EDA

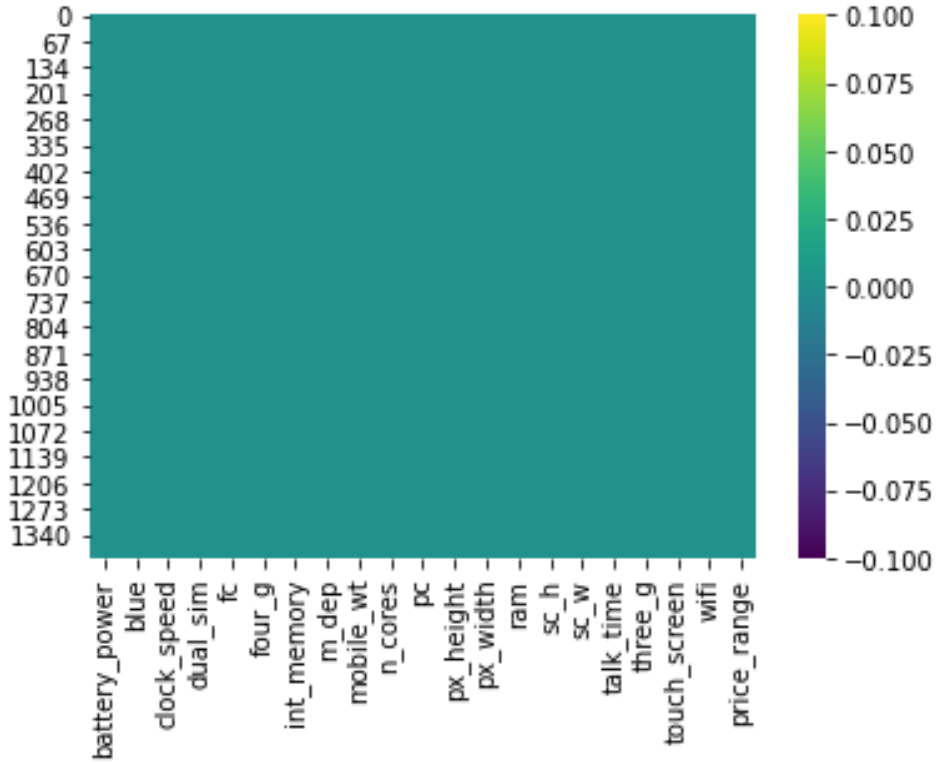
Data: mobile_data,csv

Libraries and data

```
# Jesus is my Saviour!
import os
os.chdir('C:\\Users\\Dr Vinod\\Desktop\\Diksha 12 sep21')
import pandas as pd
pd.set_option('display.max_column',None)
import warnings
warnings.filterwarnings('ignore')
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
import statsmodels.api as sm
from statsmodels.formula.api import ols
from statsmodels.stats.multicomp import pairwise_tukeyhsd
from scipy.stats import chi2, chi2_contingency


df = pd.read_csv("mobile_data.csv")
df = pd.DataFrame(df)
df.info()
```

Data

```
df.isnull().sum()

sns.heatmap(df.isnull(), cmap = 'viridis')
```

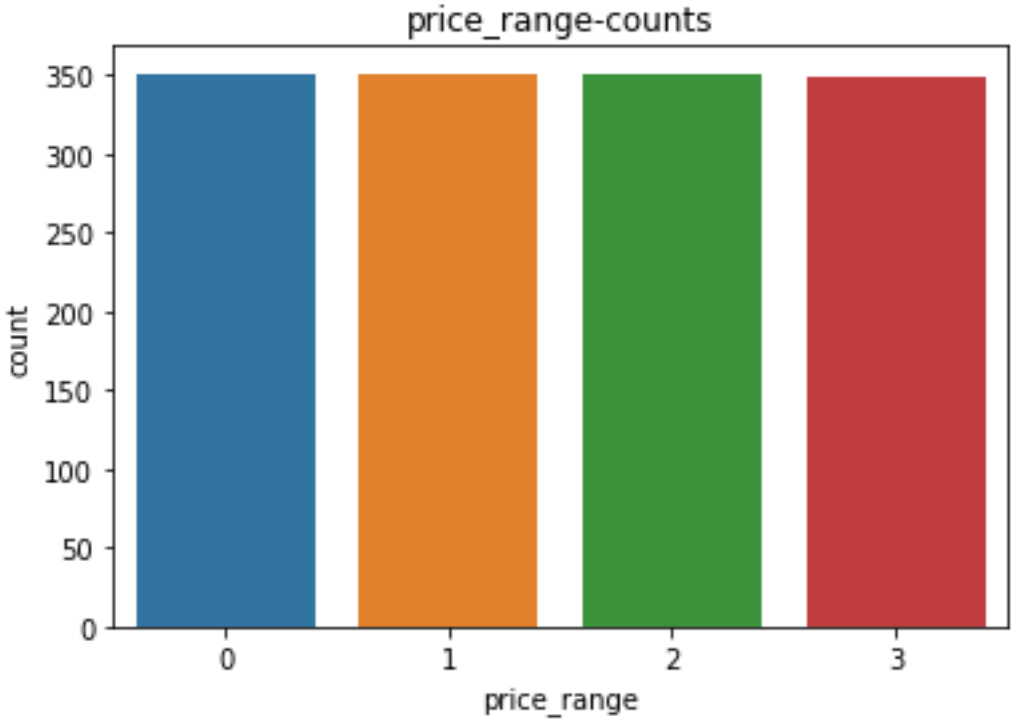



```
In [10]: df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1400 entries, 0 to 1399
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   battery_power   1400 non-null   int64
 1   blue            1400 non-null   int64
 2   clock_speed     1400 non-null   float64
 3   dual_sim        1400 non-null   int64
 4   fc              1400 non-null   int64
 5   four_g          1400 non-null   int64
 6   int_memory      1400 non-null   int64
 7   m_dep           1400 non-null   float64
 8   mobile_wt       1400 non-null   int64
 9   n_cores         1400 non-null   int64
 10  pc              1400 non-null   int64
 11  px_height       1400 non-null   int64
 12  px_width        1400 non-null   int64
 13  ram             1400 non-null   int64
 14  sc_h            1400 non-null   int64
 15  sc_w            1400 non-null   int64
 16  talk_time       1400 non-null   int64
 17  three_g         1400 non-null   int64
 18  touch_screen    1400 non-null   int64
 19  wifi            1400 non-null   int64
 20  price_range     1400 non-null   int64
dtypes: float64(2), int64(19)
memory usage: 229.8 KB
```

Target Variable: price_range

```
In [4]: df.price_range.value_counts()
Out[4]:
2    351
1    350
0    350
3    349
Name: price_range, dtype: int64
```

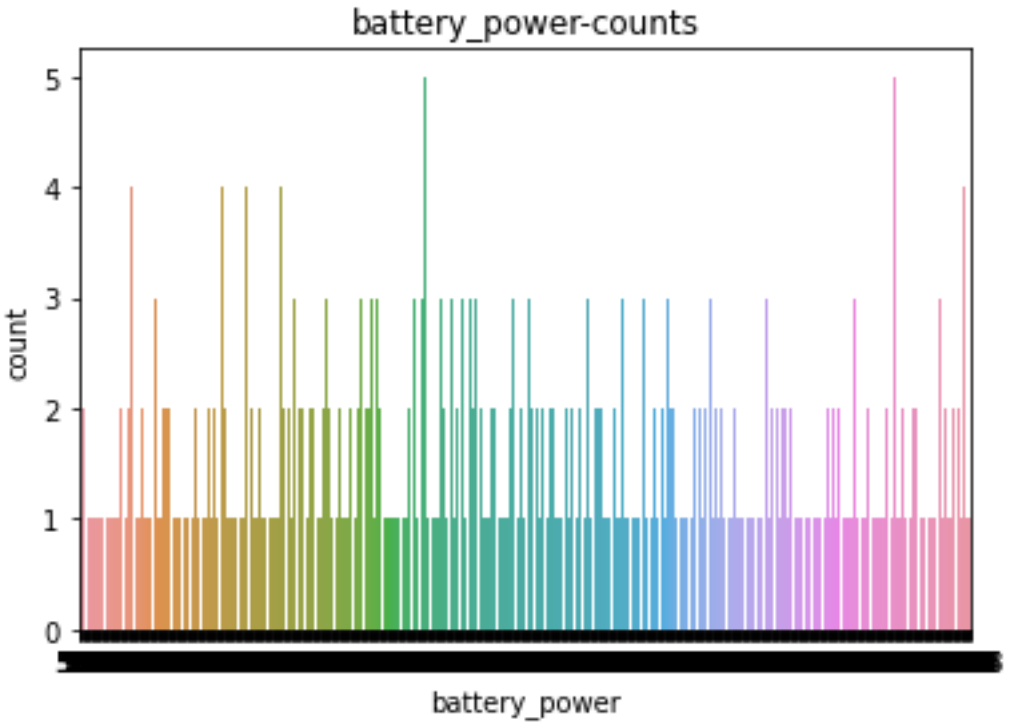
```
In [9]: df.price_range.value_counts().sum()
Out[9]: 1400
```



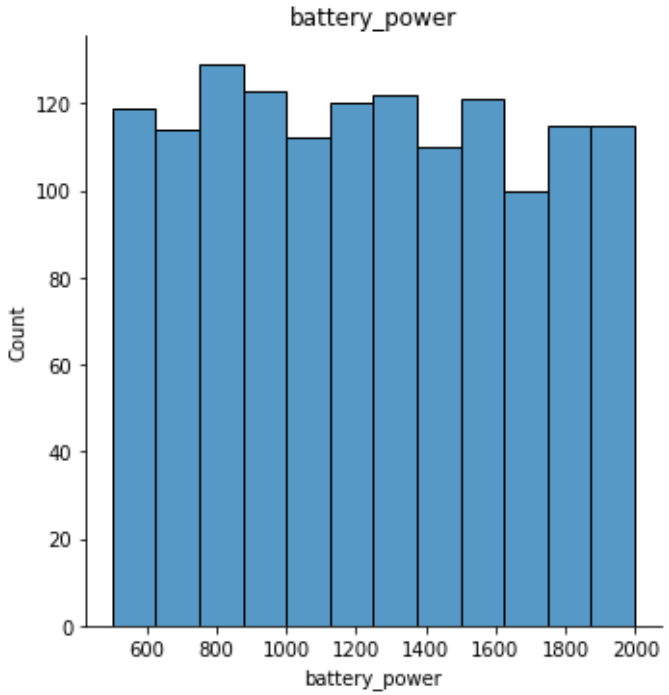
Battery_power

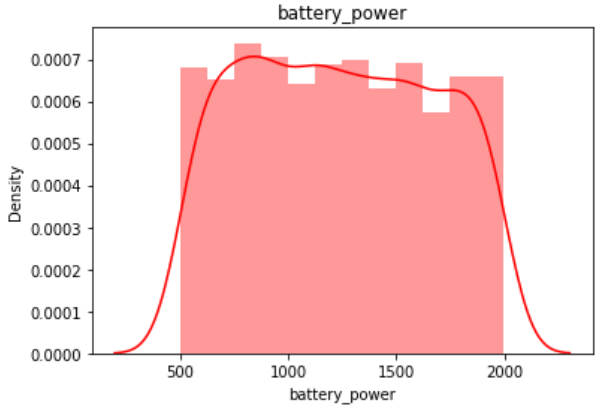
```
In [11]: df.battery_power.value_counts()
Out[11]:
1414    5
1083    5
1589    5
1872    5
930     4
       ..
1404    1
1402    1
1398    1
1397    1
501     1
Name: battery_power, Length: 911, dtype: int64
```

```
In [13]: df.battery_power.value_counts().sum()
Out[13]: 1400
```

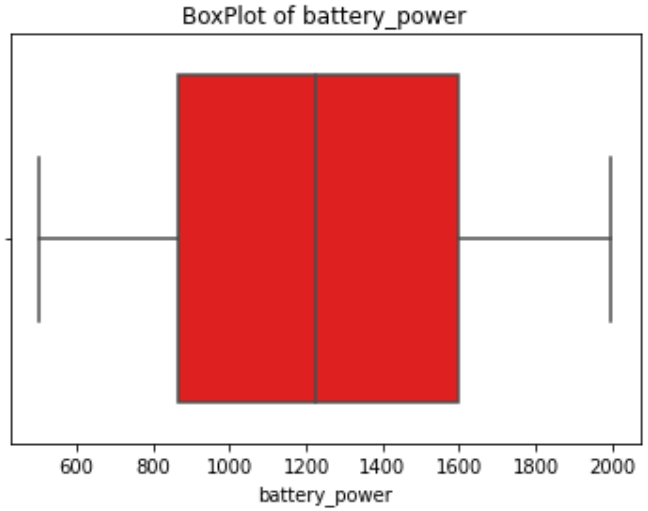


```
sns.displot(df.battery_power)
plt.title('battery_power')

sns.distplot(df.battery_power, color= 'r')
plt.title('battery_power')
```






```
sns.boxplot(df.battery_power, color = 'r')
plt.title('BoxPlot of battery_power')
```

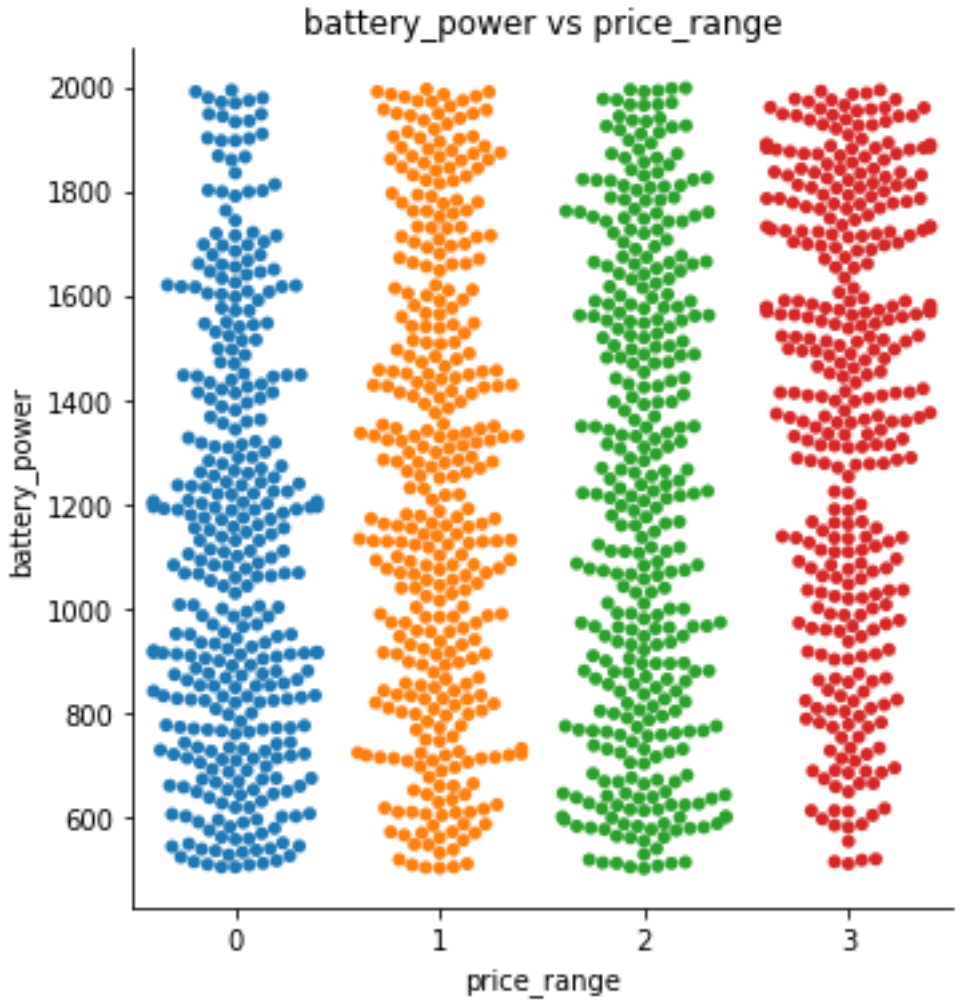


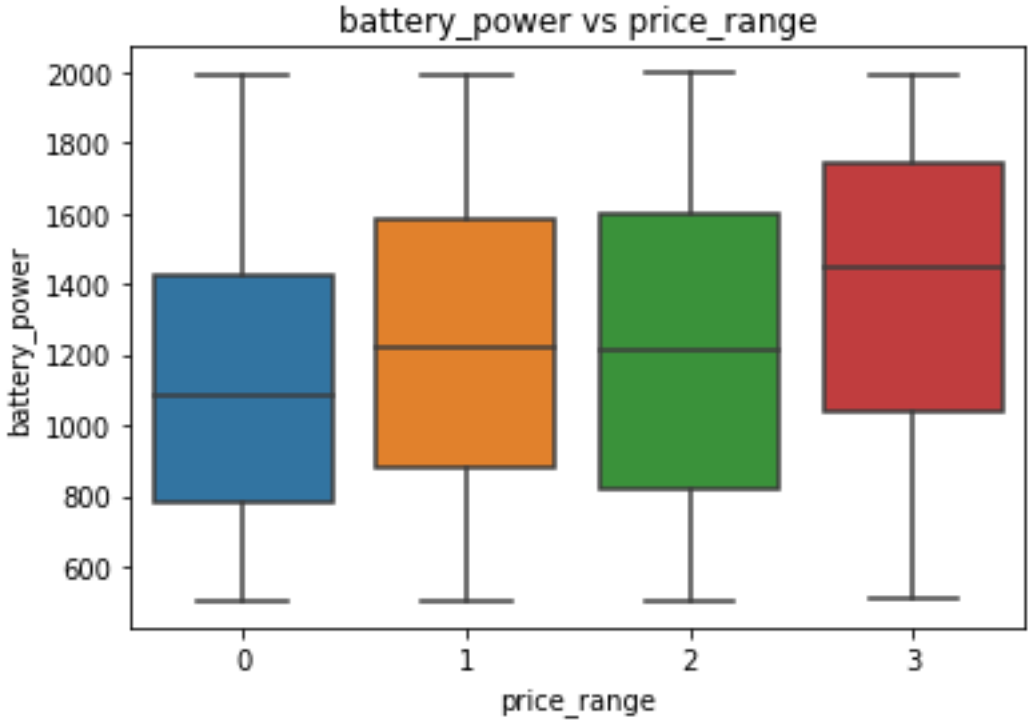
battery_power

```
In [27]: df.battery_power.groupby(df.price_range).describe()
Out[27]:
               count         mean         std    min      25%     50%      75%  \
price_range
0              350.0  1121.445714  403.061068  504.0   777.75  1085.0  1423.75
1              350.0  1238.691429  428.145460  502.0   880.50  1217.5  1587.75
2              351.0  1218.108262  443.311918  501.0   819.50  1216.0  1598.00
3              349.0  1380.699140  412.939809  510.0  1035.00  1444.0  1745.00

                 max
price_range
0             1994.0
1             1996.0
2             1998.0
3             1994.0
```

```
In [28]: df.battery_power.groupby(df.price_range).mean()
Out[28]:
price_range
0    1121.445714
1    1238.691429
2    1218.108262
3    1380.699140
Name: battery_power, dtype: float64
```

```
sns.catplot(x="price_range", y="battery_power", kind="swarm", data=df)
plt.title('battery_power vs price_range')

sns.boxplot(x="price_range", y="battery_power", data = df)
plt.title('battery_power vs price_range')
```





