1. Import Necessary libraries

In [26]:
```
import pandas as pd
from sklearn.datasets import load_wine

import warnings
warnings.filterwarnings('ignore')
```

2. Import Data

In [4]:
```
wine_data = load_wine()
```

In [6]:
```
print(wine_data.DESCR)
```

```
.. _wine_dataset:

Wine recognition dataset
------------------------

**Data Set Characteristics:**

    :Number of Instances: 178 (50 in each of three classes)
    :Number of Attributes: 13 numeric, predictive attributes and the class
    :Attribute Information:
                - Alcohol
                - Malic acid
                - Ash
                - Alcalinity of ash
                - Magnesium
                - Total phenols
                - Flavanoids
                - Nonflavanoid phenols
                - Proanthocyanins
                - Color intensity
                - Hue
                - OD280/OD315 of diluted wines
                - Proline

    - class:
            - class_0
            - class_1
            - class_2

    :Summary Statistics:

    ============================== ==== ===== ======= =====
                                    Min   Max   Mean     SD
    ============================== ==== ===== ======= =====
    Alcohol:                      11.0  14.8    13.0   0.8
    Malic Acid:                   0.74  5.80    2.34  1.12
    Ash:                          1.36  3.23    2.36  0.27
    Alcalinity of Ash:            10.6  30.0    19.5   3.3
    Magnesium:                    70.0 162.0    99.7  14.3
    Total Phenols:                0.98  3.88    2.29  0.63
    Flavanoids:                   0.34  5.08    2.03  1.00
    Nonflavanoid Phenols:         0.13  0.66    0.36  0.12
```

```
    Proanthocyanins:              0.41  3.58    1.59  0.57
    Colour Intensity:              1.3  13.0     5.1   2.3
    Hue:                          0.48  1.71    0.96  0.23
    OD280/OD315 of diluted wines: 1.27  4.00    2.61  0.71
    Proline:                       278  1680     746   315
    ============================= ==== ===== ======= =====

    :Missing Attribute Values: None
    :Class Distribution: class_0 (59), class_1 (71), class_2 (48)
    :Creator: R.A. Fisher
    :Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
    :Date: July, 1988

This is a copy of UCI ML Wine recognition datasets.
https://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data

The data is the results of a chemical analysis of wines grown in the same
region in Italy by three different cultivators. There are thirteen different
measurements taken for different constituents found in the three types of
wine.

Original Owners:

Forina, M. et al, PARVUS -
An Extendible Package for Data Exploration, Classification and Correlation.
Institute of Pharmaceutical and Food Analysis and Technologies,
Via Brigata Salerno, 16147 Genoa, Italy.

Citation:

Lichman, M. (2013). UCI Machine Learning Repository
[https://archive.ics.uci.edu/ml]. Irvine, CA: University of California,
School of Information and Computer Science.

.. topic:: References

  (1) S. Aeberhard, D. Coomans and O. de Vel,
  Comparison of Classifiers in High Dimensional Settings,
  Tech. Rep. no. 92-02, (1992), Dept. of Computer Science and Dept. of
  Mathematics and Statistics, James Cook University of North Queensland.
  (Also submitted to Technometrics).
```

```
The data was used with many others for comparing various
classifiers. The classes are separable, though only RDA
has achieved 100% correct classification.
(RDA : 100%, QDA 99.4%, LDA 98.9%, 1NN 96.1% (z-transformed data))
(All results using the leave-one-out technique)

(2) S. Aeberhard, D. Coomans and O. de Vel,
"THE CLASSIFICATION PERFORMANCE OF RDA"
Tech. Rep. no. 92-01, (1992), Dept. of Computer Science and Dept. of
Mathematics and Statistics, James Cook University of North Queensland.
(Also submitted to Journal of Chemometrics).
```

In [11]:
```
wine_data_df = pd.DataFrame(data = wine_data.data,columns = wine_data.feature_names)
wine_data_df['target'] = wine_data.target
wine_data_df
```

Out[11]:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	hue	oc
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	0.28	2.29	5.64	1.04	
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	0.26	1.28	4.38	1.05	
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	0.30	2.81	5.68	1.03	
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	0.24	2.18	7.80	0.86	
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	0.39	1.82	4.32	1.04	
...	...	...	...	...	...	...	...	...	...	...	...	
173	13.71	5.65	2.45	20.5	95.0	1.68	0.61	0.52	1.06	7.70	0.64	
174	13.40	3.91	2.48	23.0	102.0	1.80	0.75	0.43	1.41	7.30	0.70	
175	13.27	4.28	2.26	20.0	120.0	1.59	0.69	0.43	1.35	10.20	0.59	
176	13.17	2.59	2.37	20.0	120.0	1.65	0.68	0.53	1.46	9.30	0.60	
177	14.13	4.10	2.74	24.5	96.0	2.05	0.76	0.56	1.35	9.20	0.61	

178 rows × 14 columns

3. Data Understanding

In [12]:
```
wine_data_df.shape
```

Out[12]:
```
(178, 14)
```

In [13]:
```
wine_data_df.isna().sum()
```

Out[13]:
```
alcohol                         0
malic_acid                      0
ash                             0
alcalinity_of_ash               0
magnesium                       0
total_phenols                   0
flavanoids                      0
nonflavanoid_phenols            0
proanthocyanins                 0
color_intensity                 0
hue                             0
od280/od315_of_diluted_wines    0
proline                         0
target                          0
dtype: int64
```

In [14]:
```
wine_data_df.dtypes
```

Out[14]:
```
alcohol                         float64
malic_acid                      float64
ash                             float64
alcalinity_of_ash               float64
magnesium                       float64
total_phenols                   float64
flavanoids                      float64
nonflavanoid_phenols            float64
proanthocyanins                 float64
color_intensity                 float64
hue                             float64
od280/od315_of_diluted_wines    float64
proline                         float64
target                            int32
dtype: object
```

5. Model Building

In [18]:
```
X = wine_data_df.drop('target',axis = 1)
y = wine_data_df[['target']]
```

In [19]:
```
X.shape,y.shape
```

Out[19]:
```
((178, 13), (178, 1))
```

In [21]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.20,stratify=y,random_state=12)
```

In [22]:
```
X_train.shape,y_train.shape
```

Out[22]:
```
((142, 13), (142, 1))
```

In [23]:
```
X_test.shape,y_test.shape
```

Out[23]:
```
((36, 13), (36, 1))
```

6. Model Training || 7. Model Testing || 8. Model Evaluation

For k= 3

In [33]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

knn_model = KNeighborsClassifier(n_neighbors=3)
knn_model.fit(X_train,y_train)

y_pred = knn_model.predict(X_train)
accuracy_score(y_train,y_pred)
```

Out[33]:
```
0.823943661971831
```

For k= 5

In [34]:
```
knn_model = KNeighborsClassifier(n_neighbors=5)
knn_model.fit(X_train,y_train)

y_pred = knn_model.predict(X_train)
accuracy_score(y_train,y_pred)
```

Out[34]:
```
0.7605633802816901
```

For k= 7

In [35]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

knn_model = KNeighborsClassifier(n_neighbors=7)
knn_model.fit(X_train,y_train)

y_pred = knn_model.predict(X_train)
accuracy_score(y_train,y_pred)
```

Out[35]:
```
0.7535211267605634
```

For k= 9

In [36]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

knn_model = KNeighborsClassifier(n_neighbors=9)
knn_model.fit(X_train,y_train)

y_pred = knn_model.predict(X_train)
accuracy_score(y_train,y_pred)
```

Out[36]:
```
0.7676056338028169
```

==========

APPLYING STANDARIZATION

In [39]:
```
from sklearn.preprocessing import StandardScaler
std_scaler = StandardScaler()
X_scaled = std_scaler.fit_transform(X)
X_scaled = pd.DataFrame(X_scaled,columns=X.columns)
X_scaled
```

Out[39]:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	
0	1.518613	-0.562250	0.232053	-1.169593	1.913905	0.808997	1.034819	-0.659563	1.224884	0.251717	0
1	0.246290	-0.499413	-0.827996	-2.490847	0.018145	0.568648	0.733629	-0.820719	-0.544721	-0.293321	0
2	0.196879	0.021231	1.109334	-0.268738	0.088358	0.808997	1.215533	-0.498407	2.135968	0.269020	0
3	1.691550	-0.346811	0.487926	-0.809251	0.930918	2.491446	1.466525	-0.981875	1.032155	1.186068	-0
4	0.295700	0.227694	1.840403	0.451946	1.281985	0.808997	0.663351	0.226796	0.401404	-0.319276	0
...	...	...	...	...	...	...	...	...	...	...	
173	0.876275	2.974543	0.305159	0.301803	-0.332922	-0.985614	-1.424900	1.274310	-0.930179	1.142811	-1
174	0.493343	1.412609	0.414820	1.052516	0.158572	-0.793334	-1.284344	0.549108	-0.316950	0.969783	-1
175	0.332758	1.744744	-0.389355	0.151661	1.422412	-1.129824	-1.344582	0.549108	-0.422075	2.224236	-1
176	0.209232	0.227694	0.012732	0.151661	1.422412	-1.033684	-1.354622	1.354888	-0.229346	1.834923	-1
177	1.395086	1.583165	1.365208	1.502943	-0.262708	-0.392751	-1.274305	1.596623	-0.422075	1.791666	-1

178 rows × 13 columns

In [40]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X_scaled,y,test_size=0.20,stratify=y,random_state=12)
```

In [41]:
```
X_train.shape,y_train.shape
```

Out[41]:
```
((142, 13), (142, 1))
```

In [42]:
```
X_test.shape,y_test.shape
```

Out[42]:
```
((36, 13), (36, 1))
```

6. Model Training || 7. Model Testing || 8. Model Evaluation

For k= 3

In [43]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

knn_model = KNeighborsClassifier(n_neighbors=3)
knn_model.fit(X_train,y_train)

y_pred = knn_model.predict(X_train)
accuracy_score(y_train,y_pred)
```

Out[43]: 0.971830985915493

For k= 5

In [44]:
```
knn_model = KNeighborsClassifier(n_neighbors=5)
knn_model.fit(X_train,y_train)

y_pred = knn_model.predict(X_train)
accuracy_score(y_train,y_pred)
```

Out[44]: 0.9788732394366197

For k= 7

In [45]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

knn_model = KNeighborsClassifier(n_neighbors=7)
knn_model.fit(X_train,y_train)

y_pred = knn_model.predict(X_train)
accuracy_score(y_train,y_pred)
```

Out[45]: 0.9788732394366197

For k= 9

In [46]:
```
from sklearn.neighbors import KNeighborsClassifier
```

```
from sklearn.metrics import accuracy_score

knn_model = KNeighborsClassifier(n_neighbors=9)
knn_model.fit(X_train,y_train)

y_pred = knn_model.predict(X_train)
accuracy_score(y_train,y_pred)
```

Out[46]: 0.971830985915493

==========

How to find the Optimal Number of Clusters?????

In [54]:
```
X_scaled
```

Out[54]:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	
0	1.518613	-0.562250	0.232053	-1.169593	1.913905	0.808997	1.034819	-0.659563	1.224884	0.251717	0
1	0.246290	-0.499413	-0.827996	-2.490847	0.018145	0.568648	0.733629	-0.820719	-0.544721	-0.293321	0
2	0.196879	0.021231	1.109334	-0.268738	0.088358	0.808997	1.215533	-0.498407	2.135968	0.269020	0
3	1.691550	-0.346811	0.487926	-0.809251	0.930918	2.491446	1.466525	-0.981875	1.032155	1.186068	-0
4	0.295700	0.227694	1.840403	0.451946	1.281985	0.808997	0.663351	0.226796	0.401404	-0.319276	0
...	...	...	...	...	...	...	...	...	...	...	
173	0.876275	2.974543	0.305159	0.301803	-0.332922	-0.985614	-1.424900	1.274310	-0.930179	1.142811	-1
174	0.493343	1.412609	0.414820	1.052516	0.158572	-0.793334	-1.284344	0.549108	-0.316950	0.969783	-1
175	0.332758	1.744744	-0.389355	0.151661	1.422412	-1.129824	-1.344582	0.549108	-0.422075	2.224236	-1
176	0.209232	0.227694	0.012732	0.151661	1.422412	-1.033684	-1.354622	1.354888	-0.229346	1.834923	-1
177	1.395086	1.583165	1.365208	1.502943	-0.262708	-0.392751	-1.274305	1.596623	-0.422075	1.791666	-1

178 rows × 13 columns

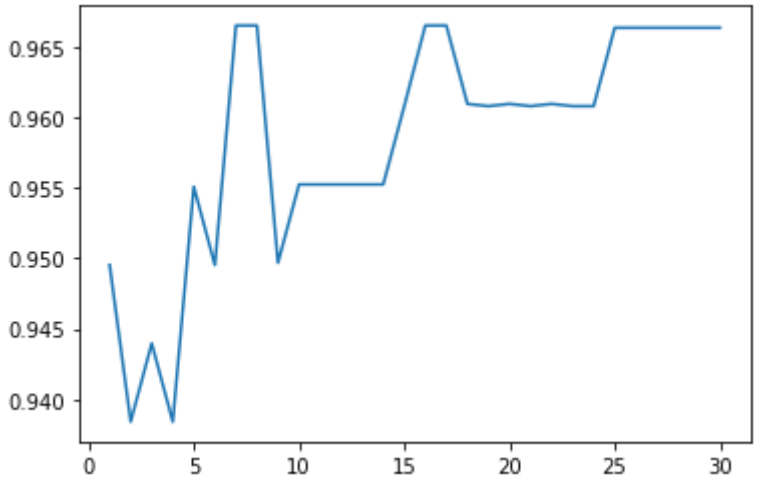
In [55]:

```
from sklearn.model_selection import cross_val_score

container_cv_scores = []

for i in range(1,31):
    knn_model = KNeighborsClassifier(n_neighbors=i)
    cv_scores = cross_val_score(estimator =knn_model, X = X_scaled,y=y,cv = 5)
    container_cv_scores.append(cv_scores.mean())
container_cv_scores
```

Out[55]:
```
[0.9495238095238095,
 0.9384126984126985,
 0.943968253968254,
 0.9384126984126985,
 0.9550793650793651,
 0.9495238095238095,
 0.9665079365079364,
 0.9665079365079364,
 0.9496825396825397,
 0.9552380952380952,
 0.9552380952380952,
 0.9552380952380952,
 0.9552380952380952,
 0.9552380952380952,
 0.9607936507936508,
 0.9665079365079364,
 0.9665079365079364,
 0.9609523809523809,
 0.9607936507936508,
 0.9609523809523809,
 0.9607936507936508,
 0.9609523809523809,
 0.9607936507936508,
 0.9607936507936508,
 0.9663492063492063,
 0.9663492063492063,
 0.9663492063492063,
 0.9663492063492063,
 0.9663492063492063,
 0.9663492063492063]
```

In [56]:
```
from matplotlib import pyplot as plt
plt.plot(range(1,31),container_cv_scores)
```

Out[56]:
```
[<matplotlib.lines.Line2D at 0x2c09d1c9c70>]
```



In [60]:
```
container_cv_scores.index(max(container_cv_scores)) #This returns the index number. Wkt, index number starts from 0.
```

Out[60]: 6

OBSERVATION - K = 7 is the optimal number.

In [57]:
```
knn_model = KNeighborsClassifier(n_neighbors=7)
knn_model.fit(X_train,y_train)

y_pred = knn_model.predict(X_train)
accuracy_score(y_train,y_pred)
```

Out[57]: 0.9788732394366197

THE END