In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

import warnings
warnings.filterwarnings('ignore')


from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression

from sklearn.metrics import classification_report,confusion_matrix
```

In [2]:
```
data=pd.read_csv('/Users/tanishq/Downloads/archive-2/Datas.csv')
```

Dataset

In [3]:
```
data.head(5)
```

Out[3]:

	battery	color	clock	sim	FC	four_g	memory	m_dep	weight	cores	...	height	widt
0	842	0	2.2	0	1	0	7	0.6	188	2	...	20	75
1	1021	1	0.5	1	0	1	53	0.7	136	3	...	905	198
2	563	1	0.5	1	2	1	41	0.9	145	5	...	1263	171
3	615	1	2.5	0	0	0	10	0.8	131	6	...	1216	178
4	1821	1	1.2	0	13	1	44	0.6	141	2	...	1208	121

5 rows × 21 columns

In [4]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 21 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   battery       2000 non-null   int64  
 1   color         2000 non-null   int64  
 2   clock         2000 non-null   float64
 3   sim           2000 non-null   int64  
 4   FC            2000 non-null   int64  
 5   four_g        2000 non-null   int64  
 6   memory        2000 non-null   int64  
 7   m_dep         2000 non-null   float64
 8   weight        2000 non-null   int64  
 9   cores         2000 non-null   int64  
 10  pc            2000 non-null   int64  
 11  height        2000 non-null   int64  
 12  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  3G            2000 non-null   int64  
 18  touch_screen  2000 non-null   int64  
 19  WIFI          2000 non-null   int64  
 20  Price         2000 non-null   int64  
dtypes: float64(2), int64(19)
memory usage: 328.2 KB
```

In [5]:
```
data.describe()
```

Out[5]:

	battery	color	clock	sim	FC	four_g	m
count	2000.000000	2000.0000	2000.000000	2000.000000	2000.000000	2000.000000	2000.0
mean	1238.518500	0.4950	1.522250	0.509500	4.309500	0.521500	32.0
std	439.418206	0.5001	0.816004	0.500035	4.341444	0.499662	18.1
min	501.000000	0.0000	0.500000	0.000000	0.000000	0.000000	2.0
25%	851.750000	0.0000	0.700000	0.000000	1.000000	0.000000	16.0
50%	1226.000000	0.0000	1.500000	1.000000	3.000000	1.000000	32.0
75%	1615.250000	1.0000	2.200000	1.000000	7.000000	1.000000	48.0
max	1998.000000	1.0000	3.000000	1.000000	19.000000	1.000000	64.0

8 rows × 21 columns

Preprocessing

In [6]:
```
data = data.drop_duplicates()
data.shape
```

Out[6]: (2000, 21)

In [7]:
```
data = data.dropna()
data.shape
```

Out[7]: (2000, 21)

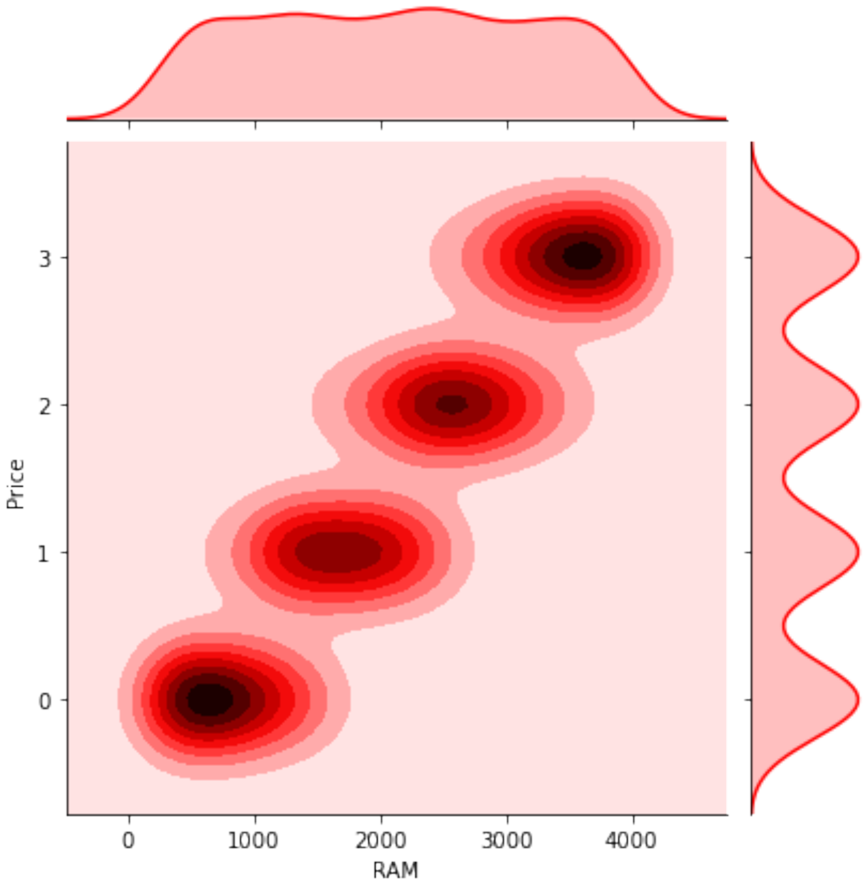
In [8]:
```
data.nunique()
data.shape
```

