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
Data:

https://drive.google.com/file/d/1ysjQubn4Rb9_dNdd_CurTQKB3LTYnUK6/view?usp=sharing
Details of features:
The columns are described as follows:

Dataset as 21 features and 2000 entries. The meanings of the features are given below:
• battery_power: Total energy a battery can store in one time measured in mAh

.

blue: Has bluetooth or not

• clock_speed: speed at which microprocessor executes instructions

• dual_sim: Has dual sim support or not

• fc: Front Camera mega pixels

• four_g: Has 4G or not

• int_memory: Internal Memory in Gigabytes

• m_dep: Mobile Depth in cm
• mobile_wt: Weight of mobile phone

• n_cores: Number of cores of processor

• pc: Primary Camera mega pixels
• px_height: Pixel Resolution Height
• px_width: Pixel Resolution Width
• ram: Random Access Memory in Mega Bytes
• sc_h: Screen Height of mobile in cm

sc_w: Screen Width of mobile in cm
• talk_time: longest time that a single battery charge will last when you are

• three_g: Has 3G or not

• touch_screen: Has touch screen or not

• wifi: Has wifi or not

• price_range: This is the target variable with value of 0(low cost), 1(medium cost),

2(high cost) and 3(very high cost).

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

In [14]:
```python
#logistic regression
df=pd.read_csv( r"C:/Users/hith6/OneDrive/Desktop/minor/mobile_price_range_data.csv"  )
 df.head()
```

Out[14]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	...	px_height	px_width	ram	sc_h	sc_w	talk_time	three_g	touch_screen	wifi	price_range
0	842	0	2.2	0	1	0	7	0.6	188	2	...	20	756	2549	9	7	19	0	0	1	1
1	1021	1	0.5	1	0	1	53	0.7	136	3	...	905	1988	2631	17	3	7	1	1	0	2
2	563	1	0.5	1	2	1	41	0.9	145	5	...	1263	1716	2603	11	2	9	1	1	0	2
3	615	1	2.5	0	0	0	10	0.8	131	6	...	1216	1786	2769	16	8	11	1	0	0	2
4	1821	1	1.2	0	13	1	44	0.6	141	2	...	1208	1212	1411	8	2	15	1	1	0	1

5 rows × 21 columns

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

In [16]:
```python
df.shape
```

Out[16]:
```
(2000, 21)
```

In [17]:
```python
df.dtypes
```

Out[17]:
```
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 [18]:
```python
x=df.iloc[:,:-1]
y=df.iloc[:,-1]
print(x.shape)
print(y.shape)
```

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

In [20]:
```python
from sklearn.model_selection import train_test_split
```

In [21]:
```python
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,)
```

In [22]:
```python
from sklearn.linear_model import LogisticRegression
```

In [24]:
```python
reg=LogisticRegression()
reg.fit(x_tr,y_tr)
```

```
C:\Users\hith6\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\linear_model\_logistic.py:444: 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
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[24]:
```
▾ LogisticRegression
LogisticRegression()
```

In [25]:
```python
print("train sc",reg.score(x_tr,y_tr))
print("test sc",reg.score(x_te,y_te))
```

```
train sc 0.634
test sc 0.626
```

In [26]:
```python
ypred=reg.predict(x_te)
print(ypred)
```

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

In [36]:
```python
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  \
1133           1528     1          1.8         1   3       1          55
919            1531     0          2.0         0   0       1          60
1311            643     1          0.7         1   3       1          23
1965            659     0          1.4         1   2       1           3
1816            584     1          1.7         1   1       1          19
...             ...   ...          ...       ...  ..     ...         ...
1615           1086     0          2.7         0   0       1          24
538             704     0          2.2         1   4       0          53
1450           1249     1          0.5         0   3       1          55
398            1323     1          2.5         1  10       1          28
711            1744     1          1.0         0   1       1          20

      m_dep  mobile_wt  n_cores  ...  px_width   ram  sc_h  sc_w  talk_time  \
1133    0.8        129        5  ...      1762  1282    11     9         11
919     0.4        162        6  ...       765  2524    13    11          7
1311    0.8        156        1  ...      1400  2096    13     6          2
1965    0.1         85        8  ...       657   967    10     1         14
1816    0.4        199        4  ...      1343  3996    12     9         17
...     ...        ...      ...  ...       ...   ...   ...   ...        ...
1615    0.1        135        8  ...      1976  3930    17     1         18
538     0.2        186        6  ...      1013  3653     7     1         15
1450    0.8        140        3  ...      1193   531    18    10         13
398     0.2        131        4  ...       619  1892    10     0         15
711     0.5        112        8  ...       718  1205    18     7         14

      three_g  touch_screen  wifi  Actual_Values  Predicted_values
1133        1             1     1              1                 1
919         1             0     0              2                 2
1311        1             0     0              1                 1
1965        1             0     1              0                 1
1816        1             1     1              3                 2
...       ...           ...   ...            ...               ...
1615        1             1     1              3                 3
538         1             1     1              2                 2
1450        1             1     1              0                 0
398         1             1     1              1                 2
711         1             0     0              1                 1

[500 rows x 22 columns]
```

