In [1]:
```
#!pip install pandas
#!pip install matplotlib
#!pip install scikit-learn
```

In [2]:
```
import pandas as pd
import matplotlib.pyplot as plt
```

In [8]:
```
data=pd.read_csv(r'C:\Users\NARENDRA\Desktop\ML\mobile_price_range_data.csv')
```

In [9]:
```
data
```

Out[9]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_
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	
...	...	...	...	...	...	...	...	...	...	
1995	794	1	0.5	1	0	1	2	0.8	106	
1996	1965	1	2.6	1	0	0	39	0.2	187	
1997	1911	0	0.9	1	1	1	36	0.7	108	
1998	1512	0	0.9	0	4	1	46	0.1	145	
1999	510	1	2.0	1	5	1	45	0.9	168	

2000 rows × 21 columns

In [11]:
```
data_mobile_price_range_data
```

Out[11]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt
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
...	...	...	...	...	...	...	...	...	...
1995	794	1	0.5	1	0	1	2	0.8	106
1996	1965	1	2.6	1	0	0	39	0.2	187
1997	1911	0	0.9	1	1	1	36	0.7	108
1998	1512	0	0.9	0	4	1	46	0.1	145
1999	510	1	2.0	1	5	1	45	0.9	168

Explortory Data Analysis

In [14]:
```
data_mobile_price_range_data.shape
```

Out[14]:
```
(2000, 21)
```

In [13]:
```
data_mobile_price_range_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_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  
```

In [15]:
```
data_mobile_price_range_data.describe()
```

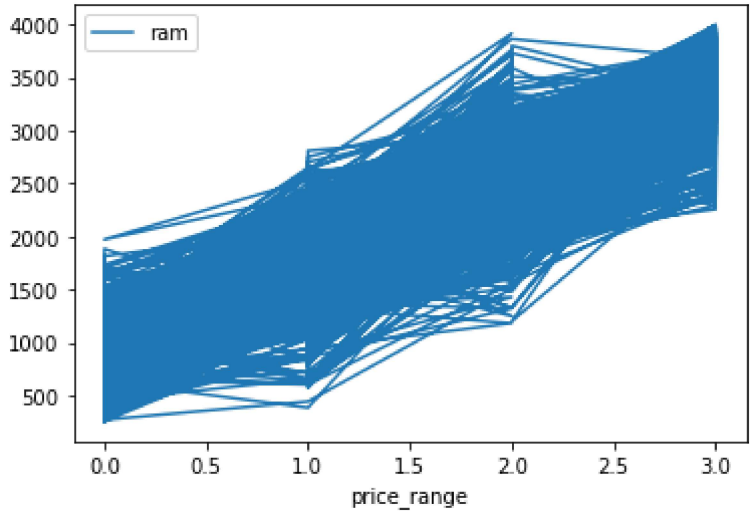
Out[15]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory
count	2000.000000	2000.0000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000
mean	1238.518500	0.4950	1.522250	0.509500	4.309500	0.521500	32.046500
std	439.418206	0.5001	0.816004	0.500035	4.341444	0.499662	18.145715
min	501.000000	0.0000	0.500000	0.000000	0.000000	0.000000	2.000000
25%	851.750000	0.0000	0.700000	0.000000	1.000000	0.000000	16.000000
50%	1226.000000	0.0000	1.500000	1.000000	3.000000	1.000000	32.000000
75%	1615.250000	1.0000	2.200000	1.000000	7.000000	1.000000	48.000000
max	1998.000000	1.0000	3.000000	1.000000	19.000000	1.000000	64.000000