Out[8]: (2000, 21)

Data Visualization

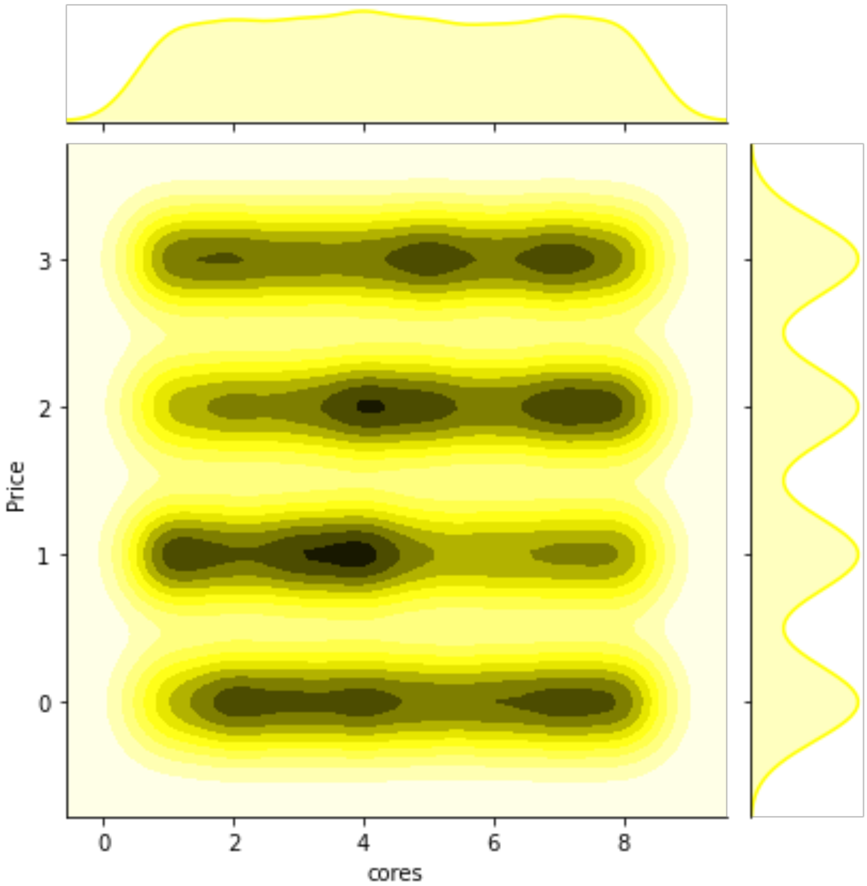
Price vs RAM

In [9]:
```
sns.jointplot(x='RAM',y='Price',data=data,color='red',kind='kde');
```