In [37]:
```python
 from sklearn.metrics import confusion_matrix,classification_report
```

In [38]:
```python
cm=confusion_matrix(y_te,ypred)
print(cm)
```

```
[[95 30  3  0]
 [17 67 35  7]
 [ 0 19 57 40]
 [ 0  1 35 94]]
```

In [41]:
```python
print(classification_report(y_te,ypred))
```

```
              precision    recall  f1-score   support

           0       0.85      0.74      0.79       128
           1       0.57      0.53      0.55       126
           2       0.44      0.49      0.46       116
           3       0.67      0.72      0.69       130

    accuracy                           0.63       500
   macro avg       0.63      0.62      0.63       500
weighted avg       0.64      0.63      0.63       500
```

In [ ]:
```python
#ACCURACY OF LOGISTIC REGRESSION MODEL IS 0.63
```

ACCURACY OF LOGISTIC REGRESSION IS 0.63

In [ ]:
```python
#knn classification
```

KNN CLASSIFICATION

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

In [49]:
```python
m1=KNeighborsClassifier(n_neighbors=19)
m1.fit(x_tr,y_tr)
```

Out[49]:
```
▾ KNeighborsClassifier
KNeighborsClassifier(n_neighbors=19)
```

In [55]:
```python
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,)
```

In [56]:
```python
print('train sc',m1.score(x_tr,y_tr))
print('testin sc',m1.score(x_te,y_te))
```

```
train sc 0.9413333333333334
testin sc 0.944
```

In [58]:
```python
ypred1=m1.predict(x_te)
print(ypred1)
```

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

In [59]:
```python
#confusion matrix
cm=confusion_matrix(y_te,ypred1)
print(cm)
```

```
[[121   0   0   0]
 [  4 121   5   0]
 [  0   6 115   5]
 [  0   0   8 115]]
```

In [60]:
```python
print(classification_report(y_te,ypred1))
```

```
              precision    recall  f1-score   support

           0       0.97      1.00      0.98       121
           1       0.95      0.93      0.94       130
           2       0.90      0.91      0.91       126
           3       0.96      0.93      0.95       123

    accuracy                           0.94       500
   macro avg       0.94      0.94      0.94       500
weighted avg       0.94      0.94      0.94       500
```

In [ ]:
```python
#ACCURACY WITH KNN is 0.94
```

ACCURACY OF KNN IS 0.94

In [ ]:
```python
 #SVM classifier
```

SVM CLASSIFIER

In [61]:
```python
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,)
```

In [63]:
```python
from sklearn.svm import SVC
```

In [66]:
```python
s=SVC(kernel='rbf',C=1,gamma=0.001)
s.fit(x_tr,y_tr)
```

Out[66]:
```
▾ SVC
SVC(C=1, gamma=0.001)
```

In [67]:
```python
print('train sc',m1.score(x_tr,y_tr))
print('testin sc',m1.score(x_te,y_te))
```

```
train sc 0.9426666666666667
testin sc 0.94
```

In [68]:
```python
ypred=s.predict(x_te)
print(ypred)
```

```
[3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3]
```

In [69]:
```python
#confusion matrix
cm=confusion_matrix(y_te,ypred)
print(cm)
```

```
[[  0   0   0 115]
 [  0   0   0 123]
 [  0   0   0 155]
 [  0   0   0 107]]
```

In [70]:
```python
print(classification_report(y_te,ypred))
```

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       115
           1       0.00      0.00      0.00       123
           2       0.00      0.00      0.00       155
           3       0.21      1.00      0.35       107

    accuracy                           0.21       500
   macro avg       0.05      0.25      0.09       500
weighted avg       0.05      0.21      0.08       500
```