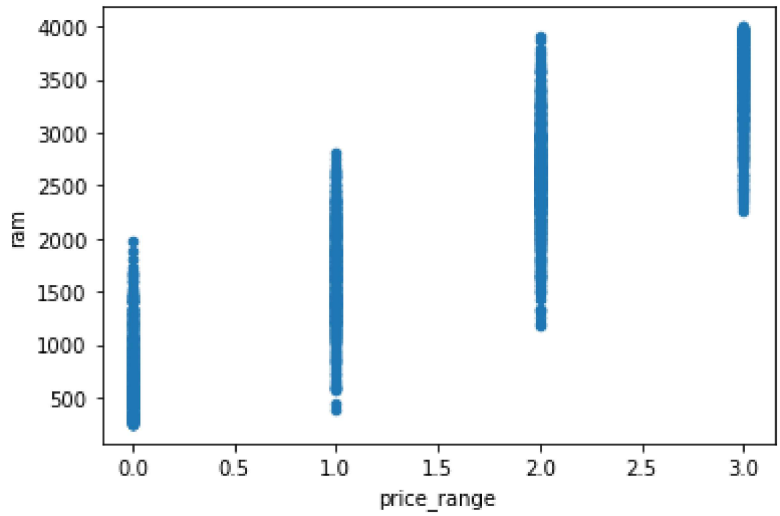
8 rows × 21 columns

In [17]:
```
data_mobile_price_range_data.plot(x='price_range',y='ram')
```

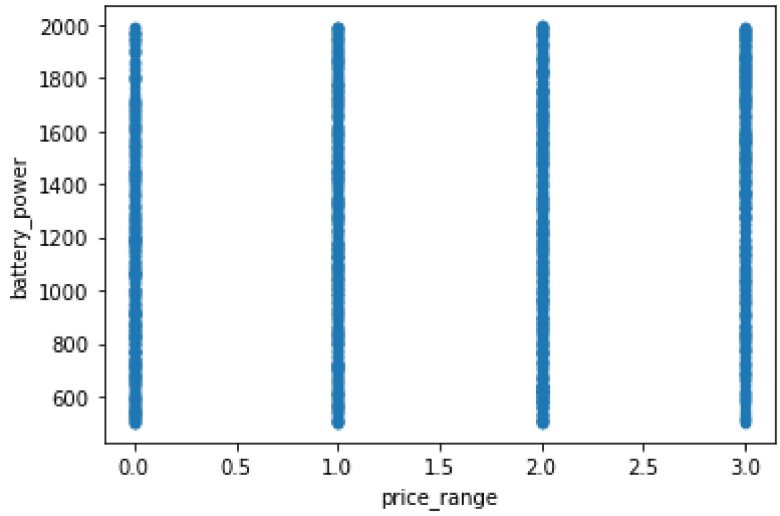
Out[17]:
```
<AxesSubplot:xlabel='price_range'>
```



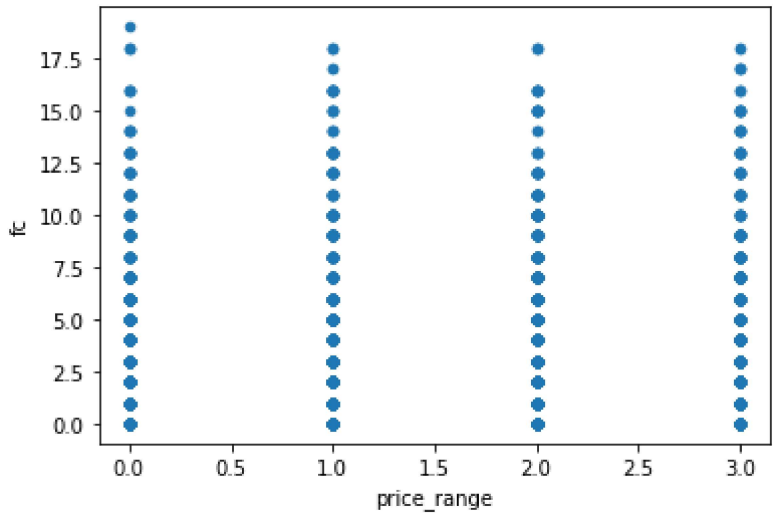
In [16]:
```
data_mobile_price_range_data.plot(x='price_range',y='ram',kind='scatter')
plt.show()
```



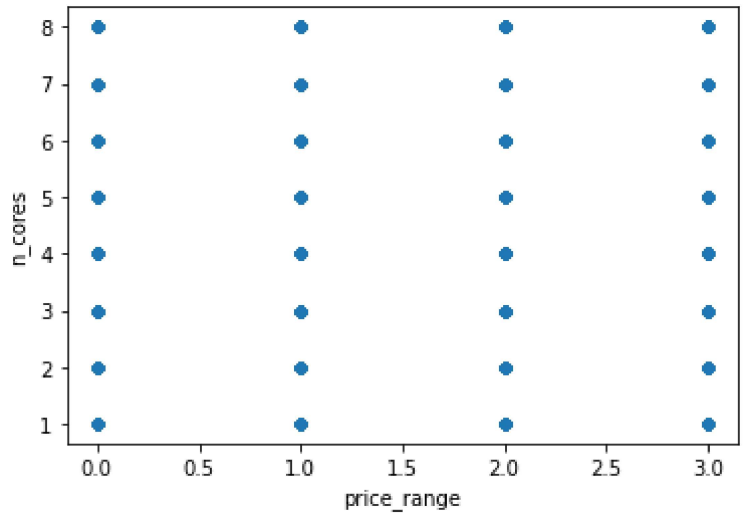
In [19]:
```
data_mobile_price_range_data.plot(x='price_range',y='battery_power',kind='scatter
plt.show()
```



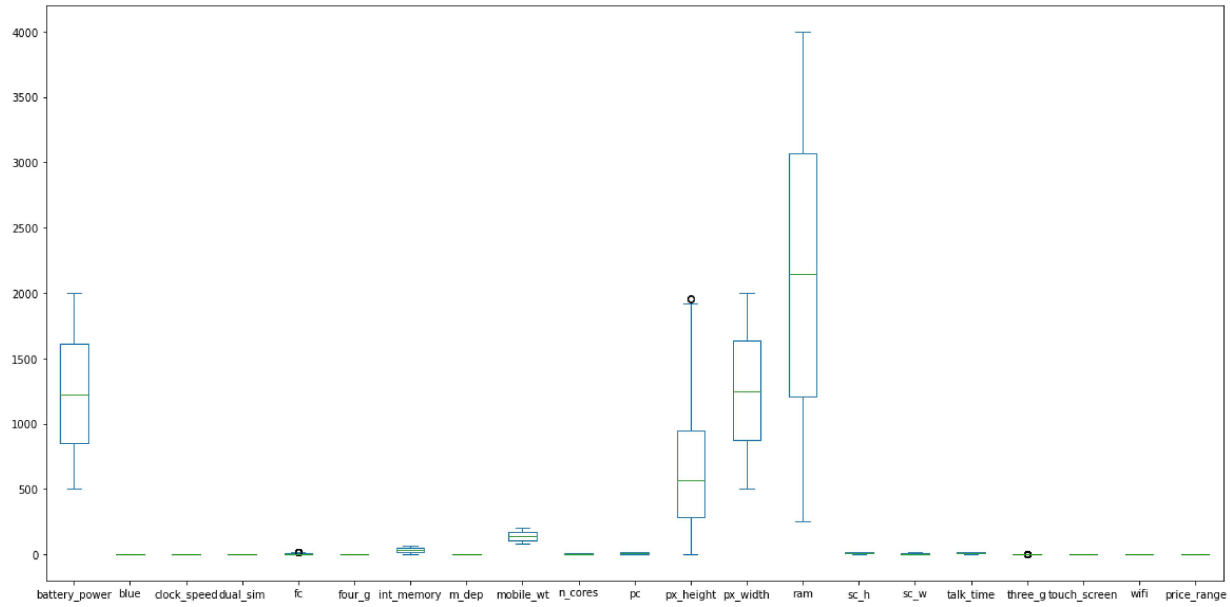
In [20]:
```
data_mobile_price_range_data.plot(x='price_range',y='fc',kind='scatter')
plt.show()
```



In [21]:
```
data_mobile_price_range_data.plot(x='price_range',y='n_cores',kind='scatter')
plt.show()
```



In [22]:
```
data_mobile_price_range_data.plot(kind='box',figsize=(20,10))
plt.show()
```



In [76]:
```
X=data_mobile_price_range_data.drop('price_range',axis=1)
```

In [77]:
```
X
```

Out[77]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_
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	
...	...	...	...	...	...	...	...	...	...	
1995	794	1	0.5	1	0	1	2	0.8	106	
1996	1965	1	2.6	1	0	0	39	0.2	187	
1997	1911	0	0.9	1	1	1	36	0.7	108	
1998	1512	0	0.9	0	4	1	46	0.1	145	
1999	510	1	2.0	1	5	1	45	0.9	168	

2000 rows × 20 columns

In [78]:
```
data_mobile_price_range_data.head()
```

Out[78]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cor
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 [79]:
```
data_mobile_price_range_data.shape
```

Out[79]:
```
(2000, 21)
```

In [81]:
```
Y=data_mobile_price_range_data['price_range']
```

In [82]:
```
Y
```

Out[82]:
```
0       1
1       2
2       2
3       2
4       1
       ..
1995    0
1996    2
1997    3
1998    0
1999    3
Name: price_range, Length: 2000, dtype: int64
```

In [83]:
```
data_mobile_price_range_data.isnull()
```

Out[83]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt
0	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...
1995	False	False	False	False	False	False	False	False	False
1996	False	False	False	False	False	False	False	False	False
1997	False	False	False	False	False	False	False	False	False
1998	False	False	False	False	False	False	False	False	False
1999	False	False	False	False	False	False	False	False	False

2000 rows × 21 columns

In [84]:
```
data_mobile_price_range_data.isnull().sum()
```

Out[84]:
```
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 [99]:
```
data_mobile_price_range_data.shape
```

Out[99]:
```
(2000, 21)
```

In [87]:
```
from sklearn.preprocessing import StandardScaler
std=StandardScaler()
```

In [ ]:
```
# if requqrd

# X_std=std.fit_transform(X)

# data_mobile_price_range_data_std=std.transform(data_mobile_price_range_data)
```

In [104]:
```
X_std
```

Out[104]:
```
array([[-0.90259726, -0.9900495 ,  0.83077942, ..., -1.78686097,
        -1.00601811,  0.98609664],
       [-0.49513857,  1.0100505 , -1.2530642 , ...,  0.55964063,
         0.99401789, -1.01409939],
       [-1.5376865 ,  1.0100505 , -1.2530642 , ...,  0.55964063,
         0.99401789, -1.01409939],
       ...,
       [ 1.53077336, -0.9900495 , -0.76274805, ...,  0.55964063,
         0.99401789, -1.01409939],
       [ 0.62252745, -0.9900495 , -0.76274805, ...,  0.55964063,
         0.99401789,  0.98609664],
       [-1.65833069,  1.0100505 ,  0.58562134, ...,  0.55964063,
         0.99401789,  0.98609664]])
```

In [ ]:

Decision Tree // Traning the model

In [93]:
```
from sklearn.tree import DecisionTreeClassifier
dt=DecisionTreeClassifier()
```

In [94]:
```
dt.fit(X_std,Y)
```

Out[94]:
```
DecisionTreeClassifier()
```

In [111]:
```
data_mobile_price_range_data
```

Out[111]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_
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	
...	...	...	...	...	...	...	...	...	...	
1995	794	1	0.5	1	0	1	2	0.8	106	
1996	1965	1	2.6	1	0	0	39	0.2	187	
1997	1911	0	0.9	1	1	1	36	0.7	108	
1998	1512	0	0.9	0	4	1	46	0.1	145	
1999	510	1	2.0	1	5	1	45	0.9	168	

2000 rows × 21 columns

In [119]:
```
X_std
```

Out[119]:
```
array([[-0.90259726, -0.9900495 ,  0.83077942, ..., -1.78686097,
        -1.00601811,  0.98609664],
       [-0.49513857,  1.0100505 , -1.2530642 , ...,  0.55964063,
         0.99401789, -1.01409939],
       [-1.5376865 ,  1.0100505 , -1.2530642 , ...,  0.55964063,
         0.99401789, -1.01409939],
       ...,
       [ 1.53077336, -0.9900495 , -0.76274805, ...,  0.55964063,
         0.99401789, -1.01409939],
       [ 0.62252745, -0.9900495 , -0.76274805, ...,  0.55964063,
         0.99401789,  0.98609664],
       [-1.65833069,  1.0100505 ,  0.58562134, ...,  0.55964063,
         0.99401789,  0.98609664]])
```

In [ ]:

KNN

In [123]:
```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier()
```

In [124]:
```
knn.fit(X_std,Y)
```

Out[124]:
```
KNeighborsClassifier()
```

In [134]:
```
# if requard
# knn.predict(data_mobile_price_range_data_std)
```

In [136]:
```
X_std
```

Out[136]:
```
array([[-0.90259726, -0.9900495 ,  0.83077942, ..., -1.78686097,
        -1.00601811,  0.98609664],
       [-0.49513857,  1.0100505 , -1.2530642 , ...,  0.55964063,
         0.99401789, -1.01409939],
       [-1.5376865 ,  1.0100505 , -1.2530642 , ...,  0.55964063,
         0.99401789, -1.01409939],
       ...,
       [ 1.53077336, -0.9900495 , -0.76274805, ...,  0.55964063,
         0.99401789, -1.01409939],
       [ 0.62252745, -0.9900495 , -0.76274805, ...,  0.55964063,
         0.99401789,  0.98609664],
       [-1.65833069,  1.0100505 ,  0.58562134, ...,  0.55964063,
         0.99401789,  0.98609664]])
```

In [ ]:

Logistic Regression

In [138]:
```
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()
```

In [139]:
```
lr.fit(X_std,Y)
```

Out[139]:
```
LogisticRegression()
```

In [142]:
```
# if req...
# lr.predict(data_mobile_price_range_data_std)
```

In [ ]:

As we predicted on mobile_price_range_data Data csv, we are not able to plot accuracy score as we dont have Ground Truth.

In [ ]:

END

In [ ]: