3, 3, 3, 3, 3, 4, 3, 3, 4, 3, 3, 4,
       3, 4, 3, 4, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 3, 4, 4, 4, 4, 3, 3, 4,
       4, 4, 4, 4, 3, 3, 3, 3, 4, 3, 4, 4, 4, 4, 4, 4, 3, 3, 3, 4, 4, 3,
       3, 3, 4, 3, 4, 3, 4, 3, 4, 4, 3, 3, 3, 3, 3, 4, 3, 3, 4, 4, 3, 4,
       3, 3, 3, 3, 3, 4, 3, 4, 4, 4, 3, 3, 3, 4, 4, 4, 4, 4, 4, 3, 4, 3,
       4, 4, 4, 3, 3, 3, 5, 5, 4, 4, 3, 3, 3, 5, 4, 4, 3, 4, 4, 3, 5, 4,
       3, 4, 5, 4, 4, 4, 4, 4, 4, 3, 4, 4, 3, 4, 3, 3, 4, 3, 4, 3, 3, 3,
       4, 3, 4, 4, 3, 3, 3, 4, 4, 3, 3, 3, 3, 3, 4, 3, 3, 4, 4, 4, 4, 4,
       4, 4, 3, 4, 3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 3, 3, 4, 3, 4, 3, 4, 4,
       4, 3, 3, 4, 3, 4, 4, 3, 4, 3, 3, 4, 5, 4, 3, 3, 3, 3])
```

```
print(metrics.accuracy_score(y_pred_train2,y_train))
```

```
0.610366398570152
```

```
print(metrics.accuracy_score(y_pred_test2,y_test))
```

```
0.5541666666666667
```

```
"""#### Log_r3"""

#________multiclass= multinomial, solver = lbfgs

Log_r3 = LogisticRegression(multi_class='multinomial',solver='lbfgs')


Log_r3 = Log_r3.fit(X_train, y_train)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:985: DataConversic
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:818: Converg
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
```

```
y_pred_train3 = Log_r3.predict(X_train)
y_pred_train3
```

```
array([4, 4, 4, ..., 4, 3, 3])
```

```
y_pred_test3 = Log_r3.predict(X_test)
y_pred_test3
```

```
array([4, 3, 4, 3, 4, 4, 3, 3, 3, 4, 3, 3, 4, 3, 3, 3, 4, 3, 3, 4, 4, 3,
       4, 3, 4, 4, 4, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 3, 4, 4, 3, 3, 4,
       3, 4, 3, 4, 4, 4, 4, 3, 4, 3, 3, 4, 3, 3, 3, 3, 4, 3, 4, 3, 4, 3,
       3, 3, 3, 4, 3, 3, 4, 4, 4, 3, 4, 4, 4, 4, 4, 4, 4, 3, 3, 4, 3, 3,
       3, 4, 4, 3, 4, 4, 4, 4, 3, 4, 3, 4, 3, 4, 4, 3, 3, 4, 3, 4, 4, 3,
       3, 3, 4, 4, 3, 3, 4, 3, 3, 3, 3, 3, 4, 4, 3, 3, 4, 3, 3, 3, 4, 4,
       4, 4, 4, 4, 3, 3, 4, 3, 4, 4, 4, 3, 3, 3, 3, 4, 3, 4, 3, 4, 4, 3,
       4, 3, 3, 3, 3, 3, 4, 3, 4, 4, 3, 3, 3, 4, 5, 3, 3, 4, 3, 3, 3, 4,
       3, 4, 3, 3, 3, 3, 3, 4, 4, 4, 3, 4, 3, 3, 4, 3, 3, 3, 3, 4, 3, 3,
       4, 3, 4, 3, 3, 3, 4, 3, 4, 4, 4, 4, 3, 3, 3, 4, 4, 4, 4, 3, 3, 4,
       3, 4, 4, 3, 4, 4, 3, 3, 3, 3, 3, 4, 4, 4, 4, 3, 4, 4, 3, 3, 4, 3,
       4, 3, 3, 4, 3, 4, 4, 4, 3, 4, 3, 3, 4, 4, 3, 3, 3, 3, 3, 3, 3, 3,
       3, 4, 3, 3, 4, 3, 4, 4, 3, 4, 3, 3, 3, 3, 3, 4, 3, 3, 4, 3, 3, 4,
       3, 4, 3, 4, 3, 3, 3, 3, 3, 3, 3, 4, 3, 4, 3, 4, 4, 3, 4, 3, 3, 4,
       4, 4, 4, 4, 3, 3, 3, 3, 4, 3, 4, 4, 4, 4, 4, 3, 4, 3, 3, 4, 4, 3,
       4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 3, 3, 3, 3, 3, 4, 3, 3, 4, 4, 3, 4,
       3, 3, 3, 3, 3, 4, 3, 4, 4, 4, 3, 3, 3, 4, 4, 4, 4, 3, 4, 4, 3, 3,
       4, 4, 4, 3, 3, 3, 4, 4, 4, 4, 3, 4, 3, 4, 4, 4, 3, 4, 4, 3, 5, 4,
       3, 4, 4, 4, 4, 4, 4, 3, 4, 3, 4, 4, 3, 4, 3, 3, 4, 3, 4, 3, 3, 3,
       3, 3, 3, 4, 3, 4, 3, 4, 4, 3, 3, 3, 3, 3, 4, 3, 3, 4, 4, 4, 3, 3,
       3, 4, 3, 4, 3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 3, 3, 3, 3, 4, 3, 4, 5,
       4, 3, 3, 4, 3, 4, 4, 3, 4, 3, 3, 4, 4, 4, 3, 4, 3, 3])
```

```
print(metrics.accuracy_score(y_pred_train3,y_train))
```

```
0.6014298480786416
```

```
print(metrics.accuracy_score(y_pred_test3,y_test))
```

```
0.5375
```

```
"""#### Log_r4"""

#________multiclass= multinomial, solver = sag

Log_r4 = LogisticRegression(multi_class='multinomial',solver='sag')

Log_r4 = Log_r4.fit(X_train, y_train)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:985: DataConversic
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_sag.py:354: ConvergenceW
  ConvergenceWarning,
```

```
y_pred_train4 = Log_r4.predict(X_train)
y_pred_train4
```

```
array([4, 4, 4, ..., 4, 3, 3])
```

```
y_pred_test4 = Log_r4.predict(X_test)
y_pred_test4
```

```
array([4, 3, 4, 3, 4, 4, 3, 3, 3, 4, 3, 3, 4, 3, 3, 3, 4, 3, 4, 4, 4, 3,
       4, 3, 4, 4, 4, 3, 3, 4, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 3, 4,
       3, 4, 3, 4, 4, 4, 4, 3, 4, 3, 3, 4, 3, 3, 3, 4, 4, 3, 4, 3, 4, 3,
       3, 3, 3, 4, 3, 4, 4, 3, 4, 3, 4, 4, 4, 4, 3, 4, 4, 3, 3, 4, 3, 3,
       3, 4, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 3, 4, 3, 3, 4, 3,
       3, 3, 4, 3, 3, 4, 4, 3, 4, 3, 3, 3, 4, 4, 3, 3, 3, 3, 3, 3, 4, 4,
       4, 4, 4, 4, 3, 3, 4, 3, 4, 4, 4, 3, 3, 4, 4, 4, 3, 4, 3, 4, 4, 3,
       4, 3, 3, 3, 3, 3, 3, 3, 4, 4, 3, 3, 3, 4, 4, 3, 3, 4, 3, 3, 3, 4,
       3, 4, 3, 3, 4, 3, 3, 3, 4, 4, 3, 4, 3, 4, 4, 4, 3, 3, 3, 4, 3, 3,
       4, 3, 4, 3, 3, 3, 4, 3, 4, 4, 4, 4, 3, 4, 3, 4, 4, 4, 4, 3, 3, 4,
       3, 4, 4, 3, 4, 4, 3, 3, 4, 3, 3, 4, 4, 4, 4, 3, 3, 4, 3, 3, 4, 3,
       4, 3, 3, 4, 4, 4, 4, 4, 3, 4, 3, 3, 4, 4, 4, 3, 3, 3, 3, 3, 3, 3,
       3, 4, 3, 3, 4, 3, 4, 4, 3, 4, 3, 3, 3, 3, 3, 4, 3, 3, 4, 3, 3, 4,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 4, 3, 4, 3, 4, 4, 3, 3, 3, 4, 3, 3, 4,
       4, 4, 4, 4, 3, 3, 3, 3, 4, 3, 4, 4, 4, 4, 4, 3, 4, 3, 3, 4, 3, 3,
       4, 3, 4, 3, 4, 3, 4, 3, 3, 3, 3, 3, 3, 3, 3, 4, 3, 3, 4, 4, 3, 4,
       3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 3, 3, 3, 4, 4, 3, 4, 3, 4, 4, 3, 3,
       4, 4, 4, 3, 3, 3, 4, 4, 4, 4, 3, 3, 3, 4, 4, 4, 3, 4, 4, 3, 4, 4,
       3, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 4, 3, 3, 3, 3, 4, 3, 4, 3, 3, 3,
       3, 3, 3, 4, 3, 4, 3, 4, 4, 3, 3, 3, 3, 4, 4, 3, 3, 4, 4, 4, 3, 3,
       3, 4, 3, 4, 3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 3, 3, 3, 3, 4, 3, 4, 4,
       4, 3, 3, 4, 3, 4, 4, 3, 4, 3, 3, 4, 4, 4, 3, 3, 3, 3])
```

```
print(metrics.accuracy_score(y_pred_train4,y_train))
```

```
0.5978552278820375
```

```
print(metrics.accuracy_score(y_pred_test4,y_test))
```

```
0.50625
```

```
"""#### Log_5"""

#________multiclass= multinomial, solver = saga

Log_r5 = LogisticRegression(multi_class='multinomial',solver='saga')

Log_r5 = Log_r5.fit(X_train, y_train)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:985: DataConversic
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_sag.py:354: ConvergenceW
  ConvergenceWarning,
```

```
y_pred_train5 = Log_r5.predict(X_train)
y_pred_train5
```

```
array([3, 4, 4, ..., 4, 3, 3])
```

```
y_pred_test5 = Log_r5.predict(X_test)
y_pred_test5
```

```
array([4, 3, 4, 3, 4, 4, 3, 3, 3, 3, 3, 3, 4, 3, 3, 3, 4, 3, 4, 4, 4, 3,
       4, 3, 4, 4, 4, 3, 3, 4, 3, 3, 4, 3, 4, 4, 4, 4, 4, 4, 4, 3, 3, 4,
       3, 4, 3, 4, 4, 4, 4, 3, 4, 3, 3, 4, 3, 3, 4, 4, 4, 3, 4, 3, 4, 4,
       3, 3, 3, 4, 3, 4, 4, 3, 4, 3, 3, 4, 4, 4, 3, 4, 4, 4, 3, 4, 3, 3,
       3, 4, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 3, 3, 4, 3,
       3, 3, 4, 3, 4, 4, 4, 3, 4, 3, 4, 4, 4, 4, 3, 3, 3, 3, 3, 3, 4, 4,
       4, 4, 4, 4, 3, 3, 3, 3, 4, 4, 4, 3, 3, 4, 4, 4, 3, 4, 3, 4, 4, 3,
       4, 3, 3, 3, 4, 4, 3, 3, 4, 4, 4, 3, 3, 4, 4, 3, 3, 4, 3, 4, 3, 4,
       3, 4, 3, 3, 4, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 3, 3, 3, 4, 3, 3,
       4, 3, 4, 4, 3, 3, 3, 3, 4, 4, 4, 4, 3, 4, 3, 4, 4, 4, 4, 3, 3, 4,
       3, 4, 4, 3, 4, 4, 3, 3, 4, 3, 3, 4, 4, 4, 4, 3, 3, 4, 3, 3, 4, 3,
       4, 3, 3, 4, 4, 4, 4, 4, 3, 4, 3, 3, 4, 4, 4, 3, 3, 3, 3, 3, 3, 3,
       3, 4, 4, 3, 4, 3, 4, 4, 3, 4, 3, 3, 3, 3, 4, 4, 3, 3, 4, 3, 3, 4,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 4, 3, 4, 3, 4, 4, 3, 3, 3, 4, 3, 3, 4,
       4, 4, 4, 4, 3, 3, 3, 3, 4, 3, 4, 4, 3, 4, 4, 3, 4, 3, 3, 4, 3, 3,
       4, 3, 4, 4, 4, 3, 4, 3, 3, 3, 3, 3, 3, 3, 3, 4, 3, 3, 4, 4, 3, 4,
       3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 3, 3, 3, 4, 4, 3, 4, 3, 4, 4, 3, 3,
       4, 4, 4, 3, 3, 3, 4, 4, 4, 4, 3, 3, 3, 4, 4, 4, 3, 4, 4, 3, 4, 4,
       3, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 4, 3, 3, 3, 3, 3, 4, 4, 4, 3, 3,
       4, 3, 3, 4, 3, 4, 3, 4, 4, 3, 3, 3, 4, 4, 4, 3, 3, 4, 4, 4, 3, 3,
       4, 4, 3, 4, 3, 3, 3, 4, 3, 3, 3, 3, 4, 4, 3, 3, 4, 3, 4, 3, 4, 4,
       4, 3, 3, 4, 3, 4, 4, 3, 4, 4, 3, 4, 4, 4, 3, 3, 3, 3])
```

```
print(metrics.accuracy_score(y_pred_train5,y_train))
```

```
0.5630026809651475
```

```
print(metrics.accuracy_score(y_pred_test5,y_test))
```

```
0.5020833333333333
```

```
#From the above experiment, We got the best accuracy from ----
# -----> Log_r2, i.e
# (multi_class='multinomial',solver='newton-cg')
#So let's find best parameters using Grid Search & CV
```

```
"""#### Grid Search (Log_r)"""
#_________gridsearch (Log_R)
from sklearn.model_selection import GridSearchCV
```

```
penalty = ['l1','l2']
max_iter=[80, 100, 140]
C = np.linspace(0.1, 1.0, num=5)
```

```
X_train.head()
```

	fixed_acidity	volatile_acidity	citric_acid	residual_sugar	chlorides	free_
318	9.8	0.660	0.39	3.2	0.083	
1364	7.2	0.605	0.02	1.9	0.096	
1007	9.1	0.300	0.34	2.0	0.064	
248	7.7	0.530	0.06	1.7	0.074	
782	9.0	0.820	0.05	2.4	0.081	

```
param_grid = dict(max_iter=max_iter, C=C, penalty=penalty)
```

```
Log_r_gs = LogisticRegression(multi_class='multinomial',solver='newton-cg')
```

