AIML ASSIGNMENT-2

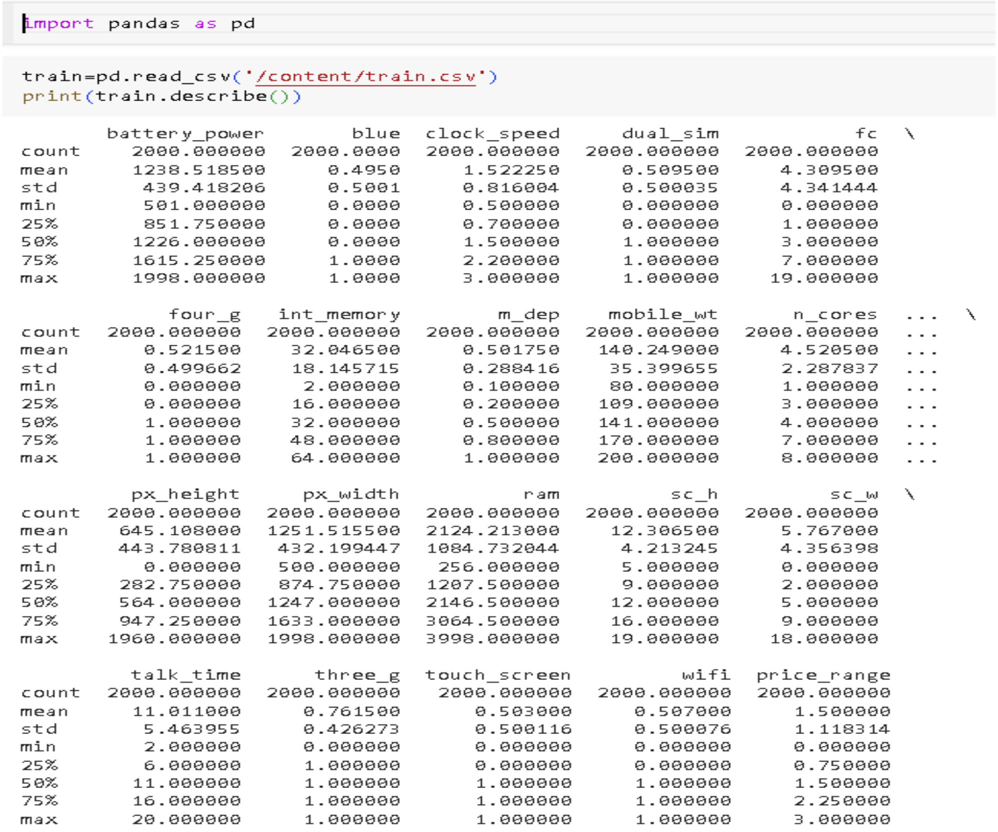
NAME: K .PAVANI

HT.NO: 2203A52101

BATCH: 12

Question 1:
https://www.kaggle.com/datasets/iabhishekofficial/mobile-price-classification?select=train.csv
From the above data:
a) Read the data with pandas and find features and target variables
b) Normalize the data with min-max scaling
c) Split the data into train and test.

