feature-selection

October 16, 2023

[1]:

```python
# Loading Libraries
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
```

[2]:

```python
# Loading Dataframe
df = pd.read_csv('mobile data.csv')
```

[3]:

```python
df.head()
```

[3]:

```
   battery_power  blue  clock_speed  dual_sim  fc  four_g  int_memory  m_dep  \
0            772     0          1.1         1  12       0          39    0.8
1           1709     1          2.1         0   1       0          13    1.0
2           1949     0          2.6         1   4       0          47    0.3
3            615     1          2.5         0   0       0          10    0.8
4           1821     1          1.2         0  13       1          44    0.6

   mobile_wt  n_cores  …  px_height  px_width   ram  sc_h  sc_w  talk_time  \
0         81        7  …       1314      1854  2819    17    15          3
1        156        2  …        974      1385  3283    17     1         15
2        199        4  …        407       822  1433    11     5         20
3        131        6  …       1216      1786  2769    16     8         11
4        141        2  …       1208      1212  1411     8     2         15

   three_g  touch_screen  wifi  price_range
0        1             1     0            3
1        1             0     0            3
2        0             0     1            1
3        1             0     0            2
4        1             1     0            1

[5 rows x 21 columns]
```

[4]:

```python
# Dataframe summery
df.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
```

[5]:

```python
# total null value
df.isnull().sum()
```

[5]:

```
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
```

[6]:

```
x = df.drop('price_range',axis=1)
y = df.price_range
```

[7]:

```
x.head()
```

[7]:

```
   battery_power  blue  clock_speed  dual_sim  fc  four_g  int_memory  m_dep  \
0            772     0          1.1         1  12       0          39    0.8
1           1709     1          2.1         0   1       0          13    1.0
2           1949     0          2.6         1   4       0          47    0.3
3            615     1          2.5         0   0       0          10    0.8
4           1821     1          1.2         0  13       1          44    0.6

   mobile_wt  n_cores  pc  px_height  px_width   ram  sc_h  sc_w  talk_time  \
0         81        7  14       1314      1854  2819    17    15          3
1        156        2   2        974      1385  3283    17     1         15
2        199        4   7        407       822  1433    11     5         20
3        131        6   9       1216      1786  2769    16     8         11
4        141        2  14       1208      1212  1411     8     2         15

   three_g  touch_screen  wifi
0        1             1     0
1        1             0     0
2        0             0     1
3        1             0     0
4        1             1     0
```

[8]:

```
y.head()
```

[8]:

```
0    3
1    3
2    1
3    2
4    1
Name: price_range, dtype: int64
```

[9]:

```
y.value_counts()
```

[9]:

```
3    501
1    501
0    500
```

```
2    498
Name: price_range, dtype: int64
```

[10]:

```
# Splitting dataframe
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size=.3,random_state=43)
```

[11]:

```
from sklearn.tree import DecisionTreeClassifier
```

0.0.1 Checking Fittings

[12]:

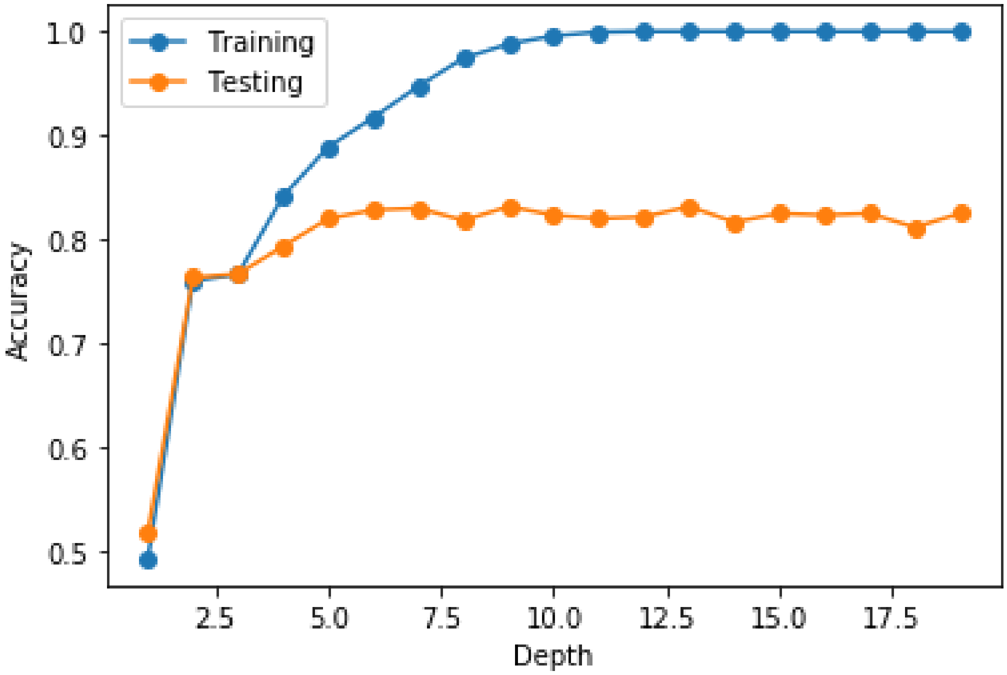
```
acc_training = []
acc_testing = []
depth = [ n for n in range(1,20)]
for i in depth:
    clf = DecisionTreeClassifier(max_depth = i)
    clf.fit(xtrain,ytrain)
    score_train = clf.score(xtrain,ytrain)
    acc_training.append(score_train)
    score_test = clf.score(xtest,ytest)
    acc_testing.append(score_test)

    print('Depth = %d , Train = %f , Test = %f ' % (i,score_train,score_test) )
```

```
Depth = 1 , Train = 0.492857 , Test = 0.518333
Depth = 2 , Train = 0.760000 , Test = 0.763333
Depth = 3 , Train = 0.766429 , Test = 0.766667
Depth = 4 , Train = 0.841429 , Test = 0.793333
Depth = 5 , Train = 0.888571 , Test = 0.820000
Depth = 6 , Train = 0.917143 , Test = 0.828333
Depth = 7 , Train = 0.947143 , Test = 0.830000
Depth = 8 , Train = 0.974286 , Test = 0.818333
Depth = 9 , Train = 0.987857 , Test = 0.831667
Depth = 10 , Train = 0.995714 , Test = 0.823333
Depth = 11 , Train = 0.999286 , Test = 0.820000
Depth = 12 , Train = 1.000000 , Test = 0.821667
Depth = 13 , Train = 1.000000 , Test = 0.831667
Depth = 14 , Train = 1.000000 , Test = 0.816667
Depth = 15 , Train = 1.000000 , Test = 0.825000
Depth = 16 , Train = 1.000000 , Test = 0.823333
Depth = 17 , Train = 1.000000 , Test = 0.825000
Depth = 18 , Train = 1.000000 , Test = 0.811667
Depth = 19 , Train = 1.000000 , Test = 0.825000
```

[13]:

```
plt.plot(depth,acc_training,"-o",label='Training')
plt.plot(depth,acc_testing,'-o',label='Testing')
plt.xlabel('Depth')
plt.ylabel('Accuracy')
plt.legend()
```

```
[13]: <matplotlib.legend.Legend at 0x284670246c8>
```



From the plot, we can see all are going smoothly and not having much difference between them, so they are good fitting

0.1 Feature Selection

0.1.1 Extra Trees Classifier

```
[14]: from sklearn.ensemble import ExtraTreesClassifier
      etc = ExtraTreesClassifier()
```

```
[15]: etc.fit(x,y)
```

```
[15]: ExtraTreesClassifier()
```

```
[16]: # Gini Feature importance
      feature_importance = etc.feature_importances_
      feature_importance #info gain
```

```
[16]: array([0.06180858, 0.01968337, 0.03293631, 0.01981819, 0.03189973,
             0.0174042 , 0.03388182, 0.03347493, 0.03604542, 0.03242053,
             0.03242006, 0.04742122, 0.04848003, 0.39814073, 0.0338445 ,
             0.034256  , 0.03390626, 0.01479553, 0.01786608, 0.01949652])
```

[17]:

```
imp = pd.DataFrame(feature_importance, columns=['Gain_Score'])
cols = pd.DataFrame(x.columns, columns=['Feature_Names'])
gains_x = pd.concat([cols,imp],axis=1)
gains_x
```

[17]:

```
    Feature_Names  Gain_Score
0   battery_power    0.061809
1            blue    0.019683
2     clock_speed    0.032936
3        dual_sim    0.019818
4              fc    0.031900
5          four_g    0.017404
6      int_memory    0.033882
7           m_dep    0.033475
8       mobile_wt    0.036045
9         n_cores    0.032421
10             pc    0.032420
11      px_height    0.047421
12       px_width    0.048480
13            ram    0.398141
14           sc_h    0.033845
15           sc_w    0.034256
16      talk_time    0.033906
17        three_g    0.014796
18   touch_screen    0.017866
19           wifi    0.019497
```

[18]:

```
gains_x.nlargest(10,'Gain_Score')
```

[18]:

```
    Feature_Names  Gain_Score
13            ram    0.398141
0   battery_power    0.061809
12       px_width    0.048480
11      px_height    0.047421
8       mobile_wt    0.036045
15           sc_w    0.034256
16      talk_time    0.033906
6      int_memory    0.033882
14           sc_h    0.033845
7           m_dep    0.033475
```

[19]:

```
gains_x.nsmallest(10,'Gain_Score')
```

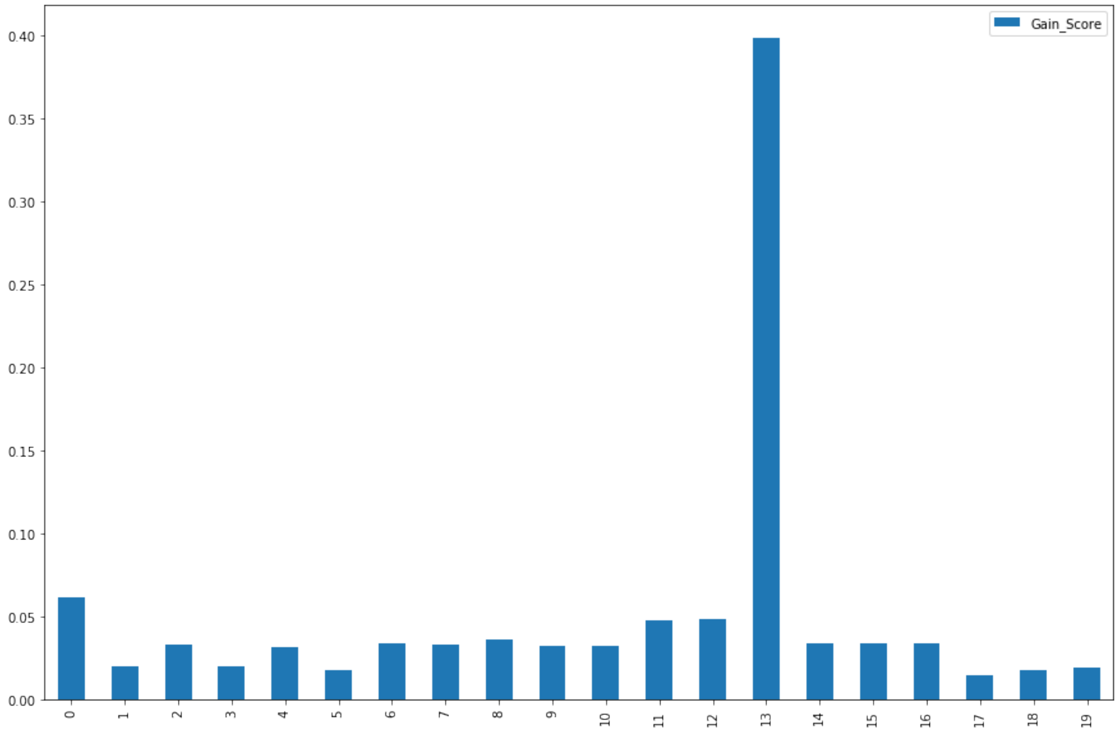
[19]:

```
   Feature_Names  Gain_Score
17       three_g    0.014796
5         four_g    0.017404
18  touch_screen    0.017866
```

```
19          wifi    0.019497
1           blue    0.019683
3       dual_sim    0.019818
4             fc    0.031900
10            pc    0.032420
9        n_cores    0.032421
2    clock_speed    0.032936
```

[21]:
```
gains_x.plot(kind='bar',figsize=(15,10))
```

[21]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x28468327508>
```



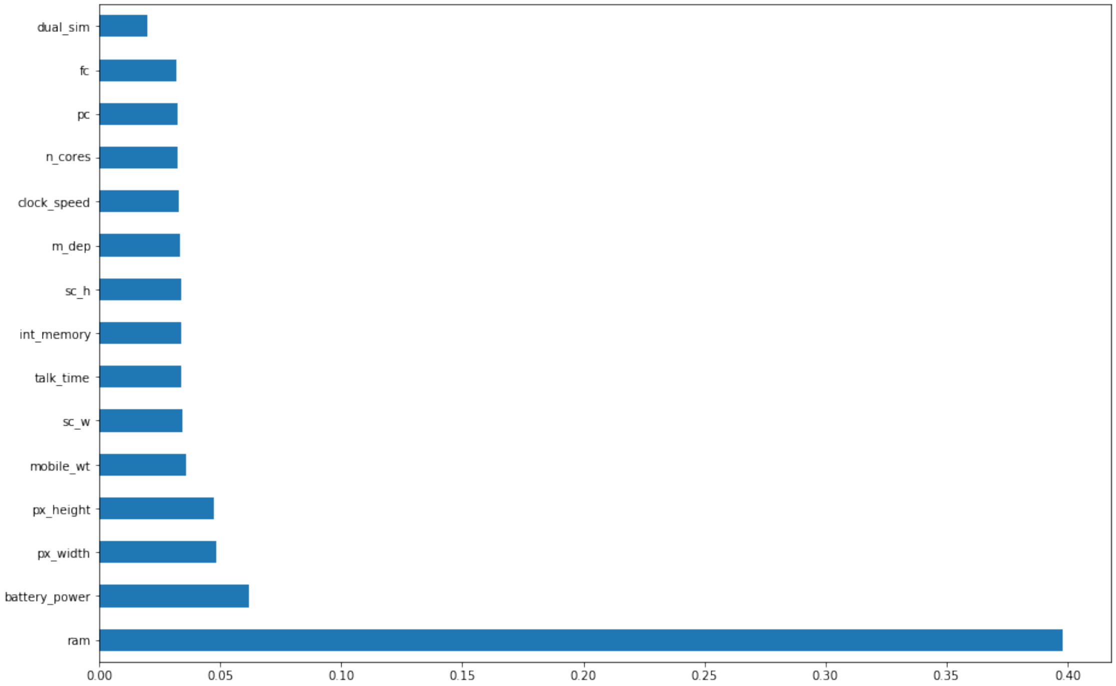
[98]:
```
features = pd.Series(etc.feature_importances_, index = x.columns)
features
```

[98]:
```
battery_power    0.061809
blue             0.019683
clock_speed      0.032936
dual_sim         0.019818
fc               0.031900
four_g           0.017404
int_memory       0.033882
m_dep            0.033475
mobile_wt        0.036045
```

```
n_cores          0.032421
pc               0.032420
px_height        0.047421
px_width         0.048480
ram              0.398141
sc_h             0.033845
sc_w             0.034256
talk_time        0.033906
three_g          0.014796
touch_screen     0.017866
wifi             0.019497
dtype: float64
```

[23]:
```
features.nlargest(15).plot(kind='barh',figsize=(15,10))
#plt.savefig('score1.png')
```

[23]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x28468720788>
```



[24]:
```
# Feature correlations
corr = x.corr()
corr
```

[24]:
```
               battery_power      blue  clock_speed  dual_sim        fc  \
battery_power       1.000000  0.011792     0.012088 -0.041499  0.031728
blue                0.011792  1.000000     0.022208  0.033198  0.004421
```

```
clock_speed         0.012088  0.022208     1.000000 -0.000013 -0.002228
dual_sim           -0.041499  0.033198    -0.000013  1.000000 -0.027660
fc                  0.031728  0.004421    -0.002228 -0.027660  1.000000
four_g              0.016252  0.010430    -0.040497  0.002204 -0.016691
int_memory         -0.003256  0.040035     0.008211 -0.016619 -0.028369
m_dep               0.035329  0.004904    -0.013967 -0.024364 -0.001180
mobile_wt           0.003405 -0.008509     0.013040 -0.008833  0.023103
n_cores            -0.029628  0.036176    -0.005288 -0.025111 -0.014089
pc                  0.029055 -0.009759    -0.006902 -0.016035  0.644736
px_height           0.015460 -0.008042    -0.012916 -0.023092 -0.010013
px_width           -0.006547 -0.043692    -0.006062  0.011648 -0.005447
ram                -0.001211  0.025296     0.003320  0.041313  0.015840
sc_h               -0.029862 -0.002829    -0.028834 -0.012072 -0.009773
sc_w               -0.020972  0.002223    -0.008453 -0.014825 -0.011747
talk_time           0.050825  0.015683    -0.014586 -0.037682 -0.005679
three_g             0.011937 -0.032583    -0.044436 -0.014008  0.002206
touch_screen       -0.011438  0.009071     0.019796 -0.018137 -0.013414
wifi               -0.008686 -0.019863    -0.025748  0.022740  0.018552

                 four_g  int_memory     m_dep  mobile_wt   n_cores        pc  \
battery_power  0.016252   -0.003256  0.035329   0.003405 -0.029628  0.029055
blue           0.010430    0.040035  0.004904  -0.008509  0.036176 -0.009759
clock_speed   -0.040497    0.008211 -0.013967   0.013040 -0.005288 -0.006902
dual_sim       0.002204   -0.016619 -0.024364  -0.008833 -0.025111 -0.016035
fc            -0.016691   -0.028369 -0.001180   0.023103 -0.014089  0.644736
four_g         1.000000    0.006831 -0.004381  -0.015238 -0.030379 -0.005887
int_memory     0.006831    1.000000  0.006267  -0.033450 -0.028415 -0.033384
m_dep         -0.004381    0.006267  1.000000   0.021180 -0.002929  0.026722
mobile_wt     -0.015238   -0.033450  0.021180   1.000000 -0.018178  0.018626
n_cores       -0.030379   -0.028415 -0.002929  -0.018178  1.000000 -0.002329
pc            -0.005887   -0.033384  0.026722   0.018626 -0.002329  1.000000
px_height     -0.021476    0.008719  0.025173   0.001784 -0.007519 -0.018958
px_width       0.005709   -0.010383  0.022626   0.001767  0.024629  0.003140
ram            0.007835    0.032136 -0.010876  -0.003159  0.004643  0.030231
sc_h           0.025434    0.037661 -0.024976  -0.033877  0.000039  0.005393
sc_w           0.037128    0.013886 -0.017654  -0.021301  0.026433 -0.023592
talk_time     -0.045850   -0.001618  0.017614   0.004002  0.013272  0.016714
three_g        0.583661   -0.010301 -0.014169   0.002776 -0.015518 -0.001586
touch_screen   0.014719   -0.028666 -0.003156  -0.014787  0.023113 -0.007426
wifi          -0.016604    0.007938 -0.026069  -0.000497 -0.009535  0.004197

               px_height  px_width       ram      sc_h      sc_w  talk_time  \
battery_power   0.015460 -0.006547 -0.001211 -0.029862 -0.020972   0.050825
blue           -0.008042 -0.043692  0.025296 -0.002829  0.002223   0.015683
clock_speed    -0.012916 -0.006062  0.003320 -0.028834 -0.008453  -0.014586
dual_sim       -0.023092  0.011648  0.041313 -0.012072 -0.014825  -0.037682
fc             -0.010013 -0.005447  0.015840 -0.009773 -0.011747  -0.005679
```

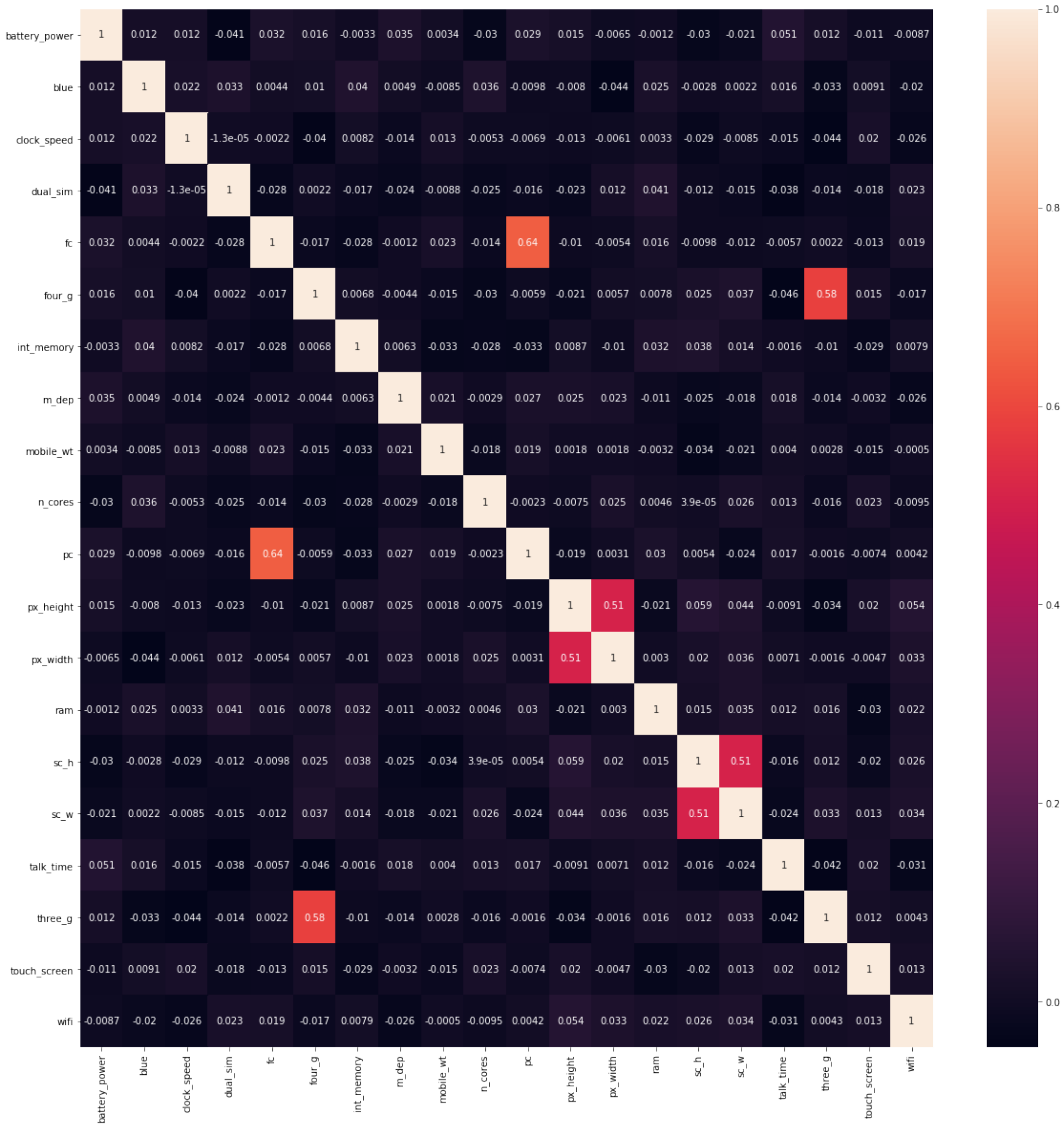
```
four_g        -0.021476  0.005709  0.007835  0.025434  0.037128  -0.045850
int_memory     0.008719 -0.010383  0.032136  0.037661  0.013886  -0.001618
m_dep          0.025173  0.022626 -0.010876 -0.024976 -0.017654   0.017614
mobile_wt      0.001784  0.001767 -0.003159 -0.033877 -0.021301   0.004002
n_cores       -0.007519  0.024629  0.004643  0.000039  0.026433   0.013272
pc            -0.018958  0.003140  0.030231  0.005393 -0.023592   0.016714
px_height      1.000000  0.509613 -0.020823  0.059052  0.043935  -0.009059
px_width       0.509613  1.000000  0.003001  0.020039  0.035830   0.007063
ram           -0.020823  0.003001  1.000000  0.015294  0.034836   0.011741
sc_h           0.059052  0.020039  0.015294  1.000000  0.507638  -0.016224
sc_w           0.043935  0.035830  0.034836  0.507638  1.000000  -0.023888
talk_time     -0.009059  0.007063  0.011741 -0.016224 -0.023888   1.000000
three_g       -0.033655 -0.001594  0.016053  0.011672  0.032786  -0.041589
touch_screen   0.020292 -0.004742 -0.029985 -0.020450  0.013447   0.020197
wifi           0.054076  0.033064  0.022397  0.026037  0.033550  -0.031255

                three_g  touch_screen      wifi
battery_power  0.011937     -0.011438 -0.008686
blue          -0.032583      0.009071 -0.019863
clock_speed   -0.044436      0.019796 -0.025748
dual_sim      -0.014008     -0.018137  0.022740
fc             0.002206     -0.013414  0.018552
four_g         0.583661      0.014719 -0.016604
int_memory    -0.010301     -0.028666  0.007938
m_dep         -0.014169     -0.003156 -0.026069
mobile_wt      0.002776     -0.014787 -0.000497
n_cores       -0.015518      0.023113 -0.009535
pc            -0.001586     -0.007426  0.004197
px_height     -0.033655      0.020292  0.054076
px_width      -0.001594     -0.004742  0.033064
ram            0.016053     -0.029985  0.022397
sc_h           0.011672     -0.020450  0.026037
sc_w           0.032786      0.013447  0.033550
talk_time     -0.041589      0.020197 -0.031255
three_g        1.000000      0.012131  0.004316
touch_screen   0.012131      1.000000  0.012904
wifi           0.004316      0.012904  1.000000
```

[25]:

```
# feature_names = x.columns
# feature_names = x.corr().index

plt.figure(figsize=(20,20))
sns.heatmap(corr,annot=True)
```

[25]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x28468469488>
```



0.1.2 Select K-Best

[26]:
```
# chi2 for non-negative feature
# f_regession for regression
# f_classif for classification set
from sklearn.feature_selection import SelectKBest, f_regression, f_classif, chi2
skb = SelectKBest(score_func = f_classif)
```

[27]:
```
skb.fit(x,y)
```

[27]:
```
SelectKBest()
```

[29]:

```
# Scores of features
feature_importance = skb.scores_
feature_importance
```

[29]:

```
array([3.11908732e+01, 4.59678478e-01, 6.13019154e-01, 4.89492949e-01,
       8.30396133e-01, 1.16710790e+00, 2.96575777e+00, 1.64410175e+00,
       3.62521255e+00, 2.58284723e+00, 8.74295515e-01, 1.95473713e+01,
       2.25200970e+01, 3.52623236e+03, 2.20350431e+00, 1.59731005e+00,
       1.66657575e+00, 4.39621377e-01, 1.45891088e+00, 2.61677203e-01])
```

[30]:

```
imp = pd.DataFrame(feature_importance,columns=['Gain Score'])
gains_kb = pd.concat([cols,imp],axis=1)
gains_kb
```

[30]:

```
   Feature_Names   Gain Score
0  battery_power    31.190873
1           blue     0.459678
2    clock_speed     0.613019
3       dual_sim     0.489493
4             fc     0.830396
5         four_g     1.167108
6     int_memory     2.965758
7          m_dep     1.644102
8      mobile_wt     3.625213
9        n_cores     2.582847
10            pc     0.874296
11     px_height    19.547371
12      px_width    22.520097
13           ram  3526.232362
14          sc_h     2.203504
15          sc_w     1.597310
16     talk_time     1.666576
17       three_g     0.439621
18  touch_screen     1.458911
19          wifi     0.261677
```

[31]:

```
gains_kb.nlargest(10,'Gain Score')
```

[31]:

```
   Feature_Names   Gain Score
13           ram  3526.232362
0  battery_power    31.190873
12      px_width    22.520097
11     px_height    19.547371
8      mobile_wt     3.625213
6     int_memory     2.965758
9        n_cores     2.582847
14          sc_h     2.203504
```

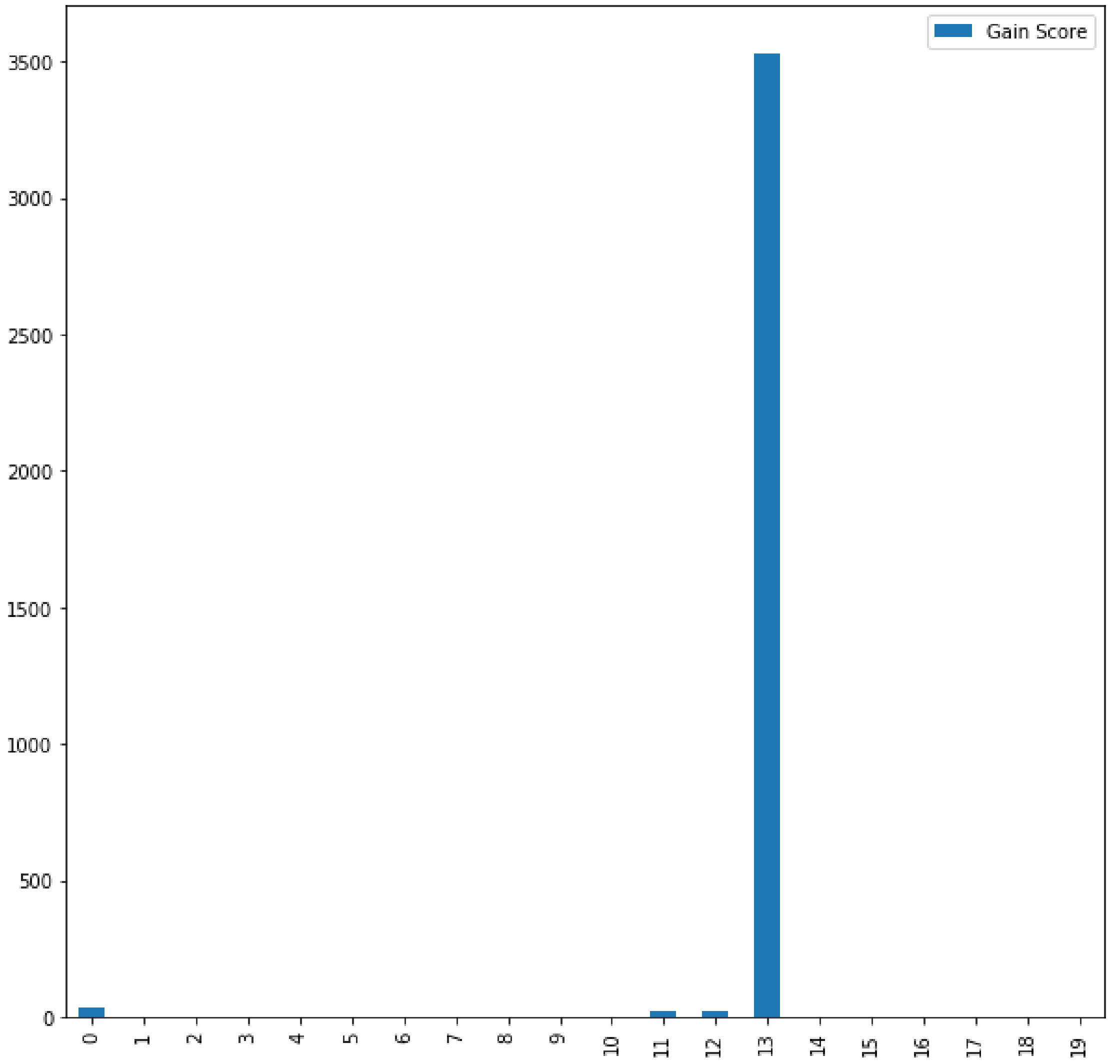
```
16       talk_time     1.666576
7            m_dep     1.644102
```

```
[32]: gains_kb.nsmallest(10,'Gain Score')
```

```
[32]:    Feature_Names  Gain Score
      19          wifi    0.261677
      17       three_g    0.439621
      1           blue    0.459678
      3       dual_sim    0.489493
      2    clock_speed    0.613019
      4             fc    0.830396
      10            pc    0.874296
      5         four_g    1.167108
      18  touch_screen    1.458911
      15          sc_w    1.597310
```

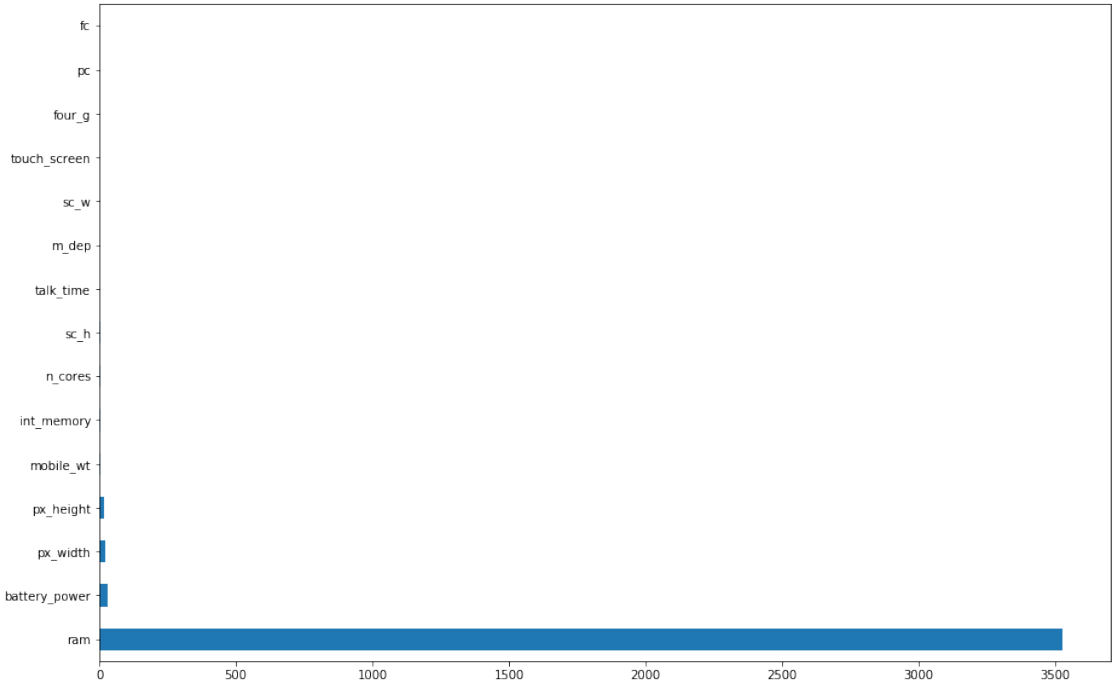
```
[33]: gains_kb.plot(kind='bar',figsize=(10,10))
```

```
[33]: <matplotlib.axes._subplots.AxesSubplot at 0x284695b1d48>
```



```
[34]: features = pd.Series(skb.scores_, index = x.columns)
      features.nlargest(15).plot(kind='barh',figsize=(15,10))
      #plt.savefig('score1.png')
```

```
[34]: <matplotlib.axes._subplots.AxesSubplot at 0x28468c7d8c8>
```



0.2 Principal Component Analysis (PCA)

[35]:
```
# Scaling Dataframe
from sklearn.preprocessing import MinMaxScaler
mms = MinMaxScaler()
```

[38]:
```
x_scaled = mms.fit_transform(x)
```

[39]:
```
x_scaled.shape
```

[39]:
```
(2000, 20)
```

[40]:
```
x.shape
```

[40]:
```
(2000, 20)
```

[41]:
```
# Applying PCA
from sklearn.decomposition import PCA
pca = PCA(n_components=3)
```

[42]:
```
x_pca = pca.fit_transform(x_scaled)
```

[43]:
```
x_pca
```

```
[43]: array([[ 0.20723861,  0.09565505, -0.00565893],
              [ 0.28947651, -0.00206982,  0.80169178],
              [ 0.93249226, -0.12077773, -0.69947386],
              …,
              [-0.56557579,  0.03097499,  0.06335067],
              [-0.56346383,  0.76939573, -0.43350877],
              [-0.51957598, -0.66046955, -0.37689763]])
```

```
[44]: features = pd.DataFrame(x_pca,columns=['pca0','pca1','pca2'])
      features
```

```
[44]:           pca0      pca1      pca2
      0     0.207239  0.095655 -0.005659
      1     0.289477 -0.002070  0.801692
      2     0.932492 -0.120778 -0.699474
      3     0.298751 -0.006629  0.774752
      4    -0.540990  0.169750  0.815691
      …          …         …         …
      1995 -0.519601 -0.543636  0.462132
      1996  0.293092 -0.554112 -0.381401
      1997 -0.565576  0.030975  0.063351
      1998 -0.563464  0.769396 -0.433509
      1999 -0.519576 -0.660470 -0.376898

      [2000 rows x 3 columns]
```

```
[45]: pip install plotly
```

```
Requirement already satisfied: plotly in d:\anaconda3\lib\site-packages (5.9.0)
Requirement already satisfied: tenacity>=6.2.0 in d:\anaconda3\lib\site-packages
(from plotly) (8.0.1)
Note: you may need to restart the kernel to use updated packages.
```

```
[46]: import plotly.express as px
      px.scatter_3d(features,x='pca0',y='pca1',z='pca2')
```

1 KNN

1.1 KNN with ExtraTreeClassifier

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

```
[83]: # list of largest scored features
      features_x = list(gains_x.nlargest(15,'Gain_Score').Feature_Names)
```

```
[84]: x_x = df[features_x]
      x_x.shape
```

```
[84]: (2000, 15)
```

```
[85]: xtrain_x,xtest_x,ytrain_x,ytest_x = train_test_split(x_x,y,train_size=.
        ↪7,random_state=43)
```

```
[86]: knn.fit(xtrain_x,ytrain_x)
```

```
[86]: KNeighborsClassifier()
```

```
[87]: # Testing accuracy
      accuracy_x = knn.score(xtest_x,ytest)
      accuracy_x
```

```
[87]: 0.9166666666666666
```

1.2 KNN with Select k-Best

```
[69]: # list of largest scored features
      features_kb = list(gains_kb.nlargest(15,'Gain Score').Feature_Names)
      features_kb
```

```
[69]: ['ram',
       'battery_power',
       'px_width',
       'px_height',
       'mobile_wt',
       'int_memory',
       'n_cores',
       'sc_h',
       'talk_time',
       'm_dep',
       'sc_w',
       'touch_screen',
       'four_g',
       'pc',
       'fc']
```

```
[71]: x_kb = df[features_kb]
      x_kb.shape
```

```
[71]: (2000, 15)
```

```
[72]: xtrain_kb,xtest_kb,ytrain_kb,ytest_kb = train_test_split(x_kb,y,train_size=.
        ↪7,random_state=43)
```

[77]:
```
xtrain_kb.shape
```

[77]:
```
(1400, 15)
```

[78]:
```
xtest_kb.shape
```

[78]:
```
(600, 15)
```

[75]:
```
knn = KNeighborsClassifier()
knn.fit(xtrain_kb,ytrain_kb)
```

[75]:
```
KNeighborsClassifier()
```

[80]:
```
# Testing accuracy
accuracy_kb = knn.score(xtest_kb,ytest_kb)
accuracy_kb
```

[80]:
```
0.9166666666666666
```

1.3 KNN with Priciple Component Analysis (PCA)

[89]:
```
xtrain_pca, xtest_pca, ytrain_pca, ytest_pca = train_test_split(features_pca,␣
 ↪y, test_size=.3, random_state=43)
```

[91]:
```
xtrain_pca.shape
```

[91]:
```
(1400, 3)
```

[92]:
```
xtest.shape
```

[92]:
```
(600, 20)
```

[90]:
```
knn = KNeighborsClassifier()
knn.fit(xtrain_pca,ytrain_pca)
```

[90]:
```
KNeighborsClassifier()
```

[94]:
```
# Testing accuracy
accuracy_pca = knn.score(xtest_pca,ytest_pca)
accuracy_pca
```

[94]:
```
0.3
```

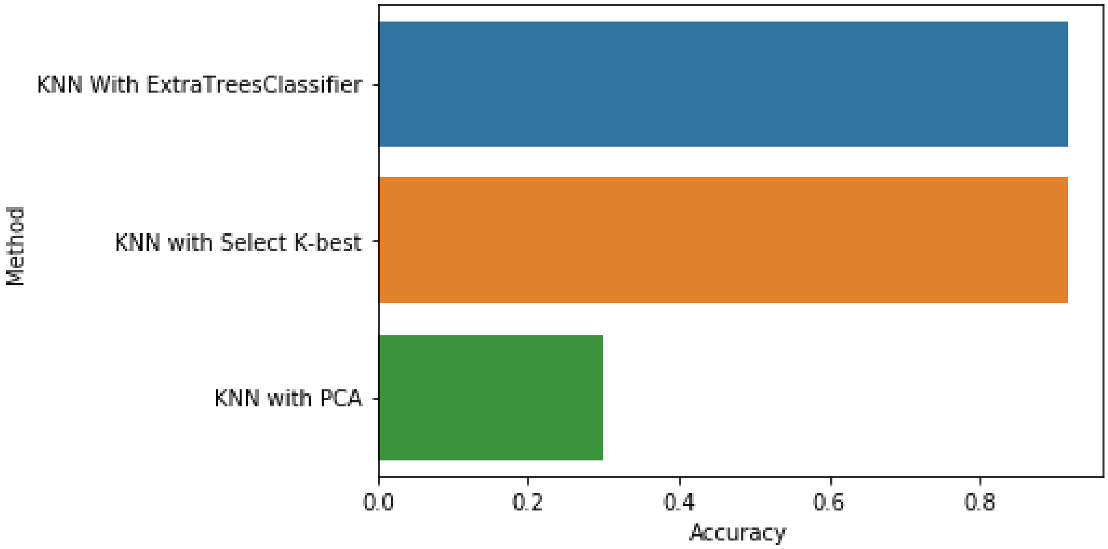
2 Comparison

```
[99]: df_comparison = pd.DataFrame([['KNN With␣
       ↪ExtraTreesClassifier',accuracy_x],['KNN with Select␣
       ↪K-best',accuracy_kb],['KNN with␣
       ↪PCA',accuracy_pca]],columns=['Method','Accuracy'])
      df_comparison
```

```
[99]:                          Method  Accuracy
      0  KNN With ExtraTreesClassifier  0.916667
      1         KNN with Select K-best  0.916667
      2                   KNN with PCA  0.300000
```

```
[101]: sns.barplot(x='Accuracy',y='Method',data=df_comparison)
```

```
[101]: <matplotlib.axes._subplots.AxesSubplot at 0x2846becbec8>
```


KNN With ExtraTreesClassifier
KNN with Select K-best
KNN with PCA
Method
Accuracy