```
g_search = GridSearchCV(estimator = Log_r_gs, param_grid = param_grid, cv = 5)
```

```
g_mod = g_search.fit(X_train, y_train)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:985: DataConv
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:985: DataConv
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:985: DataConv
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:985: DataConv
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:985: DataConv
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:985: DataConv
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:985: DataConv
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:985: DataConv
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:985: DataConv
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:985: DataConv
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:985: DataConv
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:985: DataConv
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:985: DataConv
  y = column_or_1d(y, warn=True)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:985: DataConv
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:985: DataConv
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:985: DataConv
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:985: DataConv
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:985: DataConv
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:985: DataConv
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:985: DataConv
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:985: DataConv
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:985: DataConv
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:985: DataConv
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:985: DataConv
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:985: DataConv
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:985: DataConv
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:985: DataConv
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:985: DataConv
```

```
y_pred_test_gs = g_mod.predict(X_test)
y_pred_test_gs
```

```
array([3, 3, 4, 3, 4, 4, 3, 4, 3, 4, 3, 3, 5, 3, 3, 3, 4, 3, 3, 4, 4, 3,
       4, 3, 4, 4, 4, 3, 3, 4, 3, 3, 4, 4, 4, 5, 4, 4, 3, 4, 4, 3, 3, 4,
       3, 4, 3, 4, 4, 4, 4, 3, 4, 3, 3, 4, 3, 3, 3, 3, 4, 3, 4, 3, 4, 3,
       3, 4, 3, 4, 3, 3, 4, 4, 4, 3, 4, 4, 4, 4, 3, 4, 4, 3, 3, 4, 3, 3,
       3, 4, 4, 3, 5, 4, 4, 3, 4, 4, 3, 4, 4, 4, 4, 3, 3, 4, 3, 4, 4, 3,
       3, 3, 4, 4, 3, 3, 5, 3, 3, 3, 3, 3, 4, 3, 3, 3, 3, 3, 3, 3, 4, 4,
       4, 4, 4, 4, 3, 3, 4, 3, 4, 4, 4, 3, 3, 3, 3, 4, 3, 4, 3, 4, 4, 3,
       4, 3, 3, 3, 3, 3, 4, 3, 4, 4, 3, 3, 3, 4, 4, 3, 3, 3, 3, 3, 3, 4,
       3, 4, 4, 3, 4, 3, 3, 3, 4, 4, 3, 4, 3, 3, 4, 3, 3, 3, 3, 5, 3, 3,
       4, 3, 4, 3, 3, 3, 4, 3, 4, 4, 4, 4, 3, 4, 3, 4, 4, 4, 4, 4, 3, 4,
       3, 4, 4, 3, 4, 5, 3, 3, 3, 3, 3, 4, 4, 4, 4, 3, 3, 4, 3, 3, 4, 3,
       4, 3, 3, 4, 3, 4, 4, 4, 3, 4, 3, 3, 4, 4, 4, 3, 3, 3, 3, 3, 3, 3,
       3, 4, 3, 4, 4, 3, 4, 4, 3, 4, 3, 3, 3, 3, 3, 4, 3, 3, 4, 3, 3, 4,
       3, 4, 3, 4, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 3, 4, 4, 4, 4, 3, 3, 4,
       4, 4, 4, 4, 3, 3, 3, 3, 4, 3, 4, 4, 4, 4, 4, 4, 3, 3, 3, 4, 4, 3,
       3, 3, 4, 3, 4, 3, 4, 3, 4, 4, 3, 3, 3, 3, 3, 4, 3, 3, 4, 4, 3, 4,
       3, 3, 3, 3, 3, 4, 3, 4, 4, 4, 3, 3, 3, 4, 4, 4, 4, 4, 4, 3, 4, 3,
       4, 4, 4, 3, 3, 3, 5, 5, 4, 4, 3, 3, 3, 5, 4, 4, 3, 4, 4, 3, 5, 4,
       3, 4, 5, 4, 4, 4, 4, 4, 4, 3, 4, 4, 3, 4, 3, 3, 4, 3, 4, 3, 3, 3,
       4, 3, 4, 4, 3, 3, 3, 4, 4, 3, 3, 3, 3, 3, 4, 3, 3, 4, 4, 4, 4, 4,
       4, 4, 3, 4, 3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 3, 3, 4, 3, 4, 3, 4, 4,
       4, 3, 3, 4, 3, 4, 4, 3, 4, 3, 3, 4, 5, 4, 3, 3, 3, 3])
```

```
print("Best accuracy : %f using %s" % (g_mod.best_score_, g_mod.best_params_))
```

```
Best accuracy : 0.606750 using {'C': 0.775, 'max_iter': 80, 'penalty': 'l2'}
```

```
Log_r_best = LogisticRegression(multi_class='multinomial',solver='newton-cg',
                                C= 0.775, max_iter= 80, penalty= 'l2')
```

```
Log_r_best = Log_r_best.fit(X_train, y_train)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:985: DataConversic
  y = column_or_1d(y, warn=True)
```

```
y_pred_train_best = Log_r_best.predict(X_train)
y_pred_train_best
```

```
array([4, 4, 4, ..., 4, 3, 4])
```

```
y_pred_test_best = Log_r_best.predict(X_test)
y_pred_test_best
```

```
array([3, 3, 4, 3, 4, 4, 3, 4, 3, 4, 3, 3, 5, 3, 3, 3, 4, 3, 3, 4, 4, 3,
       4, 3, 4, 4, 4, 3, 3, 4, 3, 3, 4, 4, 4, 5, 4, 4, 3, 4, 4, 3, 3, 4,
       3, 4, 3, 4, 4, 4, 4, 3, 4, 3, 3, 4, 3, 3, 3, 3, 4, 3, 4, 3, 4, 3,
       3, 4, 3, 4, 3, 3, 4, 4, 4, 3, 4, 4, 4, 4, 3, 4, 4, 3, 3, 4, 3, 3,
       3, 4, 4, 3, 5, 4, 4, 3, 4, 4, 3, 4, 4, 4, 4, 3, 3, 4, 3, 4, 4, 3,
       3, 3, 4, 4, 3, 3, 5, 3, 3, 3, 3, 3, 4, 3, 3, 3, 3, 3, 3, 3, 4, 4,
       4, 4, 4, 4, 3, 3, 4, 3, 4, 4, 4, 3, 3, 3, 3, 4, 3, 4, 3, 4, 4, 3,
       4, 3, 3, 3, 3, 3, 4, 3, 4, 4, 3, 3, 3, 4, 4, 3, 3, 3, 3, 3, 3, 4,
       3, 4, 4, 3, 4, 3, 3, 3, 4, 4, 3, 4, 3, 3, 4, 3, 3, 3, 3, 5, 3, 3,
       4, 3, 4, 3, 3, 3, 4, 3, 4, 4, 4, 4, 3, 4, 3, 4, 4, 4, 4, 4, 3, 4,
       3, 4, 4, 3, 4, 5, 3, 3, 3, 3, 3, 4, 4, 4, 4, 3, 3, 4, 3, 3, 4, 3,
       4, 3, 3, 4, 3, 4, 4, 4, 3, 4, 3, 3, 4, 4, 4, 3, 3, 3, 3, 3, 3, 3,
       3, 4, 3, 4, 4, 3, 4, 4, 3, 4, 3, 3, 3, 3, 3, 4, 3, 3, 4, 3, 3, 4,
       3, 4, 3, 4, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 3, 4, 4, 4, 4, 3, 3, 4,
       4, 4, 4, 4, 3, 3, 3, 3, 4, 3, 4, 4, 4, 4, 4, 4, 3, 3, 3, 4, 4, 3,
       3, 3, 4, 3, 4, 3, 4, 3, 4, 4, 3, 3, 3, 3, 3, 4, 3, 3, 4, 4, 3, 4,
       3, 3, 3, 3, 3, 4, 3, 4, 4, 4, 3, 3, 3, 4, 4, 4, 4, 4, 4, 3, 4, 3,
       4, 4, 4, 3, 3, 3, 5, 5, 4, 4, 3, 3, 3, 5, 4, 4, 3, 4, 4, 3, 5, 4,
       3, 4, 5, 4, 4, 4, 4, 4, 4, 3, 4, 4, 3, 4, 3, 3, 4, 3, 4, 3, 3, 3,
       4, 3, 4, 4, 3, 3, 3, 4, 4, 3, 3, 3, 3, 3, 4, 3, 3, 4, 4, 4, 4, 4,
       4, 4, 3, 4, 3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 3, 3, 4, 3, 4, 3, 4, 4,
       4, 3, 3, 4, 3, 4, 4, 3, 4, 3, 3, 4, 5, 4, 3, 3, 3, 3])
```

```
print(metrics.accuracy_score(y_pred_train_best,y_train))
```

```
0.612153708668454
```

```
print(metrics.accuracy_score(y_pred_test_best,y_test))
```

```
0.5541666666666667
```

Decision Tree Model

```
#Importing algorithm
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
```

```
dt_clf = DecisionTreeClassifier()
```

```
#Fit this classifier model on train data set
dt_clf.fit(X_train, y_train)
```

```
DecisionTreeClassifier()
```

```
y_pred_train_dt = dt_clf.predict(X_train)
```

```
y_pred_test_dt = dt_clf.predict(X_test)
```

```
# see the tree
from sklearn import tree
tree.plot_tree(dt_clf.fit(X_train, y_train))
```

```
[Text(148.51032490632974, 212.49818181818182, 'X[10] <= 10.25\ngini = 0.636\nsamples
 Text(43.0692626953125, 202.61454545454546, 'X[1] <= 0.317\ngini = 0.504\nsamples = 6
 Text(10.818367346938777, 192.73090909090908, 'X[6] <= 79.0\ngini = 0.55\nsamples = 4
 Text(9.679591836734694, 182.84727272727272, 'X[4] <= 0.104\ngini = 0.489\nsamples =
 Text(6.832653061224491, 172.96363636363637, 'X[0] <= 11.7\ngini = 0.42\nsamples = 35
 Text(4.555102040816327, 163.07999999999998, 'X[7] <= 0.996\ngini = 0.261\nsamples =
 Text(3.4163265306122454, 153.19636363636363, 'gini = 0.0\nsamples = 1\nvalue = [0, 0
 Text(5.6938775510204085, 153.19636363636363, 'X[7] <= 1.0\ngini = 0.204\nsamples = 2
 Text(4.555102040816327, 143.31272727272727, 'X[6] <= 47.5\ngini = 0.147\nsamples = 2
 Text(2.2775510204081635, 133.42909090909092, 'X[4] <= 0.089\ngini = 0.087\nsamples =
 Text(1.1387755102040817, 123.54545454545455, 'gini = 0.0\nsamples = 18\nvalue = [0,
 Text(3.4163265306122454, 123.54545454545455, 'X[5] <= 5.0\ngini = 0.375\nsamples = 4
 Text(2.2775510204081635, 113.66181818181819, 'gini = 0.0\nsamples = 3\nvalue = [0, 0
 Text(4.555102040816327, 113.66181818181819, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
 Text(6.832653061224491, 133.42909090909092, 'X[10] <= 9.85\ngini = 0.444\nsamples =
 Text(5.6938775510204085, 123.54545454545455, 'gini = 0.0\nsamples = 1\nvalue = [0, 0
 Text(7.971428571428572, 123.54545454545455, 'gini = 0.0\nsamples = 2\nvalue = [0, 0,
 Text(6.832653061224491, 143.31272727272727, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
 Text(9.110204081632654, 163.07999999999998, 'X[1] <= 0.265\ngini = 0.594\nsamples =
 Text(7.971428571428572, 153.19636363636363, 'gini = 0.0\nsamples = 3\nvalue = [0, 0,
 Text(10.248979591836736, 153.19636363636363, 'X[9] <= 0.81\ngini = 0.56\nsamples = 5
 Text(9.110204081632654, 143.31272727272727, 'gini = 0.0\nsamples = 3\nvalue = [0, 0,
 Text(11.387755102040817, 143.31272727272727, 'X[7] <= 0.999\ngini = 0.5\nsamples = 2
 Text(10.248979591836736, 133.42909090909092, 'gini = 0.0\nsamples = 1\nvalue = [0, 0
 Text(12.5265306122449, 133.42909090909092, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
 Text(12.5265306122449, 172.96363636363637, 'X[9] <= 0.815\ngini = 0.444\nsamples = 3
 Text(11.387755102040817, 163.07999999999998, 'gini = 0.0\nsamples = 2\nvalue = [0, 0
 Text(13.665306122448982, 163.07999999999998, 'gini = 0.0\nsamples = 1\nvalue = [0, 0
 Text(11.957142857142859, 182.84727272727272, 'gini = 0.0\nsamples = 4\nvalue = [0, 0
 Text(75.32015804368623, 192.73090909090908, 'X[9] <= 0.535\ngini = 0.475\nsamples =
 Text(30.284311224489798, 182.84727272727272, 'X[10] <= 9.55\ngini = 0.325\nsamples =
 Text(21.636734693877553, 172.96363636363637, 'X[10] <= 9.15\ngini = 0.167\nsamples =
 Text(17.081632653061227, 163.07999999999998, 'X[9] <= 0.485\ngini = 0.593\nsamples =
 Text(14.804081632653062, 153.19636363636363, 'X[9] <= 0.45\ngini = 0.444\nsamples =
 Text(13.665306122448982, 143.31272727272727, 'gini = 0.0\nsamples = 1\nvalue = [0, 1
 Text(15.942857142857145, 143.31272727272727, 'gini = 0.0\nsamples = 2\nvalue = [0, 0
 Text(19.35918367346939, 153.19636363636363, 'X[3] <= 3.125\ngini = 0.278\nsamples =
 Text(18.220408163265308, 143.31272727272727, 'gini = 0.0\nsamples = 5\nvalue = [0, 0
 Text(20.497959183673473, 143.31272727272727, 'gini = 0.0\nsamples = 1\nvalue = [0, 1
 Text(26.19183673469388, 163.07999999999998, 'X[0] <= 6.55\ngini = 0.084\nsamples = 6
 Text(23.914285714285718, 153.19636363636363, 'X[1] <= 0.45\ngini = 0.32\nsamples = 5
 Text(22.775510204081634, 143.31272727272727, 'gini = 0.0\nsamples = 1\nvalue = [0, 0
 Text(25.0530612244898, 143.31272727272727, 'gini = 0.0\nsamples = 4\nvalue = [0, 0,
 Text(28.469387755102044, 153.19636363636363, 'X[10] <= 9.35\ngini = 0.061\nsamples =
 Text(27.330612244897964, 143.31272727272727, 'X[7] <= 0.998\ngini = 0.208\nsamples =
 Text(26.19183673469388, 133.42909090909092, 'gini = 0.0\nsamples = 12\nvalue = [0, 0
 Text(28.469387755102044, 133.42909090909092, 'X[8] <= 3.225\ngini = 0.48\nsamples =
 Text(27.330612244897964, 123.54545454545455, 'gini = 0.0\nsamples = 3\nvalue = [0, 0
 Text(29.608163265306125, 123.54545454545455, 'gini = 0.0\nsamples = 2\nvalue = [0, 2
 Text(29.608163265306125, 143.31272727272727, 'gini = 0.0\nsamples = 47\nvalue = [0,
 Text(38.931887755102046, 172.96363636363637, 'X[1] <= 0.465\ngini = 0.474\nsamples =
 Text(35.58673469387755, 163.07999999999998, 'X[2] <= 0.245\ngini = 0.444\nsamples =
 Text(34.447959183673476, 153.19636363636363, 'gini = 0.0\nsamples = 4\nvalue = [0, 0
 Text(36.72551020408164, 153.19636363636363, 'gini = 0.0\nsamples = 2\nvalue = [0, 0,
 Text(42.27704081632653, 163.07999999999998, 'X[10] <= 9.85\ngini = 0.438\nsamples =
 Text(39.0030612244898, 153.19636363636363, 'X[6] <= 88.5\ngini = 0.579\nsamples = 31
 Text(35.871428571428574, 143.31272727272727, 'X[8] <= 3.265\ngini = 0.496\nsamples =
 Text(33.02448979591837, 133.42909090909092, 'X[4] <= 0.08\ngini = 0.142\nsamples = 1
 Text(31.88571428571429, 123.54545454545455, 'X[7] <= 0.996\ngini = 0.375\nsamples =
 Text(30.746938775510205, 113.66181818181819, 'gini = 0.0\nsamples = 3\nvalue = [0, 0
```

