Data Science February Minor Project

Project Description :

Problem Statement : Create a classification model to predict whether price range of mobile based on certain specifications

Context: An entrepreneur has started his own mobile company. He wants to give tough fight to big companies like Apple, Samsung etc. He does not know how to estimate price of mobiles his company creates. In this competitive mobile phone market, one cannot simply assume things. To solve this problem, he collects sales data of mobile phones of various companies. He wants to find out some relation between features of a mobile phone (e.g., RAM, Internal Memory etc) and its selling price. But he is not so good at Machine Learning. So, he needs your help to solve this problem. In this problem you do not have to predict actual price but a price range indicating how high the price is.

Steps to consider:

1)Remove handle null values (if any).

2)Split data into training and test data.

3)Apply the following models on the training dataset and generate the predicted value for the test dataset:

a) Logistic Regression b) KNN Classification c) SVM Classifier with linear and rbf kernel

4)Predict the price range for test data

5)Compute Confusion matrix and classification report for each of these models.

6)Report the model with the best accuracy.

LOGISTIC REGRESSION

In [35]:
```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import seaborn as sns
```

In [36]:
```
df = pd.read_csv('mobile_price_range_data (1).csv')
df.head()
```

Out[36]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	r
0	842	0	2.2	0	1	0	7	0.6	188	
1	1021	1	0.5	1	0	1	53	0.7	136	
2	563	1	0.5	1	2	1	41	0.9	145	
3	615	1	2.5	0	0	0	10	0.8	131	
4	1821	1	1.2	0	13	1	44	0.6	141	

5 rows × 21 columns

Check if there is any null values

In [37]:
```
df.isnull().sum()
```

Out[37]:
```
battery_power    0
blue             0
clock_speed      0
dual_sim         0
fc               0
four_g           0
int_memory       0
m_dep            0
mobile_wt        0
n_cores          0
pc               0
px_height        0
px_width         0
ram              0
sc_h             0
sc_w             0
talk_time        0
three_g          0
touch_screen     0
wifi             0
price_range      0
dtype: int64
```

In [38]:
```
df.shape
```

Out[38]:
```
(2000, 21)
```

In [39]:
```
df.dtypes
```

Out[39]:
```
battery_power      int64
blue               int64
clock_speed      float64
dual_sim           int64
fc                 int64
four_g             int64
int_memory         int64
m_dep            float64
mobile_wt          int64
n_cores            int64
pc                 int64
px_height          int64
px_width           int64
ram                int64
sc_h               int64
sc_w               int64
talk_time          int64
three_g            int64
touch_screen       int64
wifi               int64
price_range        int64
dtype: object
```

In [40]:
```
x = df.iloc[:,:-1]
y = df.iloc[:,-1]
print(x.shape)
print(y.shape)
```

```
(2000, 20)
(2000,)
```

Split data into test and training

In [41]:
```
from sklearn.model_selection import train_test_split
```

In [42]:
```
x_tr,x_te,y_tr,y_te=train_test_split(x,y,test_size=0.25)
print(x_te.shape)
print(x_tr.shape)
print(y_te.shape)
print(y_tr.shape)
```

```
(500, 20)
(1500, 20)
(500,)
(1500,)
```

Model for predicting price range of mobile using Logistic regression

In [43]:
```
from sklearn.linear_model import LogisticRegression
```

In [44]:
```
reg=LogisticRegression()
reg.fit(x_tr,y_tr)
```

```
D:\newfolder\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:
762: ConvergenceWarning: lbfgs failed to converge (status=1):
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 (https://sci
kit-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-regr
ession (https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression)
  n_iter_i = _check_optimize_result(
```

Out[44]:
```
LogisticRegression()
```

In [46]:
```
print("Training Score",reg.score(x_tr,y_tr))
print("Testing Score",reg.score(x_te,y_te))
```

```
Training Score 0.66
Testing Score 0.598
```

Prediction of price range

In [47]:
```
ypred=reg.predict(x_te)
print(ypred)
```

```
[1 2 2 2 1 3 1 2 2 2 1 1 0 2 0 0 1 0 0 1 0 2 1 1 3 0 0 2 3 1 0 1 2 3 3 3 1
 1 3 3 0 3 0 3 2 2 2 3 2 2 0 1 0 3 1 1 3 1 1 1 2 1 2 3 1 0 0 1 0 1 1 0 3 0
 3 1 1 0 1 3 3 2 3 0 0 0 0 3 3 1 2 1 2 0 3 3 1 1 1 3 1 1 3 1 0 1 1 2 3 1 3
 0 3 0 3 0 1 0 3 0 3 1 1 1 2 1 1 1 3 2 2 0 0 1 3 1 3 2 2 3 0 2 0 2 2 1 0 0
 2 2 0 2 0 2 0 1 2 0 2 0 2 3 0 3 2 0 1 0 1 2 1 1 3 3 0 3 2 0 1 3 1 2 0 2 0
 2 2 2 1 3 2 1 3 2 1 0 0 2 0 3 3 0 1 1 1 1 0 2 0 1 3 3 2 2 0 2 3 0 1 1 1 0
 2 2 2 3 2 1 0 2 3 3 1 3 3 0 3 1 3 3 1 2 3 1 1 2 0 3 1 0 2 3 2 3 2 3 1 3 3
 3 0 2 2 2 3 3 3 1 1 3 2 1 0 1 1 2 2 3 2 3 1 3 0 1 1 0 0 0 3 1 3 2 2 3 1 2
 2 0 3 0 3 3 1 3 1 1 2 0 3 0 0 0 1 0 2 1 3 1 2 1 1 2 1 0 3 0 3 2 1 1 2 2 1
 1 2 3 0 3 3 3 3 0 2 0 3 3 1 2 3 1 0 2 3 3 1 3 3 3 1 0 1 3 1 2 3 0 1 3 3 2
 3 2 1 1 3 2 2 0 1 1 1 0 3 0 1 3 2 3 0 2 0 3 3 2 2 1 1 3 0 2 2 2 1 3 3 0 1
 1 3 3 1 0 3 0 3 1 3 3 1 2 0 3 0 0 1 0 1 1 1 3 3 3 3 3 2 2 2 1 1 2 0 0 1 1
 1 3 1 0 1 3 0 3 1 2 0 1 1 3 1 1 2 3 1 1 3 3 2 1 2 2 0 3 1 1 2 3 1 2 1 2 2
 1 3 1 3 1 1 1 3 0 3 1 2 1 3 1 0 0 1 0]
```

In [48]:
```
x_te['Actual_Values']=y_te
x_te['Predicted_Values']=ypred
print(x_te)
```

```
      battery_power  blue  clock_speed  dual_sim  fc  four_g  int_memory  \
1762            808     1          0.5         1   3       0          46
1195           1009     1          2.8         1   7       1           2
110             783     0          1.8         1   0       1          43
259            1559     1          1.6         1   6       1           6
1978           1483     1          2.2         0   3       1          53
...             ...   ...          ...       ...  ..     ...         ...
731            1807     1          2.1         0   2       0          49
1359           1949     1          0.5         1   2       1          31
898            1372     1          2.7         0   7       0          34
1247           1566     1          0.5         1   0       0          20
279             823     1          0.5         0   0       0          39

      m_dep  mobile_wt  n_cores  ...  px_width   ram  sc_h  sc_w  talk_time
\
1762    0.5        105        8  ...       529  1082    15     5          10
1195    0.6        115        7  ...      1841  2286    19     8          13
110     1.0        106        3  ...      1471  2016    16     4          18
259     0.5        162        6  ...      1559  3352     9     1           3
1978    0.7        169        5  ...       651  1744     6     3          10
...     ...        ...      ...  ...       ...   ...   ...   ...         ...
731     0.8        125        1  ...      1384  1906    17    13          13
1359    0.1        145        5  ...      1182   832     7     3          19
898     0.4        193        4  ...       937   725    11     3          20
1247    0.5        126        2  ...      1930   659    14     9          13
279     0.4        187        8  ...       888   294    13     9          11

      three_g  touch_screen  wifi  Actual_Values  Predicted_Values
1762        1             0     1              0                 1
1195        1             0     0              2                 2
110         1             1     0              1                 2
259         1             1     1              3                 2
1978        1             0     0              1                 1
...       ...           ...   ...            ...               ...
731         0             1     1              2                 1
1359        1             0     0              1                 0
898         1             0     0              0                 0
1247        1             1     1              1                 1
279         0             0     1              0                 0

[500 rows x 22 columns]
```

```
<ipython-input-48-d7f5e6760c50>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  x_te['Actual_Values']=y_te
```

```
<ipython-input-48-d7f5e6760c50>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://p
andas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-vi
ew-versus-a-copy)
  x_te['Predicted_Values']=ypred
```