Price vs CPU cores

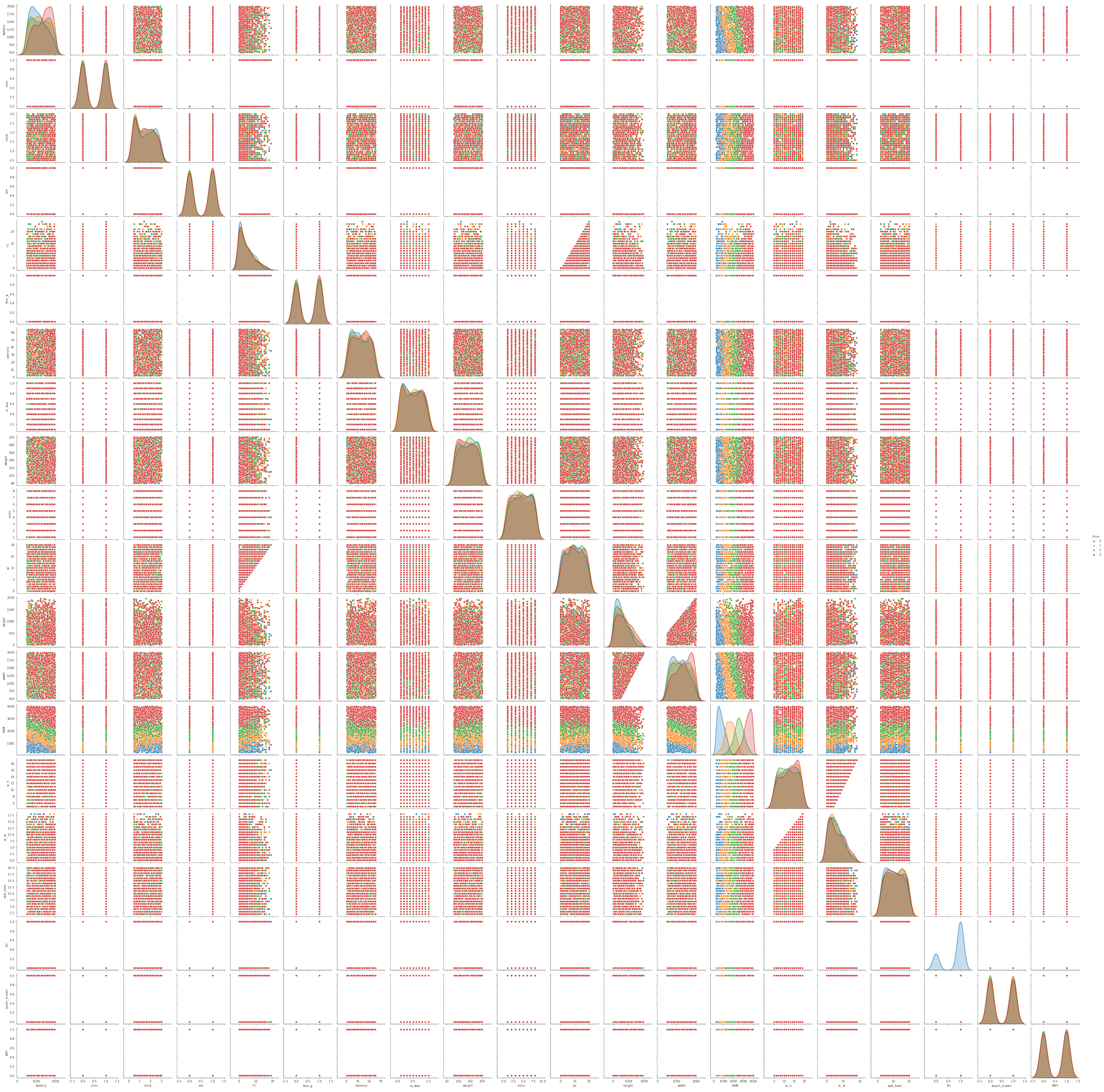
In [10]:
```
sns.jointplot(x='cores',y='Price',data=data,color='yellow',kind='kd
```



Price in Pairplot

In [11]:
```
sns.pairplot(data,hue='Price')
```

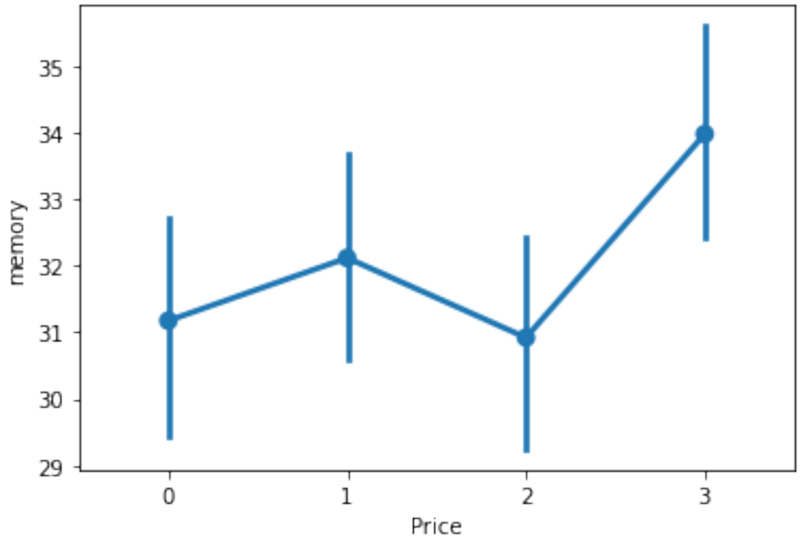
Out[11]: <seaborn.axisgrid.PairGrid at 0x7fdda6425dc0>