```
C:\Users\hith6\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\metrics\_classification.py:1334: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\hith6\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\metrics\_classification.py:1334: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\hith6\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\metrics\_classification.py:1334: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [ ]:
```python
#ACCURACY WITH SVM (RBF KERNEL):0.21
```

ACCURACY SVM 0.21

In [ ]:
```python
#SVM MODEL USING LINEAR KERNEL
```

LINEAR KERNEL SVM

In [7]:
```python
from sklearn.model_selection import  train_test_split
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import seaborn as sns
```

In [9]:
```python
df=pd.read_csv( r"C:/Users/hith6/OneDrive/Desktop/minor/mobile_price_range_data.csv"  )
df.head()
```

Out[9]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	...	px_height	px_width	ram	sc_h	sc_w	talk_time	three_g	touch_screen	wifi	price_range
0	842	0	2.2	0	1	0	7	0.6	188	2	...	20	756	2549	9	7	19	0	0	1	1
1	1021	1	0.5	1	0	1	53	0.7	136	3	...	905	1988	2631	17	3	7	1	1	0	2
2	563	1	0.5	1	2	1	41	0.9	145	5	...	1263	1716	2603	11	2	9	1	1	0	2
3	615	1	2.5	0	0	0	10	0.8	131	6	...	1216	1786	2769	16	8	11	1	0	0	2
4	1821	1	1.2	0	13	1	44	0.6	141	2	...	1208	1212	1411	8	2	15	1	1	0	1

5 rows × 21 columns

In [10]:
```python
x=df.iloc[:,:-1]
y=df.iloc[:,-1]
print(x.shape)
print(y.shape)
```

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

In [11]:
```python
x.head()
```

Out[11]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	pc	px_height	px_width	ram	sc_h	sc_w	talk_time	three_g	touch_screen	wifi
0	842	0	2.2	0	1	0	7	0.6	188	2	2	20	756	2549	9	7	19	0	0	1
1	1021	1	0.5	1	0	1	53	0.7	136	3	6	905	1988	2631	17	3	7	1	1	0
2	563	1	0.5	1	2	1	41	0.9	145	5	6	1263	1716	2603	11	2	9	1	1	0
3	615	1	2.5	0	0	0	10	0.8	131	6	9	1216	1786	2769	16	8	11	1	0	0
4	1821	1	1.2	0	13	1	44	0.6	141	2	14	1208	1212	1411	8	2	15	1	1	0

In [12]:
```python
y.head()
```

Out[12]:
```
0    1
1    2
2    2
3    2
4    1
Name: price_range, dtype: int64
```

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

In [ ]:

In [22]:
```python
x_train,x_test,y_train,y_test,=train_test_split(x,y,test_size=0.25)
print(x_test.shape)
print(x_train.shape)
print(y_test.shape)
print(y_train.shape)
```

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

In [23]:
```python
s1=SVC(kernel='linear',C=1)
s1.fit(x_train,y_train)
```

Out[23]:
```
▾ SVC
SVC(C=1, kernel='linear')
```

In [24]:
```python
SVC(C=1,kernel='linear')
```

Out[24]:
```
▾ SVC
SVC(C=1, kernel='linear')
```

In [25]:
```python
print('train sc',s1.score(x_train,y_train))
print('testin sc',s1.score(x_test,y_test))
```

```
train sc 0.992
testin sc 0.976
```

In [26]:
```python
ypred_1=s1.predict(x_test)
print(ypred_1)
```

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

In [27]:
```python
#confusion matrix
cm1=confusion_matrix(y_test,ypred_1)
print(cm1)
```

```
[[120   2   0   0]
 [  0 131   3   0]
 [  0   3 114   3]
 [  0   0   1 123]]
```

In [30]:
```python
print(classification_report(y_test,ypred_1))
```

```
              precision    recall  f1-score   support

           0       1.00      0.98      0.99       122
           1       0.96      0.98      0.97       134
           2       0.97      0.95      0.96       120
           3       0.98      0.99      0.98       124

    accuracy                           0.98       500
   macro avg       0.98      0.98      0.98       500
weighted avg       0.98      0.98      0.98       500
```

In [ ]:
```python
#ACCUARCY FOR SVM USING  linearKERNEL 0.98
```

ACCURACY IS 0.98

In [ ]:

PROJECT REPORT

ACCURACY LOGISTIC REGRESSION 0.63

ACCURACY KNN 0.94

ACCURACY WITH SVM 0.21

ACCURACY WITH SVM (LINEAR) 0.98

RESULT : THE best accuracy shown by svm using linear algorithm problem