Confusion matrix and cassification report

In [49]:
```
from sklearn.metrics import confusion_matrix,classification_report
```

In [50]:
```
cm=confusion_matrix(y_te,ypred)
print(cm)
```

```
[[84 29  1  0]
 [21 75 27  5]
 [ 0 45 55 45]
 [ 0  1 27 85]]
```

In [51]:
```
print(classification_report(y_te,ypred))
```

```
              precision    recall  f1-score   support

           0       0.80      0.74      0.77       114
           1       0.50      0.59      0.54       128
           2       0.50      0.38      0.43       145
           3       0.63      0.75      0.69       113

    accuracy                           0.60       500
   macro avg       0.61      0.61      0.61       500
weighted avg       0.60      0.60      0.59       500
```

Accuracy of Logistic regresison model : 0.60

KNN CLASSIFICATION

In [3]:
```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import seaborn as sns
```

In [5]:
```
df = pd.read_csv('mobile_price_range_data (1).csv')
df.head()
```

Out[5]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	r
0	842	0	2.2	0	1	0	7	0.6	188	
1	1021	1	0.5	1	0	1	53	0.7	136	
2	563	1	0.5	1	2	1	41	0.9	145	
3	615	1	2.5	0	0	0	10	0.8	131	
4	1821	1	1.2	0	13	1	44	0.6	141	

5 rows × 21 columns

In [6]:
```
df['price_range'].value_counts()
```

Out[6]:
```
3    500
2    500
1    500
0    500
Name: price_range, dtype: int64
```

Check if any null values

In [7]:
```
df.isnull().sum()
```

Out[7]:
```
battery_power    0
blue             0
clock_speed      0
dual_sim         0
fc               0
four_g           0
int_memory       0
m_dep            0
mobile_wt        0
n_cores          0
pc               0
px_height        0
px_width         0
ram              0
sc_h             0
sc_w             0
talk_time        0
three_g          0
touch_screen     0
wifi             0
price_range      0
dtype: int64
```

In [8]:
```
x=df.iloc[:,:-1]
y=df.iloc[:,-1]
print(x.shape)
print(y.shape)
```

```
(2000, 20)
(2000,)
```

Split into tarining and test data

In [9]:
```
from sklearn.model_selection import train_test_split
```

In [10]:
```
x_tr,x_te,y_tr,y_te=train_test_split(x,y,test_size=0.25)
```

In [11]:
```
print(x_tr.shape)
print(x_te.shape)
print(y_tr.shape)
print(y_te.shape)
```

```
(1500, 20)
(500, 20)
(1500,)
(500,)
```

Build the model

In [12]:
```
from sklearn.neighbors import KNeighborsClassifier
```

In [13]:
```
m1 = KNeighborsClassifier(n_neighbors=19)
m1.fit(x_tr,y_tr)
```

Out[13]:
```
KNeighborsClassifier(n_neighbors=19)
```

In [14]:
```
print('Training score',m1.score(x_tr,y_tr))
print('Testing score',m1.score(x_te,y_te))
```

```
Training score 0.9446666666666667
Testing score 0.922
```

Prediction of price range

In [20]:
```
ypred1 = m1.predict(x_te)
print(ypred1)
```

```
[3 3 2 2 0 3 3 2 2 2 1 2 2 3 3 1 2 3 1 0 3 3 2 0 1 1 1 0 2 3 1 2 3 0 2 2 0
 3 1 3 0 2 3 3 2 2 0 3 0 1 3 2 2 2 3 2 3 3 1 2 3 2 3 0 0 2 3 1 3 3 2 3 2 1
 3 1 2 0 0 3 2 1 2 2 1 2 0 3 3 3 2 1 2 3 0 3 0 2 0 2 3 1 0 2 2 3 1 1 3 3 0
 1 2 1 2 1 2 1 3 3 3 2 3 1 0 1 2 3 0 2 1 3 1 3 0 3 1 3 3 2 1 3 1 2 1 2 0 2
 3 0 2 0 3 3 0 2 1 3 2 0 1 3 1 1 3 1 1 2 1 1 0 1 0 0 0 1 1 3 2 1 2 1 0 0 3
 0 3 2 0 2 0 3 2 3 3 3 0 1 0 0 1 0 1 3 3 1 1 2 3 2 0 3 3 2 0 2 3 2 2 3 1 3
 0 1 3 1 1 0 3 0 0 3 2 2 0 2 0 2 2 1 2 2 2 2 1 1 1 0 1 1 2 0 2 3 0 3 1 1 1
 3 1 3 0 1 2 0 2 1 0 0 1 1 3 0 1 3 2 3 3 1 3 2 1 3 2 2 3 1 2 0 0 0 0 0 0 0
 0 1 1 0 3 0 1 1 0 2 0 2 0 2 2 2 3 0 1 2 3 3 2 2 2 3 1 2 0 0 2 1 1 3 2 1 2
 2 3 0 3 3 1 0 2 2 3 2 3 0 1 1 1 1 3 1 2 0 1 2 0 2 0 3 0 3 0 1 1 0 1 0 3 1
 0 2 1 2 3 3 2 2 1 2 2 1 1 0 0 1 1 3 0 1 2 2 2 3 0 1 1 0 0 1 3 2 0 0 3 0 3
 3 0 3 2 2 0 2 0 0 1 1 2 0 1 3 3 2 1 1 3 1 0 0 0 3 0 1 1 2 0 0 0 2 2 3 0 2
 3 1 0 3 3 0 2 1 3 2 2 1 3 0 1 1 1 1 3 3 2 1 2 0 0 0 2 1 3 2 1 0 0 0 2 3 2
 0 0 3 2 3 0 2 2 0 1 3 0 0 1 1 2 1 0 3]
```

Confusion matrix and Classification report

In [21]:
```
from sklearn.metrics import confusion_matrix,classification_report
```

In [22]:
```
cm1 = confusion_matrix(y_te,ypred1)
print(cm1)
```

```
[[115   4   0   0]
 [  6 114   5   0]
 [  0   5 115   8]
 [  0   0  11 117]]
```

In [23]:
```
print(classification_report(y_te,ypred1))
```

```
              precision    recall  f1-score   support

           0       0.95      0.97      0.96       119
           1       0.93      0.91      0.92       125
           2       0.88      0.90      0.89       128
           3       0.94      0.91      0.92       128

    accuracy                           0.92       500
   macro avg       0.92      0.92      0.92       500
weighted avg       0.92      0.92      0.92       500
```

Accuracy using KNN Classification Model : 0.92

SVM CLASSIFIER WITH RBF AND LINEAR KERNEL

In [18]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
```

In [19]:
```
df = pd.read_csv('mobile_price_range_data (1).csv')
df.head()
```

Out[19]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	r
0	842	0	2.2	0	1	0	7	0.6	188	
1	1021	1	0.5	1	0	1	53	0.7	136	
2	563	1	0.5	1	2	1	41	0.9	145	
3	615	1	2.5	0	0	0	10	0.8	131	
4	1821	1	1.2	0	13	1	44	0.6	141	

5 rows × 21 columns

checking for null values

In [20]:
```
df.isnull().sum()
```

Out[20]:
```
battery_power    0
blue             0
clock_speed      0
dual_sim         0
fc               0
four_g           0
int_memory       0
m_dep            0
mobile_wt        0
n_cores          0
pc               0
px_height        0
px_width         0
ram              0
sc_h             0
sc_w             0
talk_time        0
three_g          0
touch_screen     0
wifi             0
price_range      0
dtype: int64
```

In [21]:
```
df.dtypes  #checking datatyes
```

Out[21]:
```
battery_power       int64
blue                int64
clock_speed       float64
dual_sim            int64
fc                  int64
four_g              int64
int_memory          int64
m_dep             float64
mobile_wt           int64
n_cores             int64
pc                  int64
px_height           int64
px_width            int64
ram                 int64
sc_h                int64
sc_w                int64
talk_time           int64
three_g             int64
touch_screen        int64
wifi                int64
price_range         int64
dtype: object
```

In [22]:
```
x = df.iloc[:,:-1]
y = df.iloc[:,-1]
print(x.shape)
print(y.shape)
```

```
(2000, 20)
(2000,)
```

In [23]:
```
x.head()
```

Out[23]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	r
0	842	0	2.2	0	1	0	7	0.6	188	
1	1021	1	0.5	1	0	1	53	0.7	136	
2	563	1	0.5	1	2	1	41	0.9	145	
3	615	1	2.5	0	0	0	10	0.8	131	
4	1821	1	1.2	0	13	1	44	0.6	141	

In [24]:
```
y.head()
```

Out[24]:
```
0    1
1    2
2    2
3    2
4    1
Name: price_range, dtype: int64
```

Split into tarining and test data

In [25]:
```
from sklearn.model_selection import train_test_split
```

In [26]:
```
x_tr,x_te,y_tr,y_te = train_test_split(x,y,test_size=0.25)
print(x_tr.shape)
print(x_te.shape)
print(y_tr.shape)
print(y_te.shape)
```

```
(1500, 20)
(500, 20)
(1500,)
(500,)
```

Build the SVM model using RBF Kernel

In [27]:
```
from sklearn.svm import SVC
```

In [28]:
```
s = SVC(kernel='rbf',C=1,gamma=0.001)
s.fit(x_tr,y_tr)
```

Out[28]:
```
SVC(C=1, gamma=0.001)
```

In [30]:
```
print('Training score',s.score(x_tr,y_tr))
print('Test score',s.score(x_te,y_te))
```

```
Training score 1.0
Test score 0.246
```

Prediction of price range

In [31]:
```python
ypred = s.predict(x_te)
print(ypred)
```

```
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
```

In [32]:
```python
from sklearn.metrics import confusion_matrix, classification_report
```

Confusion matrix and classification report of RBF Kernel

In [33]:
```python
cm = confusion_matrix(y_te,ypred)
print(cm)
```

```
[[  0 126   0   0]
 [  0 119   0   0]
 [  0 118   3   0]
 [  0 133   0   1]]
```

In [34]:
```python
print(classification_report(y_te,ypred))
```

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       126
           1       0.24      1.00      0.39       119
           2       1.00      0.02      0.05       121
           3       1.00      0.01      0.01       134

    accuracy                           0.25       500
   macro avg       0.56      0.26      0.11       500
weighted avg       0.57      0.25      0.11       500
```

```
D:\newfolder\anaconda3\lib\site-packages\sklearn\metrics\_classification.p
y:1221: UndefinedMetricWarning: Precision and F-score are ill-defined and b
eing set to 0.0 in labels with no predicted samples. Use `zero_division` pa
rameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Accuracy with SVM(RBF Kernel): 0.25

Build SVM Model using Linear Kernel

In [35]:
```
s1 = SVC(kernel='linear',C=1)
s1.fit(x_tr,y_tr)
```

Out[35]:
```
SVC(C=1, kernel='linear')
```

In [36]:
```
print('Training score',s1.score(x_tr,y_tr))
print('Test score',s1.score(x_te,y_te))
```

