```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import os
import plotly as pt
from pathlib import Path
import warnings
warnings.filterwarnings('ignore')
```

Dataset columns are as follows:

- id - ID
- battery_power - Total energy a battery can store in one time measured in mAh
- blue - Has Bluetooth or not
- clock_speed - The speed at which the microprocessor executes instructions
- dual_sim - Has dual sim support or not
- fc - Front Camera megapixels
- 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 the processor
- pc - Primary Camera megapixels
- px_height - Pixel Resolution Height
- px_width - Pixel Resolution Width
- ram - Random Access Memory in Megabytes
- 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 the value of:
  - 0 (low cost)
  - 1 (medium cost)
  - 2 (high cost)
  - 3 (very high cost)

```
FOLDER_PATH = Path("/content/drive/MyDrive/maids AI test/")
train = pd.read_csv(FOLDER_PATH.joinpath("train.csv"))
test = pd.read_csv(FOLDER_PATH.joinpath("test.csv"))
```

```
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             1995 non-null   float64
 5   four_g         1995 non-null   float64
```

```
 6   int_memory     1995 non-null   float64
 7   m_dep          1995 non-null   float64
 8   mobile_wt      1996 non-null   float64
 9   n_cores        1996 non-null   float64
 10  pc             1995 non-null   float64
 11  px_height      1996 non-null   float64
 12  px_width       1998 non-null   float64
 13  ram            1998 non-null   float64
 14  sc_h           1999 non-null   float64
 15  sc_w           1999 non-null   float64
 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(13), int64(8)
memory usage: 328.2 KB
```

```
train.describe()
```

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	
count	2000.000000	2000.0000	2000.000000	2000.000000	1995.000000	1995.000000	1995.000000	1995.000000	1996.000000	1
mean	1238.518500	0.4950	1.522250	0.509500	4.310276	0.521303	32.048120	0.502256	140.266533	
std	439.418206	0.5001	0.816004	0.500035	4.335766	0.499671	18.146476	0.288530	35.384676	
min	501.000000	0.0000	0.500000	0.000000	0.000000	0.000000	2.000000	0.100000	80.000000	
25%	851.750000	0.0000	0.700000	0.000000	1.000000	0.000000	16.000000	0.200000	109.000000	
50%	1226.000000	0.0000	1.500000	1.000000	3.000000	1.000000	32.000000	0.500000	141.000000	
75%	1615.250000	1.0000	2.200000	1.000000	7.000000	1.000000	48.000000	0.800000	170.000000	
max	1998.000000	1.0000	3.000000	1.000000	19.000000	1.000000	64.000000	1.000000	200.000000	

8 rows × 21 columns

```
train.price_range.value_counts()
```

```
price_range
1    500
2    500
3    500
0    500
Name: count, dtype: int64
```

we note that the data is balanced dataset

```
train[(train.price_range == 1) & (train.ram.isna())]
```

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	...	px_height	px_wi
261	728	0	2.7	1	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	N
341	811	1	2.4	1	5.0	1.0	2.0	0.3	106.0	6.0	...	NaN	N

2 rows × 21 columns

```
train.groupby("price_range")["ram"].agg({"min","max","median","mean","count"}).reset_index()
```

	price_range	count	min	median	mean	max
0	0	500	256.0	719.5	785.314000	1974.0
1	1	498	387.0	1680.5	1677.901606	2811.0
2	2	500	1185.0	2577.0	2582.816000	3916.0
3	3	500	2259.0	3509.5	3449.232000	3998.0

drop blue

```
train[train.ram <= 1000].ram.value_counts()
```

```
train[train.ram < 1000].ram.value_counts()
```

```
ram
594.0    3
315.0    3
606.0    3
595.0    3
348.0    3
        ..
282.0    1
277.0    1
676.0    1
286.0    1
668.0    1
Name: count, Length: 322, dtype: int64
```

```
train.touch_screen.value_counts()
```

```
touch_screen
1    1006
0     994
Name: count, dtype: int64
```

```
ram_dict = {(0,350):256 , (351,750):512, (751,1500):1000, (1501,2500):2000, (2501,3500):3000, (3501,4500):4000}
def ram_bins (df:pd.DataFrame, ram_dict)-> None:
  for key in ram_dict.keys():
    df.loc[df.ram.between(key[0], key[1]) , 'ram'] = ram_dict[key]
#ram_bins(train, ram_dict)
print(train.ram.value_counts())
```

```
ram
1464.0    4
3142.0    4
2610.0    4
2227.0    4
1229.0    4
          ..
2297.0    1
2312.0    1
2167.0    1
3508.0    1
3919.0    1
Name: count, Length: 1561, dtype: int64
```

```
def explore_the_data(df:pd.DataFrame, target:str)->None:
  for col in df.columns.difference([target]):
    print(f"------------------------------------ {col} ------------------------------------")
    print("The number of unique values: ", train[col].unique().size)
    if len(train[train[col] == 0]) != 0 and train[col].unique().size > 2:
      print("There are zero values: ", len(train[train[col] == 0]))
      display(train[train[col] == 0])
explore_the_data(train,"price_range")
```