Memory vs RAM

In [12]:
```
sns.pointplot(y="memory", x="Price", data=data)
```

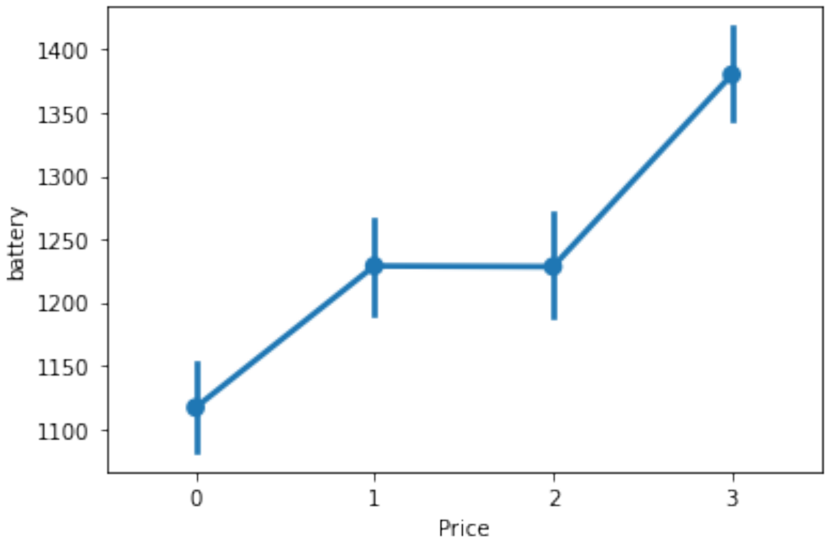
Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x7fdd95944610>



Price vs battery

In [13]:
```
sns.pointplot(y="battery", x="Price", data=data)
```

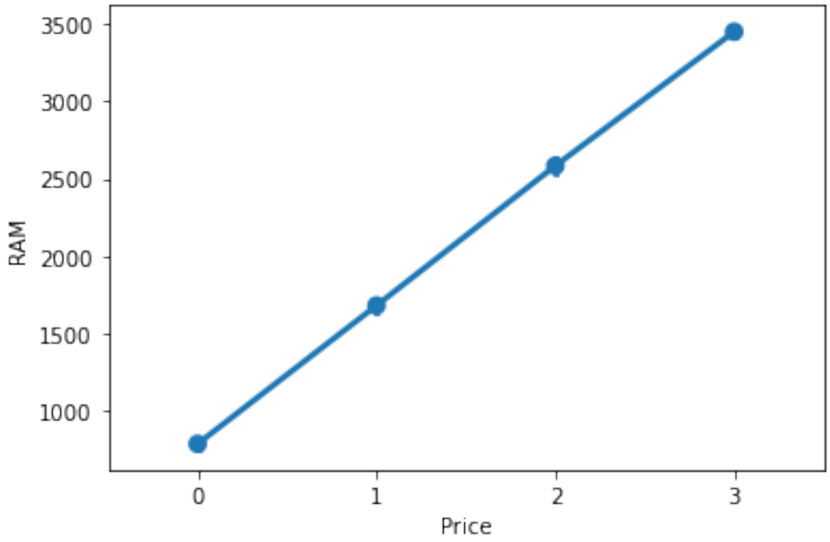
Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x7fdd9546f2b0>



Price vs RAM

In [14]:
```
sns.pointplot(y="RAM", x="Price", data=data)
```

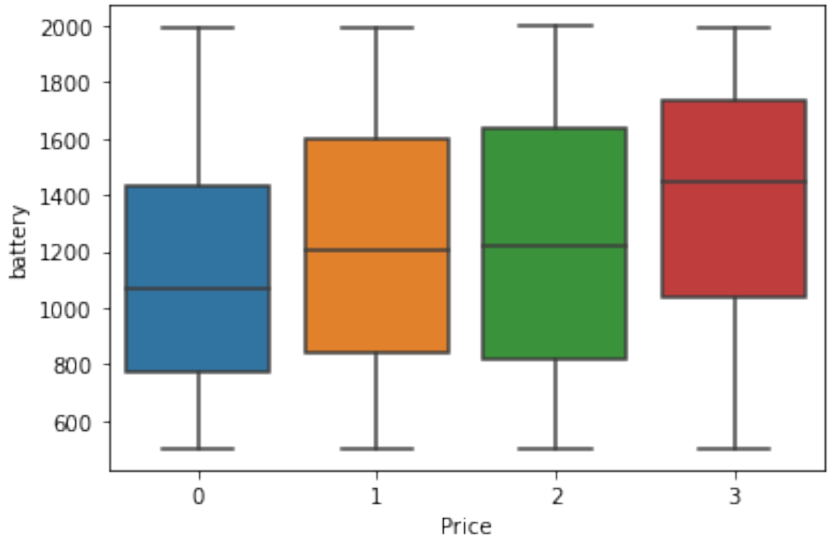
Out[14]: <matplotlib.axes._subplots.AxesSubplot at 0x7fdd95e048b0>