```
Training score 0.9953333333333333
Test score 0.972
```

Prediction of price range

In [37]:
```
ypred_1=s1.predict(x_te)
print(ypred_1)
```

```
[1 1 3 0 1 3 1 2 3 3 3 2 0 3 0 3 3 0 3 3 1 2 3 2 1 3 0 3 3 2 1 2 0 3 2 2 3
 2 2 1 0 1 1 2 3 2 0 2 1 2 0 3 0 2 0 0 2 3 3 2 2 3 2 3 3 2 0 2 1 2 0 2 2 0
 2 2 0 1 1 1 1 0 3 1 1 0 3 0 2 1 2 0 0 1 0 3 3 2 2 2 3 1 1 2 3 3 2 2 2 3 2
 3 2 0 1 1 1 3 2 1 3 2 1 0 1 1 2 3 3 3 0 3 2 0 1 1 0 1 1 1 3 2 0 2 2 2 2 1
 2 0 2 2 2 0 0 1 3 1 2 3 0 0 0 3 0 0 3 3 3 3 1 2 1 0 1 0 0 1 0 2 0 2 1 0 2
 2 0 3 3 3 1 0 3 0 3 3 0 0 0 3 0 1 1 0 0 3 2 2 3 1 1 2 0 2 3 1 2 3 3 2 1 2
 3 2 1 2 0 2 0 2 1 3 1 3 0 1 0 1 1 0 0 0 1 1 0 2 0 0 0 2 0 1 3 0 0 0 1 3 2
 1 0 1 2 1 3 3 0 0 0 0 2 3 0 3 2 3 0 0 2 2 3 1 1 1 3 3 3 2 1 3 2 2 1 1 2 0
 2 3 0 3 3 0 1 3 1 0 0 3 3 2 0 0 3 0 3 1 0 2 0 2 3 1 2 2 1 0 3 2 3 1 0 3 3
 3 3 2 2 2 2 3 2 3 3 2 2 1 1 0 3 3 2 1 3 2 3 1 3 0 3 0 2 3 1 1 3 0 0 2 0 3
 3 0 1 1 3 0 3 3 2 1 1 1 3 0 1 3 2 1 2 0 2 0 0 0 3 1 0 0 1 2 0 2 2 1 3 0 0
 2 2 0 1 0 3 1 1 0 3 2 3 0 3 2 0 2 2 0 2 1 1 2 1 0 1 2 1 1 2 2 2 0 0 0 3 2
 1 3 0 2 2 1 3 2 3 3 0 1 3 0 0 3 2 3 1 0 0 0 1 3 1 2 1 0 2 3 1 3 0 2 3 1 2
 3 3 3 3 2 1 0 1 3 0 3 3 3 1 1 1 3 0 3]
```

Confusion and Classificatio report

In [38]:
```
cm1 = confusion_matrix(y_te,ypred_1)
print(cm1)
```

```
[[125   1   0   0]
 [  2 112   5   0]
 [  0   0 118   3]
 [  0   0   3 131]]
```

In [39]:
```
print(classification_report(y_te,ypred_1))
```

```
              precision    recall  f1-score   support

           0       0.98      0.99      0.99       126
           1       0.99      0.94      0.97       119
           2       0.94      0.98      0.96       121
           3       0.98      0.98      0.98       134

    accuracy                           0.97       500
   macro avg       0.97      0.97      0.97       500
weighted avg       0.97      0.97      0.97       500
```

Accuracy with SVM(Linear Kernel): 0.97

PROJECT REPORT

Accuracy of Logistic regresison model : 0.60

Accuracy using KNN Classification Model : 0.92

Accuracy with SVM(RBF Kernel): 0.25

Accuracy with SVM(Linear Kernel): 0.97

Result : The best accuracy is shown by the SVM using Linear Kernel Algorithm for the given problem Statement.