---

int_memory	m_dep	mobile_wt	n_cores	...	px_height	px_width	ram	sc_h	sc_w	talk_time	three_g	touch_screen	wifi
53.0	0.7	136.0	3.0	...	905.0	1988.0	2631.0	17.0	3.0	7	1	1	0
10.0	0.8	131.0	6.0	...	1216.0	1786.0	2769.0	16.0	8.0	11	1	0	0
24.0	0.8	187.0	4.0	...	512.0	1149.0	700.0	16.0	3.0	5	1	1	1
53.0	0.7	174.0	7.0	...	386.0	836.0	1099.0	17.0	1.0	20	1	0	0
9.0	0.1	182.0	5.0	...	248.0	874.0	3946.0	5.0	2.0	7	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
25.0	0.3	163.0	7.0	...	455.0	537.0	2215.0	9.0	3.0	17	1	1	1
37.0	0.9	144.0	7.0	...	206.0	1167.0	2216.0	9.0	5.0	6	1	0	0
18.0	0.6	122.0	5.0	...	888.0	1099.0	3962.0	15.0	11.0	5	1	1	1
2.0	0.8	106.0	6.0	...	1222.0	1890.0	668.0	13.0	4.0	19	1	1	0
39.0	0.2	187.0	4.0	...	915.0	1965.0	2032.0	11.0	10.0	16	1	1	1

---

int_memory	m_dep	mobile_wt	n_cores	...	px_height	px_width	ram	sc_h	sc_w	talk_time	three_g	touch_screen	wifi
7.0	0.6	188.0	2.0	...	20.0	756.0	2549.0	9.0	7.0	19	0	0	1
10.0	0.8	131.0	6.0	...	1216.0	1786.0	2769.0	16.0	8.0	11	1	0	0
22.0	0.7	164.0	1.0	...	1004.0	1654.0	1067.0	17.0	1.0	10	1	0	0
24.0	0.8	187.0	4.0	...	512.0	1149.0	700.0	16.0	3.0	5	1	1	1
53.0	0.7	174.0	7.0	...	386.0	836.0	1099.0	17.0	1.0	20	1	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
15.0	0.2	83.0	3.0	...	241.0	854.0	2592.0	12.0	8.0	3	0	0	0
21.0	0.2	198.0	3.0	...	576.0	1809.0	1180.0	6.0	3.0	4	1	1	1
18.0	0.6	122.0	5.0	...	888.0	1099.0	3962.0	15.0	11.0	5	1	1	1
50.0	0.1	84.0	1.0	...	528.0	1416.0	3978.0	17.0	16.0	3	1	1	0
39.0	0.2	187.0	4.0	...	915.0	1965.0	2032.0	11.0	10.0	16	1	1	1

---

int_memory	m_dep	mobile_wt	n_cores	...	px_height	px_width	ram	sc_h	sc_w	talk_time	three_g	touch_screen	wifi
24.0	0.8	187.0	4.0	...	512.0	1149.0	700.0	16.0	3.0	5	1	1	1
51.0	0.6	100.0	4.0	...	178.0	1919.0	3845.0	7.0	0.0	12	1	1	0
43.0	0.3	109.0	2.0	...	546.0	629.0	3112.0	12.0	5.0	10	1	1	0
51.0	0.5	145.0	7.0	...	690.0	804.0	2908.0	6.0	0.0	18	0	1	0
50.0	0.8	159.0	2.0	...	322.0	547.0	470.0	7.0	0.0	15	1	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...

38.0	0.2	97.0	8.0	...	1332.0	1814.0	1069.0	7.0	6.0	7	1	0	0
13.0	0.5	97.0	2.0	...	1210.0	1989.0	340.0	17.0	13.0	4	1	0	1
33.0	0.4	195.0	1.0	...	665.0	718.0	1675.0	14.0	12.0	9	1	0	1
21.0	0.9	138.0	2.0	...	1211.0	1396.0	635.0	17.0	7.0	15	1	0	0
18.0	0.6	122.0	5.0	...	888.0	1099.0	3962.0	15.0	11.0	5	1	1	1

---

int_memory	m_dep	mobile_wt	n_cores	...	px_height	px_width	ram	sc_h	sc_w	talk_time	three_g	touch_screen	wifi
40.0	0.1	99.0	4.0	...	0.0	1987.0	3692.0	13.0	0.0	16	1	1	0
2.0	0.6	154.0	8.0	...	0.0	994.0	1958.0	7.0	5.0	7	1	1	0

---

int_memory	m_dep	mobile_wt	n_cores	...	px_height	px_width	ram	sc_h	sc_w	talk_time	three_g	touch_screen	wifi
33.0	0.6	159.0	4.0	...	607.0	748.0	1482.0	18.0	0.0	2	1	0	0
8.0	0.4	111.0	3.0	...	201.0	1245.0	2583.0	11.0	0.0	12	1	0	0
51.0	0.6	100.0	4.0	...	178.0	1919.0	3845.0	7.0	0.0	12	1	1	0
5.0	0.2	152.0	2.0	...	685.0	714.0	1878.0	15.0	0.0	4	1	1	0
14.0	0.7	198.0	3.0	...	1042.0	1832.0	2059.0	5.0	0.0	15	1	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...
23.0	0.6	97.0	7.0	...	317.0	1805.0	2782.0	7.0	0.0	17	1	0	0
16.0	0.9	90.0	6.0	...	892.0	1603.0	3746.0	5.0	0.0	5	1	1	0
32.0	0.4	141.0	6.0	...	227.0	509.0	1817.0	10.0	0.0	6	0	1	0
36.0	0.1	194.0	4.0	...	64.0	745.0	1503.0	10.0	0.0	13	0	0	0
6.0	0.4	199.0	3.0	...	698.0	1018.0	1300.0	10.0	0.0	2	0	0	1

Fix the zero value in the "px_height"

```
train.px_height.replace(0,np.NaN, inplace=True)
train.sc_w.replace(0,np.NaN, inplace=True)
display(train.iloc[12])
```

```
battery_power    1815.0
blue                0.0
clock_speed         2.8
dual_sim            0.0
fc                  2.0
four_g              0.0
int_memory         33.0
m_dep               0.6
mobile_wt         159.0
n_cores             4.0
pc                 17.0
px_height         607.0
px_width          748.0
ram              1482.0
sc_h               18.0
sc_w                NaN
talk_time           2.0
three_g             1.0
touch_screen        0.0
wifi                0.0
price_range         1.0
Name: 12, dtype: float64
```

```
cheak =["fc", "four_g", "m_dep","mobile_wt","n_cores","pc","px_height","px_width","ram","sc_h","sc_w"]
train.loc[train.int_memory.isna(),cheak]
```

	fc	four_g	m_dep	mobile_wt	n_cores	pc	px_height	px_width	ram	sc_h	sc_w
157	NaN	NaN	NaN	NaN	NaN	NaN	747.0	826.0	506.0	10.0	NaN
158	NaN	NaN	NaN	NaN	NaN	NaN	1612.0	1983.0	3702.0	17.0	NaN
261	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	5.0	1.0
276	NaN	NaN	0.8	126.0	4.0	13.0	32.0	1509.0	3760.0	9.0	5.0
371	NaN	NaN	NaN	NaN	NaN	18.0	481.0	749.0	2261.0	7.0	6.0

```
target = "price_range"
X = train.drop([target], axis =1)
y = train[target]
```

```
from sklearn.impute import KNNImputer
from sklearn.preprocessing import StandardScaler

std = StandardScaler()
X = std.fit_transform(X)
imputer = KNNImputer()
X = pd.DataFrame(imputer.fit_transform(X),columns = train.columns.difference([target]))
```

```
train[train.columns.difference([target])] = X
train
```

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	...
0	-0.902597	-0.990050	0.830779	-1.019184	-0.763673	-1.043554	-1.380676	0.338851	1.349325	-1.100361	...
1	-0.495139	1.010051	-1.253064	0.981177	-0.994370	0.958264	1.154887	0.685522	-0.120606	-0.663369	...
2	-1.537686	1.010051	-1.253064	0.981177	-0.532975	0.958264	0.493436	1.378865	0.133805	0.210614	...
3	-1.419319	1.010051	1.198517	-1.019184	-0.994370	-1.043554	-1.215313	1.032194	-0.261945	0.647606	...
4	1.325906	1.010051	-0.395011	-1.019184	2.004698	0.958264	0.658799	0.338851	0.020734	-1.100361	...
...	...	...	...	...	...	...	...	...	...	...	...
1995	-1.011860	1.010051	-1.253064	0.981177	-0.994370	0.958264	-1.656281	1.032194	-0.968643	0.647606	...
1996	1.653694	1.010051	1.321096	0.981177	-0.994370	-1.043554	0.383194	-1.047833	1.321057	-0.226378	...
1997	1.530773	-0.990050	-0.762748	0.981177	-0.763673	0.958264	0.217831	0.685522	-0.912107	1.521590	...
1998	0.622527	-0.990050	-0.762748	-1.019184	-0.071580	0.958264	0.769041	-1.394504	0.133805	0.210614	...
1999	-1.658331	1.010051	0.585621	0.981177	0.159118	0.958264	0.713920	1.378865	0.783967	0.647606	...

2000 rows × 21 columns

```
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   float64
 1   blue           2000 non-null   float64
 2   clock_speed    2000 non-null   float64
 3   dual_sim       2000 non-null   float64
 4   fc             2000 non-null   float64
 5   four_g         2000 non-null   float64
 6   int_memory     2000 non-null   float64
 7   m_dep          2000 non-null   float64
 8   mobile_wt      2000 non-null   float64
 9   n_cores        2000 non-null   float64
 10  pc             2000 non-null   float64
 11  px_height      2000 non-null   float64
 12  px_width       2000 non-null   float64
 13  ram            2000 non-null   float64
 14  sc_h           2000 non-null   float64
 15  sc_w           2000 non-null   float64
 16  talk_time      2000 non-null   float64
 17  three_g        2000 non-null   float64
 18  touch_screen   2000 non-null   float64
 19  wifi           2000 non-null   float64
 20  price_range    2000 non-null   int64  
dtypes: float64(20), int64(1)
memory usage: 328.2 KB
```

```
train.describe()
```

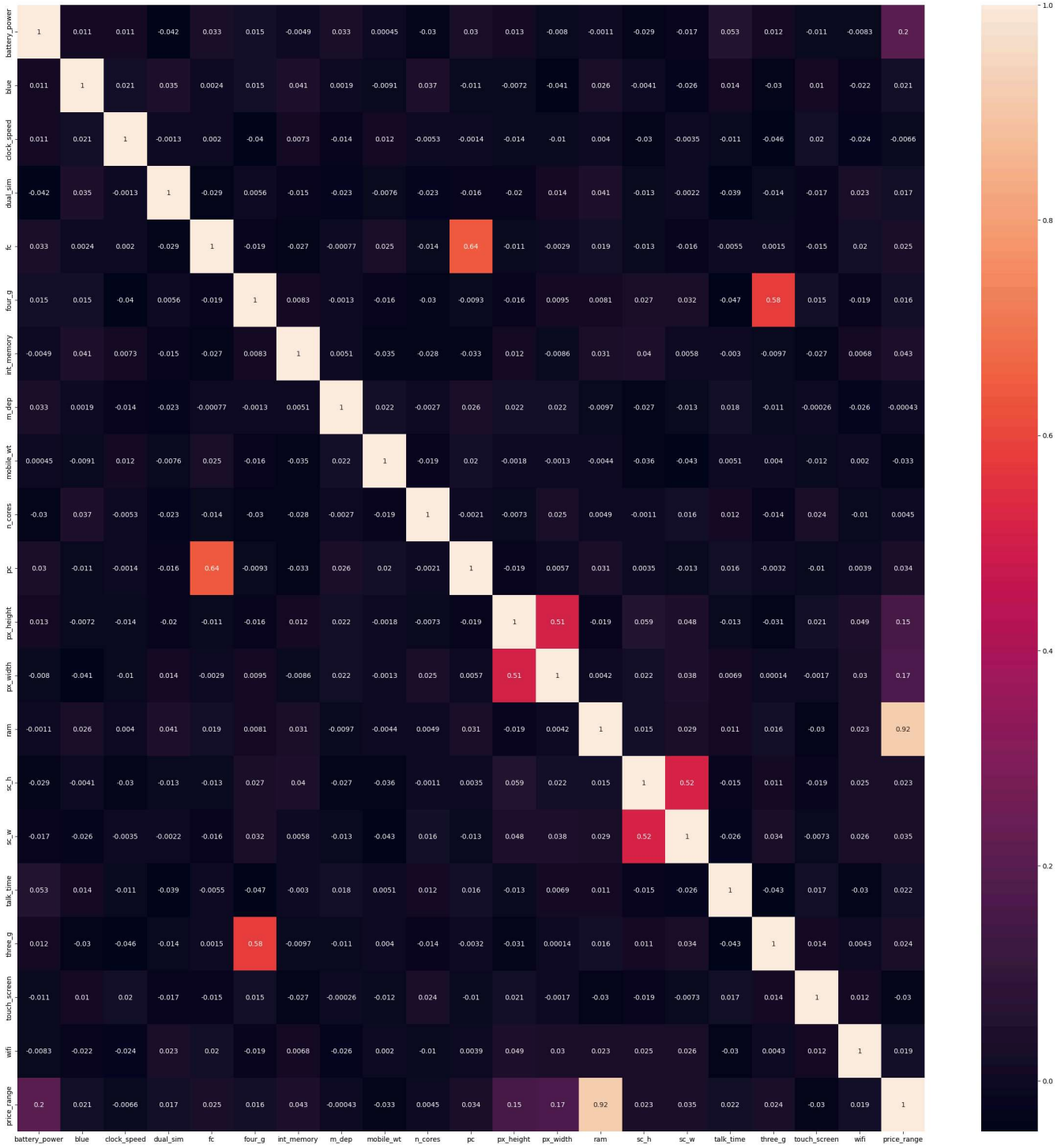
	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile
count	2.000000e+03	2.000000e+03	2.000000e+03	2.000000e+03	2000.000000	2000.000000	2000.000000	2000.000000	2000.000
mean	2.149392e-16	-1.243450e-17	-1.545430e-16	8.082424e-17	0.001044	-0.000807	-0.000062	0.000431	0.000
std	1.000250e+00	1.000250e+00	1.000250e+00	1.000250e+00	0.999337	0.999322	0.999019	0.999139	0.999
min	-1.678817e+00	-9.900495e-01	-1.253064e+00	-1.019184e+00	-0.994370	-1.043554	-1.656281	-1.394504	-1.703
25%	-8.804033e-01	-9.900495e-01	-1.007906e+00	-1.019184e+00	-0.763673	-1.043554	-0.884587	-1.047833	-0.883
50%	-2.849593e-02	-9.900495e-01	-2.727384e-02	9.811771e-01	-0.302277	0.958264	-0.002652	-0.007820	0.020
75%	8.575560e-01	1.010051e+00	8.307794e-01	9.811771e-01	0.620513	0.958264	0.879283	1.032194	0.840
max	1.728812e+00	1.010051e+00	1.811412e+00	9.811771e-01	3.388884	0.958264	1.761218	1.725536	1.688

8 rows × 21 columns

Features Importance

Correlation

```
corr = train.corr(method='spearman')#'pearson', 'kendall', 'spearman'
top_features = corr.index
plt.figure(figsize=(30,30))
sns.heatmap(train[top_features].corr(method='pearson'),annot=True)
plt.show()
```



From the correlation matrix, we see that the target (dependent variable): "price_range" is highly correlated with these features (independent variables):

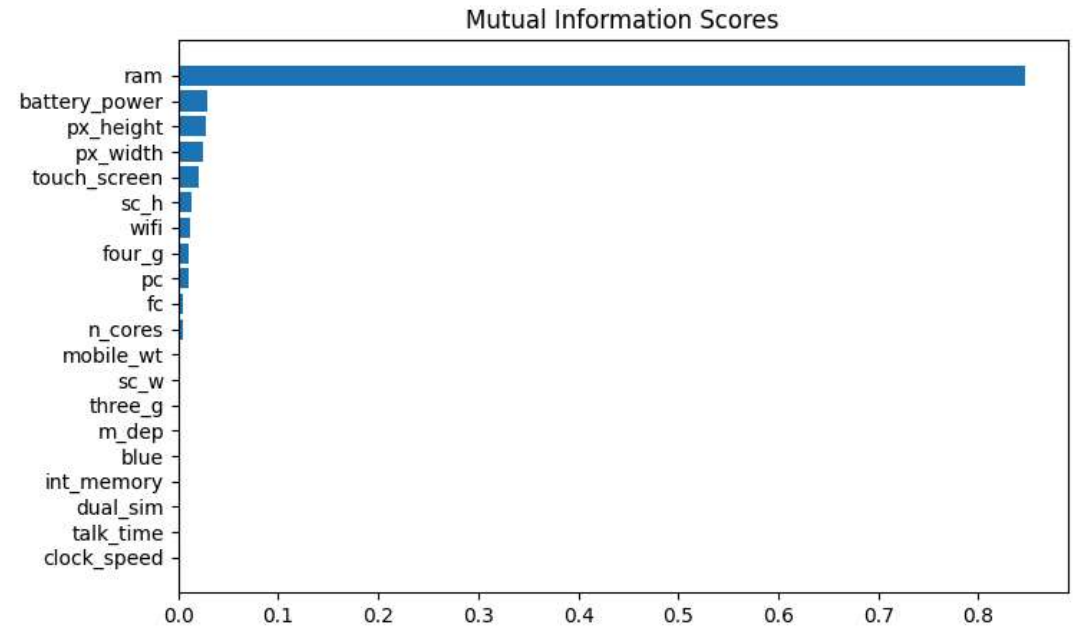
1. "ram" corr = 0.92
2. "battery_power" = 0.2
3. "px_height" = 0.17
4. "px_width" = 0.15

Other features have a correlation value close to zero, so we can drop them cause they don't give us that much information even if it can lead to the worst results. another reason to drop features causes the dimensionality of the data in the features space is 20 with 2000 data samples which can lead to overfitting in the model so to avoid that we should reduce the dimensionality.

Mutual Information

```
from sklearn.feature_selection import mutual_info_regression

def make_mi_scores(X, y):
    X = X.copy()
    for colname in X.select_dtypes(["object", "category"]):
        X[colname], _ = X[colname].factorize()
    discrete_features = [pd.api.types.is_integer_dtype(t) for t in X.dtypes]
    mi_scores = mutual_info_regression(X, y, discrete_features=discrete_features, random_state=0)
    mi_scores = pd.Series(mi_scores, name="MI Scores", index=X.columns)
    mi_scores = mi_scores.sort_values(ascending=False)
    return mi_scores


def plot_mi_scores(scores):
    scores = scores.sort_values(ascending=True)
    width = np.arange(len(scores))
    ticks = list(scores.index)
    plt.barh(width, scores)
    plt.yticks(width, ticks)
    plt.title("Mutual Information Scores")


mi_scores = make_mi_scores(X, y)


print(mi_scores.head(20))

plt.figure(dpi=100, figsize=(8, 5))
plot_mi_scores(mi_scores.head(20))
```

```
ram              0.846918
battery_power    0.028429
px_height        0.028166
px_width         0.024549
touch_screen     0.020151
sc_h             0.013973
wifi             0.012472
four_g           0.011006
pc               0.010588
fc               0.005407
n_cores          0.005392
mobile_wt        0.002501
sc_w             0.000067
m_dep            0.000000
blue             0.000000
int_memory       0.000000
dual_sim         0.000000
talk_time        0.000000
three_g          0.000000
clock_speed      0.000000
Name: MI Scores, dtype: float64
```



Features Importance based on Models (RF)

```
!pip install catboost
```

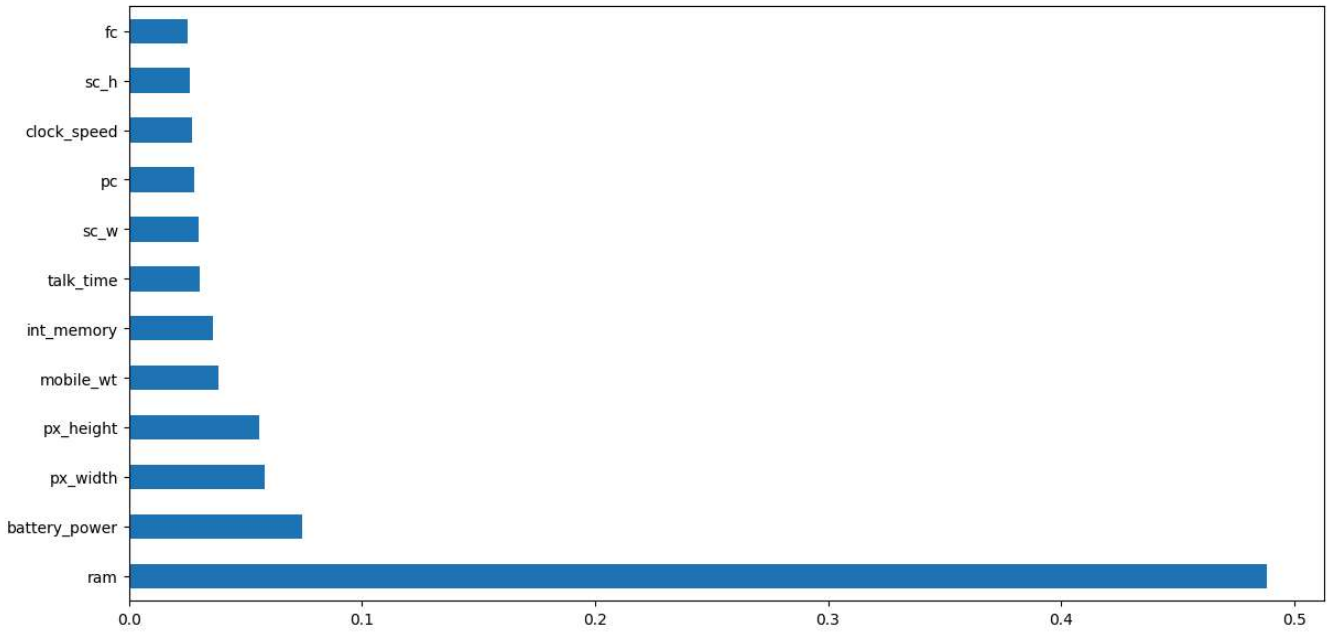
```
Collecting catboost
  Downloading catboost-1.2.3-cp310-cp310-manylinux2014_x86_64.whl (98.5 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 98.5/98.5 MB 5.9 MB/s eta 0:00:00
Requirement already satisfied: graphviz in /usr/local/lib/python3.10/dist-packages (from catboost) (0.20.3)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (from catboost) (3.7.1)
Requirement already satisfied: numpy>=1.16.0 in /usr/local/lib/python3.10/dist-packages (from catboost) (1.25.2)
Requirement already satisfied: pandas>=0.24 in /usr/local/lib/python3.10/dist-packages (from catboost) (2.0.3)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from catboost) (1.11.4)
Requirement already satisfied: plotly in /usr/local/lib/python3.10/dist-packages (from catboost) (5.15.0)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from catboost) (1.16.0)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.24->catbo
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.24->catboost) (202
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.24->catboost) (20
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (1
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (0.12.1
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (4
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (1
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (24
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (9.4.0
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (3
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from plotly->catboost) (8.2.3)
Installing collected packages: catboost
Successfully installed catboost-1.2.3
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
from sklearn.ensemble import ExtraTreesClassifier,RandomForestClassifier,VotingClassifier
from catboost import CatBoost,CatBoostClassifier
import xgboost as xgb
from xgboost import XGBClassifier
import lightgbm as lgb
from lightgbm import LGBMClassifier
```

```
model = RandomForestClassifier()
model.fit(X,y)
RF_ranked_features = pd.Series(model.feature_importances_,index=X.columns)
RF_ranked_features.nlargest(20)
```

```
ram              0.488314
battery_power    0.074474
px_width         0.058072
px_height        0.055880
mobile_wt        0.038331
int_memory       0.035861
talk_time        0.030476
sc_w             0.029870
pc               0.028169
clock_speed      0.027020
sc_h             0.025973
fc               0.025316
m_dep            0.022584
n_cores          0.022253
dual_sim         0.006981
touch_screen     0.006858
four_g           0.006239
wifi             0.005997
blue             0.005755
three_g          0.005577
dtype: float64
```

```
plt.figure(figsize=(14,7))
RF_ranked_features.nlargest(12).plot(kind = "barh")
plt.show()
```



As we saw from the three feature selection methods, the most important features were:

1. ram
2. battery_power
3. px_height
4. px_width

So, I'll rely on them in my next steps.

```
selected_features = ["ram","battery_power" ,"px_height","px_width"]
X = train[selected_features]
y= train["price_range"]
```

First shoot for model selection

lazypredict is a library that saves you time when you are trying to figure out the most suitable models for a specific machine-learning project.

```
! pip install lazypredict
```

```
Collecting lazypredict
  Downloading lazypredict-0.2.12-py2.py3-none-any.whl (12 kB)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from lazypredict) (8.1.7)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (from lazypredict) (1.2.2)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (from lazypredict) (2.0.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from lazypredict) (4.66.2)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from lazypredict) (1.3.2)
Requirement already satisfied: lightgbm in /usr/local/lib/python3.10/dist-packages (from lazypredict) (4.1.0)
Requirement already satisfied: xgboost in /usr/local/lib/python3.10/dist-packages (from lazypredict) (2.0.3)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from lightgbm->lazypredict) (1.25.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from lightgbm->lazypredict) (1.11.4)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas->lazypredict
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas->lazypredict) (2023.4
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas->lazypredict) (2024
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn->lazypre
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2->pandas->l
Installing collected packages: lazypredict
Successfully installed lazypredict-0.2.12
```

```
from lazypredict.Supervised import LazyClassifier
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,test_size=.3,random_state =123)
clf = LazyClassifier(verbose=0,ignore_warnings=True, custom_metric=None)
models,predictions = clf.fit(X_train, X_test, y_train, y_test)
models
```

```
 97%|▉| 28/29 [00:04<00:00,  3.78it/s][LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 1020
[LightGBM] [Info] Number of data points in the train set: 1400, number of used features: 4
[LightGBM] [Info] Start training from score -1.380596
[LightGBM] [Info] Start training from score -1.397789
[LightGBM] [Info] Start training from score -1.424145
[LightGBM] [Info] Start training from score -1.344330
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
100%|▉| 29/29 [00:05<00:00,  4.93it/s]
```

Model	Accuracy	Balanced Accuracy	ROC AUC	F1 Score	Time Taken
LogisticRegression	0.95	0.95	None	0.95	0.05
QuadraticDiscriminantAnalysis	0.94	0.94	None	0.94	0.05
SVC	0.94	0.94	None	0.93	0.13
NuSVC	0.93	0.93	None	0.93	0.24
LinearDiscriminantAnalysis	0.93	0.93	None	0.93	0.06
ExtraTreesClassifier	0.93	0.93	None	0.92	0.39
XGBClassifier	0.92	0.92	None	0.92	1.36
LGBMClassifier	0.91	0.91	None	0.90	1.10
RandomForestClassifier	0.90	0.90	None	0.90	0.57
LabelSpreading	0.89	0.89	None	0.88	0.27
LabelPropagation	0.88	0.89	None	0.88	0.25
BaggingClassifier	0.88	0.88	None	0.87	0.10
KNeighborsClassifier	0.87	0.87	None	0.87	0.17
DecisionTreeClassifier	0.85	0.85	None	0.85	0.03
ExtraTreeClassifier	0.84	0.84	None	0.84	0.02
LinearSVC	0.82	0.83	None	0.81	0.11
CalibratedClassifierCV	0.81	0.82	None	0.80	0.31
PassiveAggressiveClassifier	0.78	0.79	None	0.77	0.03
GaussianNB	0.77	0.78	None	0.77	0.04
Perceptron	0.77	0.78	None	0.76	0.03
AdaBoostClassifier	0.77	0.77	None	0.77	0.31
SGDClassifier	0.76	0.77	None	0.75	0.05
NearestCentroid	0.75	0.75	None	0.75	0.02
BernoulliNB	0.56	0.57	None	0.54	0.03
RidgeClassifier	0.50	0.53	None	0.37	0.04
RidgeClassifierCV	0.50	0.53	None	0.37	0.03
DummyClassifier	0.23	0.25	None	0.08	0.02

As we note these are the models and their scores.

I'll try to boost the results by using the ensemble method on models from the top 7 models.

```
classifiers_list =[LGBMClassifier,XGBClassifier,LogisticRegression,ExtraTreesClassifier,QuadraticDiscriminantAnalysis,CatBoost
clf = LazyClassifier(verbose=0,ignore_warnings=True, custom_metric=None,classifiers=classifiers_list)
models,predictions = clf.fit(X_train, X_test, y_train, y_test)
models
```

```
959:    learn: 0.0312274        total: 7.21s    remaining: 306ms
960:    learn: 0.0312061        total: 7.22s    remaining: 293ms
961:    learn: 0.0311764        total: 7.24s    remaining: 286ms
962:    learn: 0.0311341        total: 7.26s    remaining: 279ms
963:    learn: 0.0310745        total: 7.28s    remaining: 272ms
964:    learn: 0.0310411        total: 7.3s     remaining: 265ms
965:    learn: 0.0310223        total: 7.32s    remaining: 257ms
966:    learn: 0.0309835        total: 7.32s    remaining: 250ms
967:    learn: 0.0309619        total: 7.34s    remaining: 243ms
968:    learn: 0.0309306        total: 7.35s    remaining: 235ms
969:    learn: 0.0309016        total: 7.37s    remaining: 228ms
970:    learn: 0.0308522        total: 7.38s    remaining: 220ms
971:    learn: 0.0308274        total: 7.39s    remaining: 213ms
972:    learn: 0.0308064        total: 7.41s    remaining: 206ms
973:    learn: 0.0307784        total: 7.43s    remaining: 198ms
974:    learn: 0.0307237        total: 7.44s    remaining: 191ms
975:    learn: 0.0306927        total: 7.46s    remaining: 183ms
976:    learn: 0.0306468        total: 7.47s    remaining: 176ms
977:    learn: 0.0306247        total: 7.5s     remaining: 169ms
978:    learn: 0.0305839        total: 7.52s    remaining: 161ms
979:    learn: 0.0305637        total: 7.54s    remaining: 154ms
980:    learn: 0.0305502        total: 7.55s    remaining: 146ms
981:    learn: 0.0305238        total: 7.57s    remaining: 139ms
982:    learn: 0.0304630        total: 7.59s    remaining: 131ms
983:    learn: 0.0304193        total: 7.62s    remaining: 124ms
984:    learn: 0.0303777        total: 7.62s    remaining: 116ms
985:    learn: 0.0303471        total: 7.63s    remaining: 108ms
986:    learn: 0.0303069        total: 7.65s    remaining: 101ms
987:    learn: 0.0302657        total: 7.66s    remaining: 93ms
988:    learn: 0.0302387        total: 7.67s    remaining: 85.4ms
989:    learn: 0.0302141        total: 7.69s    remaining: 77.6ms
990:    learn: 0.0301750        total: 7.7s     remaining: 69.9ms
991:    learn: 0.0301338        total: 7.72s    remaining: 62.2ms
992:    learn: 0.0300897        total: 7.74s    remaining: 54.5ms
993:    learn: 0.0300604        total: 7.75s    remaining: 46.8ms
994:    learn: 0.0300324        total: 7.76s    remaining: 39ms
995:    learn: 0.0299640        total: 7.78s    remaining: 31.2ms
996:    learn: 0.0299415        total: 7.79s    remaining: 23.4ms
997:    learn: 0.0299179        total: 7.82s    remaining: 15.7ms
998:    learn: 0.0298815        total: 7.84s    remaining: 7.84ms
999:    learn: 0.0298519        total: 7.85s    remaining: 0us
100%|██████████| 6/6 [00:10<00:00,  1.77s/it]
```

Model	Accuracy	Balanced Accuracy	ROC AUC	F1 Score	Time Taken
LogisticRegression	0.95	0.95	None	0.95	0.03
QuadraticDiscriminantAnalysis	0.94	0.94	None	0.94	0.02
CatBoostClassifier	0.94	0.94	None	0.94	8.81
ExtraTreesClassifier	0.93	0.93	None	0.92	0.22
XGBClassifier	0.92	0.92	None	0.92	0.82
LGBMClassifier	0.91	0.91	None	0.90	0.71