```
Text(33.02448979591837, 113.66181818181819, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(34.163265306122454, 123.54545454545455, 'gini = 0.0\nsamples = 9\nvalue = [0, 0
Text(38.71836734693878, 133.42909090909092, 'X[2] <= 0.055\ngini = 0.681\nsamples =
Text(36.440816326530616, 123.54545454545455, 'X[7] <= 0.995\ngini = 0.375\nsamples =
Text(35.30204081632653, 113.66181818181819, 'gini = 0.0\nsamples = 1\nvalue = [1, 0,
Text(37.5795918367347, 113.66181818181819, 'gini = 0.0\nsamples = 3\nvalue = [0, 3,
Text(40.995918367346945, 123.54545454545455, 'X[3] <= 1.75\ngini = 0.531\nsamples =
Text(39.85714285714286, 113.66181818181819, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(42.13469387755102, 113.66181818181819, 'X[8] <= 3.275\ngini = 0.449\nsamples =
Text(40.995918367346945, 103.77818181818182, 'gini = 0.0\nsamples = 1\nvalue = [0, 0
Text(43.27346938775511, 103.77818181818182, 'X[2] <= 0.265\ngini = 0.278\nsamples =
Text(42.13469387755102, 93.89454545454547, 'gini = 0.0\nsamples = 4\nvalue = [0, 0,
Text(44.41224489795919, 93.89454545454547, 'X[2] <= 0.275\ngini = 0.5\nsamples = 2\n
Text(43.27346938775511, 84.01090909090911, 'gini = 0.0\nsamples = 1\nvalue = [0, 1,
Text(45.55102040816327, 84.01090909090911, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(42.13469387755102, 143.31272727272727, 'X[0] <= 6.9\ngini = 0.278\nsamples = 6\
Text(40.995918367346945, 133.42909090909092, 'gini = 0.0\nsamples = 1\nvalue = [0, 0
Text(43.27346938775511, 133.42909090909092, 'gini = 0.0\nsamples = 5\nvalue = [0, 0,
Text(45.55102040816327, 153.19636363636363, 'X[1] <= 0.51\ngini = 0.199\nsamples = 2
Text(44.41224489795919, 143.31272727272727, 'gini = 0.0\nsamples = 1\nvalue = [0, 1,
Text(46.68979591836735, 143.31272727272727, 'X[3] <= 1.4\ngini = 0.14\nsamples = 27\
Text(45.55102040816327, 133.42909090909092, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(47.828571428571436, 133.42909090909092, 'X[3] <= 3.475\ngini = 0.074\nsamples =
Text(46.68979591836735, 123.54545454545455, 'gini = 0.0\nsamples = 23\nvalue = [0, 0
Text(48.96734693877551, 123.54545454545455, 'X[0] <= 7.7\ngini = 0.444\nsamples = 3\
Text(47.828571428571436, 113.66181818181819, 'gini = 0.0\nsamples = 2\nvalue = [0, 0
Text(50.1061224489796, 113.66181818181819, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(120.35600486288267, 182.84727272727272, 'X[0] <= 9.95\ngini = 0.507\nsamples =
Text(102.06609135841838, 172.96363636363637, 'X[6] <= 91.5\ngini = 0.468\nsamples =
Text(83.99136639030613, 163.07999999999998, 'X[1] <= 0.385\ngini = 0.499\nsamples =
Text(65.87104591836736, 153.19636363636363, 'X[6] <= 53.0\ngini = 0.671\nsamples = 2
Text(61.31594387755103, 143.31272727272727, 'X[10] <= 9.5\ngini = 0.716\nsamples = 9
Text(59.03839285714286, 133.42909090909092, 'X[4] <= 0.111\ngini = 0.32\nsamples = 5
Text(57.89961734693878, 123.54545454545455, 'gini = 0.0\nsamples = 4\nvalue = [0, 0,
Text(60.177168367346944, 123.54545454545455, 'gini = 0.0\nsamples = 1\nvalue = [0, 0
Text(63.59349489795919, 133.42909090909092, 'X[3] <= 2.05\ngini = 0.625\nsamples = 4
Text(62.454719387755105, 123.54545454545455, 'X[5] <= 15.5\ngini = 0.5\nsamples = 2\
Text(61.31594387755103, 113.66181818181819, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(63.59349489795919, 113.66181818181819, 'gini = 0.0\nsamples = 1\nvalue = [0, 1,
Text(64.73227040816327, 123.54545454545455, 'gini = 0.0\nsamples = 2\nvalue = [0, 0,
Text(70.42614795918368, 143.31272727272727, 'X[6] <= 82.5\ngini = 0.403\nsamples = 1
Text(68.14859693877551, 133.42909090909092, 'X[4] <= 0.069\ngini = 0.18\nsamples = 1
Text(67.00982142857144, 123.54545454545455, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(69.2873724489796, 123.54545454545455, 'gini = 0.0\nsamples = 9\nvalue = [0, 0,
Text(72.70369897959185, 133.42909090909092, 'X[2] <= 0.51\ngini = 0.5\nsamples = 2\n
Text(71.56492346938776, 123.54545454545455, 'gini = 0.0\nsamples = 1\nvalue = [0, 1,
Text(73.84247448979592, 123.54545454545455, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(102.11168686224491, 153.19636363636363, 'X[4] <= 0.098\ngini = 0.473\nsamples =
Text(91.48459821428573, 143.31272727272727, 'X[4] <= 0.091\ngini = 0.499\nsamples =
Text(82.75695153061226, 133.42909090909092, 'X[0] <= 8.65\ngini = 0.47\nsamples = 19
Text(76.12002551020409, 123.54545454545455, 'X[7] <= 0.998\ngini = 0.497\nsamples =
Text(66.2625, 113.66181818181819, 'X[2] <= 0.16\ngini = 0.473\nsamples = 153\nvalue
Text(58.5045918367347, 103.77818181818182, 'X[1] <= 0.97\ngini = 0.4\nsamples = 95\n
Text(53.807142857142864, 93.89454545454547, 'X[4] <= 0.075\ngini = 0.359\nsamples =
Text(47.828571428571436, 84.01090909090911, 'X[6] <= 52.5\ngini = 0.518\nsamples = 3
Text(46.68979591836735, 74.12727272727273, 'X[0] <= 6.55\ngini = 0.566\nsamples = 25
Text(45.55102040816327, 64.24363636363637, 'gini = 0.0\nsamples = 5\nvalue = [0, 0,
Text(47.828571428571436, 64.24363636363637, 'X[8] <= 3.475\ngini = 0.585\nsamples =
Text(46.68979591836735, 54.360000000000014, 'X[7] <= 0.996\ngini = 0.561\nsamples =
Text(44.41224489795919, 44.47636363636366, 'X[0] <= 6.65\ngini = 0.32\nsamples = 5\n
```

```
Text(43.27346938775511, 34.592727272727274, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(45.55102040816327, 34.592727272727274, 'gini = 0.0\nsamples = 4\nvalue = [0, 0,
Text(48.96734693877551, 44.47636363636366, 'X[10] <= 9.3\ngini = 0.417\nsamples = 12
Text(47.828571428571436, 34.592727272727274, 'gini = 0.0\nsamples = 1\nvalue = [0, 1
Text(50.1061224489796, 34.592727272727274, 'X[1] <= 0.67\ngini = 0.314\nsamples = 11
Text(47.828571428571436, 24.709090909090918, 'X[9] <= 0.605\ngini = 0.198\nsamples =
Text(46.68979591836735, 14.825454545454562, 'X[3] <= 2.05\ngini = 0.5\nsamples = 2\n
Text(45.55102040816327, 4.941818181818206, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(47.828571428571436, 4.941818181818206, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(48.96734693877551, 14.825454545454562, 'gini = 0.0\nsamples = 7\nvalue = [0, 0,
Text(52.38367346938776, 24.709090909090918, 'X[10] <= 10.0\ngini = 0.5\nsamples = 2\
Text(51.24489795918368, 14.825454545454562, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(53.52244897959184, 14.825454545454562, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(48.96734693877551, 54.360000000000014, 'gini = 0.0\nsamples = 3\nvalue = [0, 0,
Text(48.96734693877551, 74.12727272727273, 'gini = 0.0\nsamples = 6\nvalue = [0, 0,
Text(59.78571428571429, 84.01090909090911, 'X[7] <= 0.995\ngini = 0.238\nsamples = 6
Text(58.64693877551021, 74.12727272727273, 'gini = 0.0\nsamples = 1\nvalue = [0, 1,
Text(60.924489795918376, 74.12727272727273, 'X[0] <= 7.65\ngini = 0.213\nsamples = 5
Text(58.077551020408166, 64.24363636363637, 'X[6] <= 87.0\ngini = 0.092\nsamples = 4
Text(55.800000000000004, 54.360000000000014, 'X[10] <= 10.15\ngini = 0.049\nsamples
Text(54.66122448979593, 44.47636363636366, 'gini = 0.0\nsamples = 35\nvalue = [0, 0,
Text(56.93877551020409, 44.47636363636366, 'X[9] <= 0.64\ngini = 0.32\nsamples = 5\n
Text(55.800000000000004, 34.592727272727274, 'X[4] <= 0.081\ngini = 0.5\nsamples = 2
Text(54.66122448979593, 24.709090909090918, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(56.93877551020409, 24.709090909090918, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(58.077551020408166, 34.592727272727274, 'gini = 0.0\nsamples = 3\nvalue = [0, 0
Text(60.355102040816334, 54.360000000000014, 'X[9] <= 0.565\ngini = 0.5\nsamples = 2
Text(59.21632653061225, 44.47636363636366, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(61.49387755102041, 44.47636363636366, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(63.77142857142858, 64.24363636363637, 'X[8] <= 3.3\ngini = 0.415\nsamples = 17\
Text(62.632653061224495, 54.360000000000014, 'gini = 0.0\nsamples = 10\nvalue = [0,
Text(64.91020408163266, 54.360000000000014, 'X[5] <= 13.5\ngini = 0.408\nsamples = 7
Text(63.77142857142858, 44.47636363636366, 'X[7] <= 0.997\ngini = 0.444\nsamples = 3
Text(62.632653061224495, 34.592727272727274, 'gini = 0.0\nsamples = 1\nvalue = [0, 0
Text(64.91020408163266, 34.592727272727274, 'gini = 0.0\nsamples = 2\nvalue = [0, 0,
Text(66.04897959183674, 44.47636363636366, 'gini = 0.0\nsamples = 4\nvalue = [0, 0,
Text(63.20204081632654, 93.89454545454547, 'X[8] <= 3.335\ngini = 0.375\nsamples = 4
Text(62.06326530612245, 84.01090909090911, 'gini = 0.0\nsamples = 3\nvalue = [0, 0,
Text(64.34081632653061, 84.01090909090911, 'gini = 0.0\nsamples = 1\nvalue = [0, 1,
Text(74.02040816326532, 103.77818181818182, 'X[10] <= 9.45\ngini = 0.515\nsamples =
Text(68.89591836734695, 93.89454545454547, 'X[6] <= 37.5\ngini = 0.397\nsamples = 22
Text(66.61836734693878, 84.01090909090911, 'X[5] <= 5.5\ngini = 0.444\nsamples = 6\n
Text(65.4795918367347, 74.12727272727273, 'gini = 0.0\nsamples = 2\nvalue = [0, 0, 2
Text(67.75714285714287, 74.12727272727273, 'gini = 0.0\nsamples = 4\nvalue = [0, 0,
Text(71.1734693877551, 84.01090909090911, 'X[6] <= 85.5\ngini = 0.219\nsamples = 16\
Text(70.03469387755102, 74.12727272727273, 'X[9] <= 0.545\ngini = 0.124\nsamples = 1
Text(68.89591836734695, 64.24363636363637, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(71.1734693877551, 64.24363636363637, 'gini = 0.0\nsamples = 14\nvalue = [0, 0,
Text(72.31224489795919, 74.12727272727273, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(79.14489795918368, 93.89454545454547, 'X[1] <= 0.495\ngini = 0.508\nsamples = 3
Text(75.72857142857144, 84.01090909090911, 'X[0] <= 6.95\ngini = 0.524\nsamples = 15
Text(74.58979591836736, 74.12727272727273, 'gini = 0.0\nsamples = 3\nvalue = [0, 0,
Text(76.86734693877551, 74.12727272727273, 'X[5] <= 24.0\ngini = 0.403\nsamples = 12
Text(75.72857142857144, 64.24363636363637, 'gini = 0.0\nsamples = 7\nvalue = [0, 0,
Text(78.0061224489796, 64.24363636363637, 'X[7] <= 0.997\ngini = 0.64\nsamples = 5\n
Text(76.86734693877551, 54.360000000000014, 'gini = 0.0\nsamples = 2\nvalue = [0, 0,
Text(79.14489795918368, 54.360000000000014, 'X[6] <= 56.0\ngini = 0.444\nsamples = 3
Text(78.0061224489796, 44.47636363636366, 'gini = 0.0\nsamples = 1\nvalue = [0, 0, 0
Text(80.28367346938776, 44.47636363636366, 'gini = 0.0\nsamples = 2\nvalue = [0, 0,
Text(82.56122448979593, 84.01090909090911, 'X[0] <= 6.45\ngini = 0.363\nsamples = 21
```

```
Text(81.42244897959185, 74.12727272727273, 'gini = 0.0\nsamples = 2\nvalue = [0, 0,
Text(83.7, 74.12727272727273, 'X[3] <= 2.15\ngini = 0.266\nsamples = 19\nvalue = [0,
Text(82.56122448979593, 64.24363636363637, 'gini = 0.0\nsamples = 12\nvalue = [0, 0,
Text(84.83877551020409, 64.24363636363637, 'X[2] <= 0.215\ngini = 0.49\nsamples = 7\
Text(83.7, 54.360000000000014, 'X[9] <= 0.595\ngini = 0.375\nsamples = 4\nvalue = [0
Text(82.56122448979593, 44.47636363636366, 'gini = 0.0\nsamples = 3\nvalue = [0, 0,
Text(84.83877551020409, 44.47636363636366, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(85.97755102040817, 54.360000000000014, 'gini = 0.0\nsamples = 3\nvalue = [0, 0,
Text(85.97755102040817, 113.66181818181819, 'X[6] <= 68.0\ngini = 0.32\nsamples = 15
Text(84.83877551020409, 103.77818181818182, 'X[8] <= 3.64\ngini = 0.142\nsamples = 1
Text(83.7, 93.89454545454547, 'gini = 0.0\nsamples = 11\nvalue = [0, 0, 0, 11, 0, 0]
Text(85.97755102040817, 93.89454545454547, 'X[7] <= 0.999\ngini = 0.5\nsamples = 2\n
Text(84.83877551020409, 84.01090909090911, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(87.11632653061226, 84.01090909090911, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(87.11632653061226, 103.77818181818182, 'gini = 0.0\nsamples = 2\nvalue = [0, 0,
Text(89.39387755102042, 123.54545454545455, 'X[5] <= 4.5\ngini = 0.14\nsamples = 27\
Text(88.25510204081634, 113.66181818181819, 'gini = 0.0\nsamples = 1\nvalue = [0, 1,
Text(90.5326530612245, 113.66181818181819, 'X[9] <= 0.71\ngini = 0.074\nsamples = 26
Text(89.39387755102042, 103.77818181818182, 'gini = 0.0\nsamples = 25\nvalue = [0, 0
Text(91.67142857142858, 103.77818181818182, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(100.2122448979592, 133.42909090909092, 'X[6] <= 51.0\ngini = 0.535\nsamples = 2
Text(97.36530612244898, 123.54545454545455, 'X[8] <= 3.405\ngini = 0.41\nsamples = 1
Text(95.08775510204083, 113.66181818181819, 'X[7] <= 1.0\ngini = 0.24\nsamples = 15\
Text(93.94897959183675, 103.77818181818182, 'X[2] <= 0.3\ngini = 0.133\nsamples = 14
Text(92.81020408163266, 93.89454545454547, 'gini = 0.0\nsamples = 13\nvalue = [0, 0,
Text(95.08775510204083, 93.89454545454547, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(96.2265306122449, 103.77818181818182, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(99.64285714285715, 113.66181818181819, 'X[2] <= 0.18\ngini = 0.375\nsamples = 4
Text(98.50408163265307, 103.77818181818182, 'gini = 0.0\nsamples = 3\nvalue = [0, 0,
Text(100.78163265306124, 103.77818181818182, 'gini = 0.0\nsamples = 1\nvalue = [0, 0
Text(103.05918367346939, 123.54545454545455, 'X[9] <= 0.615\ngini = 0.54\nsamples =
Text(101.92040816326532, 113.66181818181819, 'gini = 0.0\nsamples = 5\nvalue = [0, 0
Text(104.19795918367348, 113.66181818181819, 'X[6] <= 62.0\ngini = 0.56\nsamples = 5
Text(103.05918367346939, 103.77818181818182, 'X[0] <= 7.95\ngini = 0.5\nsamples = 2\
Text(101.92040816326532, 93.89454545454547, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(104.19795918367348, 93.89454545454547, 'gini = 0.0\nsamples = 1\nvalue = [0, 1,
Text(105.33673469387756, 103.77818181818182, 'gini = 0.0\nsamples = 3\nvalue = [0, 0
Text(112.73877551020409, 143.31272727272727, 'X[5] <= 7.5\ngini = 0.292\nsamples = 5
Text(109.89183673469388, 133.42909090909092, 'X[7] <= 0.997\ngini = 0.595\nsamples =
Text(107.61428571428573, 123.54545454545455, 'X[1] <= 0.818\ngini = 0.245\nsamples =
Text(106.47551020408164, 113.66181818181819, 'gini = 0.0\nsamples = 6\nvalue = [0, 0
Text(108.75306122448981, 113.66181818181819, 'gini = 0.0\nsamples = 1\nvalue = [0, 0
Text(112.16938775510205, 123.54545454545455, 'X[8] <= 3.345\ngini = 0.375\nsamples =
Text(111.03061224489797, 113.66181818181819, 'gini = 0.0\nsamples = 3\nvalue = [0, 0
Text(113.30816326530613, 113.66181818181819, 'gini = 0.0\nsamples = 1\nvalue = [0, 0
Text(115.5857142857143, 133.42909090909092, 'X[8] <= 2.962\ngini = 0.172\nsamples =
Text(114.44693877551022, 123.54545454545455, 'gini = 0.0\nsamples = 1\nvalue = [0, 0
Text(116.72448979591837, 123.54545454545455, 'X[7] <= 0.996\ngini = 0.136\nsamples =
Text(115.5857142857143, 113.66181818181819, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(117.86326530612246, 113.66181818181819, 'X[6] <= 46.5\ngini = 0.095\nsamples =
Text(116.72448979591837, 103.77818181818182, 'X[6] <= 45.5\ngini = 0.245\nsamples =
Text(115.5857142857143, 93.89454545454547, 'X[10] <= 9.15\ngini = 0.142\nsamples = 1
Text(114.44693877551022, 84.01090909090911, 'X[5] <= 11.5\ngini = 0.5\nsamples = 2\n
Text(113.30816326530613, 74.12727272727273, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(115.5857142857143, 74.12727272727273, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(116.72448979591837, 84.01090909090911, 'gini = 0.0\nsamples = 11\nvalue = [0, 0
Text(117.86326530612246, 93.89454545454547, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(119.00204081632654, 103.77818181818182, 'gini = 0.0\nsamples = 26\nvalue = [0,
Text(120.14081632653063, 163.07999999999998, 'X[8] <= 3.005\ngini = 0.235\nsamples =
Text(117.86326530612246, 153.19636363636363, 'X[2] <= 0.285\ngini = 0.375\nsamples =
Text(116.72448979591837, 143.31272727272727, 'gini = 0.0\nsamples = 3\nvalue = [0, 0
```

```
Text(119.00204081632654, 143.31272727272727, 'gini = 0.0\nsamples = 1\nvalue = [0, 0
Text(122.4183673469388, 153.19636363636363, 'X[3] <= 1.6\ngini = 0.165\nsamples = 56
Text(121.27959183673471, 143.31272727272727, 'gini = 0.0\nsamples = 1\nvalue = [0, 1
Text(123.55714285714286, 143.31272727272727, 'X[4] <= 0.067\ngini = 0.135\nsamples =
Text(122.4183673469388, 133.42909090909092, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(124.69591836734695, 133.42909090909092, 'X[4] <= 0.076\ngini = 0.105\nsamples =
Text(121.27959183673471, 123.54545454545455, 'X[9] <= 0.555\ngini = 0.408\nsamples =
Text(120.14081632653063, 113.66181818181819, 'gini = 0.0\nsamples = 1\nvalue = [0, 0
Text(122.4183673469388, 113.66181818181819, 'X[6] <= 96.0\ngini = 0.278\nsamples = 6
Text(121.27959183673471, 103.77818181818182, 'gini = 0.0\nsamples = 1\nvalue = [0, 0
Text(123.55714285714286, 103.77818181818182, 'gini = 0.0\nsamples = 5\nvalue = [0, 0
Text(128.1122448979592, 123.54545454545455, 'X[6] <= 98.5\ngini = 0.042\nsamples = 4
Text(126.97346938775512, 113.66181818181819, 'X[6] <= 97.0\ngini = 0.198\nsamples =
Text(125.83469387755103, 103.77818181818182, 'gini = 0.0\nsamples = 7\nvalue = [0, 0
Text(128.1122448979592, 103.77818181818182, 'X[0] <= 8.25\ngini = 0.5\nsamples = 2\n
Text(126.97346938775512, 93.89454545454547, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(129.25102040816327, 93.89454545454547, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(129.25102040816327, 113.66181818181819, 'gini = 0.0\nsamples = 38\nvalue = [0,
Text(138.64591836734695, 172.96363636363637, 'X[2] <= 0.485\ngini = 0.622\nsamples =
Text(134.9448979591837, 163.07999999999998, 'X[1] <= 0.615\ngini = 0.482\nsamples =
Text(132.66734693877552, 153.19636363636363, 'X[9] <= 0.835\ngini = 0.426\nsamples =
Text(131.52857142857144, 143.31272727272727, 'X[9] <= 0.555\ngini = 0.381\nsamples =
Text(130.38979591836735, 133.42909090909092, 'gini = 0.0\nsamples = 1\nvalue = [0, 0
Text(132.66734693877552, 133.42909090909092, 'X[4] <= 0.062\ngini = 0.344\nsamples =
Text(131.52857142857144, 123.54545454545455, 'gini = 0.0\nsamples = 1\nvalue = [0, 0
Text(133.8061224489796, 123.54545454545455, 'X[3] <= 2.25\ngini = 0.299\nsamples = 2
Text(132.66734693877552, 113.66181818181819, 'gini = 0.0\nsamples = 10\nvalue = [0,
Text(134.9448979591837, 113.66181818181819, 'X[7] <= 0.999\ngini = 0.462\nsamples =
Text(132.66734693877552, 103.77818181818182, 'X[7] <= 0.998\ngini = 0.444\nsamples =
Text(131.52857142857144, 93.89454545454547, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(133.8061224489796, 93.89454545454547, 'gini = 0.0\nsamples = 2\nvalue = [0, 0,
Text(137.22244897959186, 103.77818181818182, 'X[0] <= 10.55\ngini = 0.18\nsamples =
Text(136.08367346938778, 93.89454545454547, 'gini = 0.0\nsamples = 8\nvalue = [0, 0,
Text(138.36122448979594, 93.89454545454547, 'X[1] <= 0.545\ngini = 0.5\nsamples = 2\
Text(137.22244897959186, 84.01090909090911, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(139.5, 84.01090909090911, 'gini = 0.0\nsamples = 1\nvalue = [0, 0, 0, 1, 0, 0]
Text(133.8061224489796, 143.31272727272727, 'gini = 0.0\nsamples = 1\nvalue = [1, 0,
Text(137.22244897959186, 153.19636363636363, 'X[4] <= 0.078\ngini = 0.375\nsamples =
Text(136.08367346938778, 143.31272727272727, 'gini = 0.0\nsamples = 1\nvalue = [0, 0
Text(138.36122448979594, 143.31272727272727, 'gini = 0.0\nsamples = 3\nvalue = [0, 0
Text(142.3469387755102, 163.07999999999998, 'X[3] <= 1.7\ngini = 0.669\nsamples = 30
Text(141.20816326530613, 153.19636363636363, 'gini = 0.0\nsamples = 2\nvalue = [0, 0
Text(143.4857142857143, 153.19636363636363, 'X[8] <= 3.035\ngini = 0.635\nsamples =
Text(142.3469387755102, 143.31272727272727, 'gini = 0.0\nsamples = 5\nvalue = [0, 0,
Text(144.62448979591838, 143.31272727272727, 'X[0] <= 12.125\ngini = 0.677\nsamples
Text(141.77755102040817, 133.42909090909092, 'X[1] <= 0.577\ngini = 0.602\nsamples =
Text(139.5, 123.54545454545455, 'X[5] <= 18.5\ngini = 0.32\nsamples = 10\nvalue = [0
Text(138.36122448979594, 113.66181818181819, 'gini = 0.0\nsamples = 7\nvalue = [0, 0
Text(140.63877551020408, 113.66181818181819, 'X[10] <= 9.55\ngini = 0.444\nsamples =
Text(139.5, 103.77818181818182, 'gini = 0.0\nsamples = 1\nvalue = [0, 0, 1, 0, 0, 0]
Text(141.77755102040817, 103.77818181818182, 'gini = 0.0\nsamples = 2\nvalue = [0, 0
Text(144.05510204081634, 123.54545454545455, 'X[10] <= 9.25\ngini = 0.722\nsamples =
Text(142.91632653061225, 113.66181818181819, 'gini = 0.0\nsamples = 2\nvalue = [2, 0
Text(145.19387755102042, 113.66181818181819, 'X[9] <= 0.595\ngini = 0.625\nsamples =
Text(144.05510204081634, 103.77818181818182, 'X[1] <= 0.585\ngini = 0.5\nsamples = 2
Text(142.91632653061225, 93.89454545454547, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(145.19387755102042, 93.89454545454547, 'gini = 0.0\nsamples = 1\nvalue = [0, 1,
Text(146.3326530612245, 103.77818181818182, 'gini = 0.0\nsamples = 2\nvalue = [0, 0,
Text(147.4714285714286, 133.42909090909092, 'X[3] <= 3.425\ngini = 0.408\nsamples =
Text(146.3326530612245, 123.54545454545455, 'gini = 0.0\nsamples = 5\nvalue = [0, 0,
```

