

Practical implementation of Decision Tree Algorithm

Submitted by:- Ambarish Singh

Importing required libraries

```
In [62]: 1  ## Importing Necessary Library
2  import pandas as pd
3  import numpy as np
4  from sklearn.model_selection import train_test_split
5  from sklearn.tree import DecisionTreeClassifier
6  from sklearn.metrics import accuracy_score
7  from sklearn.model_selection import GridSearchCV
8
9  import warnings
10 warnings.filterwarnings('ignore')
11
```

Importing CSV File

```
In [4]: 1  ## Importing CSV File
2  df = pd.read_csv("https://raw.githubusercontent.com/shrikant-temburwar/Wine-
```

```
In [5]: 1  ## Checking Top 5 Rows
2  df.head()
```

Out[5]:

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

```
In [6]: 1 ## Checking Bottom 5 Rows
        2 df.tail()
```

Out[6]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcoh
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	11
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	11

```
In [7]: 1 ## Checking Shape of the Dataset, here there are total 1599 Rows and 12 Columns
        2 df.shape
```

Out[7]: (1599, 12)

```
In [8]: 1 ## Checking Target Column Unique Details.
        2 df['quality'].unique()
```

Out[8]: array([5, 6, 7, 4, 8, 3], dtype=int64)

```
In [10]: 1 ## Checking Length of target Columns Unique Details.
         2 len(df['quality'].unique())
```

Out[10]: 6

```
In [9]: 1 ## Checking Total Count of Target Columns Unique Details
        2 df['quality'].value_counts()
        3
```

Out[9]:

5	681
6	638
7	199
4	53
8	18
3	10

Name: quality, dtype: int64

```
In [11]: 1 ## Checking Basic Statistics Analysis via Transpose
2 df.describe().T
3
```

Out[11]:

	count	mean	std	min	25%	50%	75%	max
fixed acidity	1599.0	8.319637	1.741096	4.60000	7.1000	7.90000	9.200000	15.90000
volatile acidity	1599.0	0.527821	0.179060	0.12000	0.3900	0.52000	0.640000	1.58000
citric acid	1599.0	0.270976	0.194801	0.00000	0.0900	0.26000	0.420000	1.00000
residual sugar	1599.0	2.538806	1.409928	0.90000	1.9000	2.20000	2.600000	15.50000
chlorides	1599.0	0.087467	0.047065	0.01200	0.0700	0.07900	0.090000	0.61100
free sulfur dioxide	1599.0	15.874922	10.460157	1.00000	7.0000	14.00000	21.000000	72.00000
total sulfur dioxide	1599.0	46.467792	32.895324	6.00000	22.0000	38.00000	62.000000	289.00000
density	1599.0	0.996747	0.001887	0.99007	0.9956	0.99675	0.997835	1.00369
pH	1599.0	3.311113	0.154386	2.74000	3.2100	3.31000	3.400000	4.01000
sulphates	1599.0	0.658149	0.169507	0.33000	0.5500	0.62000	0.730000	2.00000
alcohol	1599.0	10.422983	1.065668	8.40000	9.5000	10.20000	11.100000	14.90000
quality	1599.0	5.636023	0.807569	3.00000	5.0000	6.00000	6.000000	8.00000

```
In [12]: 1 ## Checking Duplicate Rows in Dataset and there are total 240 Duplicate rows
2 df.duplicated().sum()
3
```

Out[12]: 240

```
In [13]: 1 ## Dropping Duplicate Rows from dataset.
2 df = df.drop_duplicates()
3
```

```
In [14]: 1 ## Again Checking Duplicate Rows in dataset, Now Duplicate Row is Zero after
2 df.duplicated().sum()
3
```

Out[14]: 0

Creating Independent Feature

```
In [16]: 1 X = df.drop("quality", axis =1)
```

Creating Dependent Feature

```
In [17]: 1 y = df['quality']
```

Model Building :-

```
In [18]: 1 from sklearn.model_selection import train_test_split, GridSearchCV
```

```
In [36]: 1 ## Splitting the Dataset  
2 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random  
3
```

```
In [37]: 1 ## Scaling is not Required in Decision tree Problem  
2  
3 '''  
4 from sklearn.preprocessing import StandardScaler  
5 sclr = StandardScaler()  
6 sclr.fit()  
7 sclr.transform()  
8 '''
```

```
Out[37]: '\nfrom sklearn.preprocessing import StandardScaler\nsclr = StandardScaler()\nsclr.fit()\nsclr.transform()\n'
```

Model 1 :- Decision Tree classifier Algorithm

```
In [38]: 1 from sklearn.tree import DecisionTreeClassifier  
2 model = DecisionTreeClassifier()
```

```
In [39]: 1 ## Model Fitting with Decision Tree Classifier on Training dataset  
2 model.fit(X_train, y_train)
```

```
Out[39]: ▾ DecisionTreeClassifier  
DecisionTreeClassifier()
```

```
In [40]: 1 model.score(X_train, y_train)
```

```
Out[40]: 1.0
```

```
In [41]: 1 ## Model Prediction on Testing Dataset  
2 y_predict = model.predict(X_test)
```

```
In [42]: 1 from sklearn.metrics import accuracy_score
```

```
In [43]: 1 ## Accuracy of the Model using Decision Tree Classifier Algorithm
        2 accuracy_score(y_test,y_predict)
        3
```

Out[43]: 0.5073529411764706

Observation

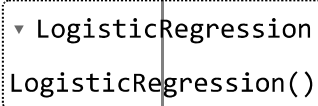
- By using Decision Tree Classifier ,we get Approx 51% Accuracy.