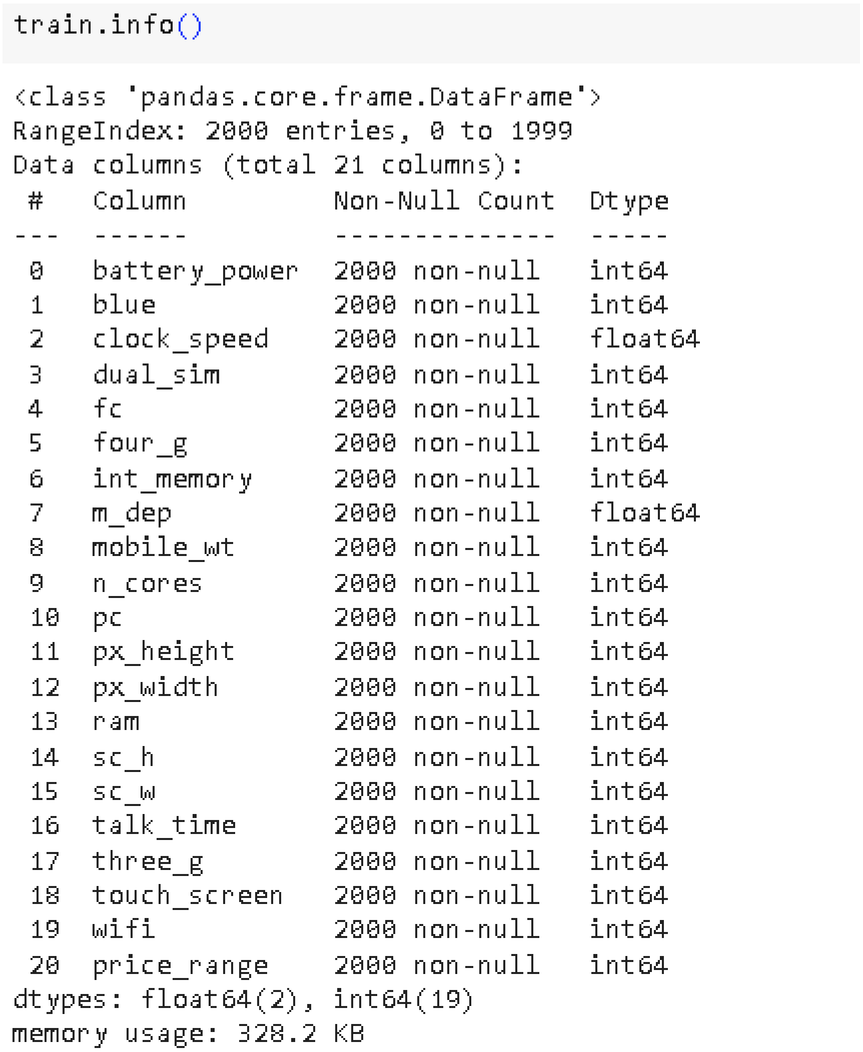
a)reading with pandas and finding feature and target variables



```
import pandas as pd
```