```
Text(148.61020408163267, 123.54545454545455, 'gini = 0.0\nsamples = 2\nvalue = [0, 0
Text(253.95138711734697, 202.61454545454546, 'X[10] <= 11.45\ngini = 0.65\nsamples =
Text(207.94218750000002, 192.73090909090908, 'X[6] <= 97.0\ngini = 0.637\nsamples =
Text(206.80341198979593, 182.84727272727272, 'X[9] <= 0.645\ngini = 0.633\nsamples =
Text(169.4106505102041, 172.96363636363637, 'X[8] <= 3.165\ngini = 0.63\nsamples = 1
Text(153.16530612244898, 163.07999999999998, 'X[9] <= 0.5\ngini = 0.32\nsamples = 16
Text(152.02653061224493, 153.19636363636363, 'gini = 0.0\nsamples = 1\nvalue = [0, 0
Text(154.30408163265307, 153.19636363636363, 'X[2] <= 0.67\ngini = 0.24\nsamples = 1
Text(153.16530612244898, 143.31272727272727, 'X[4] <= 0.073\ngini = 0.133\nsamples =
Text(152.02653061224493, 133.42909090909092, 'X[9] <= 0.585\ngini = 0.5\nsamples = 2
Text(150.88775510204084, 123.54545454545455, 'gini = 0.0\nsamples = 1\nvalue = [0, 0
Text(153.16530612244898, 123.54545454545455, 'gini = 0.0\nsamples = 1\nvalue = [0, 1
Text(154.30408163265307, 133.42909090909092, 'gini = 0.0\nsamples = 12\nvalue = [0,
Text(155.44285714285715, 143.31272727272727, 'gini = 0.0\nsamples = 1\nvalue = [0, 0
Text(185.6559948979592, 163.07999999999998, 'X[5] <= 12.5\ngini = 0.63\nsamples = 14
Text(168.32525510204084, 153.19636363636363, 'X[3] <= 1.85\ngini = 0.703\nsamples =
Text(158.8591836734694, 143.31272727272727, 'X[8] <= 3.37\ngini = 0.475\nsamples = 1
Text(156.58163265306123, 133.42909090909092, 'X[2] <= 0.405\ngini = 0.469\nsamples =
Text(155.44285714285715, 123.54545454545455, 'gini = 0.0\nsamples = 5\nvalue = [0, 0
Text(157.72040816326532, 123.54545454545455, 'gini = 0.0\nsamples = 3\nvalue = [0, 0
Text(161.13673469387757, 133.42909090909092, 'X[5] <= 8.5\ngini = 0.18\nsamples = 10
Text(159.9979591836735, 123.54545454545455, 'gini = 0.0\nsamples = 9\nvalue = [0, 0,
Text(162.27551020408166, 123.54545454545455, 'gini = 0.0\nsamples = 1\nvalue = [0, 0
Text(177.79132653061225, 143.31272727272727, 'X[1] <= 0.655\ngini = 0.699\nsamples =
Text(168.82346938775513, 133.42909090909092, 'X[0] <= 7.2\ngini = 0.611\nsamples = 3
Text(164.55306122448982, 123.54545454545455, 'X[4] <= 0.061\ngini = 0.716\nsamples =
Text(163.41428571428574, 113.66181818181819, 'gini = 0.0\nsamples = 3\nvalue = [0, 0
Text(165.69183673469388, 113.66181818181819, 'X[9] <= 0.56\ngini = 0.611\nsamples =
Text(164.55306122448982, 103.77818181818182, 'gini = 0.0\nsamples = 3\nvalue = [0, 0
Text(166.83061224489796, 103.77818181818182, 'X[3] <= 2.3\ngini = 0.444\nsamples = 3
Text(165.69183673469388, 93.89454545454547, 'gini = 0.0\nsamples = 2\nvalue = [0, 2,
Text(167.96938775510205, 93.89454545454547, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(173.09387755102043, 123.54545454545455, 'X[7] <= 0.996\ngini = 0.48\nsamples =
Text(170.24693877551022, 113.66181818181819, 'X[7] <= 0.994\ngini = 0.153\nsamples =
Text(169.10816326530613, 103.77818181818182, 'gini = 0.0\nsamples = 1\nvalue = [0, 0
Text(171.3857142857143, 103.77818181818182, 'gini = 0.0\nsamples = 11\nvalue = [0, 0
Text(175.94081632653064, 113.66181818181819, 'X[4] <= 0.09\ngini = 0.549\nsamples =
Text(173.66326530612247, 103.77818181818182, 'X[3] <= 3.3\ngini = 0.346\nsamples = 9
Text(172.52448979591838, 93.89454545454547, 'gini = 0.0\nsamples = 6\nvalue = [0, 0,
Text(174.80204081632655, 93.89454545454547, 'X[7] <= 0.998\ngini = 0.444\nsamples =
Text(173.66326530612247, 84.01090909090911, 'gini = 0.0\nsamples = 2\nvalue = [0, 0,
Text(175.94081632653064, 84.01090909090911, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(178.2183673469388, 103.77818181818182, 'X[3] <= 2.7\ngini = 0.494\nsamples = 9\
Text(177.07959183673472, 93.89454545454547, 'gini = 0.0\nsamples = 5\nvalue = [0, 0,
Text(179.35714285714286, 93.89454545454547, 'X[0] <= 10.65\ngini = 0.625\nsamples =
Text(178.2183673469388, 84.01090909090911, 'gini = 0.0\nsamples = 2\nvalue = [0, 0,
Text(180.49591836734695, 84.01090909090911, 'X[6] <= 23.5\ngini = 0.5\nsamples = 2\r
Text(179.35714285714286, 74.12727272727273, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(181.63469387755103, 74.12727272727273, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(186.7591836734694, 133.42909090909092, 'X[10] <= 10.85\ngini = 0.688\nsamples =
Text(183.9122448979592, 123.54545454545455, 'X[1] <= 0.89\ngini = 0.531\nsamples = 8
Text(181.63469387755103, 113.66181818181819, 'X[7] <= 0.998\ngini = 0.278\nsamples =
Text(180.49591836734695, 103.77818181818182, 'gini = 0.0\nsamples = 5\nvalue = [0, 0
Text(182.7734693877551, 103.77818181818182, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(186.18979591836737, 113.66181818181819, 'X[7] <= 0.996\ngini = 0.5\nsamples = 2
Text(185.05102040816328, 103.77818181818182, 'gini = 0.0\nsamples = 1\nvalue = [0, 0
Text(187.32857142857145, 103.77818181818182, 'gini = 0.0\nsamples = 1\nvalue = [1, 0
Text(189.60612244897962, 123.54545454545455, 'X[2] <= 0.005\ngini = 0.406\nsamples =
Text(188.46734693877553, 113.66181818181819, 'gini = 0.0\nsamples = 1\nvalue = [0, 0
Text(190.7448979591837, 113.66181818181819, 'X[9] <= 0.52\ngini = 0.245\nsamples = 7
```

```
Text(189.60612244897962, 103.77818181818182, 'gini = 0.0\nsamples = 1\nvalue = [1, 0
Text(191.8836734693878, 103.77818181818182, 'gini = 0.0\nsamples = 6\nvalue = [0, 6,
Text(202.98673469387757, 153.19636363636363, 'X[6] <= 43.5\ngini = 0.484\nsamples =
Text(195.3, 143.31272727272727, 'X[9] <= 0.5\ngini = 0.329\nsamples = 36\nvalue = [0
Text(193.02244897959184, 133.42909090909092, 'X[4] <= 0.066\ngini = 0.375\nsamples =
Text(191.8836734693878, 123.54545454545455, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(194.16122448979593, 123.54545454545455, 'gini = 0.0\nsamples = 3\nvalue = [0, 0
Text(197.57755102040818, 133.42909090909092, 'X[10] <= 10.35\ngini = 0.227\nsamples
Text(196.4387755102041, 123.54545454545455, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(198.71632653061226, 123.54545454545455, 'X[4] <= 0.069\ngini = 0.179\nsamples =
Text(197.57755102040818, 113.66181818181819, 'X[6] <= 26.0\ngini = 0.611\nsamples =
Text(196.4387755102041, 103.77818181818182, 'gini = 0.0\nsamples = 3\nvalue = [0, 0,
Text(198.71632653061226, 103.77818181818182, 'X[10] <= 11.2\ngini = 0.444\nsamples =
Text(197.57755102040818, 93.89454545454547, 'gini = 0.0\nsamples = 2\nvalue = [0, 2,
Text(199.85510204081635, 93.89454545454547, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(199.85510204081635, 113.66181818181819, 'gini = 0.0\nsamples = 25\nvalue = [0,
Text(210.67346938775512, 143.31272727272727, 'X[6] <= 62.0\ngini = 0.529\nsamples =
Text(206.68775510204082, 133.42909090909092, 'X[5] <= 32.5\ngini = 0.42\nsamples = 2
Text(205.54897959183674, 123.54545454545455, 'X[3] <= 2.05\ngini = 0.346\nsamples =
Text(202.13265306122452, 113.66181818181819, 'X[5] <= 27.5\ngini = 0.153\nsamples =
Text(200.99387755102043, 103.77818181818182, 'gini = 0.0\nsamples = 9\nvalue = [0, 0
Text(203.2714285714286, 103.77818181818182, 'X[5] <= 28.5\ngini = 0.444\nsamples = 3
Text(202.13265306122452, 93.89454545454547, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(204.41020408163268, 93.89454545454547, 'gini = 0.0\nsamples = 2\nvalue = [0, 0,
Text(208.965306122449, 113.66181818181819, 'X[9] <= 0.605\ngini = 0.5\nsamples = 6\n
Text(207.8265306122449, 103.77818181818182, 'X[9] <= 0.47\ngini = 0.375\nsamples = 4
Text(206.68775510204082, 93.89454545454547, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(208.965306122449, 93.89454545454547, 'gini = 0.0\nsamples = 3\nvalue = [0, 0, 0
Text(210.10408163265308, 103.77818181818182, 'gini = 0.0\nsamples = 2\nvalue = [0, 0
Text(207.8265306122449, 123.54545454545455, 'gini = 0.0\nsamples = 2\nvalue = [0, 0,
Text(214.6591836734694, 133.42909090909092, 'X[3] <= 1.7\ngini = 0.403\nsamples = 12
Text(212.38163265306125, 123.54545454545455, 'X[6] <= 76.0\ngini = 0.5\nsamples = 2\
Text(211.24285714285716, 113.66181818181819, 'gini = 0.0\nsamples = 1\nvalue = [0, 0
Text(213.52040816326533, 113.66181818181819, 'gini = 0.0\nsamples = 1\nvalue = [0, 1
Text(216.93673469387758, 123.54545454545455, 'X[1] <= 0.792\ngini = 0.18\nsamples =
Text(215.7979591836735, 113.66181818181819, 'gini = 0.0\nsamples = 9\nvalue = [0, 0,
Text(218.07551020408167, 113.66181818181819, 'gini = 0.0\nsamples = 1\nvalue = [0, 0
Text(244.1961734693878, 172.96363636363637, 'X[1] <= 0.375\ngini = 0.585\nsamples =
Text(228.89387755102044, 163.07999999999998, 'X[4] <= 0.061\ngini = 0.621\nsamples =
Text(220.3530612244898, 153.19636363636363, 'X[0] <= 8.05\ngini = 0.379\nsamples = 1
Text(219.21428571428572, 143.31272727272727, 'X[7] <= 0.996\ngini = 0.625\nsamples =
Text(218.07551020408167, 133.42909090909092, 'gini = 0.0\nsamples = 2\nvalue = [0, 0
Text(220.3530612244898, 133.42909090909092, 'X[9] <= 0.845\ngini = 0.5\nsamples = 2\
Text(219.21428571428572, 123.54545454545455, 'gini = 0.0\nsamples = 1\nvalue = [0, 0
Text(221.4918367346939, 123.54545454545455, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(221.4918367346939, 143.31272727272727, 'gini = 0.0\nsamples = 9\nvalue = [0, 0,
Text(237.43469387755104, 153.19636363636363, 'X[0] <= 10.45\ngini = 0.615\nsamples =
Text(230.60204081632656, 143.31272727272727, 'X[0] <= 8.1\ngini = 0.576\nsamples = 3
Text(226.04693877551023, 133.42909090909092, 'X[1] <= 0.355\ngini = 0.667\nsamples =
Text(223.76938775510206, 123.54545454545455, 'X[2] <= 0.46\ngini = 0.58\nsamples = 1
Text(222.63061224489797, 113.66181818181819, 'X[3] <= 2.35\ngini = 0.571\nsamples =
Text(221.4918367346939, 103.77818181818182, 'gini = 0.0\nsamples = 4\nvalue = [0, 0,
Text(223.76938775510206, 103.77818181818182, 'X[6] <= 35.0\ngini = 0.444\nsamples =
Text(222.63061224489797, 93.89454545454547, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(224.90816326530614, 93.89454545454547, 'gini = 0.0\nsamples = 2\nvalue = [0, 0,
Text(224.90816326530614, 113.66181818181819, 'gini = 0.0\nsamples = 3\nvalue = [0, 0
Text(228.3244897959184, 123.54545454545455, 'X[4] <= 0.08\ngini = 0.32\nsamples = 5\
Text(227.1857142857143, 113.66181818181819, 'gini = 0.0\nsamples = 4\nvalue = [0, 0,
Text(229.46326530612248, 113.66181818181819, 'gini = 0.0\nsamples = 1\nvalue = [0, 0
Text(235.15714285714287, 133.42909090909092, 'X[2] <= 0.515\ngini = 0.42\nsamples =
Text(232.87959183673471, 123.54545454545455, 'X[0] <= 9.67\ngini = 0.278\nsamples =
```