Battery power vs Price Range

In [15]:
```
sns.boxplot(x="Price", y="battery", data=data)
```

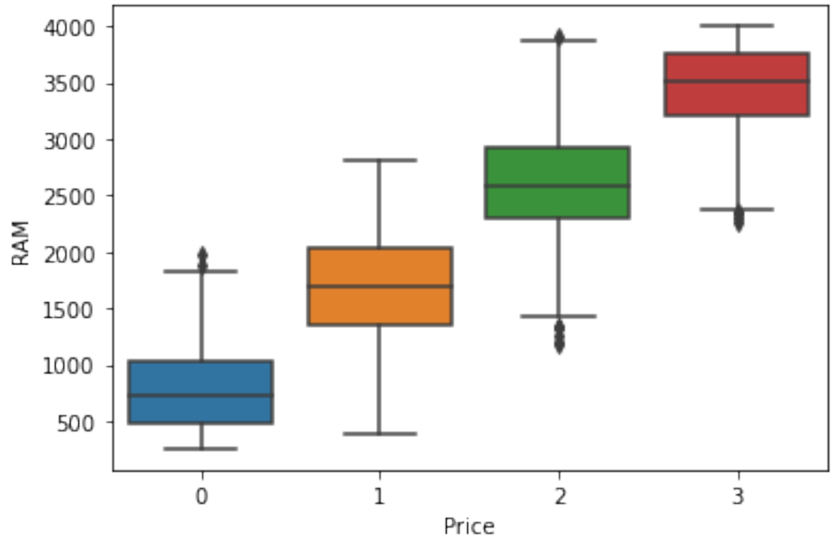
Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x7fdd99cd75e0>



Battery power vs RAM

In [16]:
```
sns.boxplot(x="Price", y="RAM", data=data)
```

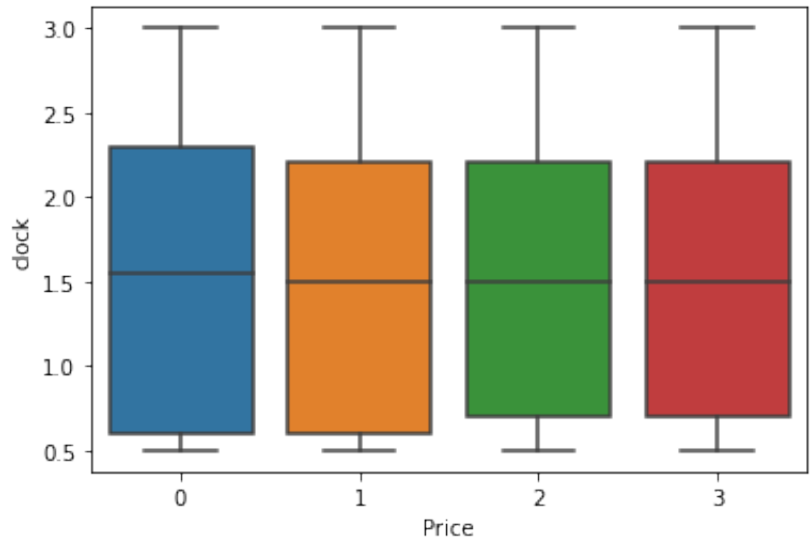
Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x7fdd99bfb610>



Battery power vs RAM

In [17]:
```
sns.boxplot(x="Price", y="clock", data=data)
```

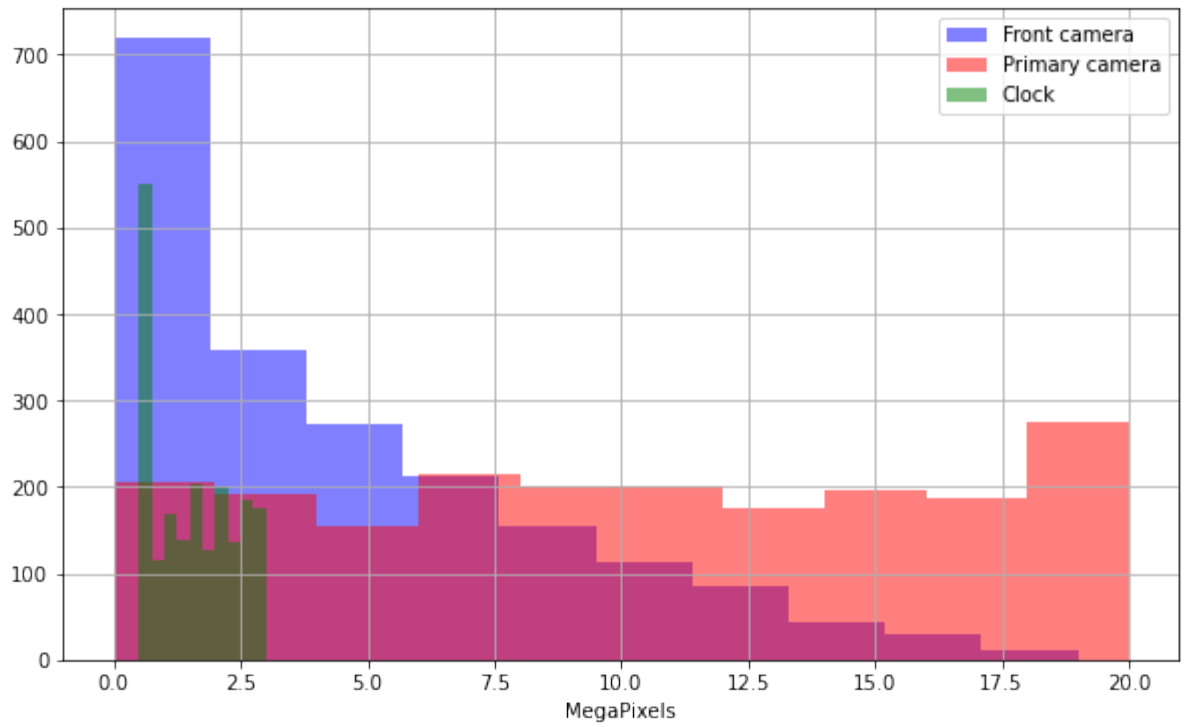
Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0x7fdd99cb7850>



No of Phones vs Camera megapixels of front and primary camera vs Clock speed

In [18]:
```
plt.figure(figsize=(10,6))
data['FC'].hist(alpha=0.5,color='blue',label='Front camera')
data['pc'].hist(alpha=0.5,color='red',label='Primary camera')
data['clock'].hist(alpha=0.5,color='green',label='Clock')

plt.legend()
plt.xlabel('MegaPixels')
```

Out[18]:
```
Text(0.5, 0, 'MegaPixels')
```



Splitting Data

In [19]:
```
a=data[['battery','color','clock','sim','RAM','touch_screen','3G']]
b=data['Price']
```

In [20]:
```
X_train, X_test, y_train, y_test = train_test_split(a, b, test_size
```

Linear Regression Model

In [21]:
```
lrm = LinearRegression()
```

In [22]:
```
lrm.fit(X_train,y_train)
```

Out[22]:
```
LinearRegression()
```

In [23]:
```
lrm.score(X_test,y_test)*100
```

Out[23]:
```
87.11897187595284
```

K Nearlest Neighbour Model

In [24]:
```
knn = KNeighborsClassifier(n_neighbors=10)
knn.fit(X_train,y_train)
```