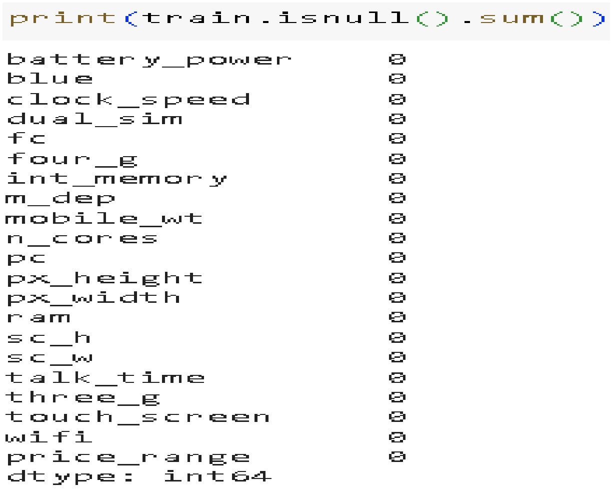
```
train=pd.read_csv('/content/train.csv')
print(train.describe())
```

```
       battery_power       blue  clock_speed     dual_sim           fc  \
count    2000.000000  2000.0000  2000.000000  2000.000000  2000.000000
mean     1238.518500     0.4950     1.522250     0.509500     4.309500
std       439.418206     0.5001     0.816004     0.500035     4.341444
min       501.000000     0.0000     0.500000     0.000000     0.000000
25%       851.750000     0.0000     0.700000     0.000000     1.000000
50%      1226.000000     0.0000     1.500000     1.000000     3.000000
75%      1615.250000     1.0000     2.200000     1.000000     7.000000
max      1998.000000     1.0000     3.000000     1.000000    19.000000

            four_g   int_memory        m_dep    mobile_wt      n_cores  ...  \
count  2000.000000  2000.000000  2000.000000  2000.000000  2000.000000  ...
mean      0.521500    32.046500     0.501750   140.249000     4.520500  ...
std       0.499662    18.145715     0.288416    35.399655     2.287837  ...
min       0.000000     2.000000     0.100000    80.000000     1.000000  ...
25%       0.000000    16.000000     0.200000   109.000000     3.000000  ...
50%       1.000000    32.000000     0.500000   141.000000     4.000000  ...
75%       1.000000    48.000000     0.800000   170.000000     7.000000  ...
max       1.000000    64.000000     1.000000   200.000000     8.000000  ...

         px_height     px_width          ram         sc_h         sc_w  \
count  2000.000000  2000.000000  2000.000000  2000.000000  2000.000000
mean    645.108000  1251.515500  2124.213000    12.306500     5.767000
std     443.780811   432.199447  1084.732044     4.213245     4.356398
min       0.000000   500.000000   256.000000     5.000000     0.000000
25%     282.750000   874.750000  1207.500000     9.000000     2.000000
50%     564.000000  1247.000000  2146.500000    12.000000     5.000000
75%     947.250000  1633.000000  3064.500000    16.000000     9.000000
max    1960.000000  1998.000000  3998.000000    19.000000    18.000000

         talk_time      three_g  touch_screen         wifi  price_range
count  2000.000000  2000.000000   2000.000000  2000.000000  2000.000000
mean     11.011000     0.761500      0.503000     0.507000     1.500000
std       5.463955     0.426273      0.500116     0.500076     1.118314
min       2.000000     0.000000      0.000000     0.000000     0.000000
25%       6.000000     1.000000      0.000000     0.000000     0.750000
50%      11.000000     1.000000      1.000000     1.000000     1.500000
75%      16.000000     1.000000      1.000000     1.000000     2.250000
max      20.000000     1.000000      1.000000     1.000000     3.000000
```



```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   battery_power  2000 non-null   int64  
 1   blue           2000 non-null   int64  
 2   clock_speed    2000 non-null   float64
 3   dual_sim       2000 non-null   int64  
 4   fc             2000 non-null   int64  
 5   four_g         2000 non-null   int64  
 6   int_memory     2000 non-null   int64  
 7   m_dep          2000 non-null   float64
 8   mobile_wt      2000 non-null   int64  
 9   n_cores        2000 non-null   int64  
 10  pc             2000 non-null   int64  
 11  px_height      2000 non-null   int64  
 12  px_width       2000 non-null   int64  
 13  ram            2000 non-null   int64  
 14  sc_h           2000 non-null   int64  
 15  sc_w           2000 non-null   int64  
 16  talk_time      2000 non-null   int64  
 17  three_g        2000 non-null   int64  
 18  touch_screen   2000 non-null   int64  
 19  wifi           2000 non-null   int64  
 20  price_range    2000 non-null   int64  
dtypes: float64(2), int64(19)
memory usage: 328.2 KB
```



```
print(train.isnull().sum())
```

```
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
```

b)normalizing data

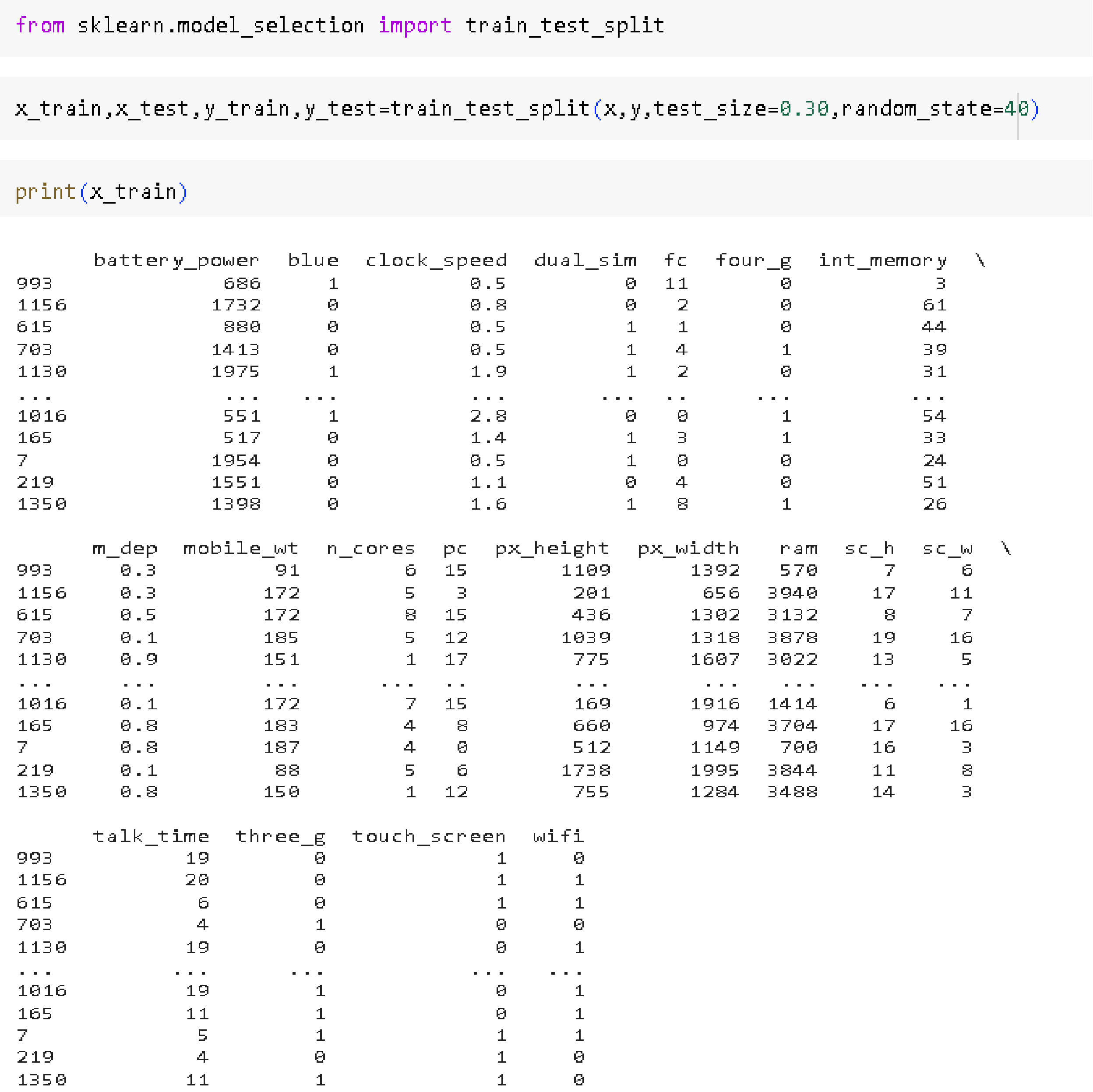
```
from sklearn.preprocessing import MinMaxScaler
```

```
d=MinMaxScaler()
```

```
y=train['price_range']
x=train.drop('price_range',axis=1)
print(x)
print(y)
```

```
      battery_power  blue  clock_speed  dual_sim  fc  four_g  int_memory  \
0               842     0          2.2         0   1       0           7   
1              1021     1          0.5         1   0       1          53   
2               563     1          0.5         1   2       1          41   
3               615     1          2.5         0   0       0          10   
4              1821     1          1.2         0  13       1          44   
...             ...   ...          ...       ...  ..     ...         ...   
1995            794     1          0.5         1   0       1           2   
1996           1965     1          2.6         1   0       0          39   
1997           1911     0          0.9         1   1       1          36   
1998           1512     0          0.9         0   4       1          46   
1999            510     1          2.0         1   5       1          45   

      m_dep  mobile_wt  n_cores  pc  px_height  px_width   ram  sc_h  sc_w  \
0       0.6        188        2   2         20       756  2549     9     7   
1       0.7        136        3   6        905      1988  2631    17     3   
2       0.9        145        5   6       1263      1716  2603    11     2   
3       0.8        131        6   9       1216      1786  2769    16     8   
4       0.6        141        2  14       1208      1212  1411     8     2   
...     ...        ...      ...  ..        ...       ...   ...   ...   ...   
1995    0.8        106        6  14       1222      1890   668    13     4   
1996    0.2        187        4   3        915      1965  2032    11    10   
1997    0.7        108        8   3        868      1632  3057     9     1   
1998    0.1        145        5   5        336       670   869    18    10   
1999    0.9        168        6  16        483       754  3919    19     4   

      talk_time  three_g  touch_screen  wifi  
0            19        0             0     1  
1             7        1             1     0  
2             9        1             1     0  
3            11        1             0     0  
4            15        1             1     0  
...         ...      ...           ...   ...  
1995         19        1             1     0  
1996         16        1             1     1  
1997          5        1             1     0  
```

c)splitting data into train and test



```
from sklearn.model_selection import train_test_split
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=40)
```

```
print(x_train)
```

```
      battery_power  blue  clock_speed  dual_sim  fc  four_g  int_memory  \
993             686     1          0.5         0  11       0           3   
1156           1732     0          0.8         0   2       0          61   
615             880     0          0.5         1   1       0          44   
703            1413     0          0.5         1   4       1          39   
1130           1975     1          1.9         1   2       0          31   
...             ...   ...          ...       ...  ..     ...         ...   
1016            551     1          2.8         0   0       1          54   
165             517     0          1.4         1   3       1          33   
7              1954     0          0.5         1   0       0          24   
219            1551     0          1.1         0   4       0          51   
1350           1398     0          1.6         1   8       1          26   

      m_dep  mobile_wt  n_cores  pc  px_height  px_width   ram  sc_h  sc_w  \
993     0.3         91        6  15       1109      1392   570     7     6   
1156    0.3        172        5   3        201       656  3940    17    11   
615     0.5        172        8  15        436      1302  3132     8     7   
703     0.1        185        5  12       1039      1318  3878    19    16   
1130    0.9        151        1  17        775      1607  3022    13     5   
...     ...        ...      ...  ..        ...       ...   ...   ...   ...   
1016    0.1        172        7  15        169      1916  1414     6     1   
165     0.8        183        4   8        660       974  3704    17    16   
7       0.8        187        4   0        512      1149   700    16     3   
219     0.1         88        5   6       1738      1995  3844    11     8   
1350    0.8        150        1  12        755      1284  3488    14     3   

      talk_time  three_g  touch_screen  wifi  
993          19        0             1     0  
1156         20        0             1     1  
615           6        0             1     1  
703           4        1             0     0  
1130         19        0             0     1  
...         ...      ...           ...   ...  
1016         19        1             0     1  
165          11        1             0     1  
7             5        1             1     1  
219           4        0             1     0  
1350         11        1             1     0  
```

```
print(x_test)
```

```
      battery_power  blue  clock_speed  dual_sim  fc  four_g  int_memory  \
423            1681     1          2.5         0   2       0          11   
1495           1472     0          3.0         0   4       1          20   
1618            502     0          0.8         0   7       0          52   
1099           1697     0          0.5         0   0       1          60   
1307            831     0          1.7         1   7       1          26   
...             ...   ...          ...       ...  ..     ...         ...   
14             1866     0          0.5         0  13       1          52   
282            1839     1          1.2         0   9       1          54   
952            1444     1          2.1         0   9       0          38   
1079           1893     1          0.5         1   1       0          23   
486            1089     1          0.9         1  12       1           2   

      m_dep  mobile_wt  n_cores  pc  px_height  px_width   ram  sc_h  sc_w  \
423     0.4        158        2  13        195      1205  1122    12     6   
1495    0.3        169        2   6        443       892   797     6     1   
1618    1.0         82        6   8        281      1159  2666     5     4   
1099    0.1         90        4   0         88      1046   441    15     1   
1307    0.7        177        5  11        511       621  1704     6     5   
...     ...        ...      ...  ..        ...       ...   ...   ...   ...   
14      0.7        185        1  17        356       563   373    14     9   
282     0.5        200        7  11        475      1493   927    19    10   
952     0.4        104        7  16        624       917  3764    14     9   
1079    0.1        179        8   3       1203      1432  1482    15     7   
486     0.7        145        5  15        636      1259  2765    13    12   

      talk_time  three_g  touch_screen  wifi  
423          16        0             1     1  
1495         11        1             1     0  
1618         20        1             1     0  
1099         11        1             1     0  
1307         20        1             1     1  
...         ...      ...           ...   ...  
14            3        1             0     1  
282          18        1             0     1  
952          10        0             0     0  
1079         17        0             1     0  
486          10        1             0     1  

[600 rows x 20 columns]
```

```
print(y_test)
```

```
423     0
1495    0
1618    2
1099    0
1307    0
       ..
14      0
282     1
952     3
1079    2
486     2
Name: price_range, Length: 600, dtype: int64
```

```
print(y_train)
```

```
993     0
1156    3
615     2
703     3
1130    3
       ..
1016    0
165     3
7       0
219     3
1350    3
Name: price_range, Length: 1400, dtype: int64
```