https://seaborn.pydata.org/tutorial/categorical.html

battery_power is a good predictor?

```
In [38]: mod = ols('battery_power ~ price_range', data = df).fit()

In [39]: sm.stats.anova_lm(mod)
Out[39]:
                df         sum_sq        mean_sq           F         PR(>F)
price_range    1.0   1.001085e+07   1.001085e+07   55.805844   1.403146e-13
Residual    1398.0   2.507832e+08   1.793871e+05         NaN            NaN

In [40]: from statsmodels.stats.multicomp import pairwise_tukeyhsd

In [41]: rslt = pairwise_tukeyhsd(df.battery_power, df.price_range, alpha =
0.05)

In [42]: print(rslt)
  Multiple Comparison of Means - Tukey HSD, FWER=0.05
====================================================
group1 group2 meandiff p-adj    lower    upper   reject
----------------------------------------------------
     0      1 117.2457 0.0014   35.1622 199.3292   True
     0      2  96.6625 0.0132   14.6376 178.6875   True
     0      3 259.2534  0.001  177.1112 341.3957   True
     1      2 -20.5832    0.9 -102.6082  61.4418  False
     1      3 142.0077  0.001   59.8655   224.15   True
     2      3 162.5909  0.001   80.5071 244.6747   True
----------------------------------------------------
```

Ho reject, there is difference

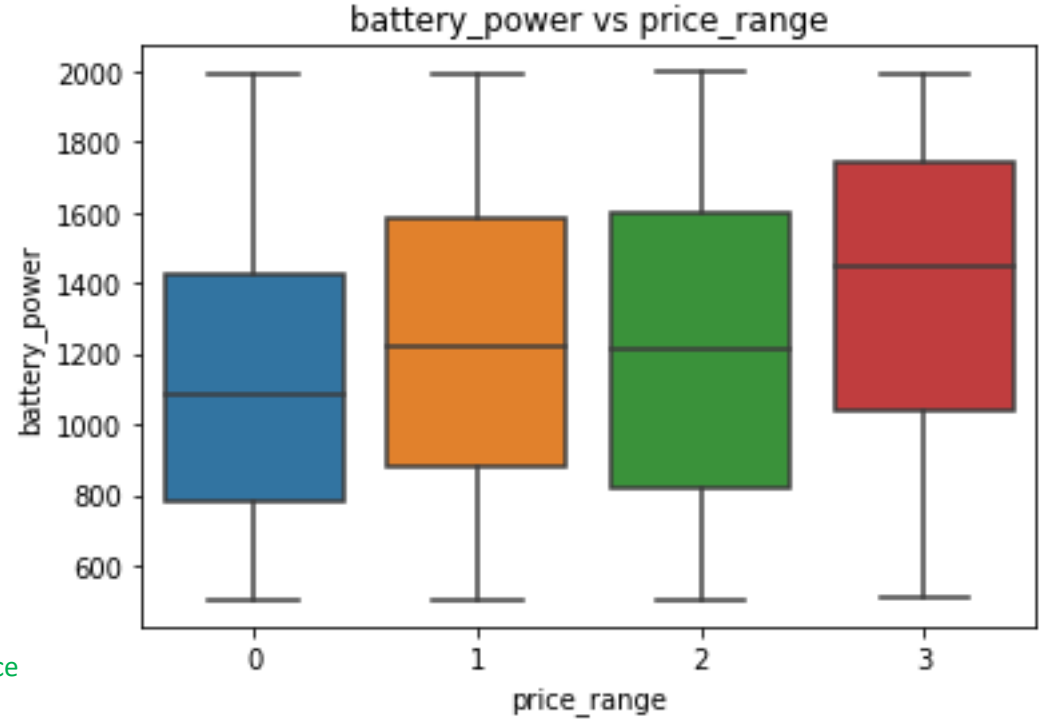
Ho reject, there is difference

Ho reject, there is difference

Ho accepted, there is NO difference

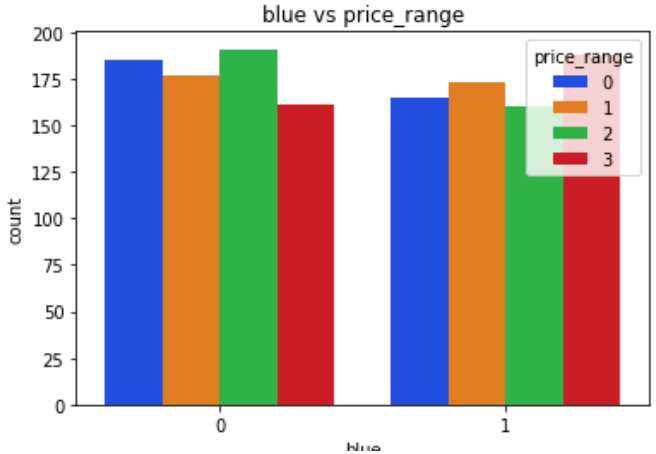
Ho reject, there is difference

Ho reject, there is difference



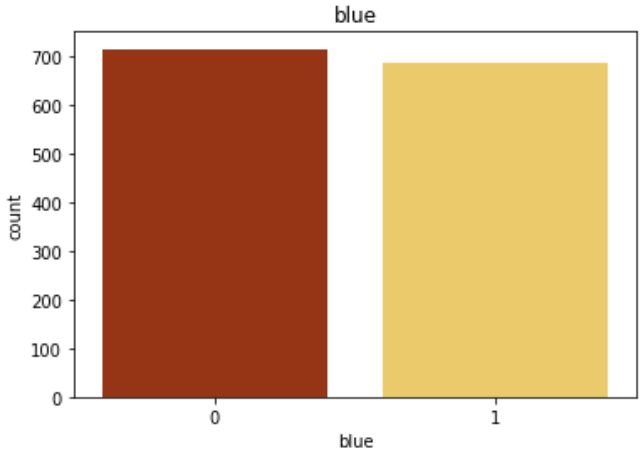
blue

```
sns.countplot(df.blue, hue = df.price_range, palette = 'bright')
plt.title('blue vs price_range')
```




```
In [6]: df.blue.value_counts()
Out[6]:
0    714
1    686
Name: blue, dtype: int64
```

```
#_____________3 blue
df.blue.value_counts()
sns.countplot(df.blue, palette = 'afmhot')
plt.title('blue')
df.blue.value_counts().sum() #1400
```

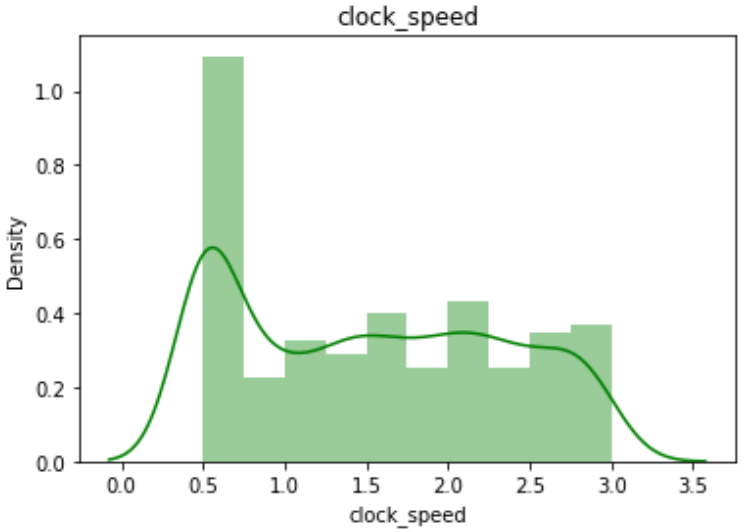


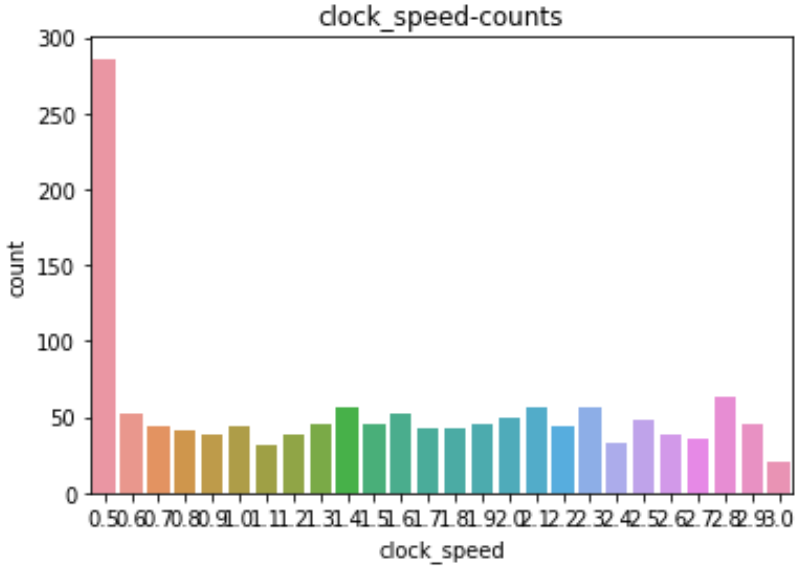

```
In [17]: from scipy.stats import chi2, chi2_contingency

In [18]: ct_blue = pd.crosstab(df.blue, df.price_range)

In [19]: ct_blue
Out[19]:
price_range    0    1    2    3
blue
0            185  177  191  161
1            165  173  160  188

In [20]: chi2_contingency(ct_blue, correction=False)
Out[20]:
(5.45747136991085,
 0.14120454392255236,
 3,
 array([[178.5 , 178.5 , 179.01, 177.99],
        [171.5 , 171.5 , 171.99, 171.01]]))
```

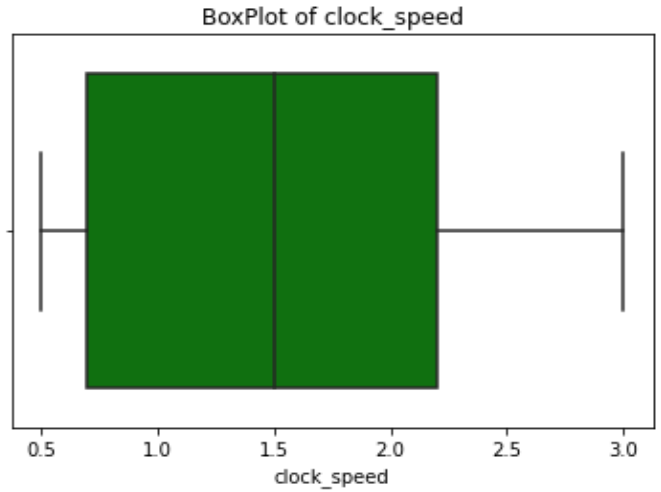
Not Good predictor!

Clock_speed

```
df.clock_speed.value_counts() # continuous
df.clock_speed.value_counts().sum() #1400
sns.countplot(df.clock_speed)
plt.title('clock_speed-counts')

sns.distplot(df.clock_speed, color= 'g')
plt.title('clock_speed')

sns.boxplot(df.clock_speed, color = 'g')
plt.title('BoxPlot of clock_speed')
```

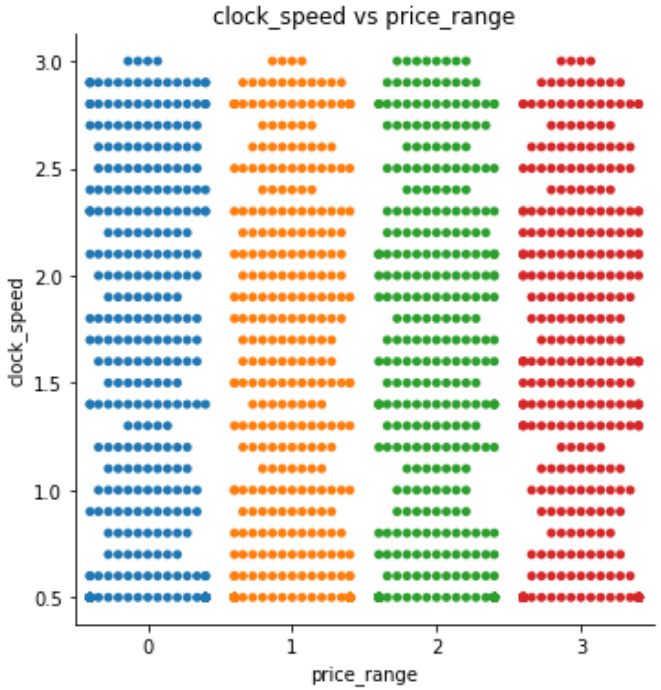


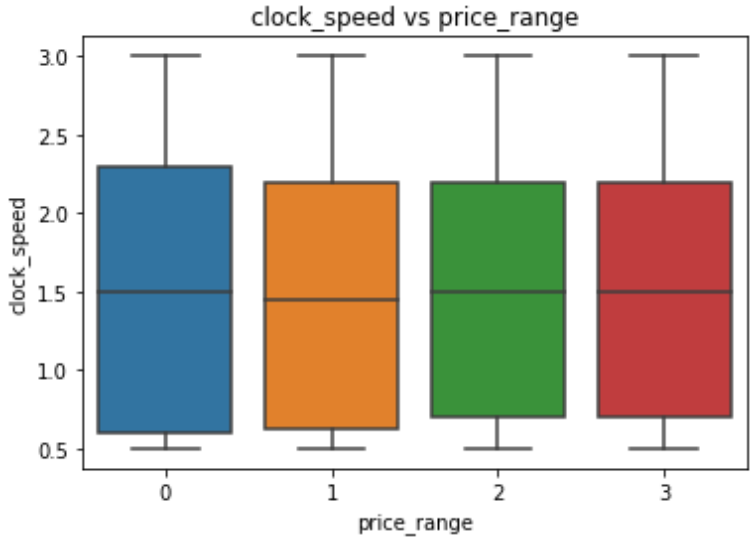




```
#_________groupby plot
sns.catplot(x="price_range", y="clock_speed", kind="swarm", data=df)
plt.title('clock_speed vs price_range')

sns.boxplot(x="price_range", y="clock_speed", data = df)
plt.title('clock_speed vs price_range')
```

```
In [34]: df.clock_speed.groupby(df.price_range).mean()
Out[34]:
price_range
0    1.543714
1    1.490571
2    1.537607
3    1.536676
Name: clock_speed, dtype: float64
```

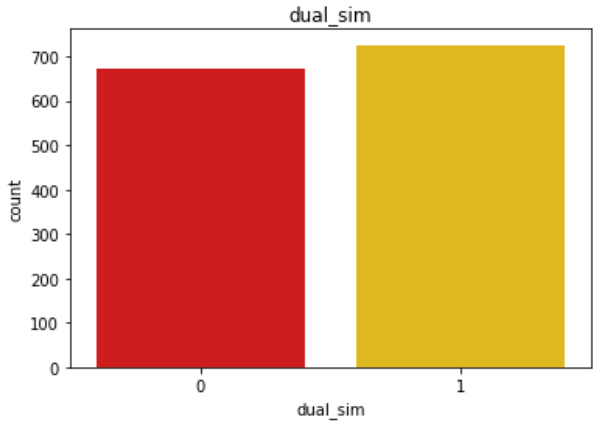


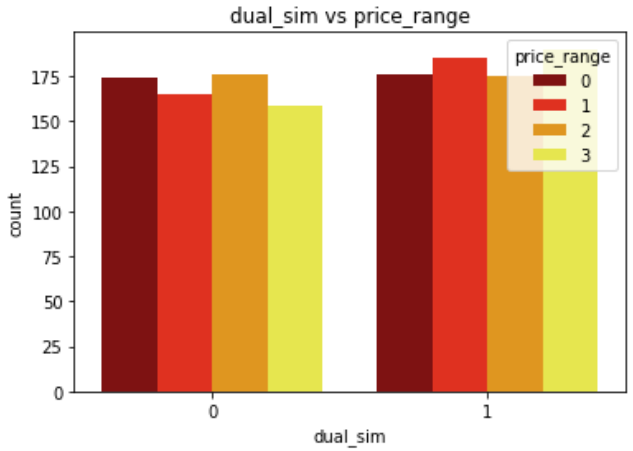



```
In [38]: from statsmodels.formula.api import ols

In [39]: mod = ols('clock_speed ~ price_range', data = df).fit()

In [40]: sm.stats.anova_lm(mod)
Out[40]:
                df      sum_sq   mean_sq         F    PR(>F)
price_range    1.0    0.011725  0.011725  0.017593  0.894498
Residual    1398.0  931.696847  0.666450       NaN       NaN

In [41]: from statsmodels.stats.multicomp import pairwise_tukeyhsd

In [42]: rslt = pairwise_tukeyhsd(df.clock_speed, df.price_range, alpha = 0.05)

In [43]: print(rslt)
Multiple Comparison of Means - Tukey HSD, FWER=0.05
=================================================
group1 group2 meandiff p-adj   lower  upper  reject
-------------------------------------------------
     0      1  -0.0531  0.802 -0.2119 0.1056  False
     0      2  -0.0061    0.9 -0.1648 0.1526  False
     0      3   -0.007    0.9 -0.1659 0.1519  False
     1      2    0.047 0.8572 -0.1116 0.2057  False
     1      3   0.0461 0.8663 -0.1128  0.205  False
     2      3  -0.0009    0.9 -0.1597 0.1579  False
-------------------------------------------------
```

dual_sim

```
In [54]: df.dual_sim.value_counts()
Out[54]:
1    726
0    674
Name: dual_sim, dtype: int64

In [55]: df.dual_sim.value_counts().sum() #1400
Out[55]: 1400
```





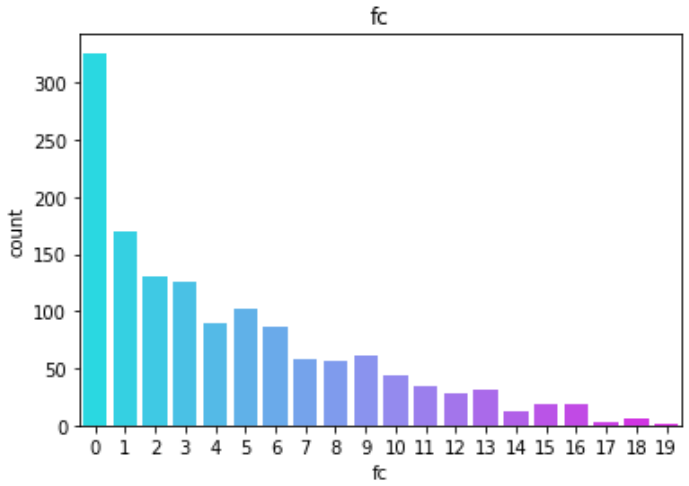

```
sns.countplot(df.dual_sim, palette = 'hot')
plt.title('dual_sim')

sns.countplot(df.dual_sim, hue = df.price_range, palette = 'hot')
plt.title('dual_sim vs price_range')
```

```
In [59]: from scipy.stats import chi2, chi2_contingency

In [60]: ct_dual_sim = pd.crosstab(df.dual_sim, df.price_range)

In [61]: ct_dual_sim
Out[61]:
price_range    0    1    2    3
dual_sim
0            174  165  176  159
1            176  185  175  190

In [62]: chi2_contingency(ct_dual_sim, correction=False)
Out[62]:
(1.9820221892297483,
 0.5761464389580644,
 3,
 array([[168.5       , 168.5       , 168.98142857, 168.01857143],
        [181.5       , 181.5       , 182.01857143, 180.98142857]]))
```

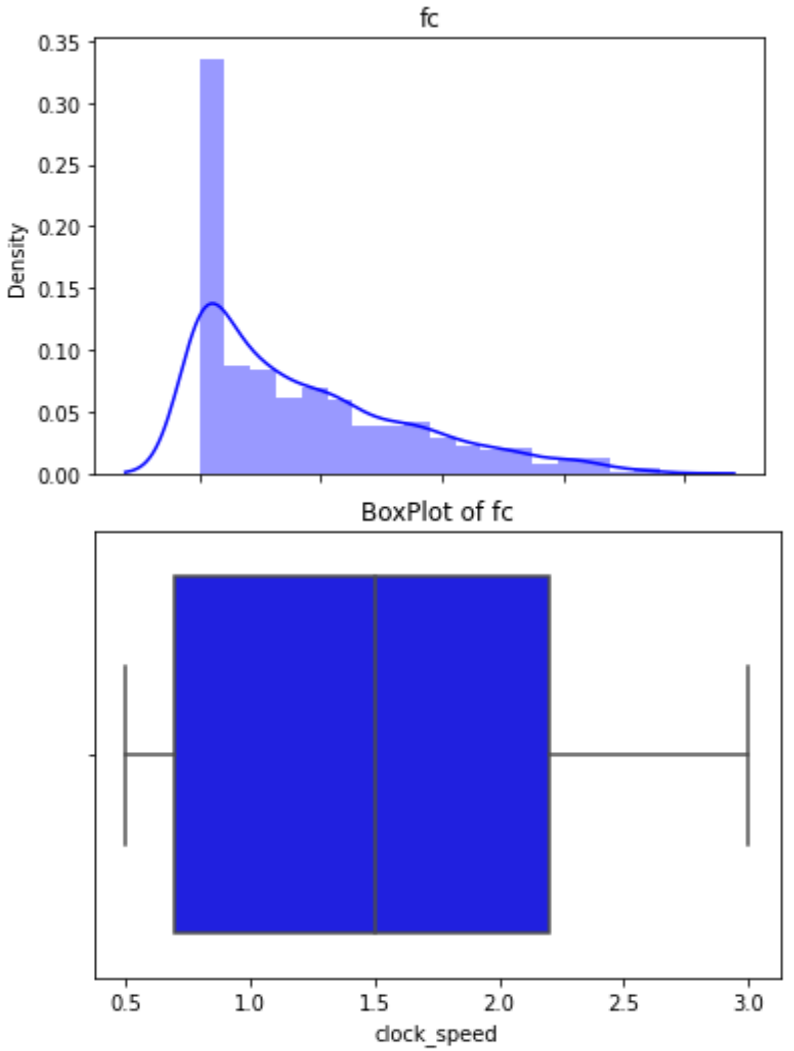
Not Good predictor!

fc

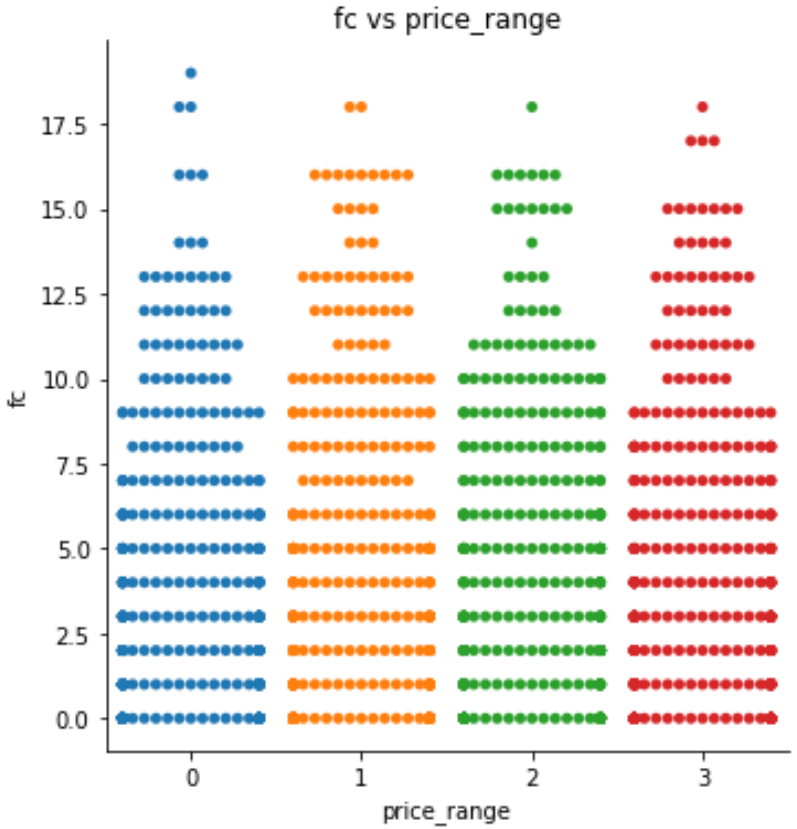
```
In [64]: df.fc.value_counts()
Out[64]:
0     326
1     170
2     130
3     125
5     102
4      90
6      87
9      61
7      58
8      57
10     43
11     34
13     31
12     28
15     18
16     18
14     12
18      6
17      3
19      1
Name: fc, dtype: int64
```

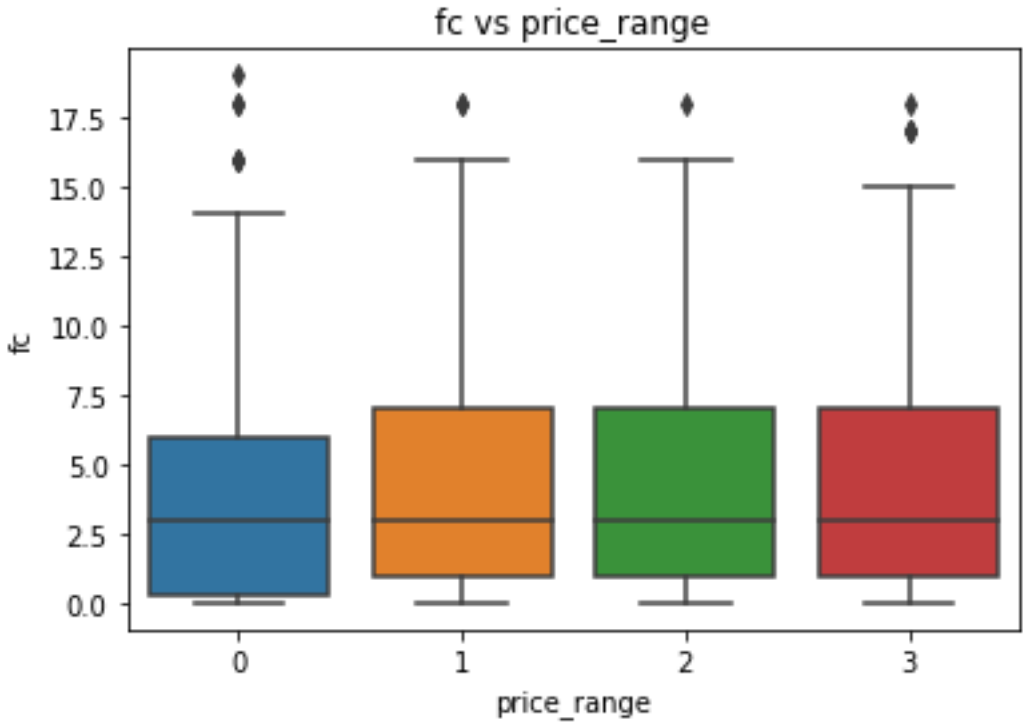



```
sns.distplot(df.fc, color='blue')
plt.title('fc')

sns.boxplot(df.clock_speed, color = 'blue')
plt.title('BoxPlot of fc')
```

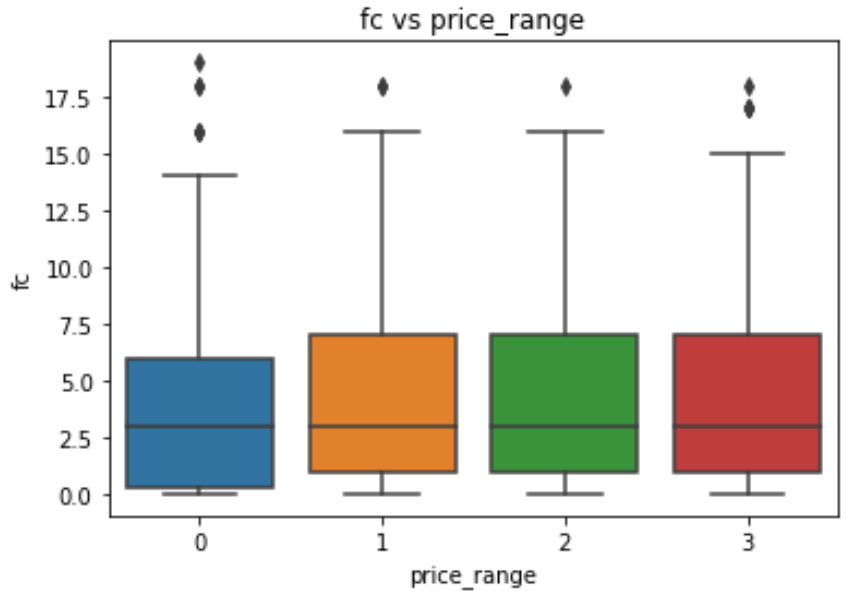


```
#_________groupby plot
sns.catplot(x="price_range", y="fc", kind="swarm", data=df)
plt.title('fc vs price_range')

sns.boxplot(x="price_range", y="fc", data = df)
plt.title('fc vs price_range')
```





```
In [78]: df.fc.groupby(df.price_range).describe()
Out[78]:
             count      mean       std  min   25%  50%  75%   max
price_range
0            350.0  4.005714  4.142860  0.0  0.25  3.0  6.0  19.0
1            350.0  4.431429  4.545174  0.0  1.00  3.0  7.0  18.0
2            351.0  4.450142  4.275805  0.0  1.00  3.0  7.0  18.0
3            349.0  4.412607  4.288646  0.0  1.00  3.0  7.0  18.0

In [79]: df.fc.groupby(df.price_range).mean()
Out[79]:
price_range
0    4.005714
1    4.431429
2    4.450142
3    4.412607
Name: fc, dtype: float64
```

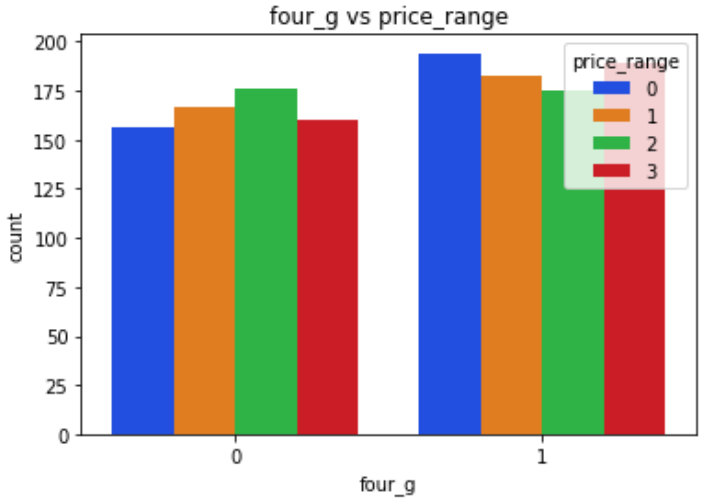



```
In [80]: import statsmodels.api as sm

In [81]: from statsmodels.formula.api import ols

In [82]: mod = ols('fc ~ price_range', data = df).fit()

In [83]: sm.stats.anova_lm(mod)
Out[83]:
                 df        sum_sq    mean_sq         F    PR(>F)
price_range     1.0     26.895366  26.895366  1.445019  0.229532
Residual     1398.0  26020.229634  18.612468       NaN       NaN

In [84]: from statsmodels.stats.multicomp import pairwise_tukeyhsd

In [85]: rslt = pairwise_tukeyhsd(df.fc, df.price_range, alpha = 0.05)

In [86]: print(rslt)
Multiple Comparison of Means - Tukey HSD, FWER=0.05
=================================================
group1 group2 meandiff p-adj   lower  upper  reject
-------------------------------------------------
     0      1   0.4257 0.5528 -0.4134 1.2648  False
     0      2   0.4444 0.5201 -0.3941 1.2829  False
     0      3   0.4069 0.5857 -0.4328 1.2466  False
     1      2   0.0187    0.9 -0.8198 0.8572  False
     1      3  -0.0188    0.9 -0.8585 0.8209  False
     2      3  -0.0375    0.9 -0.8766 0.8016  False
-------------------------------------------------
```

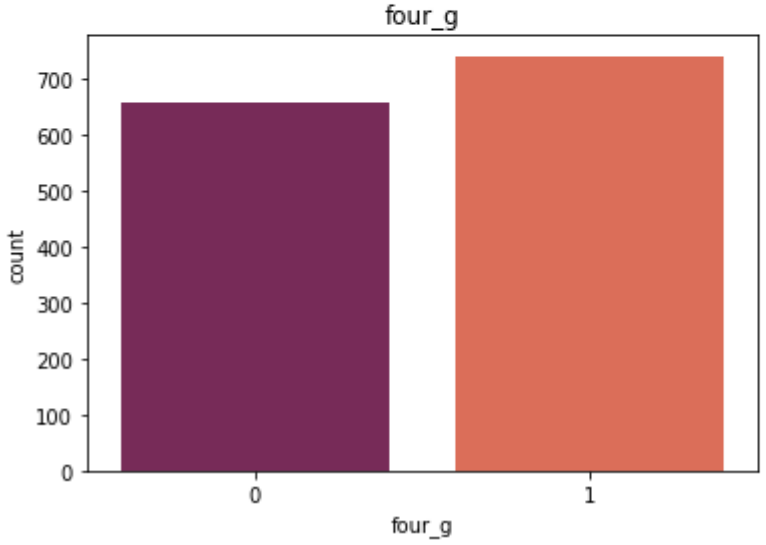
Not Good predictor!

four_g

```
sns.countplot(df.four_g, hue=df.price_range, palette='bright')
plt.title('four_g vs price_range')
```




```
In [88]: df.four_g.value_counts()
Out[88]:
1    741
0    659
Name: four_g, dtype: int64

In [89]: sns.countplot(df.four_g, palette = 'rocket')
    ...: plt.title('four_g')
Out[89]: Text(0.5, 1.0, 'four_g')
```

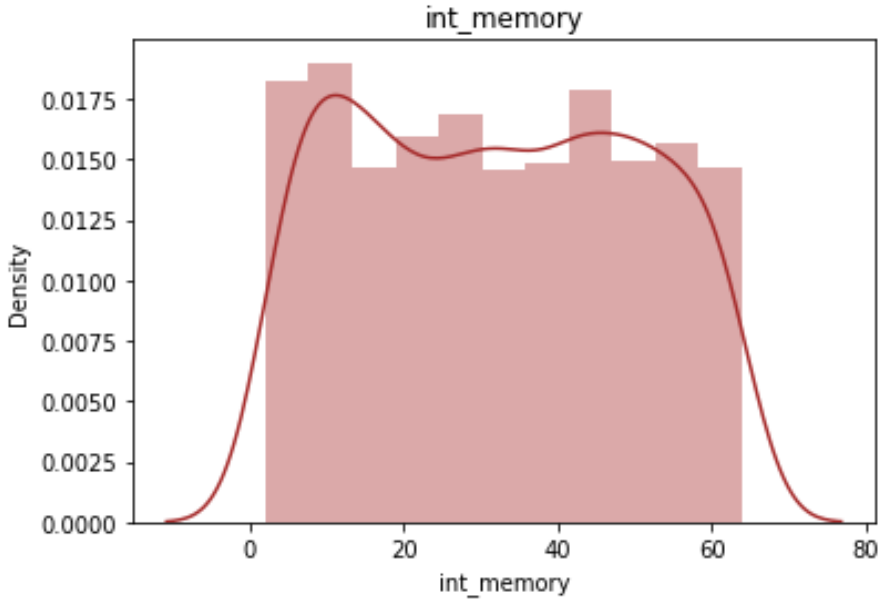


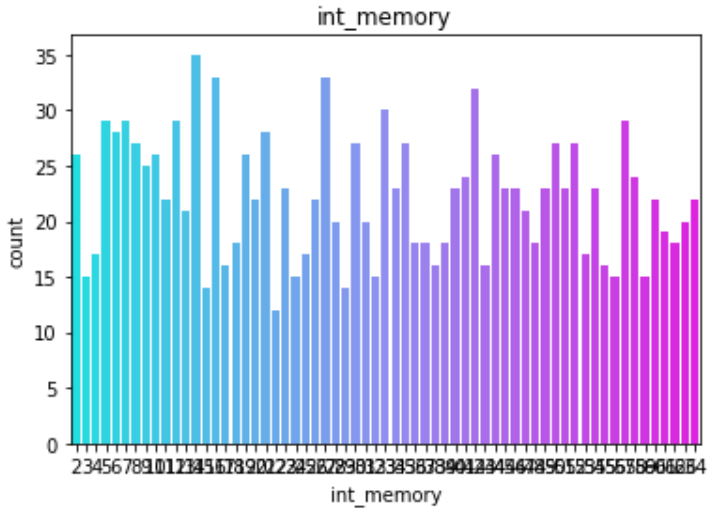

```
In [92]: from scipy.stats import chi2, chi2_contingency

In [93]: ct_four_g = pd.crosstab(df.four_g, df.price_range)

In [94]: ct_four_g
Out[94]:
price_range    0    1    2    3
four_g
0            156  167  176  160
1            194  183  175  189

In [95]: chi2_contingency(ct_four_g, correction=False)
Out[95]:
(2.475368868228543,
 0.479758931049657 94,
 3,
 array([[164.75      , 164.75      , 165.22071429, 164.27928571],
        [185.25      , 185.25      , 185.77928571, 184.72071429]]))
```

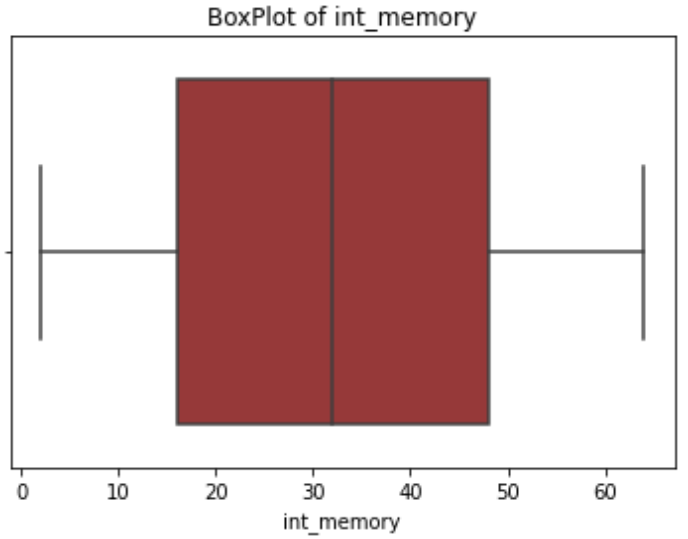
Not Good predictor!

int_memory

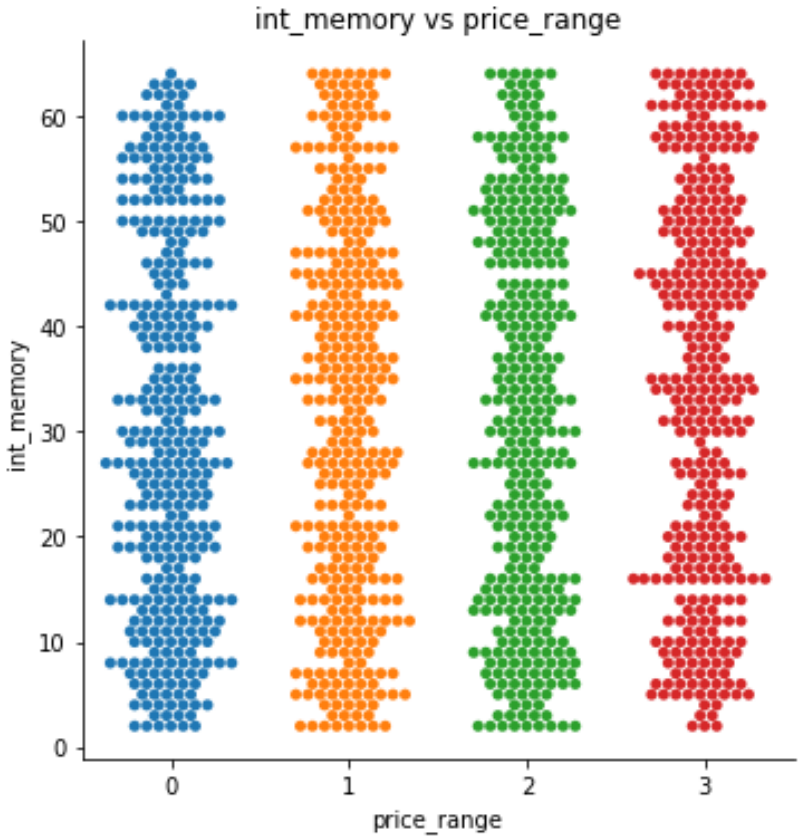
```
df.int_memory.value_counts()

sns.countplot(df.int_memory, palette = 'cool')
plt.title('int_memory')

sns.distplot(df.int_memory, color='brown')
plt.title('int_memory')

sns.boxplot(df.int_memory, color = 'brown')
plt.title('BoxPlot of int_memory')
```

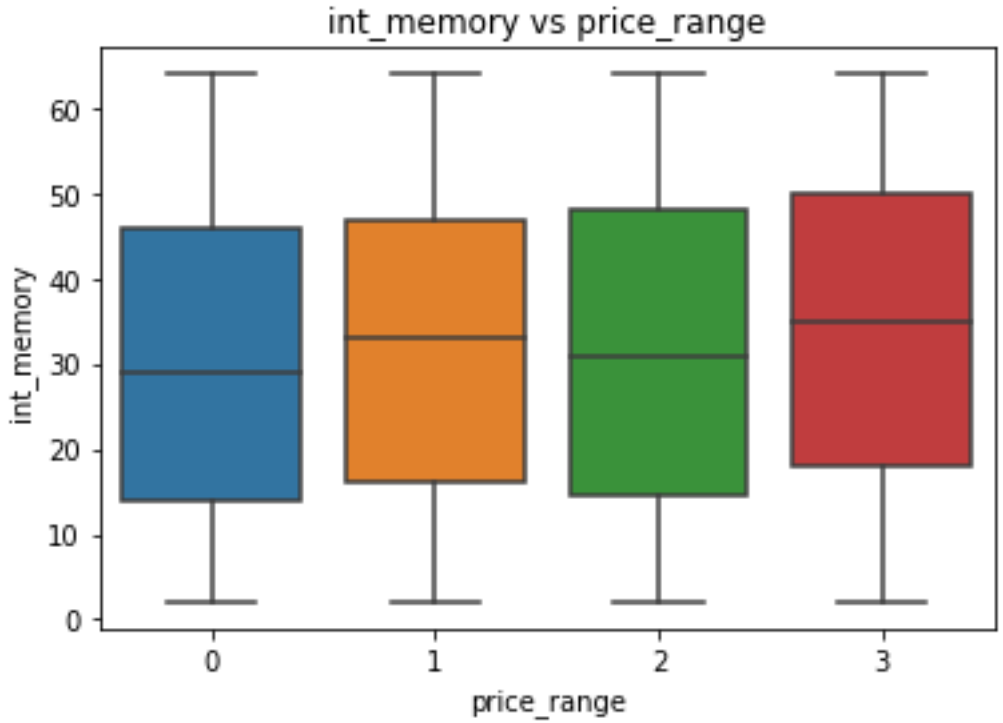




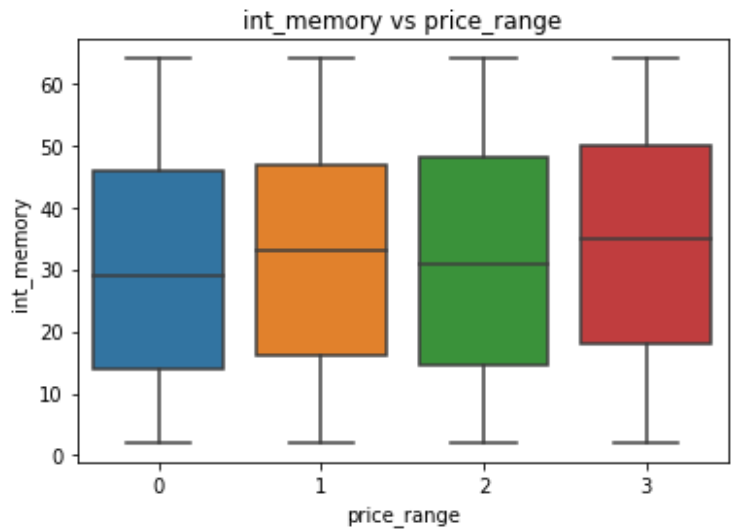


```
sns.catplot(x="price_range", y="int_memory", kind="swarm", data=df)
plt.title('int_memory vs price_range')

sns.boxplot(x="price_range", y="int_memory", data = df)
plt.title('int_memory vs price_range')
```





```
In [106]: df.int_memory.groupby(df.price_range).describe()
Out[106]:
             count       mean        std  min   25%   50%    75%   max
price_range
0            350.0  30.637143  17.952095  2.0  14.0  29.0  46.00  64.0
1            350.0  31.885714  18.172323  2.0  16.0  33.0  46.75  64.0
2            351.0  31.484330  18.268448  2.0  14.5  31.0  48.00  64.0
3            349.0  34.524355  18.453813  2.0  18.0  35.0  50.00  64.0

In [107]: df.int_memory.groupby(df.price_range).mean()
Out[107]:
price_range
0    30.637143
1    31.885714
2    31.484330
3    34.524355
Name: int_memory, dtype: float64
```

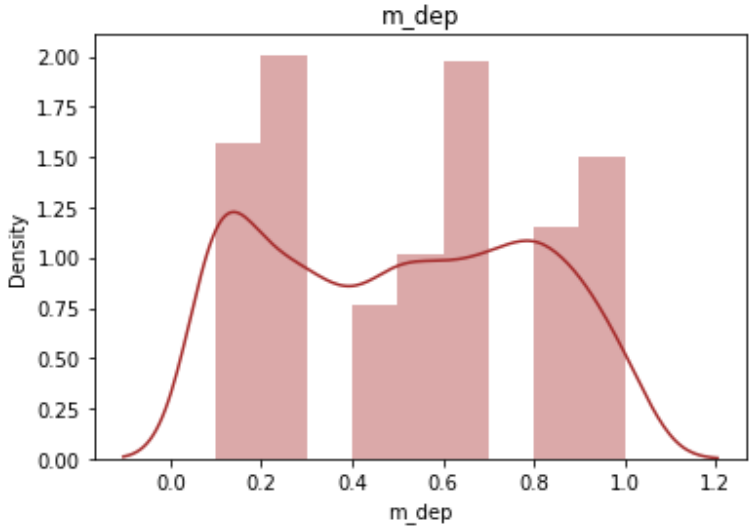


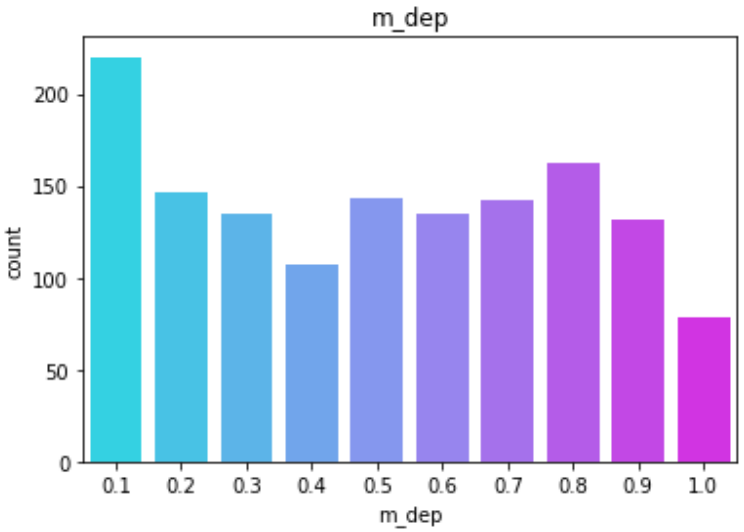
Good predictor!

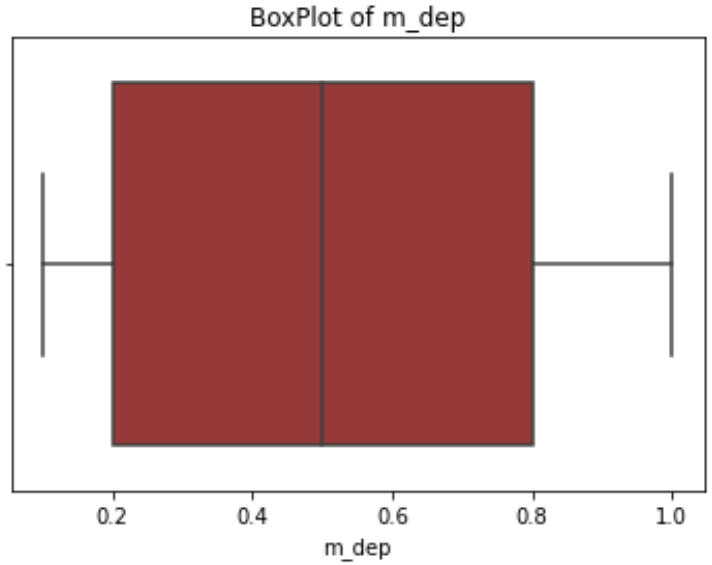
```
In [108]: import statsmodels.api as sm

In [109]: from statsmodels.formula.api import ols

In [110]: mod = ols('int_memory ~ price_range', data = df).fit()

In [111]: sm.stats.anova_lm(mod)
Out[111]:
                df          sum_sq      mean_sq         F    PR(>F)
price_range    1.0     2212.607521  2212.607521  6.669618  0.009908
Residual    1398.0   463778.471765   331.744257       NaN       NaN

In [112]: from statsmodels.stats.multicomp import pairwise_tukeyhsd

In [113]: rslt = pairwise_tukeyhsd(df.int_memory, df.price_range, alpha = 0.05)

In [114]: print(rslt)
Multiple Comparison of Means - Tukey HSD, FWER=0.05
=====================================================
group1 group2 meandiff p-adj   lower  upper  reject
-----------------------------------------------------
     0      1   1.2486 0.7762 -2.2926 4.7897  False
     0      2   0.8472    0.9 -2.6914 4.3858  False
     0      3   3.8872  0.025  0.3435 7.4309   True
     1      2  -0.4014    0.9   -3.94 3.1372  False
     1      3   2.6386 0.2222  -0.905 6.1823  False
     2      3     3.04 0.1217 -0.5011 6.5812  False
-----------------------------------------------------
```

m_dep

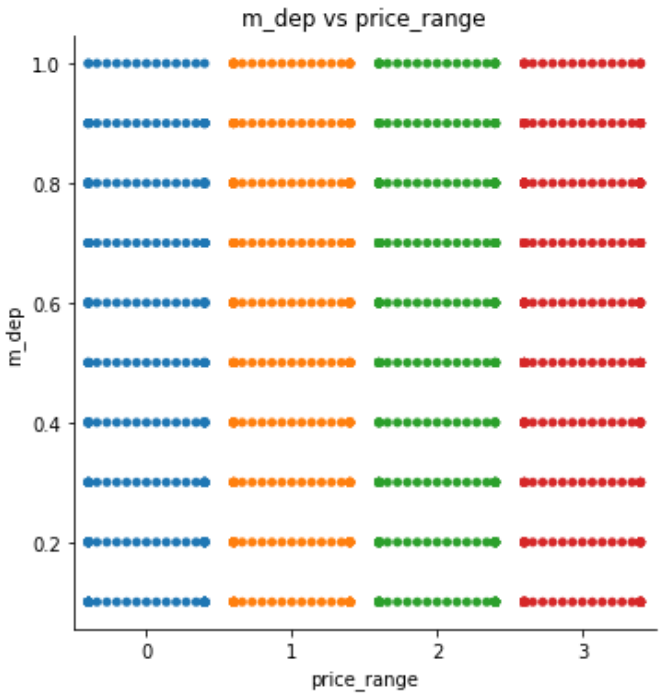
```
df.m_dep.value_counts() # cont
sns.countplot(df.m_dep, palette = 'cool')
plt.title('m_dep')

sns.distplot(df.m_dep, color='brown')
plt.title('m_dep')

sns.boxplot(df.m_dep, color = 'brown')
plt.title('BoxPlot of m_dep')
```

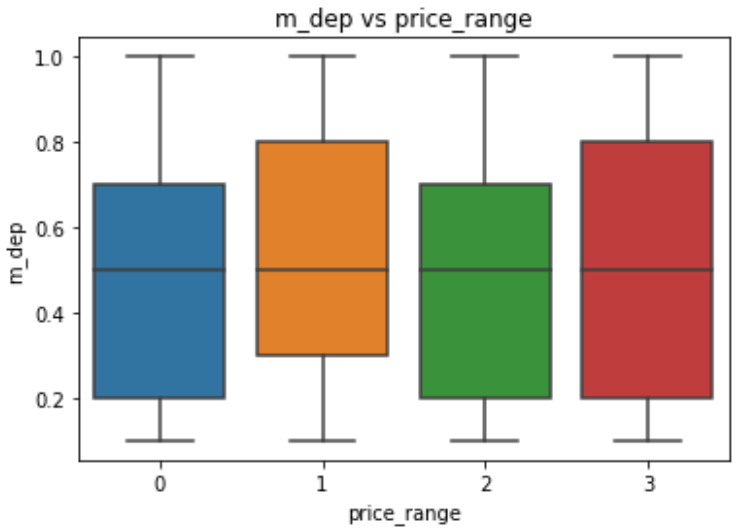


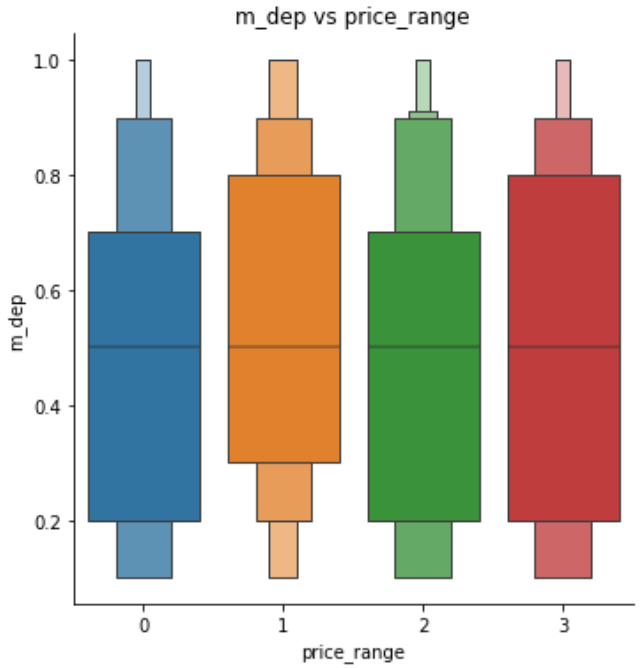




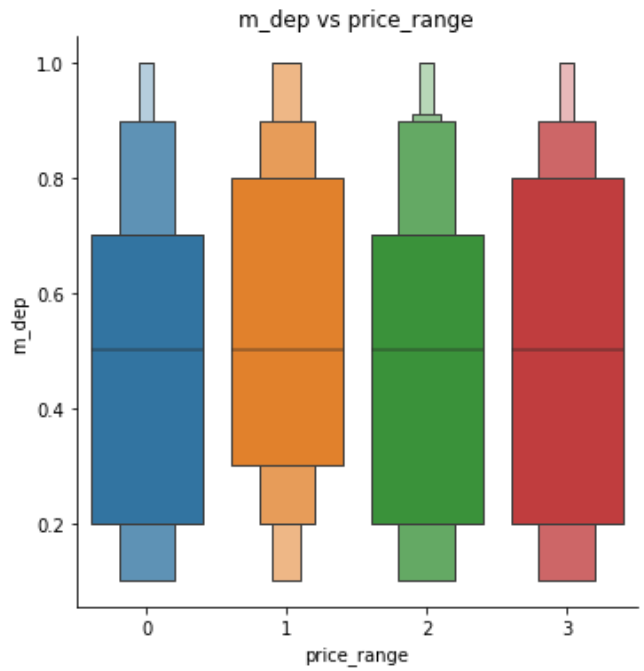
```
#_________groupby plot
sns.catplot(x="price_range", y="m_dep", kind="swarm", data=df)
plt.title('m_dep vs price_range')

sns.boxplot(x="price_range", y="m_dep", data = df)
plt.title('m_dep vs price_range')

sns.catplot(x="price_range", y="m_dep", kind="boxen", data=df)
plt.title('m_dep vs price_range')
```







```
In [130]: df.m_dep.groupby(df.price_range).mean()
Out[130]:
price_range
0    0.489429
1    0.535429
2    0.500285
3    0.511461
Name: m_dep, dtype: float64
```

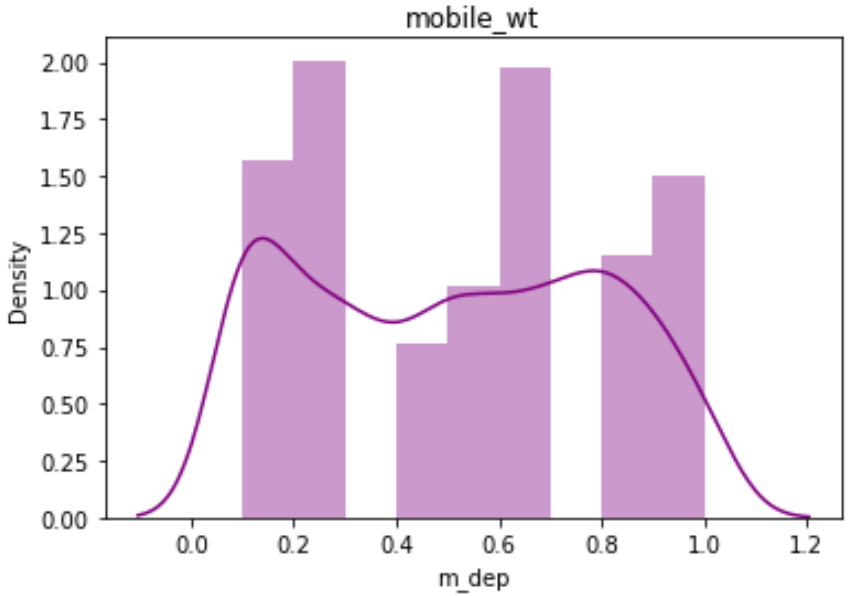


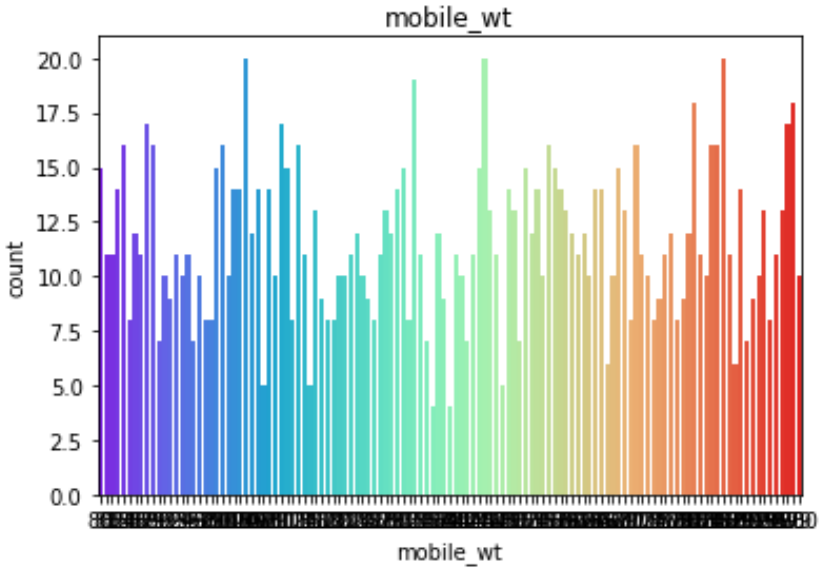

```
In [123]: import statsmodels.api as sm

In [124]: from statsmodels.formula.api import ols

In [125]: mod = ols('m_dep ~ price_range', data = df).fit()

In [126]: sm.stats.anova_lm(mod)
Out[126]:
                df      sum_sq   mean_sq         F    PR(>F)
price_range    1.0    0.016738  0.016738  0.200164  0.654658
Residual    1398.0  116.906233  0.083624       NaN       NaN

In [127]: from statsmodels.stats.multicomp import pairwise_tukeyhsd

In [128]: rslt = pairwise_tukeyhsd(df.m_dep, df.price_range, alpha = 0.05)

In [129]: print(rslt)
Multiple Comparison of Means - Tukey HSD, FWER=0.05
=================================================
group1 group2 meandiff p-adj   lower  upper  reject
-------------------------------------------------
     0      1    0.046 0.1516 -0.0102 0.1022  False
     0      2   0.0109    0.9 -0.0453  0.067  False
     0      3    0.022 0.7194 -0.0342 0.0782  False
     1      2  -0.0351 0.3739 -0.0913  0.021  False
     1      3   -0.024 0.6697 -0.0802 0.0322  False
     2      3   0.0112    0.9  -0.045 0.0673  False
```

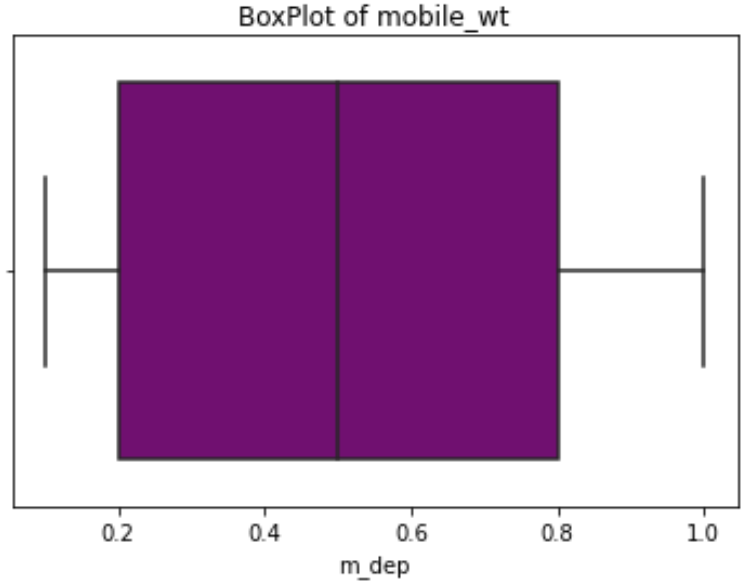
Not Good predictor!

mobile_wt

```
df.mobile_wt.value_counts()

sns.countplot(df.mobile_wt, palette='rainbow')
plt.title('mobile_wt')

sns.distplot(df.m_dep, color='purple')
plt.title('mobile_wt')

sns.boxplot(df.m_dep, color='purple')
plt.title('BoxPlot of mobile_wt')
```

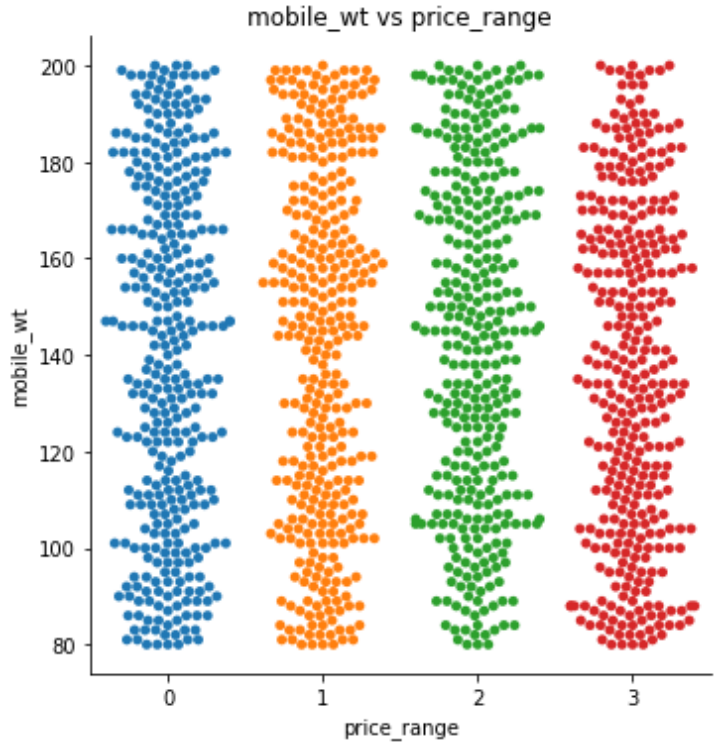


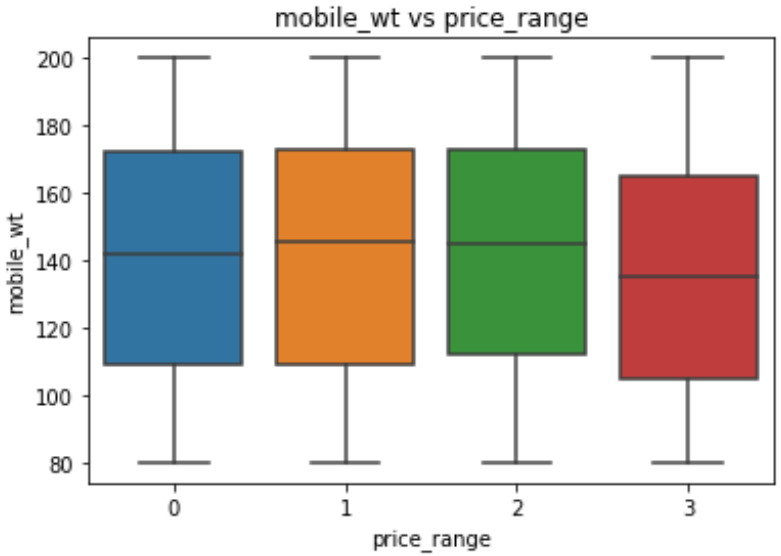


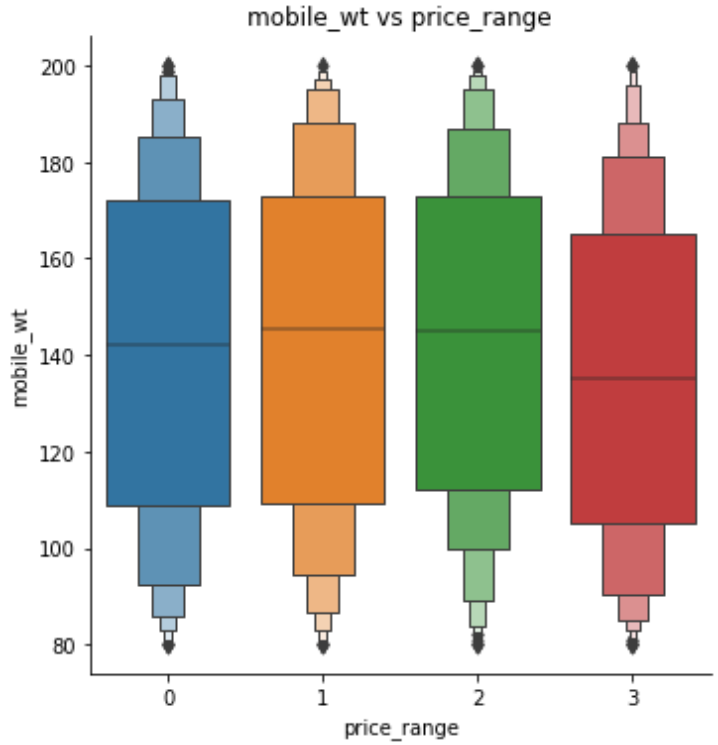


```
#_________groupby plot
sns.catplot(x="price_range", y="mobile_wt", kind="swarm", data=df)
plt.title('mobile_wt vs price_range')

sns.boxplot(x="price_range", y="mobile_wt", data = df)
plt.title('mobile_wt vs price_range')

sns.catplot(x="price_range", y="mobile_wt", kind="boxen", data=df)
plt.title('mobile_wt vs price_range')
```

```
In [141]: df.mobile_wt.groupby(df.price_range).mean()
Out[141]:
price_range
0    140.094286
1    141.905714
2    143.105413
3    136.406877
Name: mobile_wt, dtype: float64
```



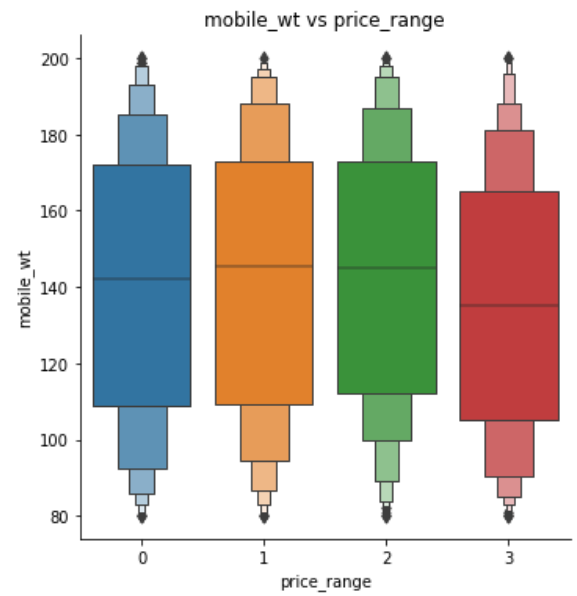




```
In [142]: import statsmodels.api as sm

In [143]: from statsmodels.formula.api import ols

In [144]: mod = ols('mobile_wt ~ price_range', data = df).fit()

In [145]: sm.stats.anova_lm(mod)
Out[145]:
                 df         sum_sq      mean_sq          F    PR(>F)
price_range     1.0   1.689729e+03  1689.728885   1.347108  0.245983
Residual     1398.0   1.753565e+06  1254.338383        NaN       NaN

In [146]: from statsmodels.stats.multicomp import pairwise_tukeyhsd

In [147]: rslt = pairwise_tukeyhsd(df.mobile_wt, df.price_range, alpha = 0.05)

In [148]: print(rslt)
Multiple Comparison of Means - Tukey HSD, FWER=0.05
===================================================
group1 group2 meandiff p-adj   lower   upper  reject
---------------------------------------------------
     0      1   1.8114    0.9  -5.0655 8.6883  False
     0      2   3.0111 0.6527  -3.8609 9.8831  False
     0      3  -3.6874 0.5117 -10.5692 3.1944  False
     1      2   1.1997    0.9  -5.6723 8.0717  False
     1      3  -5.4988 0.1687 -12.3806  1.383  False
     2      3  -6.6985 0.0596 -13.5755 0.1784  False
---------------------------------------------------
```

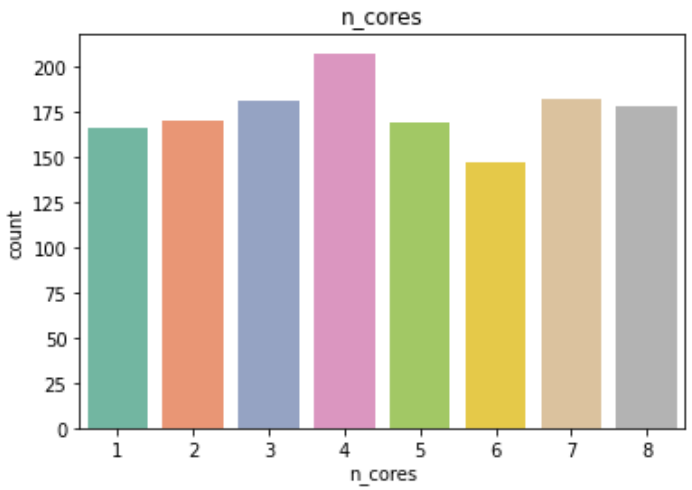
Not Good predictor!

```
In [141]: df.mobile_wt.groupby(df.price_range).mean()
Out[141]:
price_range
0    140.094286
1    141.905714
2    143.105413
3    136.406877
Name: mobile_wt, dtype: float64
```

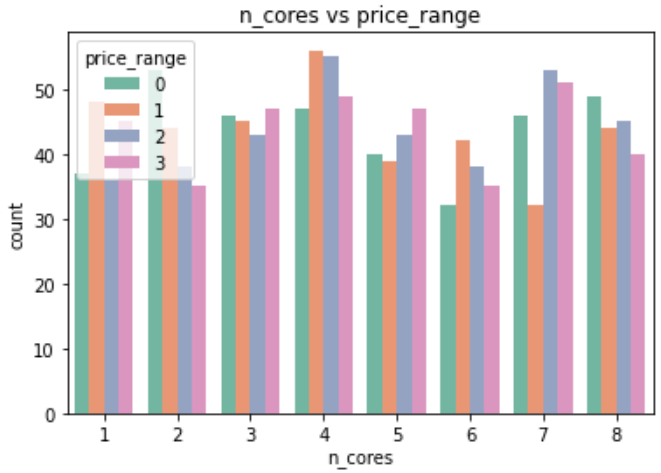


n_cores

```
In [150]: df.n_cores.value_counts()
Out[150]:
4    207
7    182
3    181
8    178
2    170
5    169
1    166
6    147
Name: n_cores, dtype: int64
```




```
df.n_cores.value_counts()

sns.countplot(df.n_cores, palette='Set2')
plt.title('n_cores')

sns.countplot(df.n_cores, hue=df.price_range, palette='Set2')
plt.title('n_cores vs price_range')
```

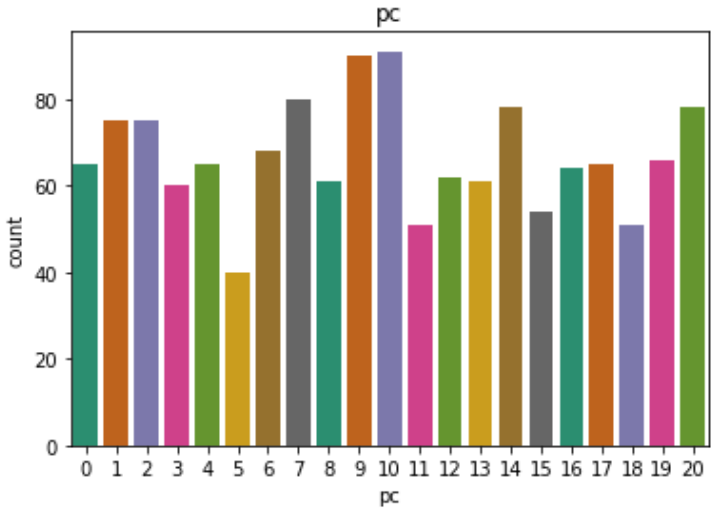


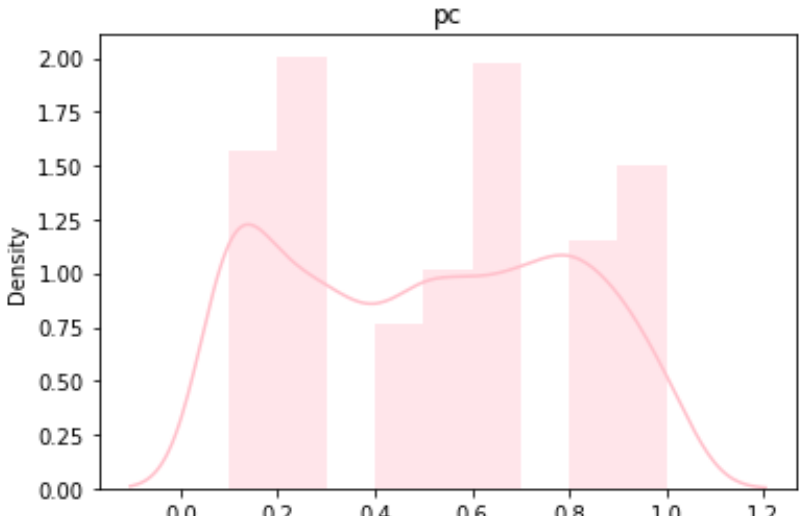

```
In [153]: from scipy.stats import chi2, chi2_contingency

In [154]: ct_n_cores = pd.crosstab(df.n_cores, df.price_range)

In [155]: ct_n_cores
Out[155]:
price_range   0   1   2   3
n_cores
1            37  48  36  45
2            53  44  38  35
3            46  45  43  47
4            47  56  55  49
5            40  39  43  47
6            32  42  38  35
7            46  32  53  51
8            49  44  45  40

In [156]: chi2_contingency(ct_n_cores, correction=False)
Out[156]:
(17.542689748111123,
 0.6777314982666398,
 21,
 array([[41.5       , 41.5       , 41.61857143, 41.38142857],
        [42.5       , 42.5       , 42.62142857, 42.37857143],
```

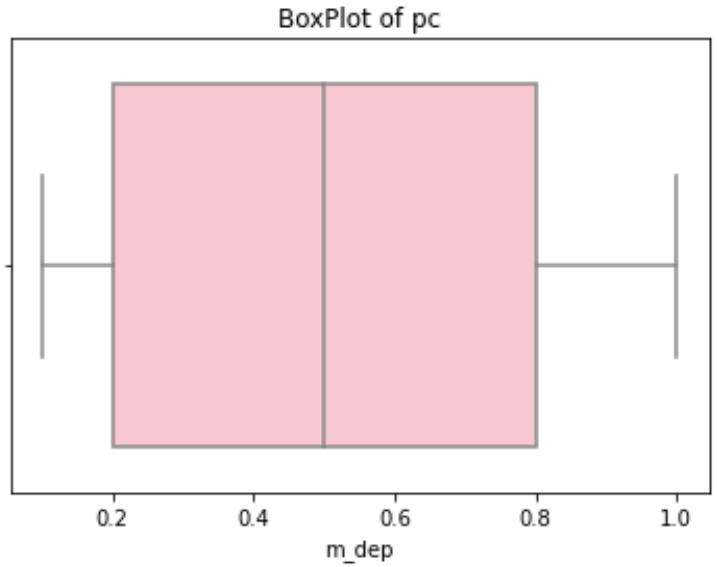
pc

```
In [158]: df.pc.value_counts()
Out[158]:
10    91
9     90
7     80
14    78
20    78
2     75
1     75
6     68
19    66
17    65
4     65
0     65
16    64
12    62
13    61
8     61
3     60
15    54
11    51
18    51
5     40
Name: pc, dtype: int64
```

```
df.pc.value_counts()

sns.countplot(df.pc, palette='Dark2')
plt.title('pc')

sns.distplot(df.m_dep, color='pink')
plt.title('pc')

sns.boxplot(df.m_dep, color='pink')
plt.title('BoxPlot of pc')
```

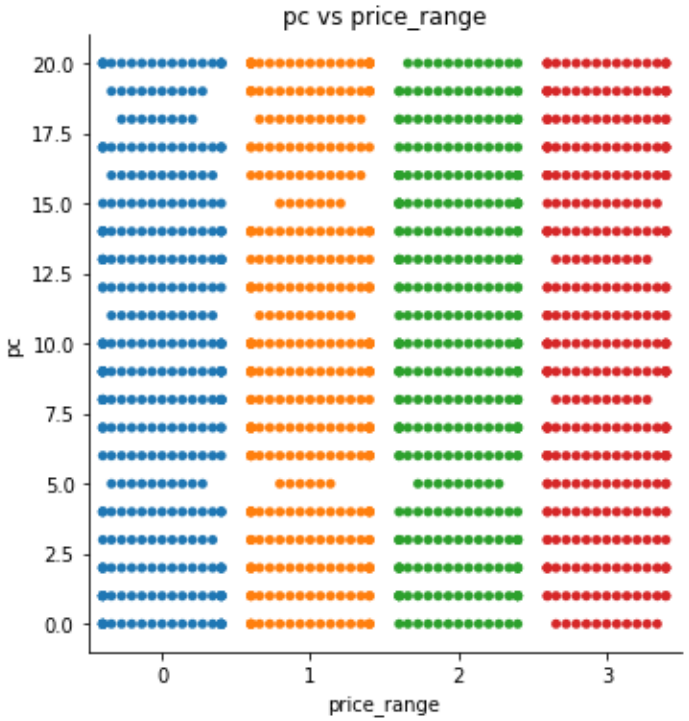


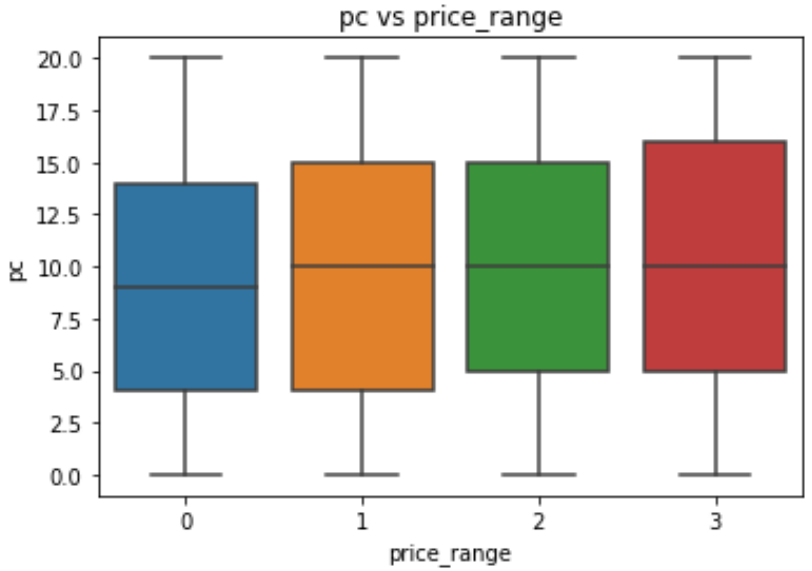


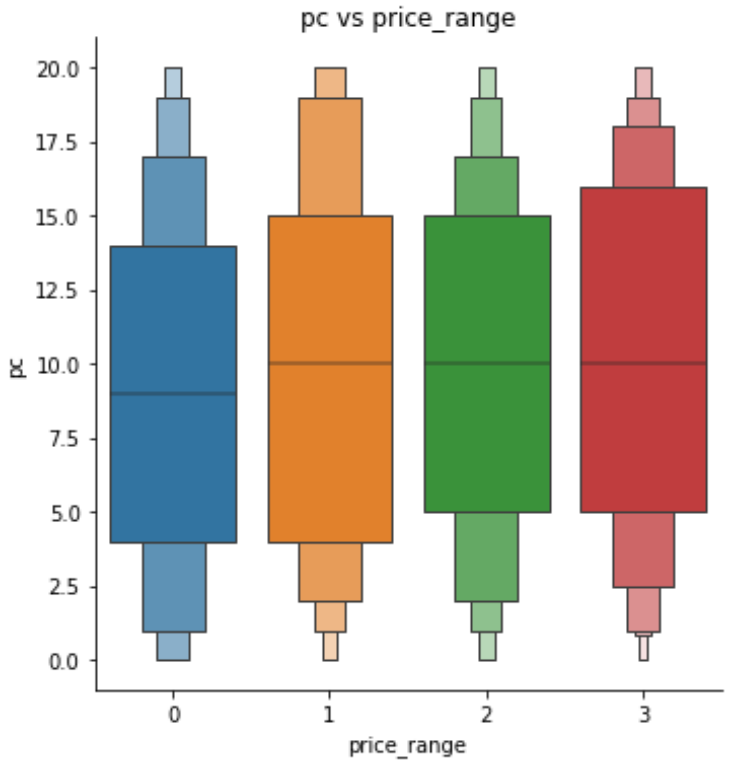


```
sns.catplot(x="price_range", y="pc", kind="swarm", data=df)
plt.title('pc vs price_range')

sns.boxplot(x="price_range", y="pc", data = df)
plt.title('pc vs price_range')

sns.catplot(x="price_range", y="pc", kind="boxen", data=df)
plt.title('pc vs price_range')
```

```
In [166]: df.pc.groupby(df.price_range).mean()
Out[166]:
price_range
0     9.457143
1     9.945714
2     9.980057
3    10.338109
Name: pc, dtype: float64
```

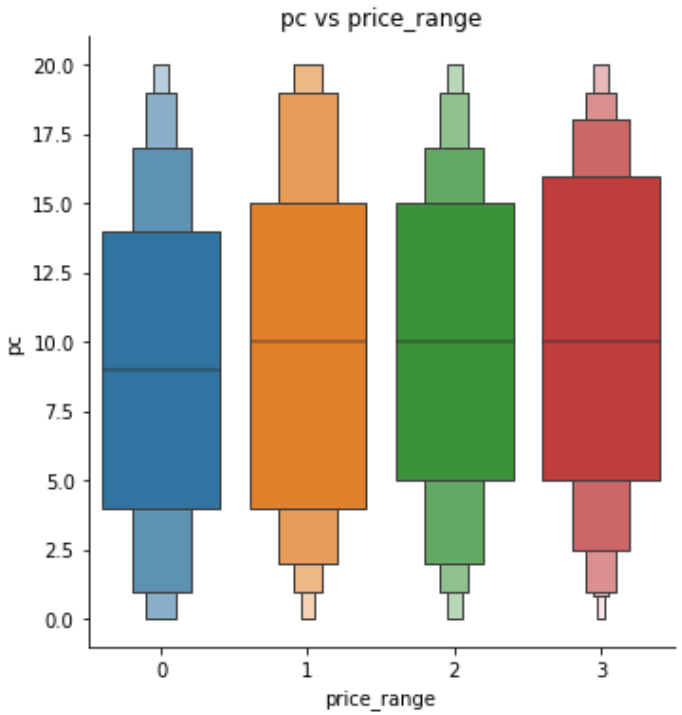






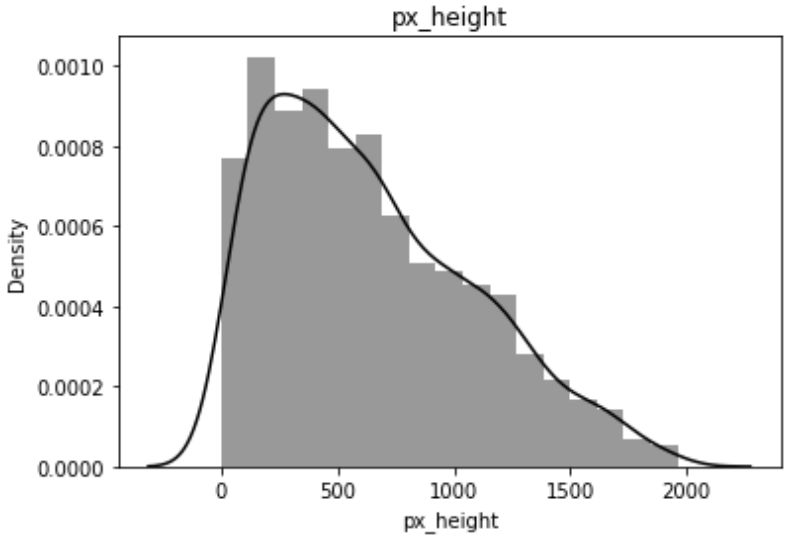
```
In [167]: import statsmodels.api as sm

In [168]: from statsmodels.formula.api import ols

In [169]: mod = ols('pc ~ price_range', data = df).fit()

In [170]: sm.stats.anova_lm(mod)
Out[170]:
                df        sum_sq     mean_sq          F    PR(>F)
price_range    1.0    125.262342  125.262342   3.432997  0.064117
Residual    1398.0  51009.877658   36.487752        NaN       NaN

In [171]: from statsmodels.stats.multicomp import pairwise_tukeyhsd

In [172]: rslt = pairwise_tukeyhsd(df.pc, df.price_range, alpha = 0.05)

In [173]: print(rslt)
Multiple Comparison of Means - Tukey HSD, FWER=0.05
===================================================
group1 group2 meandiff p-adj   lower  upper  reject
---------------------------------------------------
     0      1   0.4886  0.685 -0.6866 1.6638  False
     0      2   0.5229 0.6424 -0.6514 1.6973  False
     0      3    0.881 0.2173 -0.2951  2.057  False
     1      2   0.0343    0.9   -1.14 1.2087  False
     1      3   0.3924 0.8035 -0.7836 1.5684  False
     2      3   0.3581 0.8453 -0.8171 1.5332  False
---------------------------------------------------
```

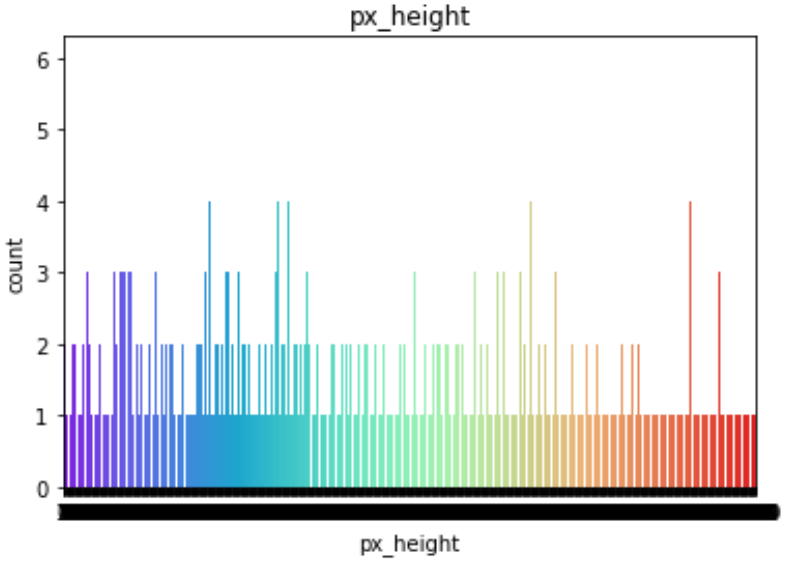
Not Good predictor!

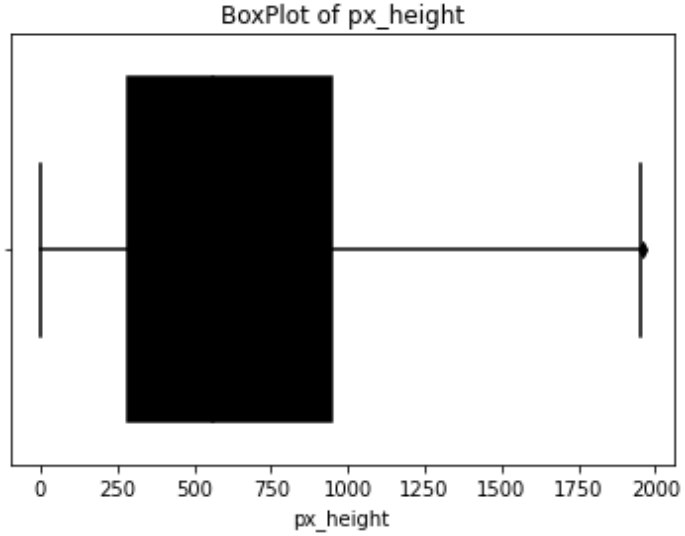


px_height

```
df.px_height.value_counts()

sns.countplot(df.px_height, palette='rainbow')
plt.title('px_height')

sns.distplot(df.px_height, color='k')
plt.title('px_height')

sns.boxplot(df.px_height, color='k')
plt.title('BoxPlot of px_height')
```

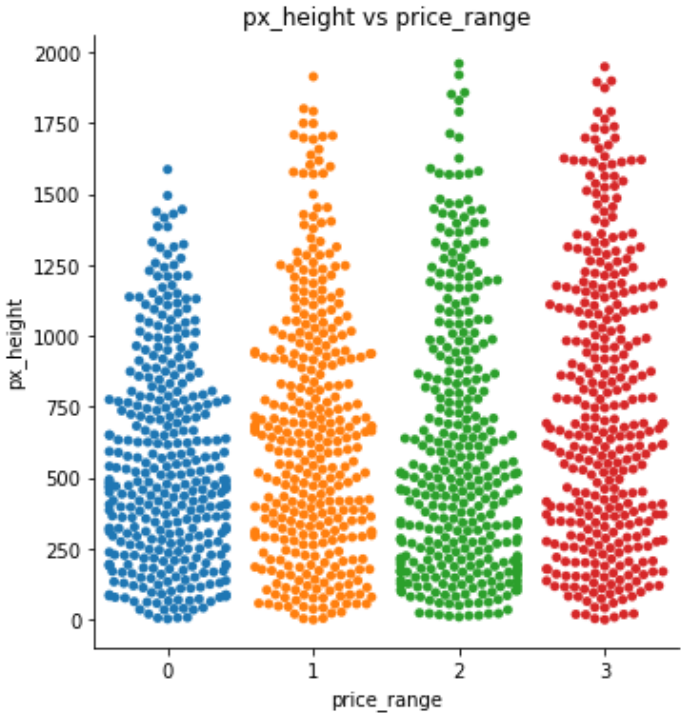


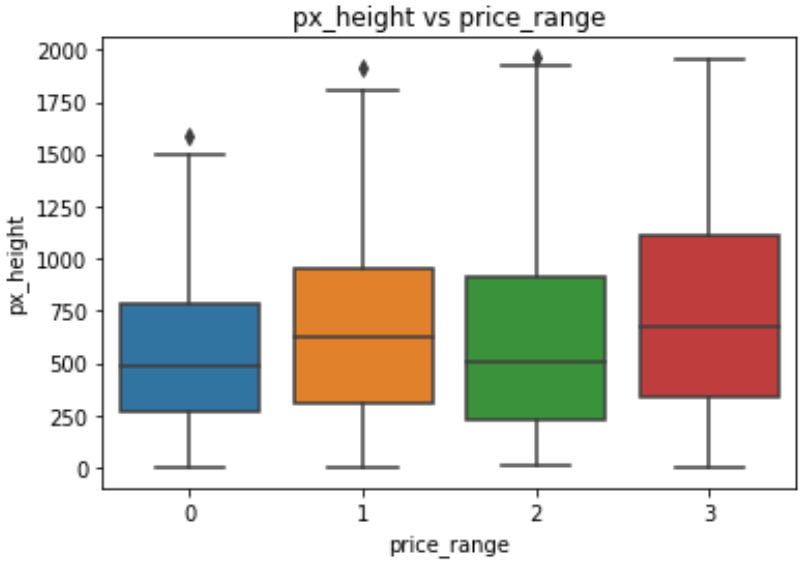


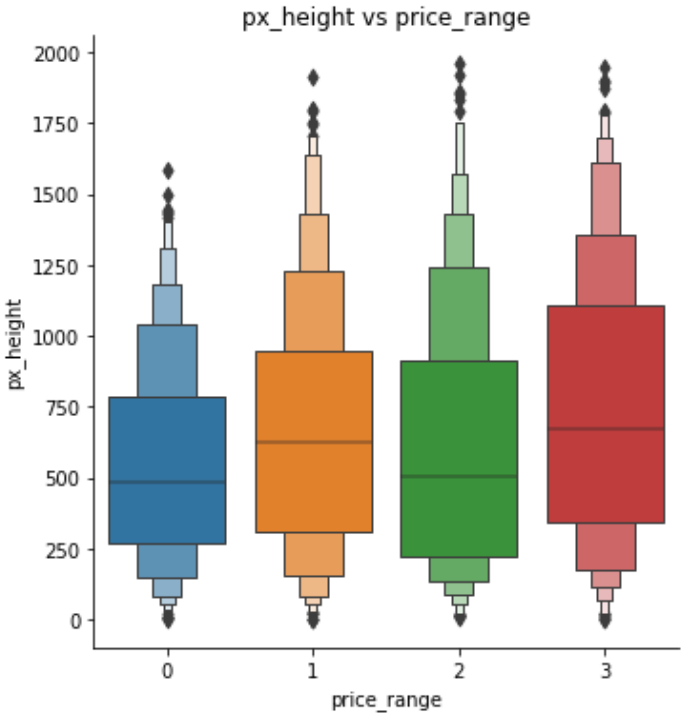


```
#_________groupby plot
sns.catplot(x="price_range", y="px_height", kind="swarm", data=df)
plt.title('px_height vs price_range')

sns.boxplot(x="price_range", y="px_height", data = df)
plt.title('px_height vs price_range')

sns.catplot(x="price_range", y="px_height", kind="boxen", data=df)
plt.title('px_height vs price_range')
```

```
In [185]: df.px_height.groupby(df.price_range).mean()
Out[185]:
price_range
0    555.634286
1    666.722857
2    614.333333
3    746.117479
Name: px_height, dtype: float64
```

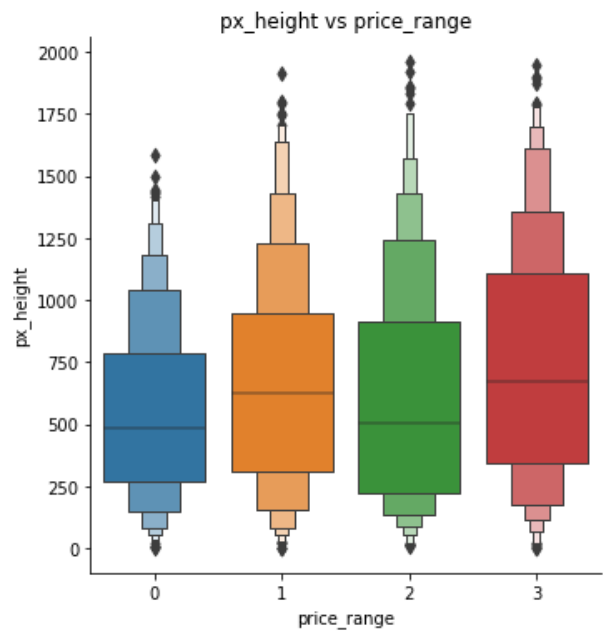






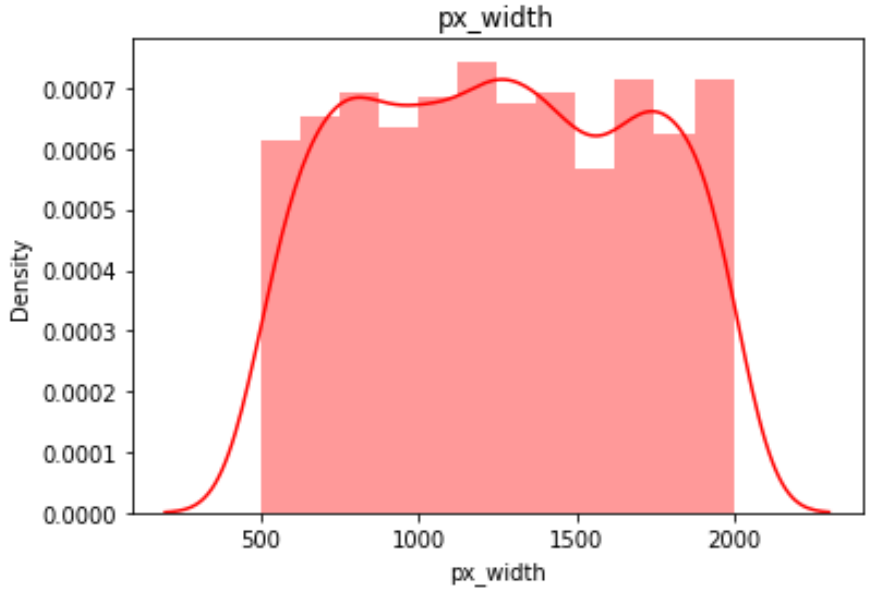
```
In [186]: import statsmodels.api as sm

In [187]: from statsmodels.formula.api import ols

In [188]: mod = ols('px_height ~ price_range', data = df).fit()

In [189]: sm.stats.anova_lm(mod)
Out[189]:
                 df         sum_sq       mean_sq           F    PR(>F)
price_range     1.0   4.703027e+06  4.703027e+06   24.131544  0.000001
Residual     1398.0   2.724580e+08  1.948912e+05         NaN       NaN

In [190]: from statsmodels.stats.multicomp import pairwise_tukeyhsd

In [191]: rslt = pairwise_tukeyhsd(df.px_height, df.price_range, alpha = 0.05)

In [192]: print(rslt)
  Multiple Comparison of Means - Tukey HSD, FWER=0.05
====================================================
group1 group2 meandiff p-adj    lower    upper   reject
----------------------------------------------------
     0      1 111.0886 0.0048   25.5311 196.6461   True
     0      2   58.699 0.2905  -26.7975 144.1956  False
     0      3 190.4832  0.001  104.8644  276.102   True
     1      2 -52.3895 0.3938 -137.8861   33.107  False
     1      3  79.3946 0.0805   -6.2242 165.0134  False
     2      3 131.7841  0.001   46.2263  217.342   True
----------------------------------------------------
```

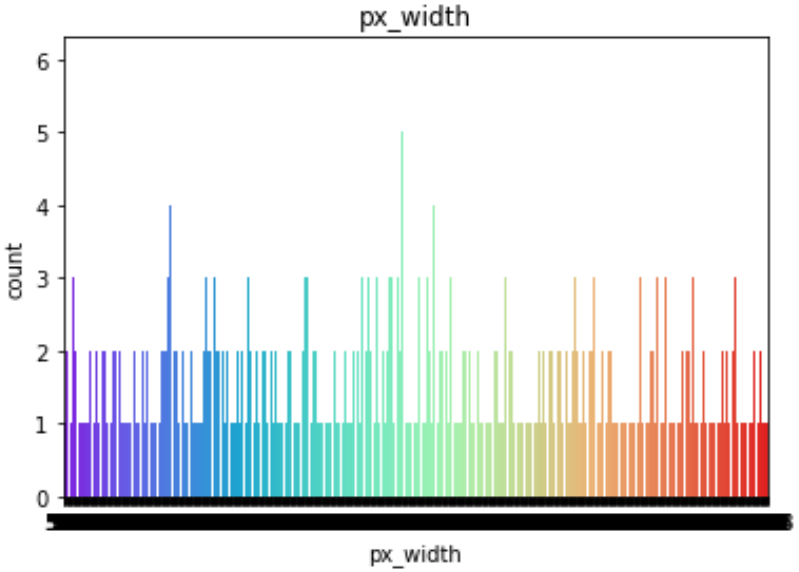
Good predictor!