Model 2:- Logistic Regression Algorithm

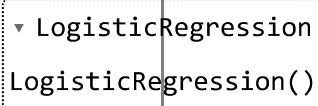
```
In [58]: 1 ## Importing Logistic Regression Algorithm
        2 from sklearn.linear_model import LogisticRegression
        3
```

```
In [59]: 1 LR = LogisticRegression()
```

```
In [60]: 1 LR
```

Out[60]: 

```
In [63]: 1 ## Fitting the Training Data in Logistic Regression
        2 LR.fit(X_train,y_train)
        3
```

Out[63]: 

```
In [65]: 1 ## Predicting Testing data Value using Logistic regression
        2 y_predict1 = LR.predict(X_test)
```

```
In [66]: 1 ## Checking Accuracy score using Logistic Regression
        2 accuracy_score(y_test,y_predict1)
```

Out[66]: 0.5784313725490197

Observation

- By using Logistic Regression ,we get Approx 57% Accuracy.

Model 3:- SVC (Support Vector Classifier) Algorithm.

```
In [67]: 1  ## Importing SVC
        2  from sklearn.svm import SVC
```

```
In [68]: 1  svc = SVC()
```

```
In [69]: 1  svc
```

```
Out[69]: ▾ SVC
          SVC()
```

```
In [70]: 1  ## Fitting the Training Data into SVC Model
        2  svc.fit(X_train,y_train)
        3
```

```
Out[70]: ▾ SVC
          SVC()
```

```
In [73]: 1  ## Predicting Testing data Value using SVC Model
        2  y_predict2 = svc.predict(X_test)
        3
```

```
In [74]: 1  ## Checking Accuracy score using Logistic Regression
        2  accuracy_score(y_test,y_predict2)
```

```
Out[74]: 0.5049019607843137
```

Observation

- By using SVC Model ,we get Approx 50% Accuracy.

Applying GridSearchCV Hyperparameter tuning

```
In [46]: 1  ## Creating Parameter for GridSearch CV
        2
        3  grid_param = {
        4      'criterion': ['gini', 'entropy'],
        5      'max_depth' : range(2,32,1),
        6      'min_samples_leaf' : range(1,10,1),
        7      'min_samples_split': range(2,10,1),
        8      'splitter' : ['best', 'random']
        9
        10 }
```

```
In [47]: 1  ## Importing GridSearchCV Library
          2
          3  from sklearn.model_selection import GridSearchCV
          4  grid_search = GridSearchCV(estimator= model, param_grid = grid_param, cv = 5)
          5
```

```
In [49]: 1 ## Fitting our Training Data into GridSearch.
          2 grid_search.fit(X_train,y_train)      ## GridSearchCV take a Lot of time to RU
          3
```

```
Out[49]:
```

```
GridSearchCV
estimator: DecisionTreeClassifier
DecisionTreeClassifier
```

```
In [50]: 1 ## Finding Best param Model using GridSearch
          2 grid_search.best_params_
          3
```

```
Out[50]: {'criterion': 'gini',
          'max_depth': 4,
          'min_samples_leaf': 6,
          'min_samples_split': 7,
          'splitter': 'random'}
```

Observation

- In GridSearch CV , we found that Best Parameter are as:-
 - a) criterion = 'gini',
 - b) max_depth = 4,
 - c) min_samples_leaf = 6,
 - d) min_samples_split = 7,
 - e) splitter = 'random'

```
In [53]: 1 ## Fitting Best Param Model to Training Dataset.
          2 model_with_best_params.fit(X_train,y_train)
          3
```

```
Out[53]: DecisionTreeClassifier
          ▼
          DecisionTreeClassifier(max_depth=4, min_samples_leaf=6, min_samples_split=7,
                                splitter='random')
```

```
In [54]: 1 ## Predictin the Value
          2 y_prediction2 = model_with_best_params.predict(X_test)
          3
```

```
In [55]: 1 ## Checking Accuracy of Model After Doing HyperParameter Tuning using GridSe
          2 accuracy_score(y_test, y_prediction2)
          3
```

```
Out[55]: 0.571078431372549
```

Observation

- After Doing Hyperparameter Tuning with GridSearch CV, we get Approx. 57% Accuracy.

Thank You