Question 2:

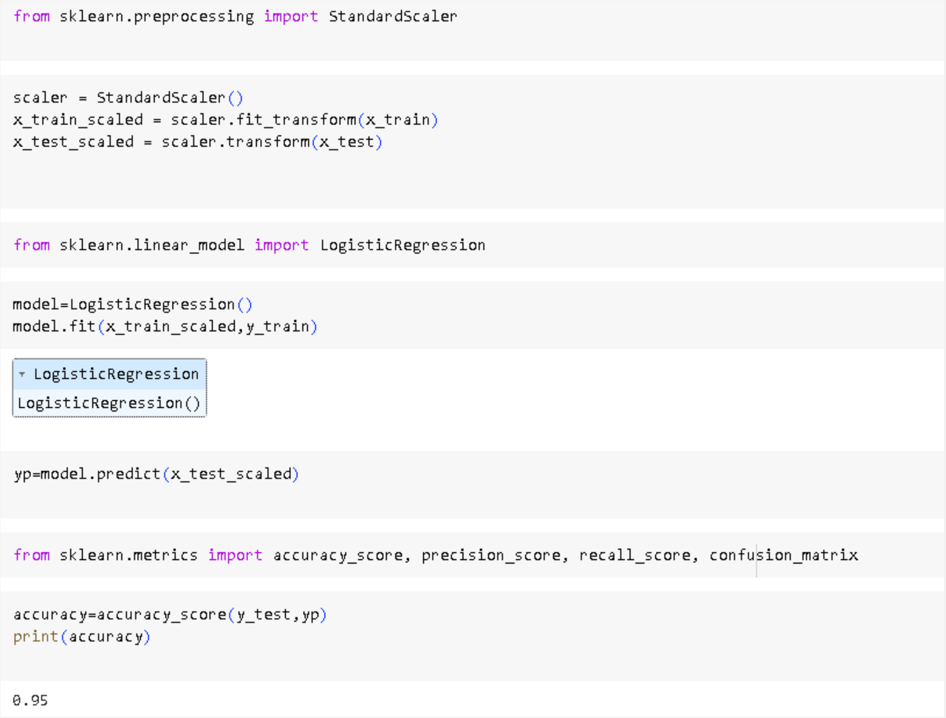
Bob has started his own mobile company. He wants to give tough fight to big companies like Apple,Samsung etc.

He does not know how to estimate price of mobiles his company creates. In this competitive mobile phone market you cannot simply assume things. To solve this problem he collects sales data of mobile phones of various companies.

Bob wants to find out some relation between features of a mobile phone (eg:- 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. And provide the results like accuracy, precision recall and confusion matrix.

https://www.kaggle.com/datasets/iabhishekofficial/mobile-price-classification?select=train.csv

Hint: apply logistic regression



```
from sklearn.preprocessing import StandardScaler
```

```
scaler = StandardScaler()
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test)
```

```
from sklearn.linear_model import LogisticRegression
```

```
model=LogisticRegression()
model.fit(x_train_scaled,y_train)
```

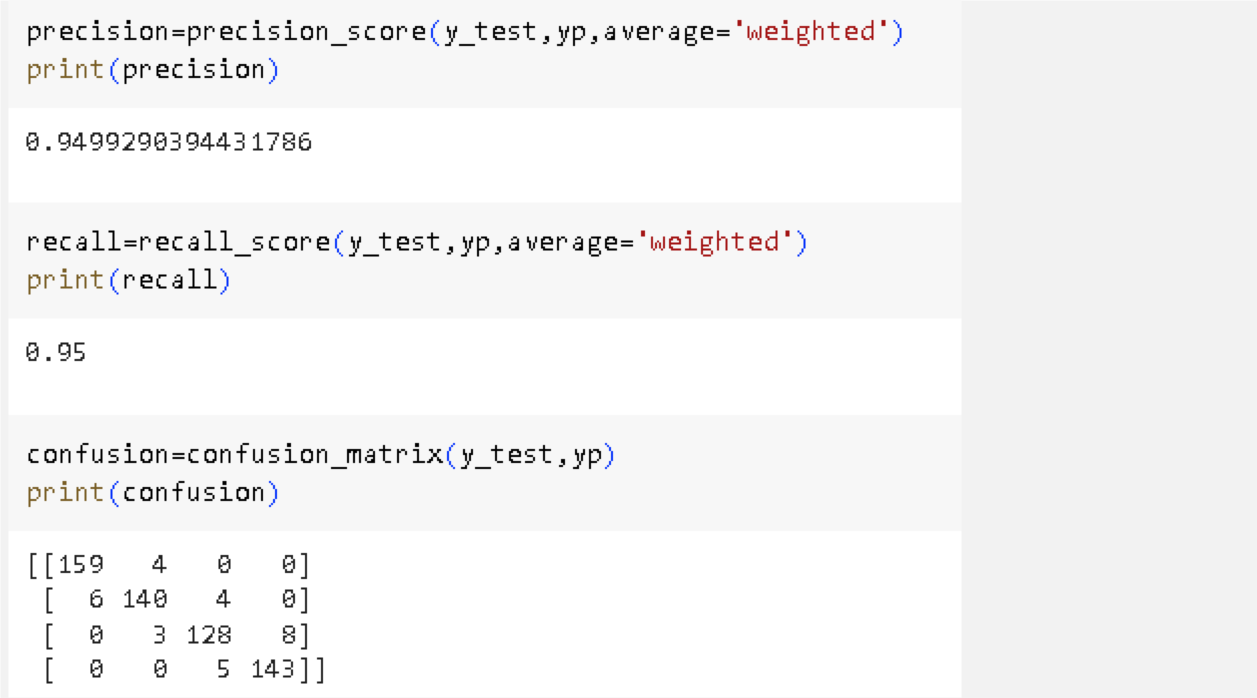
```
▾ LogisticRegression
LogisticRegression()
```

```
yp=model.predict(x_test_scaled)
```

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, confusion_matrix
```

```
accuracy=accuracy_score(y_test,yp)
print(accuracy)
```

```
0.95
```



```
precision=precision_score(y_test,yp,average='weighted')
print(precision)
```

```
0.9499290394431786
```

```
recall=recall_score(y_test,yp,average='weighted')
print(recall)
```

```
0.95
```

```
confusion=confusion_matrix(y_test,yp)
print(confusion)
```

```
[[159   4   0   0]
 [  6 140   4   0]
 [  0   3 128   8]
 [  0   0   5 143]]
```