```
Text(232.87959183673347, 123.54545454545455, 'X[9] <= 0.67\ngini = 0.278\nsamples = 1
Text(231.74081632653065, 113.66181818181819, 'gini = 0.0\nsamples = 1\nvalue = [0, 0
Text(234.0183673469388, 113.66181818181819, 'X[5] <= 41.5\ngini = 0.208\nsamples = 1
Text(232.8795918367347, 103.77818181818182, 'X[0] <= 9.7\ngini = 0.117\nsamples = 16
Text(231.74081632653065, 93.89454545454547, 'gini = 0.0\nsamples = 13\nvalue = [0, 0
Text(234.0183673469388, 93.89454545454547, 'X[2] <= 0.415\ngini = 0.444\nsamples = 3
Text(232.8795918367347, 84.01090909090911, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(235.15714285714287, 84.01090909090911, 'gini = 0.0\nsamples = 2\nvalue = [0, 0,
Text(235.15714285714287, 103.77818181818182, 'gini = 0.0\nsamples = 1\nvalue = [0, 0
Text(237.43469387755104, 123.54545454545455, 'X[8] <= 3.225\ngini = 0.48\nsamples =
Text(236.29591836734696, 113.66181818181819, 'gini = 0.0\nsamples = 2\nvalue = [0, 0
Text(238.57346938775513, 113.66181818181819, 'gini = 0.0\nsamples = 3\nvalue = [0, 0
Text(244.26734693877555, 143.31272727272727, 'X[1] <= 0.335\ngini = 0.486\nsamples =
Text(243.12857142857146, 133.42909090909092, 'X[6] <= 21.5\ngini = 0.34\nsamples = 1
Text(241.98979591836738, 123.54545454545455, 'X[8] <= 3.28\ngini = 0.5\nsamples = 2\
Text(240.8510204081633, 113.66181818181819, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(243.12857142857146, 113.66181818181819, 'gini = 0.0\nsamples = 1\nvalue = [0, 0
Text(244.26734693877555, 123.54545454545455, 'gini = 0.0\nsamples = 8\nvalue = [0, 0
Text(245.4061224489796, 133.42909090909092, 'gini = 0.0\nsamples = 2\nvalue = [0, 0,
Text(259.4984693877551, 163.07999999999998, 'X[3] <= 3.475\ngini = 0.496\nsamples =
Text(253.6622448979592, 153.19636363636363, 'X[7] <= 0.996\ngini = 0.415\nsamples =
Text(248.82244897959185, 143.31272727272727, 'X[4] <= 0.119\ngini = 0.263\nsamples =
Text(247.68367346938777, 133.42909090909092, 'X[2] <= 0.16\ngini = 0.222\nsamples =
Text(246.5448979591837, 123.54545454545455, 'gini = 0.0\nsamples = 21\nvalue = [0, 0
Text(248.82244897959185, 123.54545454545455, 'X[9] <= 0.815\ngini = 0.514\nsamples =
Text(247.68367346938777, 113.66181818181819, 'X[10] <= 11.15\ngini = 0.72\nsamples =
Text(245.4061224489796, 103.77818181818182, 'X[4] <= 0.077\ngini = 0.444\nsamples =
Text(244.26734693877555, 93.89454545454547, 'gini = 0.0\nsamples = 2\nvalue = [0, 0,
Text(246.5448979591837, 93.89454545454547, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(249.96122448979594, 103.77818181818182, 'X[6] <= 28.5\ngini = 0.5\nsamples = 2\
Text(248.82244897959185, 93.89454545454547, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(251.10000000000002, 93.89454545454547, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(249.96122448979594, 113.66181818181819, 'gini = 0.0\nsamples = 7\nvalue = [0, 0
Text(249.96122448979594, 133.42909090909092, 'gini = 0.0\nsamples = 1\nvalue = [0, 0
Text(258.50204081632654, 143.31272727272727, 'X[8] <= 3.46\ngini = 0.495\nsamples =
Text(256.2244897959184, 133.42909090909092, 'X[3] <= 1.55\ngini = 0.46\nsamples = 40
Text(255.08571428571432, 123.54545454545455, 'gini = 0.0\nsamples = 1\nvalue = [0, 0
Text(257.36326530612246, 123.54545454545455, 'X[7] <= 0.997\ngini = 0.437\nsamples =
Text(253.3775510204082, 113.66181818181819, 'X[4] <= 0.067\ngini = 0.642\nsamples =
Text(252.2387755102041, 103.77818181818182, 'gini = 0.0\nsamples = 2\nvalue = [0, 0,
Text(254.51632653061228, 103.77818181818182, 'X[2] <= 0.21\ngini = 0.49\nsamples = 7
Text(253.3775510204082, 93.89454545454547, 'X[9] <= 0.695\ngini = 0.32\nsamples = 5\
Text(252.2387755102041, 84.01090909090911, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(254.51632653061228, 84.01090909090911, 'gini = 0.0\nsamples = 4\nvalue = [0, 0,
Text(255.65510204081636, 93.89454545454547, 'gini = 0.0\nsamples = 2\nvalue = [0, 0,
Text(261.3489795918368, 113.66181818181819, 'X[9] <= 0.865\ngini = 0.331\nsamples =
Text(259.0714285714286, 103.77818181818182, 'X[6] <= 71.0\ngini = 0.261\nsamples = 2
Text(257.9326530612245, 93.89454545454547, 'X[4] <= 0.089\ngini = 0.21\nsamples = 26
Text(256.79387755102044, 84.01090909090911, 'gini = 0.0\nsamples = 17\nvalue = [0, 0
Text(259.0714285714286, 84.01090909090911, 'X[2] <= 0.395\ngini = 0.494\nsamples = 9
Text(257.9326530612245, 74.12727272727273, 'X[4] <= 0.095\ngini = 0.625\nsamples = 4
Text(256.79387755102044, 64.24363636363637, 'X[4] <= 0.091\ngini = 0.5\nsamples = 2\
Text(255.65510204081636, 54.360000000000014, 'gini = 0.0\nsamples = 1\nvalue = [0, 0
Text(257.9326530612245, 54.360000000000014, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(259.0714285714286, 64.24363636363637, 'gini = 0.0\nsamples = 2\nvalue = [0, 0,
Text(260.2102040816327, 74.12727272727273, 'gini = 0.0\nsamples = 5\nvalue = [0, 0,
Text(260.2102040816327, 93.89454545454547, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(263.62653061224495, 103.77818181818182, 'X[7] <= 1.0\ngini = 0.444\nsamples = 3
Text(262.48775510204086, 93.89454545454547, 'gini = 0.0\nsamples = 2\nvalue = [0, 0,
Text(264.76530612244903, 93.89454545454547, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(260.7795918367347, 133.42909090909092, 'X[4] <= 0.076\ngini = 0.375\nsamples =
```

```
Text(259.6408163265306, 123.54545454545455, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(261.9183673469388, 123.54545454545455, 'gini = 0.0\nsamples = 3\nvalue = [0, 0,
Text(265.33469387755105, 153.19636363636363, 'X[1] <= 0.425\ngini = 0.678\nsamples =
Text(264.19591836734696, 143.31272727272727, 'gini = 0.0\nsamples = 3\nvalue = [0, 0
Text(266.47346938775513, 143.31272727272727, 'X[9] <= 0.735\ngini = 0.531\nsamples =
Text(265.33469387755105, 133.42909090909092, 'X[4] <= 0.103\ngini = 0.444\nsamples =
Text(264.19591836734696, 123.54545454545455, 'gini = 0.0\nsamples = 2\nvalue = [0, 0
Text(266.47346938775513, 123.54545454545455, 'gini = 0.0\nsamples = 1\nvalue = [0, 1
Text(267.6122448979592, 133.42909090909092, 'gini = 0.0\nsamples = 5\nvalue = [0, 0,
Text(209.0809630102041, 182.84727272727272, 'gini = 0.0\nsamples = 9\nvalue = [0, 0,
Text(299.9605867346939, 192.73090909090908, 'X[0] <= 7.85\ngini = 0.614\nsamples = 1
Text(280.20994897959184, 182.84727272727272, 'X[2] <= 0.355\ngini = 0.549\nsamples =
Text(279.07117346938776, 172.96363636363637, 'X[9] <= 0.86\ngini = 0.605\nsamples =
Text(277.9323979591837, 163.07999999999998, 'X[6] <= 15.0\ngini = 0.589\nsamples = 7
Text(269.8897959183674, 153.19636363636363, 'X[3] <= 1.75\ngini = 0.542\nsamples = 1
Text(268.7510204081633, 143.31272727272727, 'gini = 0.0\nsamples = 3\nvalue = [0, 0,
Text(271.02857142857147, 143.31272727272727, 'X[7] <= 0.996\ngini = 0.37\nsamples =
Text(269.8897959183674, 133.42909090909092, 'X[8] <= 3.335\ngini = 0.219\nsamples =
Text(268.7510204081633, 123.54545454545455, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(271.02857142857147, 123.54545454545455, 'gini = 0.0\nsamples = 7\nvalue = [0, 0
Text(272.16734693877555, 133.42909090909092, 'gini = 0.0\nsamples = 1\nvalue = [0, 0
Text(285.975, 153.19636363636363, 'X[6] <= 78.5\ngini = 0.545\nsamples = 62\nvalue =
Text(279.28469387755104, 143.31272727272727, 'X[6] <= 27.5\ngini = 0.442\nsamples =
Text(274.4448979591837, 133.42909090909092, 'X[0] <= 6.35\ngini = 0.551\nsamples = 1
Text(273.30612244897964, 123.54545454545455, 'gini = 0.0\nsamples = 4\nvalue = [0, 0
Text(275.5836734693878, 123.54545454545455, 'X[3] <= 3.325\ngini = 0.43\nsamples = 1
Text(273.30612244897964, 113.66181818181819, 'X[3] <= 1.775\ngini = 0.198\nsamples =
Text(272.16734693877555, 103.77818181818182, 'gini = 0.0\nsamples = 1\nvalue = [0, 0
Text(274.4448979591837, 103.77818181818182, 'gini = 0.0\nsamples = 8\nvalue = [0, 0,
Text(277.861224489796, 113.66181818181819, 'X[8] <= 3.44\ngini = 0.5\nsamples = 2\nv
Text(276.7224489795919, 103.77818181818182, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(279.0, 103.77818181818182, 'gini = 0.0\nsamples = 1\nvalue = [0, 1, 0, 0, 0, 0]
Text(284.1244897959184, 133.42909090909092, 'X[0] <= 4.95\ngini = 0.323\nsamples = 3
Text(281.27755102040817, 123.54545454545455, 'X[9] <= 0.65\ngini = 0.5\nsamples = 2\
Text(280.1387755102041, 113.66181818181819, 'gini = 0.0\nsamples = 1\nvalue = [0, 1,
Text(282.41632653061225, 113.66181818181819, 'gini = 0.0\nsamples = 1\nvalue = [0, 0
Text(286.9714285714286, 123.54545454545455, 'X[1] <= 0.8\ngini = 0.251\nsamples = 35
Text(284.6938775510204, 113.66181818181819, 'X[8] <= 3.27\ngini = 0.174\nsamples = 3
Text(283.55510204081634, 103.77818181818182, 'gini = 0.0\nsamples = 1\nvalue = [0, 0
Text(285.8326530612245, 103.77818181818182, 'X[0] <= 5.05\ngini = 0.123\nsamples = 3
Text(283.55510204081634, 93.89454545454547, 'X[2] <= 0.125\ngini = 0.444\nsamples =
Text(282.41632653061225, 84.01090909090911, 'gini = 0.0\nsamples = 2\nvalue = [0, 0,
Text(284.6938775510204, 84.01090909090911, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(288.1102040816327, 93.89454545454547, 'X[6] <= 30.0\ngini = 0.069\nsamples = 28
Text(286.9714285714286, 84.01090909090911, 'X[10] <= 11.9\ngini = 0.32\nsamples = 5\
Text(285.8326530612245, 74.12727272727273, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(288.1102040816327, 74.12727272727273, 'gini = 0.0\nsamples = 4\nvalue = [0, 0,
Text(289.24897959183676, 84.01090909090911, 'gini = 0.0\nsamples = 23\nvalue = [0, 0
Text(289.24897959183676, 113.66181818181819, 'X[2] <= 0.05\ngini = 0.444\nsamples =
Text(288.1102040816327, 103.77818181818182, 'gini = 0.0\nsamples = 2\nvalue = [0, 0,
Text(290.38775510204084, 103.77818181818182, 'gini = 0.0\nsamples = 1\nvalue = [0, 0
Text(292.665306122449, 143.31272727272727, 'X[2] <= 0.07\ngini = 0.78\nsamples = 10\
Text(290.38775510204084, 133.42909090909092, 'X[1] <= 0.485\ngini = 0.5\nsamples = 4
Text(289.24897959183676, 123.54545454545455, 'gini = 0.0\nsamples = 2\nvalue = [0, 0
Text(291.5265306122449, 123.54545454545455, 'gini = 0.0\nsamples = 2\nvalue = [0, 0,
Text(294.9428571428572, 133.42909090909092, 'X[9] <= 0.61\ngini = 0.611\nsamples = 6
Text(293.8040816326531, 123.54545454545455, 'X[6] <= 109.5\ngini = 0.444\nsamples =
Text(292.665306122449, 113.66181818181819, 'gini = 0.0\nsamples = 2\nvalue = [0, 0,
Text(294.9428571428572, 113.66181818181819, 'gini = 0.0\nsamples = 1\nvalue = [0, 1,
Text(296.08163265306126, 123.54545454545455, 'gini = 0.0\nsamples = 3\nvalue = [0, 0
```

