Random Forest ¶

In [3]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [4]:

```
mydata=pd.read_excel("wine (2).xlsx")
```

In [5]:

```
mydata.isnull().sum()
```

Out[5]:

```
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
```

In [6]:
```
mydata.head()
```

Out[6]:

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

In [7]:
```
y_dep=mydata.quality
```

In [8]:
```
y_dep
```

Out[8]:
```
0       5
1       5
2       5
3       6
4       5
       ..
1594    5
1595    6
1596    6
1597    5
1598    6
Name: quality, Length: 1599, dtype: int64
```

In [9]:
```
x_ind=mydata.drop("quality",axis=1)
```

In [10]:

```
x_ind
```

Out[10]:

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

1599 rows × 11 columns

In [11]:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
```

In [12]:

```
x_train,x_test,y_train,y_test=train_test_split(x_ind,y_dep,train_size=0.8,random_state=3)
```

In [13]:

```
model_rf=RandomForestClassifier()
```

In [14]:

```
model_r=model_rf.fit(x_train,y_train)
```

In [15]:
```
y_pred=model_rf.predict(x_test)
```

In [16]:
```
y_pred
```

Out[16]:
```
array([5, 6, 6, 6, 5, 6, 5, 6, 5, 5, 5, 5, 5, 5, 7, 5, 6, 5, 7, 7, 7, 5,
       6, 6, 6, 5, 5, 6, 5, 6, 5, 6, 5, 6, 5, 5, 6, 5, 5, 6, 6, 5, 7, 6,
       5, 5, 6, 5, 5, 6, 5, 6, 6, 6, 6, 5, 5, 5, 5, 5, 5, 5, 5, 6, 6, 6,
       5, 6, 5, 6, 6, 5, 5, 5, 6, 6, 6, 5, 6, 6, 6, 4, 5, 6, 5, 5, 5, 5,
       6, 5, 6, 6, 6, 5, 5, 6, 5, 5, 6, 7, 6, 5, 6, 6, 6, 5, 5, 5, 5, 6,
       6, 5, 7, 6, 5, 5, 5, 5, 6, 6, 6, 5, 6, 5, 6, 6, 6, 6, 6, 5, 5, 6,
       6, 5, 7, 5, 6, 6, 5, 6, 5, 7, 5, 5, 6, 5, 6, 6, 6, 5, 5, 6, 6, 5,
       5, 6, 6, 5, 5, 6, 6, 6, 6, 7, 5, 6, 5, 6, 5, 5, 6, 7, 5, 6, 6, 5,
       5, 5, 6, 6, 7, 6, 6, 6, 5, 6, 5, 6, 7, 6, 6, 7, 5, 5, 6, 5, 6, 6,
       6, 5, 5, 5, 7, 6, 6, 5, 5, 7, 5, 5, 6, 5, 5, 5, 6, 6, 6, 5, 5, 5,
       5, 5, 6, 5, 6, 5, 6, 5, 6, 5, 6, 5, 5, 6, 5, 5, 5, 7, 6, 5, 6, 6,
       6, 5, 8, 7, 6, 5, 5, 6, 5, 6, 6, 6, 5, 6, 7, 5, 6, 5, 6, 7, 6, 6,
       5, 8, 5, 6, 5, 5, 6, 6, 6, 6, 7, 6, 5, 5, 5, 5, 6, 5, 5, 6, 5, 6,
       5, 5, 6, 6, 5, 7, 5, 5, 6, 6, 5, 6, 6, 7, 6, 7, 6, 5, 6, 6, 5, 6,
       5, 5, 6, 5, 6, 6, 6, 5, 5, 6, 5, 5], dtype=int64)
```

In [17]:
```
from sklearn.metrics import confusion_matrix,accuracy_score
```

In [18]:
```
cm=confusion_matrix(y_test,y_pred)
```

In [19]:
```
cm
```

Out[19]:
```
array([[  0,   0,   2,   0,   0,   0],
       [  0,   0,   7,   5,   0,   0],
       [  0,   1, 113,  24,   0,   0],
       [  0,   0,  26,  99,   6,   0],
       [  0,   0,   1,  15,  18,   2],
       [  0,   0,   0,   1,   0,   0]], dtype=int64)
```

In [20]:
```
accuracy_score(y_test,y_pred)
```

Out[20]:
```
0.71875
```

Hyper Parameter Tuning

In [21]:
```
from sklearn.model_selection import RandomizedSearchCV
```

In [22]:
```
parameters={"n_estimators":(200,300,400,500,600,7000),"criterion":("gini","entropy"),
           "max_features":("auto","sqrt","log2"),"min_samples_split":(2,4,6),"random_state":(0,1,2,3)}
```

In [23]:
```
RF=RandomizedSearchCV(RandomForestClassifier(),param_distributions=parameters,cv=5)
```

In [24]:
```
RF.fit(x_train,y_train)
```

Out[24]:
```
RandomizedSearchCV(cv=5, estimator=RandomForestClassifier(),
                   param_distributions={'criterion': ('gini', 'entropy'),
                                        'max_features': ('auto', 'sqrt',
                                                         'log2'),
                                        'min_samples_split': (2, 4, 6),
                                        'n_estimators': (200, 300, 400, 500,
                                                         600, 7000),
                                        'random_state': (0, 1, 2, 3)})
```

In [25]:
```
RF.best_estimator_
```

Out[25]:
```
RandomForestClassifier(n_estimators=400, random_state=3)
```

In [26]:
```
# hyper parameter is used to choose the best option for the model wheather gini or entropy
```

In [27]:
```
# best estimator is entropy
```

In [28]:
```
model_after_hp=RandomForestClassifier(criterion='entropy', max_features='log2',
                       min_samples_split=6, n_estimators=600, random_state=3)
```

Type *Markdown* and LaTeX: α^2

In [29]:
```
model_after_hp=model_after_hp.fit(x_train,y_train)
```

```
In [30]: y_pred_after_hp=model_after_hp.predict(x_test)
```

```
In [33]: from sklearn.metrics import confusion_matrix,accuracy_score
```

```
In [34]: confusion_matrix(y_test,y_pred_after_hp)
```

```
Out[34]: array([[  0,   0,   2,   0,   0,   0],
                [  0,   0,   9,   3,   0,   0],
                [  0,   0, 116,  22,   0,   0],
                [  0,   0,  28,  98,   5,   0],
                [  0,   0,   1,  16,  19,   0],
                [  0,   0,   0,   1,   0,   0]], dtype=int64)
```

```
In [35]: accuracy_score(y_test,y_pred_after_hp)
```

```
Out[35]: 0.728125
```

```
In [ ]: 
```