Out[24]:
```
KNeighborsClassifier(n_neighbors=10)
```

In [25]:
```
knn.score(X_test,y_test)*100
```

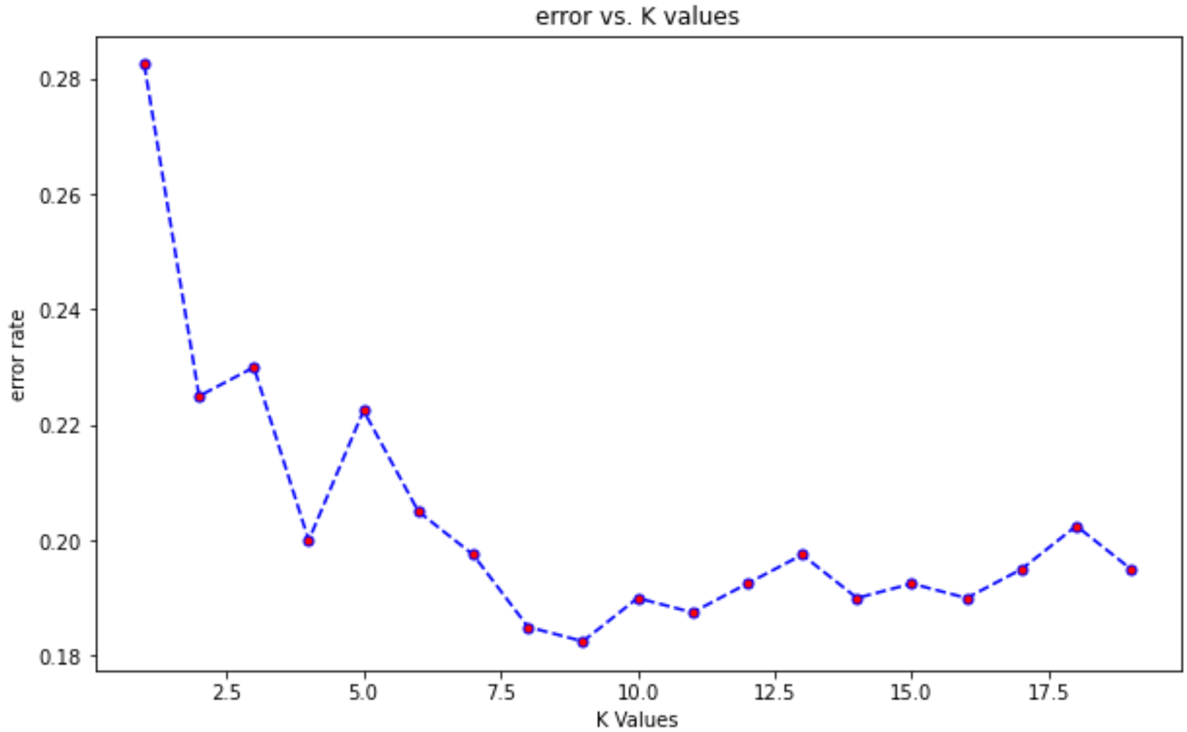
Out[25]:
```
81.0
```

Elbow Method For optimum value of K

In [26]:
```
error_rate = []
for i in range(1,20):

    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train,y_train)
    pred_i = knn.predict(X_test)
    error_rate.append(np.mean(pred_i != y_test))
```

In [27]:
```
plt.figure(figsize=(10,6))
plt.plot(range(1,20),error_rate,color='blue', linestyle='dashed', m
         markerfacecolor='red', markersize=5)
plt.title('error vs. K values')
plt.xlabel('K Values')
plt.ylabel('error rate')
```

Out[27]: Text(0, 0.5, 'error rate')



Logistic Regression Model

In [28]:
```
lr = LogisticRegression()
lr.fit(X_train,y_train)
```

Out[28]: LogisticRegression()

In [29]:
```
lr.score(X_test,y_test)*100
```

Out[29]: 59.75

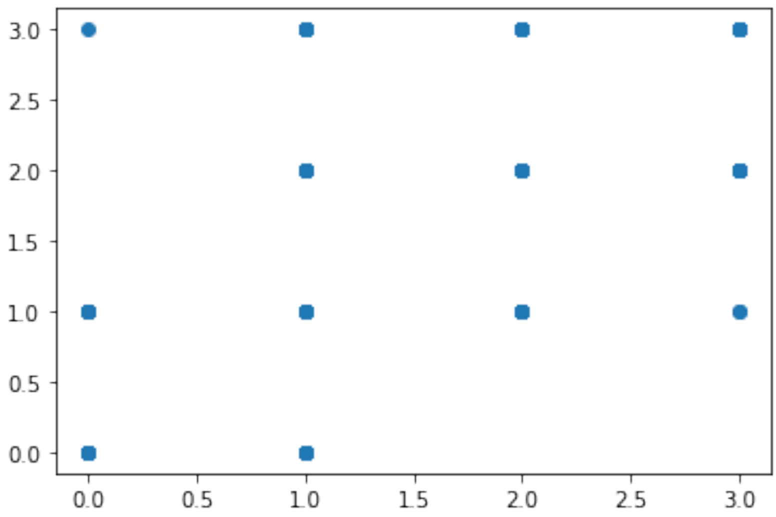
Conclusion

Linear Regression

In [30]:
```
y_pred=lr.predict(X_test)
```

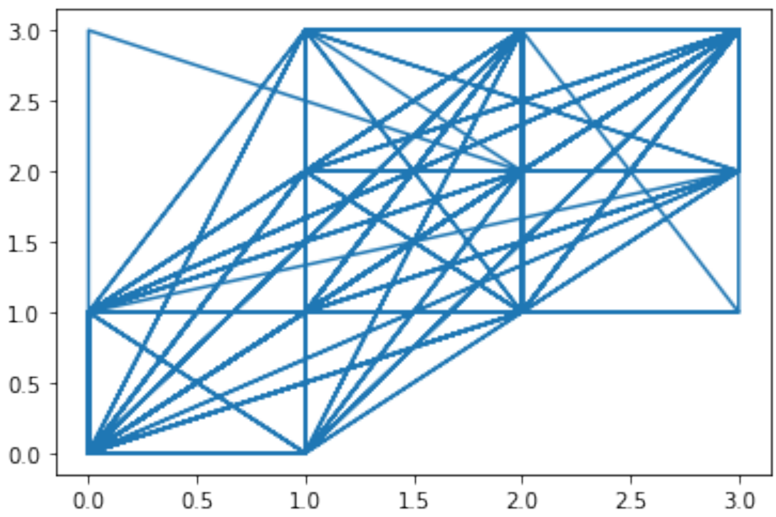
In [31]:
```
plt.scatter(y_test,y_pred)
```

Out[31]: <matplotlib.collections.PathCollection at 0x7fdd7f3ba4c0>



In [32]:
```
plt.plot(y_test,y_pred)
```

Out[32]: [<matplotlib.lines.Line2D at 0x7fdd8017c0a0>]



K Nearlest Neighbour Model

In [33]:
```
pred = knn.predict(X_test)
```

In [34]:
```
print(classification_report(y_test,pred))
```

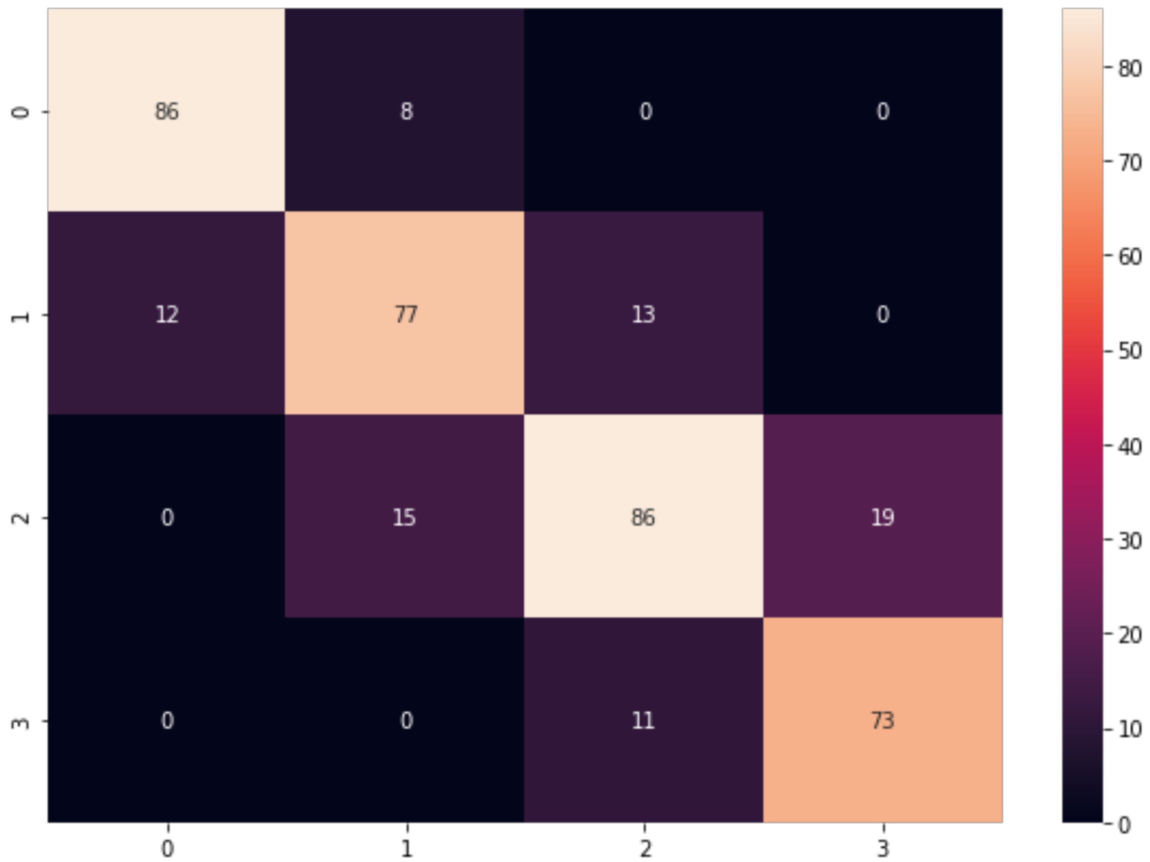
```
              precision    recall  f1-score   support

           0       0.88      0.91      0.90        94
           1       0.77      0.75      0.76       102
           2       0.78      0.72      0.75       120
           3       0.79      0.87      0.83        84

    accuracy                           0.81       400
   macro avg       0.81      0.81      0.81       400
weighted avg       0.80      0.81      0.80       400
```

In [35]:
```
matrix=confusion_matrix(y_test,pred)
print(matrix)
```

```
[[86  8  0  0]
 [12 77 13  0]
 [ 0 15 86 19]
 [ 0  0 11 73]]
```

In [36]:
```
plt.figure(figsize = (10,7))
sns.heatmap(matrix,annot=True)
```

Out[36]: <matplotlib.axes._subplots.AxesSubplot at 0x7fdd801931c0>



In [ ]:

In [ ]: