DELHI TECHNOLOGICAL UNIVERSITY

PATTERN RECOGNITION

CO-324

Submitted to:
Mr. Anurag Goel
Department of Computer Science and Engineering

Submitted by:
Saksham Gera (2K18/EE/179)
Shivam Shaurya (2K18/EE/194)

Que1.

- **Data Preprocessing**

Dataset and Libraries used:

Libraries: Sklearn (import classifier,Gridsearch KNN), Numpy (convert to numpy array), Matplotlib.

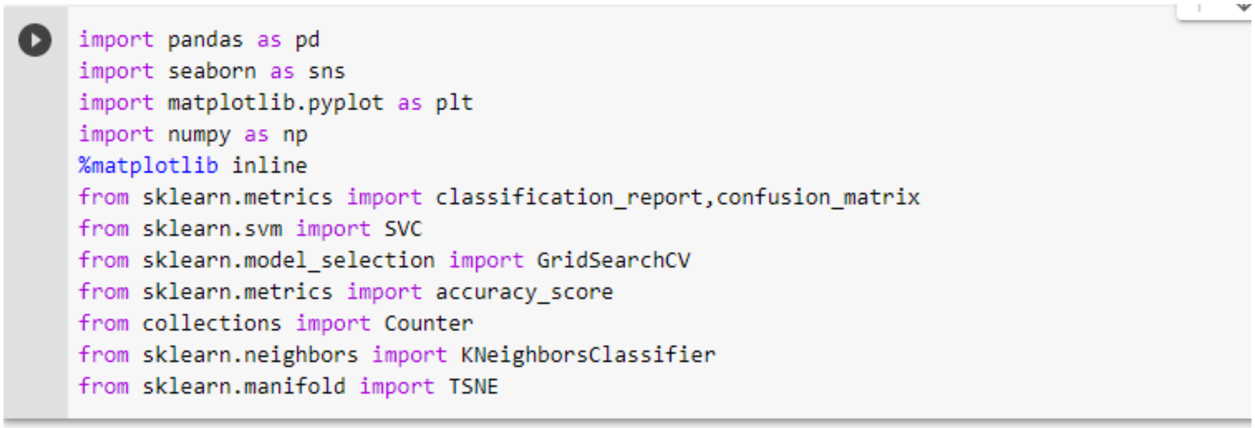
Dataset: Database consists of multi-spectral values of pixels in 3x3 neighbourhoods in a satellite image, and classification associated with the central pixel in each neighbourhood.

There are 6 decision classes: 1,2,3,4,5,7.

1	Red soil
2	cotton crop
3	grey soil
4	damp grey soil
5	soil with vegetation stubble
6	mixture class
7	very damp grey soil

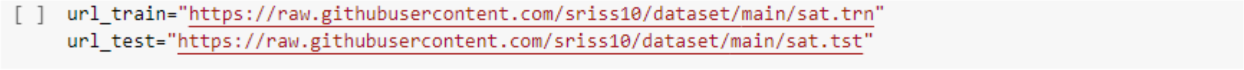
Training set=4435 examples and testing set 2000 examples

Importing Libraries:



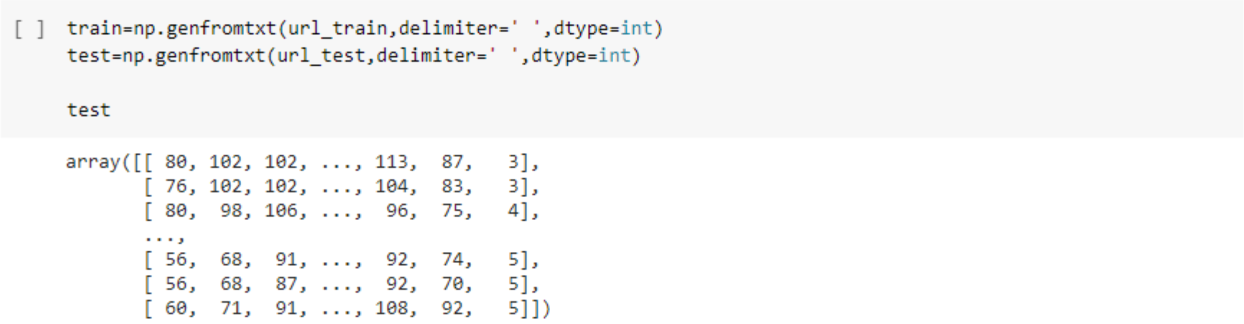
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline
from sklearn.metrics import classification_report,confusion_matrix
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score
from collections import Counter
from sklearn.neighbors import KNeighborsClassifier
from sklearn.manifold import TSNE
```

Loading Dataset:



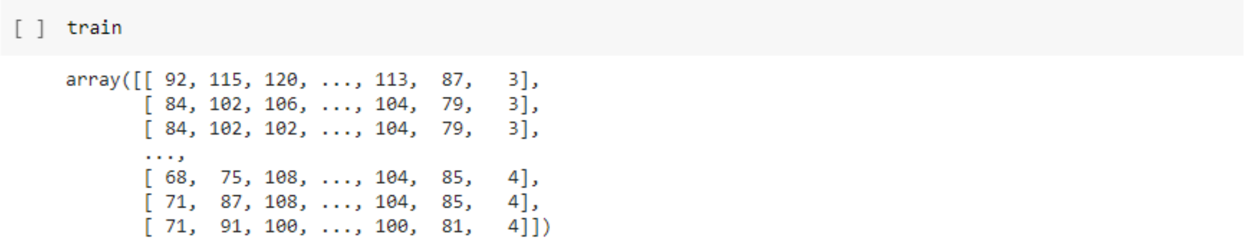
```
url_train="https://raw.githubusercontent.com/sriss10/dataset/main/sat.trn"
url_test="https://raw.githubusercontent.com/sriss10/dataset/main/sat.tst"
```

Converting Dataset to numpy array:



```
train=np.genfromtxt(url_train,delimiter=' ',dtype=int)
test=np.genfromtxt(url_test,delimiter=' ',dtype=int)

test
```

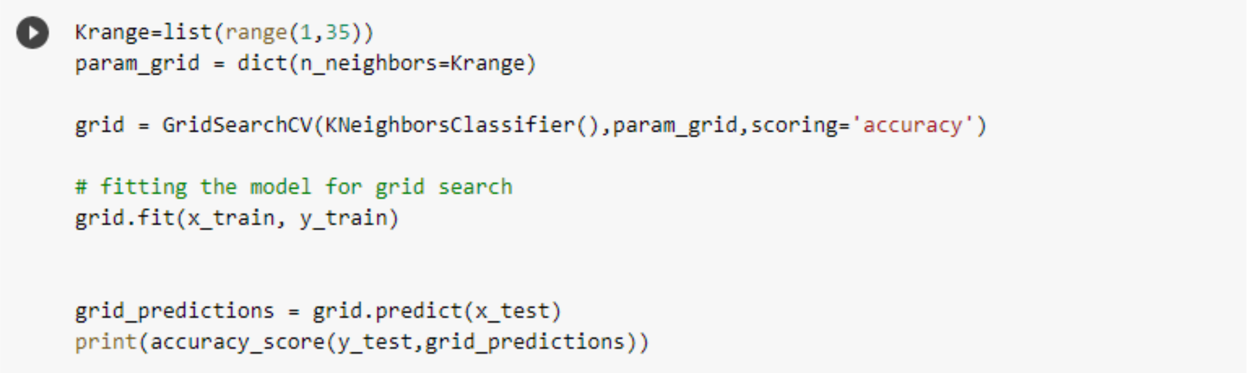
```
array([[ 80, 102, 102, ..., 113,  87,   3],
       [ 76, 102, 102, ..., 104,  83,   3],
       [ 80,  98, 106, ...,  96,  75,   4],
       ...,
       [ 56,  68,  91, ...,  92,  74,   5],
       [ 56,  68,  87, ...,  92,  70,   5],
       [ 60,  71,  91, ..., 108,  92,   5]])
```



```
train
```

```
array([[ 92, 115, 120, ..., 113,  87,   3],
       [ 84, 102, 106, ..., 104,  79,   3],
       [ 84, 102, 102, ..., 104,  79,   3],
       ...,
       [ 68,  75, 108, ..., 104,  85,   4],
       [ 71,  87, 108, ..., 104,  85,   4],
       [ 71,  91, 100, ..., 100,  81,   4]])
```

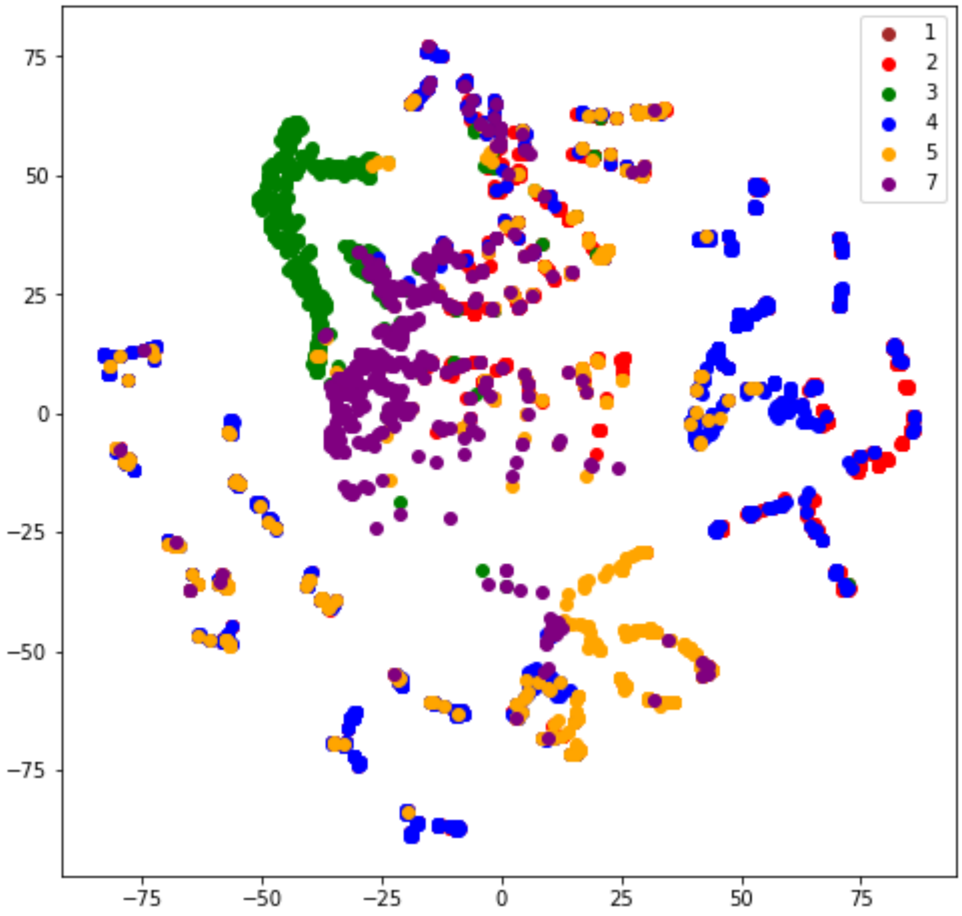
GridSearch Method:



```
Krange=list(range(1,35))
param_grid = dict(n_neighbors=Krange)

grid = GridSearchCV(KNeighborsClassifier(),param_grid,scoring='accuracy')

# fitting the model for grid search
grid.fit(x_train, y_train)


grid_predictions = grid.predict(x_test)
print(accuracy_score(y_test,grid_predictions))
```

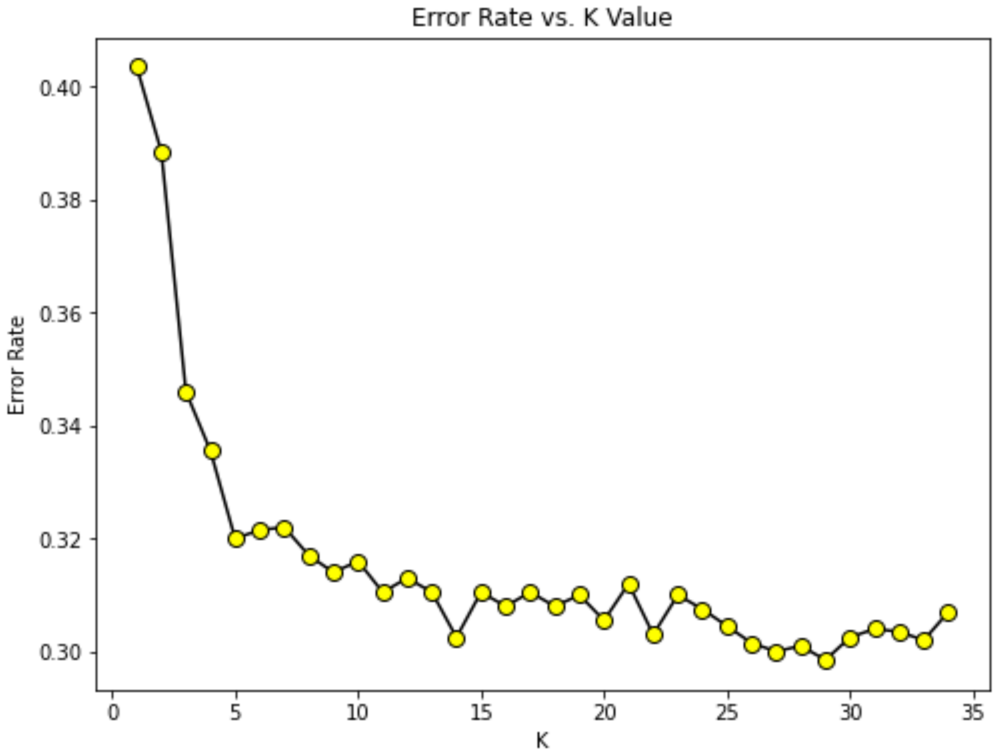
RESULTS:

Tsne plot to visualise the dataset



Error Vs K value plot (to find optimal value of K)

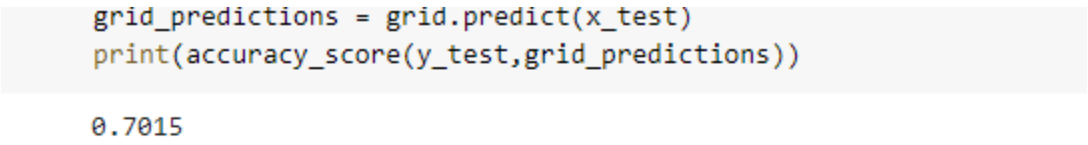
```
Minimum error at K = 28
```



Minimum error at K=28. Therefore, optimal value of K obtained =28

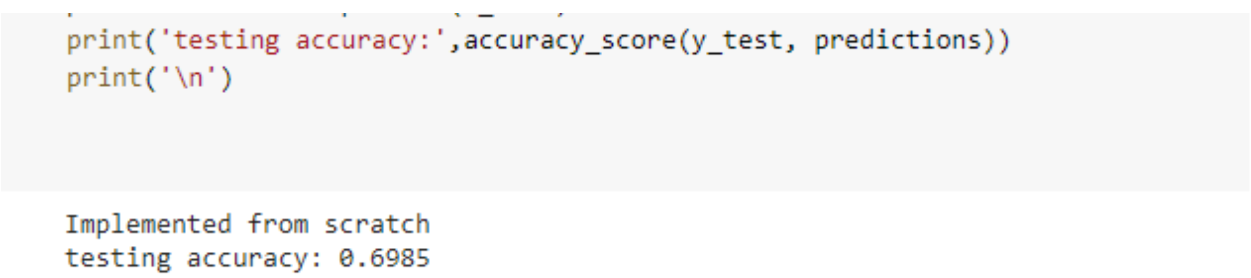
<u>Accuracy:</u>

- ➢ Testing Accuracy when knn implemented using Grid search method.



```
grid_predictions = grid.predict(x_test)
print(accuracy_score(y_test,grid_predictions))

0.7015
```

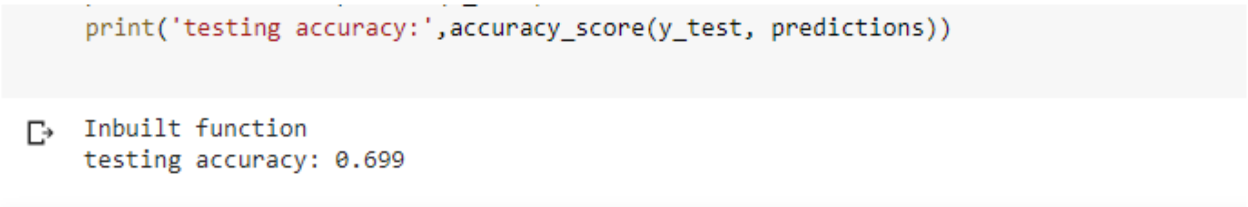
- ➢ Testing accuracy for the dataset when knn implemented from scratch.



```
print('testing accuracy:',accuracy_score(y_test, predictions))
print('\n')
```

```
Implemented from scratch
testing accuracy: 0.6985
```

- ➢ Testing accuracy for the dataset when knn is implemented using inbuilt sklearn function.



```
print('testing accuracy:',accuracy_score(y_test, predictions))
```

```
Inbuilt function
testing accuracy: 0.699
```

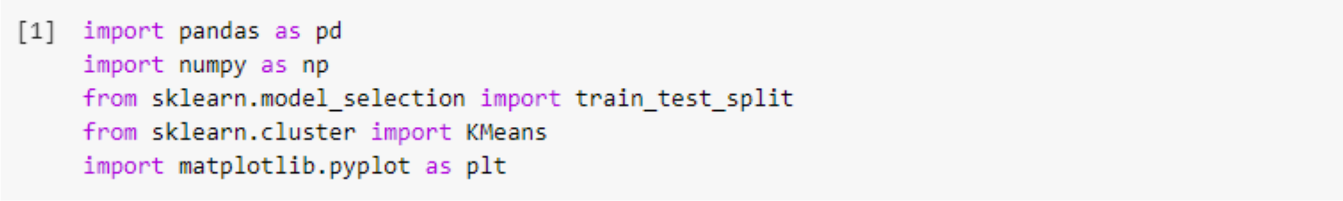
Therefore, for optimal value of the number of neighbours (i.e. K) implementing knn using sklearn inbuilt function yields better accuracy than implementing knn from scratch.

Question2.

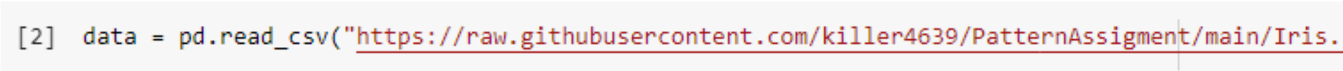
Data Preprocessing

Dataset and Libraries used:

Libraries: Sklearn (for KMeans , train_test split), Numpy, Matplotlib.



```
[1]  import pandas as pd
     import numpy as np
     from sklearn.model_selection import train_test_split
     from sklearn.cluster import KMeans
     import matplotlib.pyplot as plt
```



```
[2]  data = pd.read_csv("https://raw.githubusercontent.com/killer4639/PatternAssigment/main/Iris.
```

Dataset:

The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are NOT linearly separable from each other.

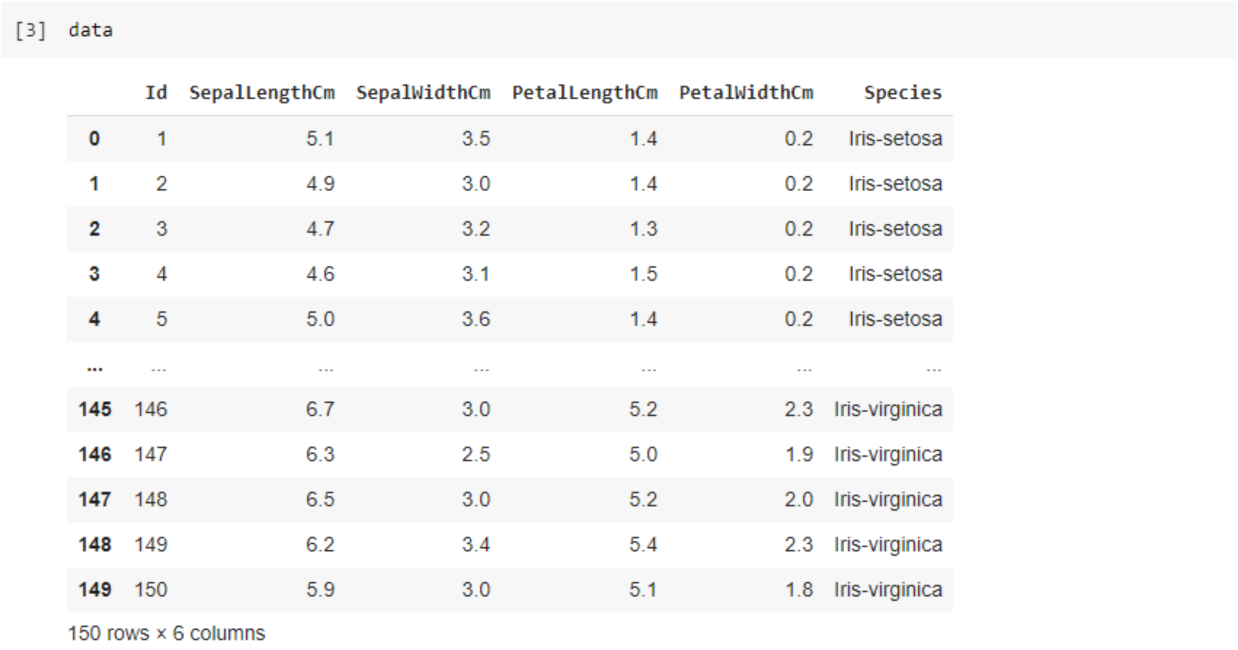
Three classes are:

-> Iris Setosa

->Iris versicolour

->Iris Virginica

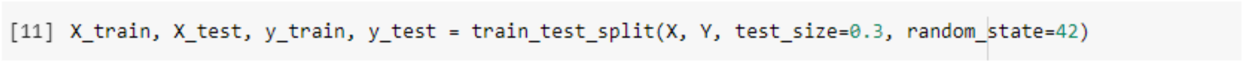
We will split our data for training and testing in a 70:30 ratio.



```
[3] data
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

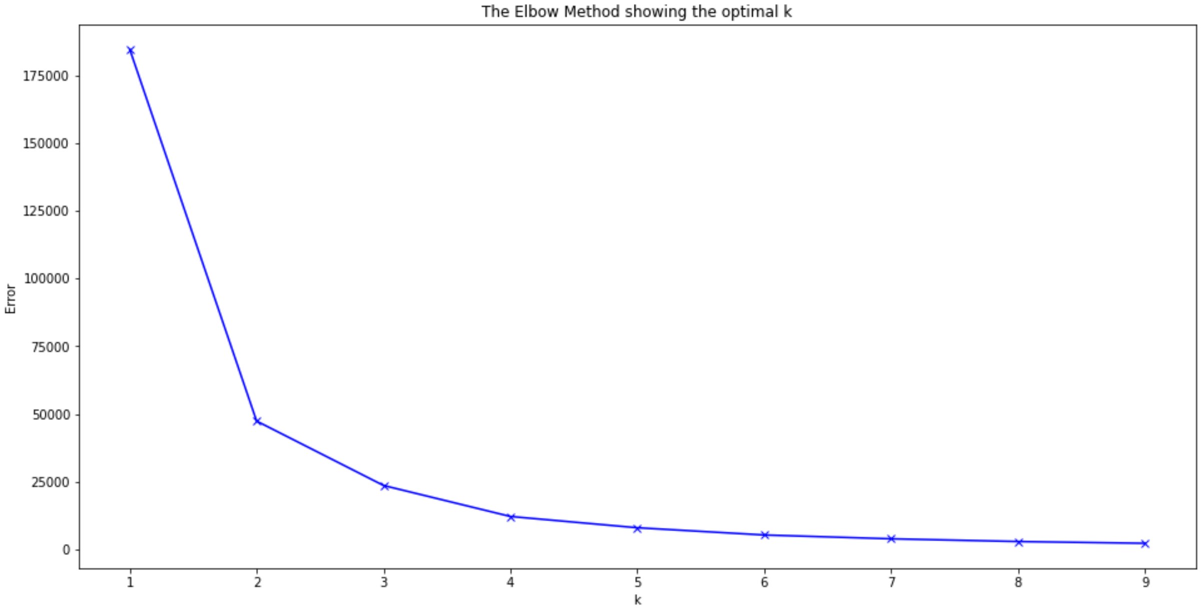
150 rows × 6 columns



```
[11] X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3, random_state=42)
```

RESULTS:

Error V/s Number of Cluster plot

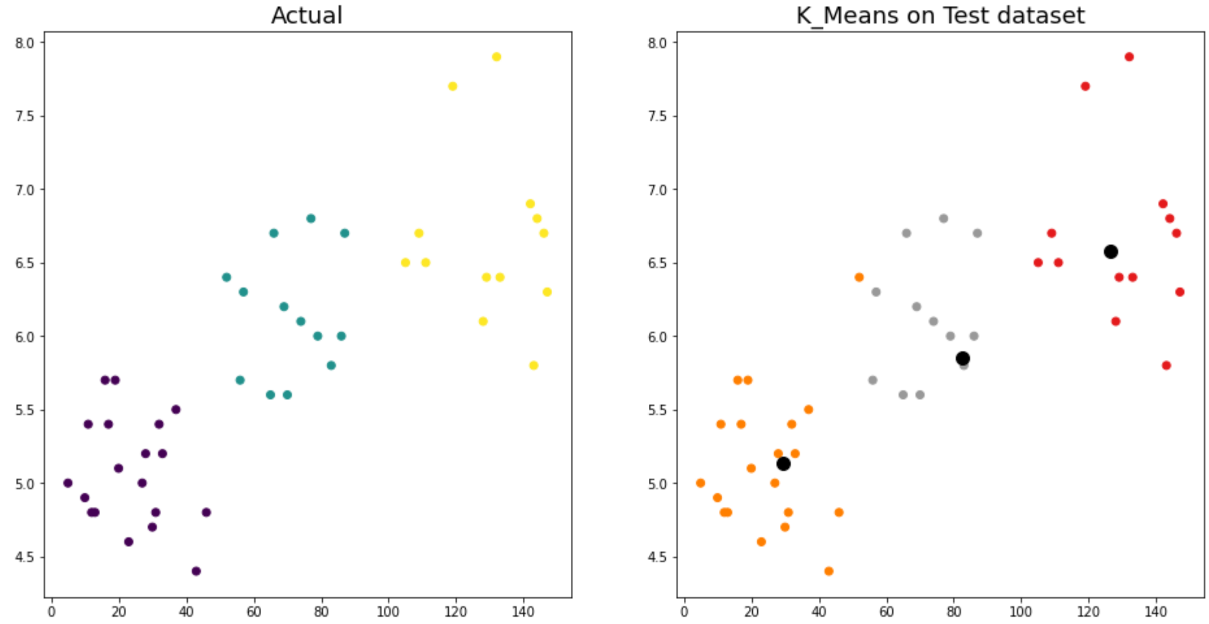


Optimal number of clusters= 3 (i.e Elbow point of the curve from the above graph)

Scatter plot to visualize the dataset to depict the clusters formed (optimal).

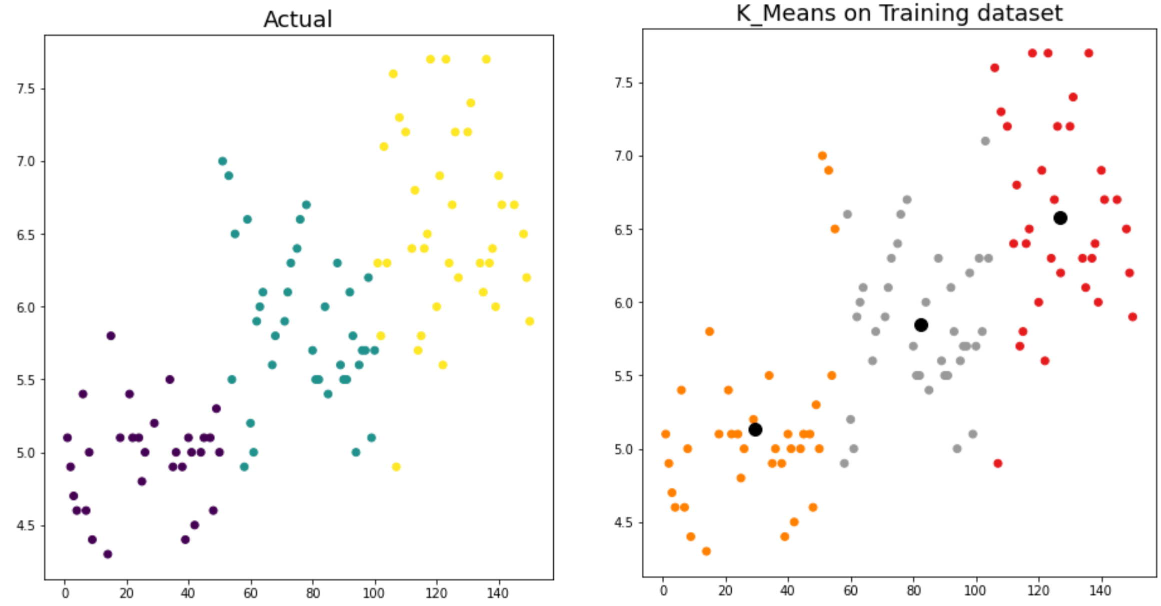
visualization of actual and cluster formed for Testing dataset

```
<matplotlib.collections.PathCollection at 0x7f1122609a90>
```



visualization of actual and cluster formed for Training dataset

```
<matplotlib.collections.PathCollection at 0x7f1122af8e90>
```


Actual
K_Means on Training dataset