```
Text(280.20994897959184, 163.07999999999998, 'gini = 0.0\nsamples = 3\nvalue = [0, 0
Text(281.3487244897959, 172.96363636363637, 'gini = 0.0\nsamples = 13\nvalue = [0, 0
Text(319.71122448979594, 182.84727272727272, 'X[0] <= 9.95\ngini = 0.613\nsamples =
Text(309.7469387755102, 172.96363636363637, 'X[5] <= 13.5\ngini = 0.583\nsamples = 0
Text(302.3448979591837, 163.07999999999998, 'X[1] <= 0.415\ngini = 0.539\nsamples =
Text(299.4979591836735, 153.19636363636363, 'X[4] <= 0.098\ngini = 0.335\nsamples =
Text(298.35918367346943, 143.31272727272727, 'X[6] <= 7.5\ngini = 0.25\nsamples = 29
Text(297.22040816326535, 133.42909090909092, 'gini = 0.0\nsamples = 1\nvalue = [0, 0
Text(299.4979591836735, 133.42909090909092, 'X[4] <= 0.051\ngini = 0.196\nsamples =
Text(298.35918367346943, 123.54545454545455, 'gini = 0.0\nsamples = 1\nvalue = [0, 1
Text(300.6367346938776, 123.54545454545455, 'X[2] <= 0.545\ngini = 0.137\nsamples =
Text(299.4979591836735, 113.66181818181819, 'X[1] <= 0.39\ngini = 0.074\nsamples = 2
Text(298.35918367346943, 103.77818181818182, 'gini = 0.0\nsamples = 22\nvalue = [0,
Text(300.6367346938776, 103.77818181818182, 'X[7] <= 0.995\ngini = 0.375\nsamples =
Text(299.4979591836735, 93.89454545454547, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(301.7755102040817, 93.89454545454547, 'gini = 0.0\nsamples = 3\nvalue = [0, 0,
Text(301.7755102040817, 113.66181818181819, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(300.6367346938776, 143.31272727272727, 'gini = 0.0\nsamples = 2\nvalue = [0, 0,
Text(305.1918367346939, 153.19636363636363, 'X[9] <= 0.635\ngini = 0.694\nsamples =
Text(302.91428571428577, 143.31272727272727, 'X[4] <= 0.095\ngini = 0.5\nsamples = 4
Text(301.7755102040817, 133.42909090909092, 'gini = 0.0\nsamples = 2\nvalue = [0, 0,
Text(304.05306122448985, 133.42909090909092, 'gini = 0.0\nsamples = 2\nvalue = [0, 0
Text(307.46938775510205, 143.31272727272727, 'X[2] <= 0.42\ngini = 0.408\nsamples =
Text(306.33061224489796, 133.42909090909092, 'X[5] <= 10.0\ngini = 0.444\nsamples =
Text(305.1918367346939, 123.54545454545455, 'gini = 0.0\nsamples = 2\nvalue = [0, 0,
Text(307.46938775510205, 123.54545454545455, 'gini = 0.0\nsamples = 1\nvalue = [0, 0
Text(308.60816326530613, 133.42909090909092, 'gini = 0.0\nsamples = 4\nvalue = [0, 0
Text(317.1489795918368, 163.07999999999998, 'X[3] <= 3.15\ngini = 0.518\nsamples = 2
Text(314.30204081632655, 153.19636363636363, 'X[4] <= 0.073\ngini = 0.375\nsamples =
Text(312.0244897959184, 143.31272727272727, 'X[4] <= 0.048\ngini = 0.142\nsamples =
Text(310.8857142857143, 133.42909090909092, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(313.16326530612247, 133.42909090909092, 'gini = 0.0\nsamples = 12\nvalue = [0,
Text(316.5795918367347, 143.31272727272727, 'X[3] <= 2.7\ngini = 0.49\nsamples = 7\n
Text(315.44081632653064, 133.42909090909092, 'gini = 0.0\nsamples = 4\nvalue = [0, 0
Text(317.7183673469388, 133.42909090909092, 'gini = 0.0\nsamples = 3\nvalue = [0, 0,
Text(319.995918367347, 153.19636363636363, 'X[10] <= 12.7\ngini = 0.278\nsamples = 0
Text(318.8571428571429, 143.31272727272727, 'gini = 0.0\nsamples = 5\nvalue = [0, 0,
Text(321.13469387755106, 143.31272727272727, 'gini = 0.0\nsamples = 1\nvalue = [0, 0
Text(329.67551020408166, 172.96363636363637, 'X[2] <= 0.7\ngini = 0.596\nsamples = 4
Text(326.8285714285715, 163.07999999999998, 'X[10] <= 12.3\ngini = 0.554\nsamples =
Text(324.5510204081633, 153.19636363636363, 'X[10] <= 12.05\ngini = 0.592\nsamples =
Text(323.4122448979592, 143.31272727272727, 'X[1] <= 0.275\ngini = 0.566\nsamples =
Text(321.13469387755106, 133.42909090909092, 'X[6] <= 49.5\ngini = 0.5\nsamples = 2\
Text(319.995918367347, 123.54545454545455, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
Text(322.27346938775514, 123.54545454545455, 'gini = 0.0\nsamples = 1\nvalue = [0, 0
Text(325.6897959183674, 133.42909090909092, 'X[3] <= 2.3\ngini = 0.485\nsamples = 20
Text(324.5510204081633, 123.54545454545455, 'gini = 0.0\nsamples = 6\nvalue = [0, 0,
Text(326.8285714285715, 123.54545454545455, 'X[7] <= 0.998\ngini = 0.561\nsamples =
Text(324.5510204081633, 113.66181818181819, 'X[1] <= 0.39\ngini = 0.408\nsamples = 7
Text(323.4122448979592, 103.77818181818182, 'gini = 0.0\nsamples = 2\nvalue = [0, 0,
Text(325.6897959183674, 103.77818181818182, 'gini = 0.0\nsamples = 5\nvalue = [0, 0,
Text(329.10612244897965, 113.66181818181819, 'X[6] <= 13.5\ngini = 0.449\nsamples =
Text(327.96734693877556, 103.77818181818182, 'gini = 0.0\nsamples = 1\nvalue = [0, 0
```

```
# Model has learnt everything that is nothing but overfitting
# we need to optimize

from sklearn.metrics import accuracy_score
```

```
Text(330.24489795918373, 143.31272727272727, 'gini = 0.0\nsamples = 9\nvalue = [0, 0
```

```
print(round(accuracy_score(y_train, y_pred_train_dt),2))
```

```
1.0
```

```
print(round(accuracy_score(y_test, y_pred_test_dt),2))
```

```
0.6
```

```
from sklearn import tree
path = dt_clf.cost_complexity_pruning_path(X_train, y_train)
```

```
alphas = path['ccp_alphas']
alphas
```

```
array([0.        , 0.00058088, 0.00058309, 0.00074471, 0.00077179,
       0.00080429, 0.00081918, 0.00082216, 0.00082491, 0.00082491,
       0.00082491, 0.00082982, 0.0008378 , 0.00085041, 0.00085216,
       0.00085928, 0.00085928, 0.00087303, 0.00089366, 0.00089366,
       0.00089366, 0.00089366, 0.00089366, 0.00089366, 0.00089366,
       0.00089366, 0.00089366, 0.00089366, 0.00089366, 0.00089366,
       0.00089366, 0.00089366, 0.00103612, 0.00110492, 0.0011543 ,
       0.00116175, 0.00119154, 0.00119154, 0.00119154, 0.00119154,
       0.00119154, 0.00119154, 0.00119154, 0.00119154, 0.00119154,
       0.00119154, 0.00119154, 0.00119154, 0.00119154, 0.00119154,
       0.00130738, 0.00132393, 0.00134048, 0.00134048, 0.00134048,
       0.00134048, 0.00134048, 0.00134048, 0.00134048, 0.00134048,
       0.00134048, 0.00134048, 0.00134048, 0.00134048, 0.00134048,
       0.00134048, 0.00136176, 0.00136176, 0.0013811 , 0.00139634,
       0.00140432, 0.00141977, 0.00142985, 0.00142985, 0.00142985,
       0.00142985, 0.0014412 , 0.00145219, 0.00145219, 0.0014546 ,
       0.00146017, 0.00148943, 0.00148943, 0.00148943, 0.00148943,
       0.00153198, 0.00153624, 0.00154544, 0.0015639 , 0.0015639 ,
       0.00156525, 0.0015693 , 0.00158424, 0.00158741, 0.00158796,
       0.00158872, 0.00158872, 0.00160858, 0.00160858, 0.00160858,
       0.00160858, 0.00160858, 0.00160858, 0.00161975, 0.00163018,
       0.00163701, 0.00163837, 0.00163837, 0.00164982, 0.00166816,
       0.00168802, 0.00168878, 0.00170167, 0.00173766, 0.00174218,
       0.00174263, 0.00178731, 0.00178731, 0.0018171 , 0.0018171 ,
       0.00185695, 0.00186923, 0.00190088, 0.00191774, 0.00194618,
       0.00199583, 0.00207872, 0.0020852 , 0.0020852 , 0.00209512,
       0.00209603, 0.0021145 , 0.00214477, 0.00214649, 0.00216539,
       0.00217456, 0.00221727, 0.00224277, 0.00224903, 0.00226906,
       0.00229349, 0.00233281, 0.00233569, 0.00238308, 0.00239301,
       0.00243904, 0.00248238, 0.00248366, 0.00249674, 0.00250223,
       0.00251784, 0.00252629, 0.00259663, 0.00260649, 0.00263972,
       0.00268249, 0.00269364, 0.0027095 , 0.00275389, 0.00275544,
       0.00281119, 0.00286376, 0.00299374, 0.00301857, 0.00306348,
       0.00312779, 0.00320841, 0.00331025, 0.00338361, 0.00339294,
       0.00370056, 0.00383892, 0.00385893, 0.00400584, 0.00403601,
       0.0040543 , 0.00415053, 0.0041823 , 0.0044017 , 0.00500447,
       0.0054185 , 0.00651323, 0.00655364, 0.0070582 , 0.01033366,
       0.01251074, 0.06471415])
```

```
accuracy_train, accuracy_test = [],[]

for i in alphas:
    dt_clf = DecisionTreeClassifier(ccp_alpha=i)

    dt_clf.fit(X_train, y_train)
    y_pred_train1 = dt_clf.predict(X_train)
    y_pred_test1 = dt_clf.predict(X_test)

    accuracy_train.append(accuracy_score(y_train, y_pred_train1))
    accuracy_test.append(accuracy_score(y_test,y_pred_test1))



accuracy_train
```

```
[1.0,
 0.9991063449508489,
 0.998212689901698,
 0.9973190348525469,
 0.9955317247542449,
 0.9946380697050938,
 0.9937444146559428,
 0.9919571045576407,
 0.9892761394101877,
 0.9892761394101877,
 0.9892761394101877,
 0.9883824843610366,
 0.9874888293118856,
 0.9857015192135835,
 0.9848078641644326,
 0.9830205540661304,
 0.9830205540661304,
 0.9821268990169795,
 0.968722073279714,
 0.968722073279714,
 0.968722073279714,
 0.9696157283288651,
 0.968722073279714,
 0.9696157283288651,
 0.968722073279714,
 0.968722073279714,
 0.968722073279714,
 0.9696157283288651,
 0.968722073279714,
 0.968722073279714,
 0.968722073279714,
 0.967828418230563,
 0.9651474530831099,
 0.9651474530831099,
 0.9615728328865059,
 0.9606791778373548,
 0.9490616621983914,
 0.9490616621983914,
 0.9463806970509383,
 0.9463806970509383,
 0.9490616621983914,
 0.9454870420017873,
 0.9463806970509383,
```

```
        0.9463806970509383,
        0.9463806970509383,
        0.9472743521000894,
        0.9481680071492404,
        0.9472743521000894,
        0.9472743521000894,
        0.9490616621983914,
        0.9436997319034852,
        0.9419124218051832,
        0.9302949061662198,
        0.9294012511170688,
        0.9294012511170688,
        0.9294012511170688,
        0.9302949061662198,
        0.9302949061662198,
```

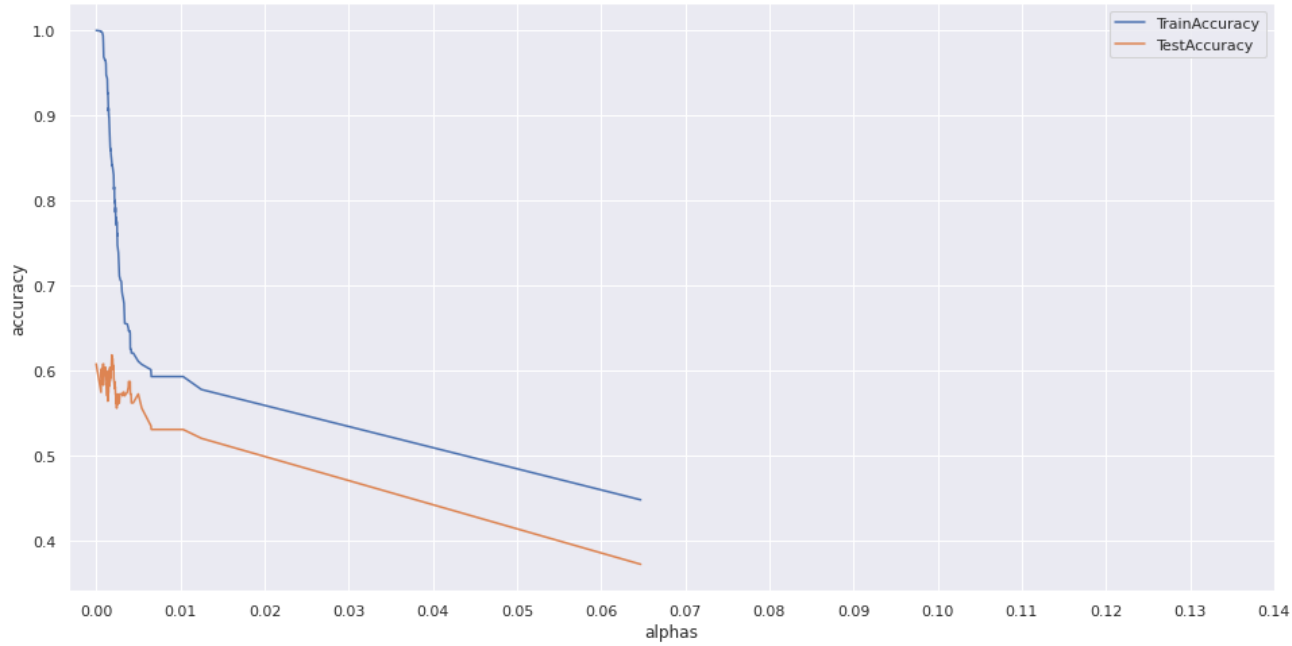
```
accuracy_test
```

```
     [0.6083333333333333,
      0.575,
      0.6020833333333333,
      0.5895833333333333,
      0.6,
      0.5854166666666667,
      0.5875,
      0.59375,
      0.5979166666666667,
      0.6,
      0.59375,
      0.6083333333333333,
      0.6020833333333333,
      0.5833333333333334,
      0.6,
      0.5979166666666667,
      0.6104166666666667,
      0.5958333333333333,
      0.5875,
      0.58125,
      0.6083333333333333,
      0.5895833333333333,
      0.5916666666666667,
      0.59375,
      0.6125,
      0.6041666666666666,
      0.6,
      0.6083333333333333,
      0.5729166666666666,
      0.5875,
      0.6041666666666666,
      0.5958333333333333,
      0.5979166666666667,
      0.6041666666666666,
      0.5979166666666667,
      0.59375,
      0.5833333333333334,
      0.5958333333333333,
      0.6,
      0.5958333333333333,
      0.59375,
```

```
        0.58125,
        0.60625,
        0.60625,
        0.5895833333333333,
        0.59375,
        0.6,
        0.5916666666666667,
        0.59375,
        0.5729166666666666,
        0.5708333333333333,
        0.5791666666666667,
        0.6,
        0.58125,
        0.5854166666666667,
        0.5770833333333333,
        0.5791666666666667,
```

```
# now we have scores
# lets, plot

sns.set()
plt.figure(figsize = (16,8))
sns.lineplot(y =accuracy_train, x = alphas, label = 'TrainAccuracy')
sns.lineplot(y =accuracy_test, x = alphas, label = 'TestAccuracy')
plt.xticks(ticks=np.arange(0.00,0.15,0.01))
plt.xlabel('alphas')
plt.ylabel('accuracy')
plt.show()
```



```
#_________with ccp = 0.01
dt_clf = DecisionTreeClassifier(ccp_alpha=0.01, random_state = 14)
```

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

```
     DecisionTreeClassifier(ccp_alpha=0.01, random_state=14)
```

```
y_pred_train2 = dt_clf.predict(X_train)
```

```
y_pred_test2 = dt_clf.predict(X_test)
```

```
from sklearn.metrics import accuracy_score
print(round(accuracy_score(y_train,y_pred_train2), 2))
```

```
     0.59
```

```
print(round(accuracy_score(y_test,y_pred_test2), 2))
```

```
     0.53
```

```
#Earlier before optimization accuracys were 1 and 0.6 (train & test)
#Now after optimization accuracys are 0.59 and 0.53 (train & test)
```

```
from sklearn.metrics import confusion_matrix
confusion_matrix = confusion_matrix(y_test, y_pred_test2)
print(confusion_matrix)
```

```
     [[  0   0   3   1   0   0]
      [  0   0  10   8   0   0]
      [  0   0 129  50   0   0]
      [  0   0  82 126   0   0]
      [  0   0   3  64   0   0]
      [  0   0   0   4   0   0]]
```

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred_test2))
```

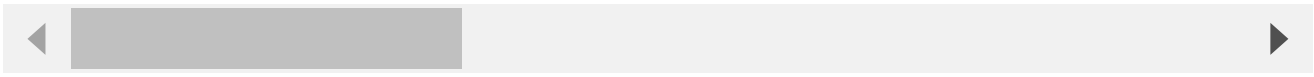
```
                   precision    recall  f1-score   support

                1       0.00      0.00      0.00         4
                2       0.00      0.00      0.00        18
```

```
              3       0.57      0.72      0.64       179
              4       0.50      0.61      0.55       208
              5       0.00      0.00      0.00        67
              6       0.00      0.00      0.00         4

       accuracy                           0.53       480
      macro avg       0.18      0.22      0.20       480
   weighted avg       0.43      0.53      0.47       480

/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1308: Undef
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1308: Undef
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1308: Undef
  _warn_prf(average, modifier, msg_start, len(result))
```



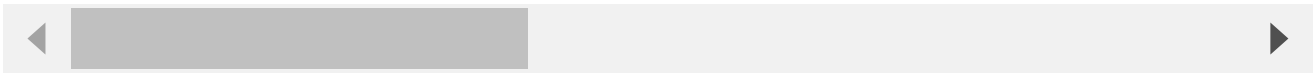
#Random Forest

```
#import the classifier
from sklearn.ensemble import RandomForestClassifier
```

```
#in our previous experiment, we found ccp_alphas = 0.013 has the best accuarcy
rf_clf = RandomForestClassifier(n_estimators =100, ccp_alpha= 0.01, random_state = 14)
```

```
#fit the classifier with x and y  data = train
rf_mod = rf_clf.fit(X_train, y_train)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: DataConversionWarning
```



```
#Prediction
y_train_pred_rf1 = rf_mod.predict(X_train)
y_train_pred_rf1
```

```
array([4, 4, 4, ..., 4, 3, 4])
```

```
#Prediction
y_test_pred_rf1 = rf_mod.predict(X_test)
y_test_pred_rf1
```

```
array([4, 3, 4, 3, 4, 4, 3, 3, 3, 4, 3, 3, 4, 3, 3, 3, 4, 3, 3, 4, 4, 3,
       4, 3, 4, 4, 4, 3, 3, 4, 3, 3, 4, 4, 4, 4, 4, 4, 3, 4, 4, 3, 3, 4,
       3, 4, 3, 4, 4, 4, 4, 3, 4, 3, 3, 4, 3, 3, 3, 4, 4, 3, 4, 3, 4, 3,
       3, 4, 3, 4, 3, 3, 4, 3, 4, 3, 4, 4, 4, 4, 3, 4, 4, 3, 3, 4, 3, 3,
```

```
       3, 4, 4, 3, 4, 3, 4, 3, 4, 4, 3, 4, 4, 4, 4, 3, 3, 4, 3, 4, 4, 3,
       3, 3, 4, 4, 3, 3, 4, 3, 3, 3, 3, 3, 4, 3, 3, 3, 4, 3, 4, 3, 4, 4,
       4, 4, 4, 3, 3, 3, 4, 3, 4, 4, 4, 3, 3, 3, 3, 4, 3, 4, 3, 4, 4, 3,
       4, 3, 3, 3, 3, 3, 4, 4, 4, 4, 3, 3, 3, 4, 4, 3, 3, 3, 3, 3, 3, 4,
       3, 4, 4, 3, 4, 3, 3, 3, 4, 4, 3, 4, 3, 3, 4, 4, 3, 3, 4, 4, 3, 3,
       4, 3, 4, 3, 3, 3, 4, 3, 4, 4, 4, 4, 3, 4, 3, 4, 3, 4, 4, 4, 3, 4,
       3, 4, 4, 3, 4, 4, 3, 3, 3, 3, 3, 4, 4, 4, 4, 3, 4, 4, 3, 3, 4, 3,
       4, 3, 3, 4, 3, 4, 4, 4, 3, 4, 3, 3, 4, 4, 3, 4, 3, 4, 3, 3, 3, 3,
       3, 4, 3, 4, 4, 3, 4, 4, 4, 4, 3, 3, 3, 3, 3, 4, 3, 3, 4, 3, 3, 4,
       3, 4, 3, 4, 3, 3, 3, 4, 3, 3, 3, 4, 4, 4, 3, 4, 4, 4, 4, 3, 3, 4,
       4, 4, 3, 4, 3, 3, 3, 3, 4, 3, 4, 4, 4, 4, 4, 3, 3, 3, 3, 4, 4, 3,
       3, 3, 4, 3, 4, 3, 4, 3, 4, 3, 3, 3, 3, 3, 3, 4, 3, 3, 4, 4, 3, 4,
       3, 3, 3, 4, 3, 4, 3, 4, 4, 4, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 3,
       4, 4, 4, 3, 3, 3, 4, 4, 4, 4, 3, 4, 3, 4, 4, 4, 3, 4, 4, 3, 4, 4,
       4, 4, 4, 4, 4, 4, 4, 3, 4, 3, 4, 4, 3, 4, 3, 3, 4, 3, 4, 3, 3, 3,
       3, 3, 4, 4, 3, 3, 3, 4, 4, 3, 3, 3, 3, 3, 4, 3, 3, 4, 4, 4, 4, 3,
       3, 4, 3, 4, 3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 3, 3, 4, 3, 4, 3, 4, 4,
       4, 3, 3, 4, 3, 4, 4, 3, 4, 3, 4, 4, 4, 4, 3, 4, 3, 3])
```

```
from sklearn.metrics import accuracy_score
```

```
print(round(accuracy_score(y_train,y_train_pred_rf1), 2))
```

```
0.62
```

```
print(round(accuracy_score(y_test,y_test_pred_rf1), 2))
```

```
0.55
```

Confusion Matrix

```
from sklearn.metrics import confusion_matrix
```

```
confusion_matrix = confusion_matrix(y_test, y_test_pred_rf1)
print(confusion_matrix)
```

```
[[  0   0   3   1   0   0]
 [  0   0  13   5   0   0]
 [  0   0 139  40   0   0]
 [  0   0  85 123   0   0]
 [  0   0   3  64   0   0]
 [  0   0   0   4   0   0]]
```

Classification Report

```
from sklearn.metrics import classification_report
```

```
print(classification_report(y_test, y_test_pred_rf1))
```

```
                   precision    recall  f1-score   support

                1       0.00      0.00      0.00         4
                2       0.00      0.00      0.00        18
                3       0.57      0.78      0.66       179
                4       0.52      0.59      0.55       208
                5       0.00      0.00      0.00        67
                6       0.00      0.00      0.00         4

         accuracy                           0.55       480
        macro avg       0.18      0.23      0.20       480
     weighted avg       0.44      0.55      0.49       480

     /usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1308: Undef
       _warn_prf(average, modifier, msg_start, len(result))
     /usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1308: Undef
       _warn_prf(average, modifier, msg_start, len(result))
     /usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1308: Undef
       _warn_prf(average, modifier, msg_start, len(result))
```

```
#_______________________________ Extract Feature Importance
fi = pd.DataFrame({'feature': list(X_train.columns),
                   'importance': rf_mod.feature_importances_}).\
    sort_values('importance', ascending = False)
```

```
fi.head()
```

	feature	importance
10	alcohol	0.448663
1	volatile_acidity	0.150800
6	total_sulfur_dioxide	0.116731
9	sulphates	0.113725
7	density	0.070888

```
# Accuarcy above is 0.62 & 0.55 for Train & test (respectively)
# This accuracy is for having all columns as features in our model
# Lets build a model keeping 4 best features
# that is keeping volatile_acidity, total_sulfur_dioxide, sulphates, alcohol only
```

```
#Create Classifier object
#in our previous experiment Decision Tree model,
#we found ccp_alphas = 0.01 has the best accuarcy
clf_rf1 = RandomForestClassifier(n_estimators =100, ccp_alpha= 0.01, random_state = 14)
```

```
# fit the classifier with x and y data=TRAIN,
#this time with Failure_Type only
X_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1119 entries, 318 to 619
Data columns (total 11 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed_acidity         1119 non-null   float64
 1   volatile_acidity      1119 non-null   float64
 2   citric_acid           1119 non-null   float64
 3   residual_sugar        1119 non-null   float64
 4   chlorides             1119 non-null   float64
 5   free_sulfur_dioxide   1119 non-null   float64
 6   total_sulfur_dioxide  1119 non-null   float64
 7   density               1119 non-null   float64
 8   pH                    1119 non-null   float64
 9   sulphates             1119 non-null   float64
 10  alcohol               1119 non-null   float64
dtypes: float64(11)
memory usage: 104.9 KB
```

```
X_train1 = X_train.iloc[ : ,[1,6,9,10]]
X_train1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1119 entries, 318 to 619
Data columns (total 4 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   volatile_acidity      1119 non-null   float64
 1   total_sulfur_dioxide  1119 non-null   float64
 2   sulphates             1119 non-null   float64
 3   alcohol               1119 non-null   float64
dtypes: float64(4)
memory usage: 43.7 KB
```

```
rf_mod1 = clf_rf1.fit(X_train1, y_train)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: DataConversionWarning
  """Entry point for launching an IPython kernel.
```

```
#Prediction
y_train_pred1 = rf_mod1.predict(X_train1)
y_train_pred1
```

```
array([4, 4, 4, ..., 4, 3, 3])
```

```
X_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 480 entries, 667 to 1589
Data columns (total 11 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed_acidity         480 non-null    float64
```

```
 1   volatile_acidity      480 non-null    float64
 2   citric_acid           480 non-null    float64
 3   residual_sugar        480 non-null    float64
 4   chlorides             480 non-null    float64
 5   free_sulfur_dioxide   480 non-null    float64
 6   total_sulfur_dioxide  480 non-null    float64
 7   density               480 non-null    float64
 8   pH                    480 non-null    float64
 9   sulphates             480 non-null    float64
 10  alcohol               480 non-null    float64
dtypes: float64(11)
memory usage: 45.0 KB
```

```
X_test1 = X_test.iloc[ : ,[1,6,9,10]]
X_test1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 480 entries, 667 to 1589
Data columns (total 4 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   volatile_acidity      480 non-null    float64
 1   total_sulfur_dioxide  480 non-null    float64
 2   sulphates             480 non-null    float64
 3   alcohol               480 non-null    float64
dtypes: float64(4)
memory usage: 18.8 KB
```

```
#Prediction
y_test_pred1 = rf_mod1.predict(X_test1)
y_test_pred1
```

```
array([3, 3, 4, 3, 4, 4, 3, 3, 3, 4, 3, 3, 4, 3, 3, 3, 4, 3, 4, 4, 4, 3,
       4, 3, 4, 4, 4, 3, 3, 4, 3, 3, 3, 4, 4, 4, 4, 3, 3, 4, 4, 3, 3, 4,
       3, 4, 3, 4, 4, 4, 4, 4, 4, 3, 3, 4, 3, 3, 3, 4, 4, 3, 4, 3, 4, 3,
       3, 3, 3, 4, 3, 3, 4, 4, 4, 3, 4, 4, 4, 4, 3, 4, 4, 3, 3, 4, 3, 3,
       3, 4, 4, 3, 4, 3, 4, 3, 4, 4, 3, 4, 4, 4, 4, 3, 3, 4, 3, 4, 4, 3,
       3, 3, 4, 4, 3, 3, 4, 3, 4, 3, 3, 3, 4, 3, 3, 3, 4, 3, 4, 3, 4, 4,
       4, 4, 4, 3, 3, 3, 4, 3, 4, 4, 4, 3, 4, 4, 4, 4, 3, 4, 3, 4, 4, 3,
       4, 3, 3, 3, 3, 3, 4, 4, 4, 4, 3, 3, 3, 4, 4, 3, 4, 3, 3, 3, 3, 4,
       3, 4, 4, 3, 3, 3, 3, 3, 4, 4, 3, 4, 3, 3, 4, 4, 3, 3, 4, 4, 3, 3,
       4, 3, 4, 3, 3, 3, 4, 3, 4, 4, 4, 4, 3, 4, 3, 4, 3, 4, 4, 4, 3, 4,
       3, 4, 4, 3, 4, 4, 3, 3, 3, 3, 3, 4, 4, 4, 4, 3, 4, 4, 3, 3, 4, 3,
       4, 3, 3, 4, 3, 4, 4, 4, 3, 4, 3, 3, 4, 4, 3, 4, 3, 4, 3, 3, 3, 3,
       3, 4, 3, 4, 4, 3, 4, 4, 4, 4, 3, 3, 3, 3, 3, 4, 3, 3, 4, 3, 3, 4,
       3, 4, 3, 4, 3, 3, 3, 4, 4, 3, 3, 4, 4, 4, 3, 4, 4, 4, 4, 3, 3, 4,
       4, 4, 3, 4, 3, 3, 3, 3, 4, 3, 4, 4, 4, 4, 4, 3, 3, 3, 3, 4, 4, 3,
       3, 3, 4, 3, 4, 3, 4, 3, 4, 3, 3, 3, 3, 3, 3, 4, 3, 3, 4, 4, 3, 4,
       3, 3, 3, 4, 3, 4, 4, 4, 4, 4, 3, 3, 3, 4, 4, 4, 4, 4, 4, 3, 4, 3,
       4, 4, 4, 3, 3, 3, 4, 4, 4, 4, 3, 4, 3, 4, 4, 4, 3, 4, 4, 3, 4, 4,
       4, 4, 4, 4, 4, 4, 4, 3, 4, 3, 4, 4, 3, 4, 3, 3, 4, 3, 3, 3, 3, 3,
       3, 3, 4, 4, 3, 3, 3, 4, 4, 3, 3, 3, 3, 3, 4, 3, 3, 4, 4, 4, 4, 3,
       3, 4, 3, 4, 3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 3, 3, 4, 3, 4, 3, 3, 4,
       4, 3, 3, 4, 3, 4, 4, 3, 4, 3, 3, 4, 4, 4, 3, 4, 3, 3])
```

```
from sklearn.metrics import accuracy_score
```