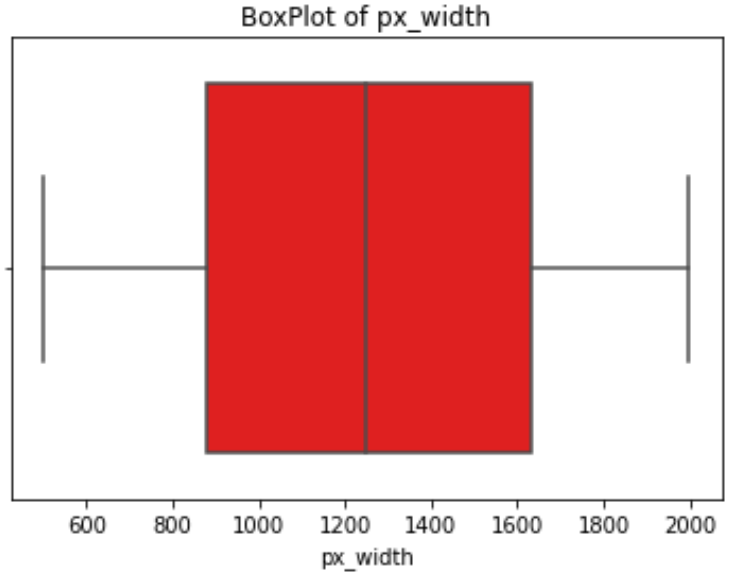


px_width

```
df.px_width.value_counts()

sns.countplot(df.px_width, palette='rainbow')
plt.title('px_width')

sns.distplot(df.px_width, color='r')
plt.title('px_width')

sns.boxplot(df.px_width, color='r')
plt.title('BoxPlot of px_width')
```

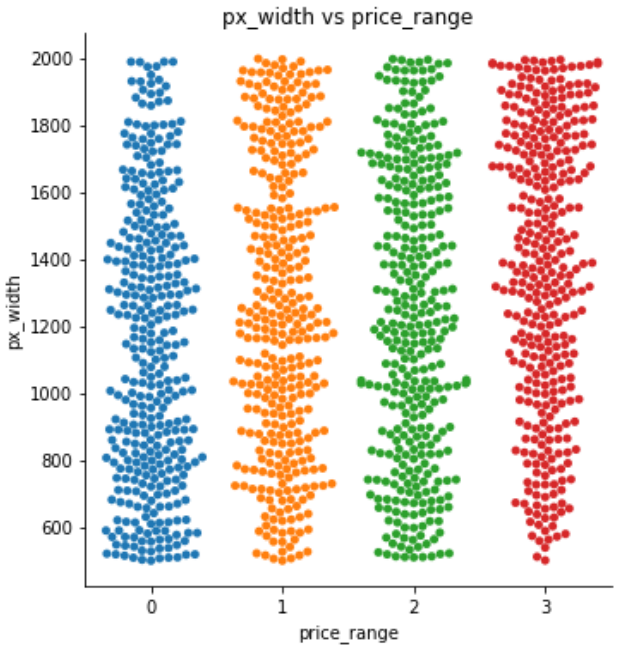


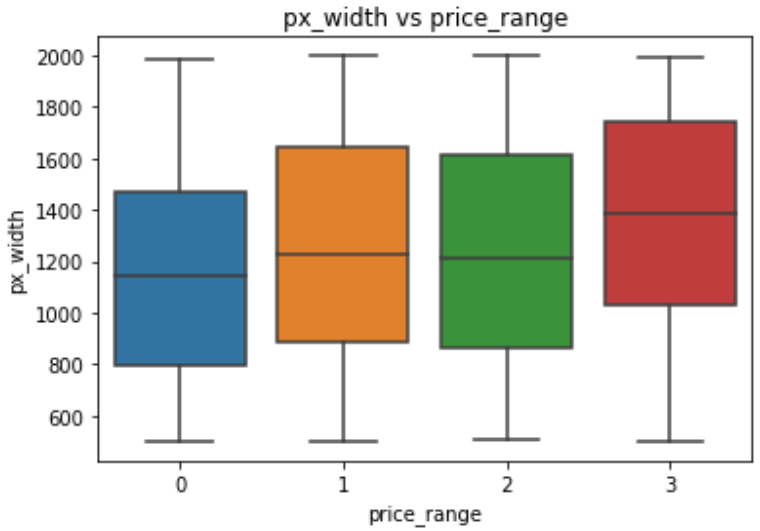


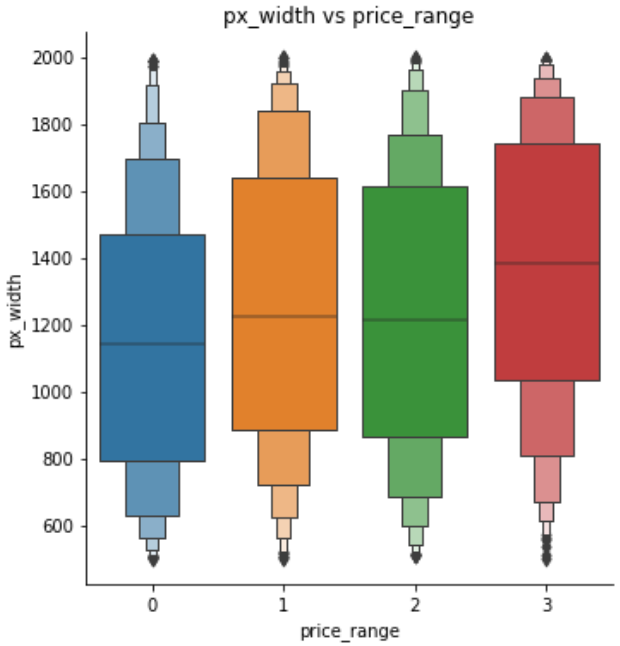


```
#_________groupby plot
sns.catplot(x="price_range", y="px_width", kind="swarm", data=df)
plt.title('px_width vs price_range')

sns.boxplot(x="price_range", y="px_width", data = df)
plt.title('px_width vs price_range')

sns.catplot(x="price_range", y="px_width", kind="boxen", data=df)
plt.title('px_width vs price_range')
```

```
In [202]: df.px_width.groupby(df.price_range).mean()
Out[202]:
price_range
0    1152.845714
1    1257.645714
2    1235.339031
3    1367.845272
Name: px_width, dtype: float64
```

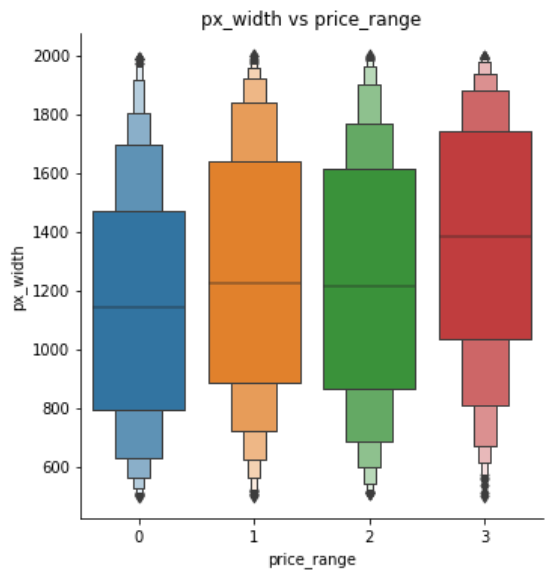






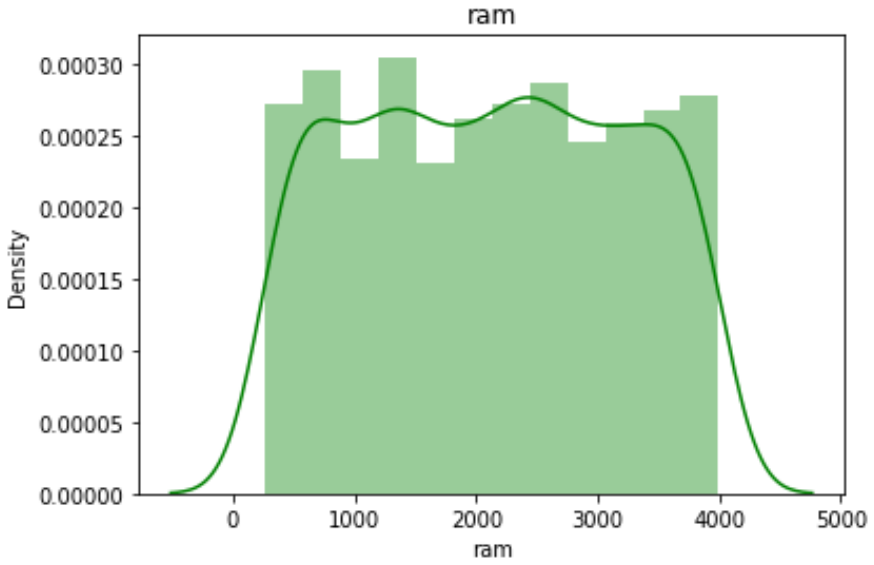
```
In [203]: import statsmodels.api as sm

In [204]: from statsmodels.formula.api import ols

In [205]: mod = ols('px_width ~ price_range', data = df).fit()

In [206]: sm.stats.anova_lm(mod)
Out[206]:
                  df         sum_sq        mean_sq           F         PR(>F)
price_range      1.0   6.770789e+06   6.770789e+06   37.606622   1.124277e-09
Residual      1398.0   2.516994e+08   1.800425e+05         NaN            NaN

In [207]: from statsmodels.stats.multicomp import pairwise_tukeyhsd

In [208]: rslt = pairwise_tukeyhsd(df.px_width, df.price_range, alpha = 0.05)

In [209]: print(rslt)
  Multiple Comparison of Means - Tukey HSD, FWER=0.05
====================================================
group1 group2 meandiff p-adj    lower    upper   reject
----------------------------------------------------
     0      1    104.8  0.006    22.479  187.121    True
     0      2  82.4933 0.0491    0.2309 164.7557    True
     0      3 214.9996  0.001  132.6196 297.3795    True
     1      2 -22.3067 0.8937 -104.5691  59.9557   False
     1      3 110.1996 0.0033   27.8196 192.5795    True
     2      3 132.5062  0.001   50.1849 214.8276    True
----------------------------------------------------
```

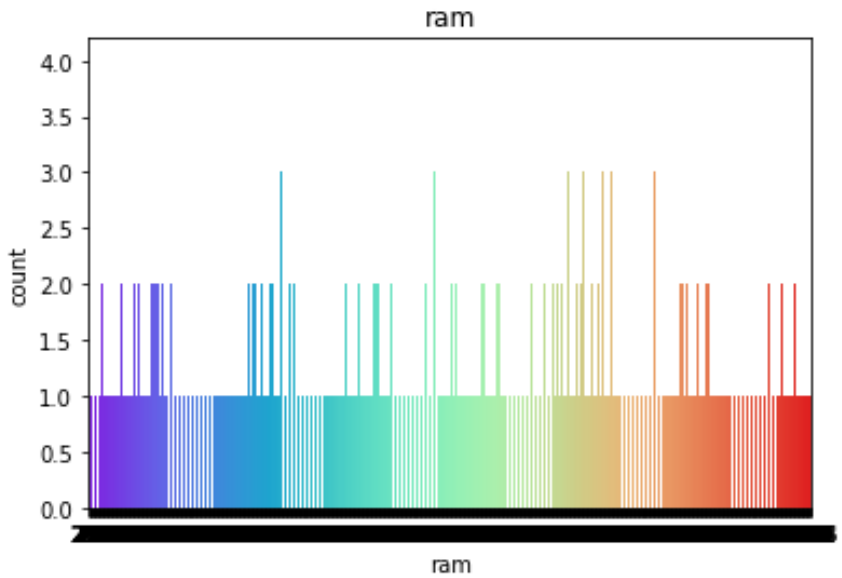
Good predictor!

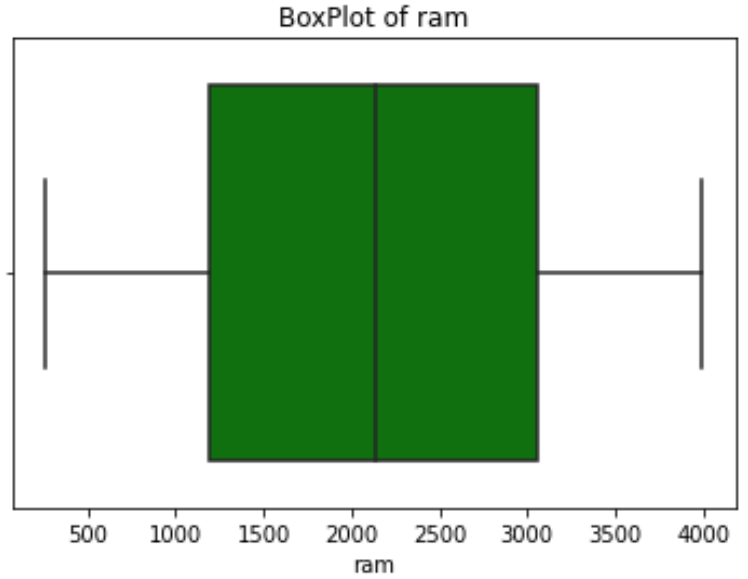


ram

```
#_______ 15 ram cont
df.ram.value_counts()

sns.countplot(df.ram, palette='rainbow')
plt.title('ram')

sns.distplot(df.ram, color='g')
plt.title('ram')

sns.boxplot(df.ram, color='g')
plt.title('BoxPlot of ram')
```

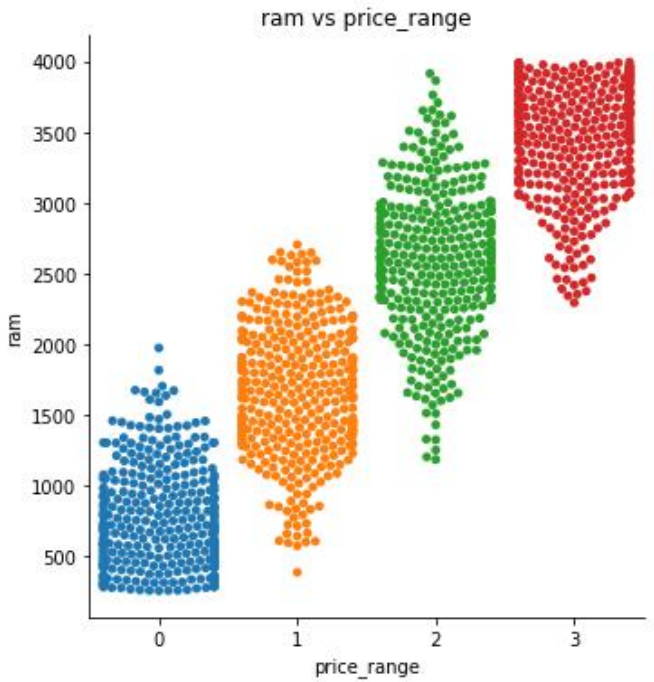


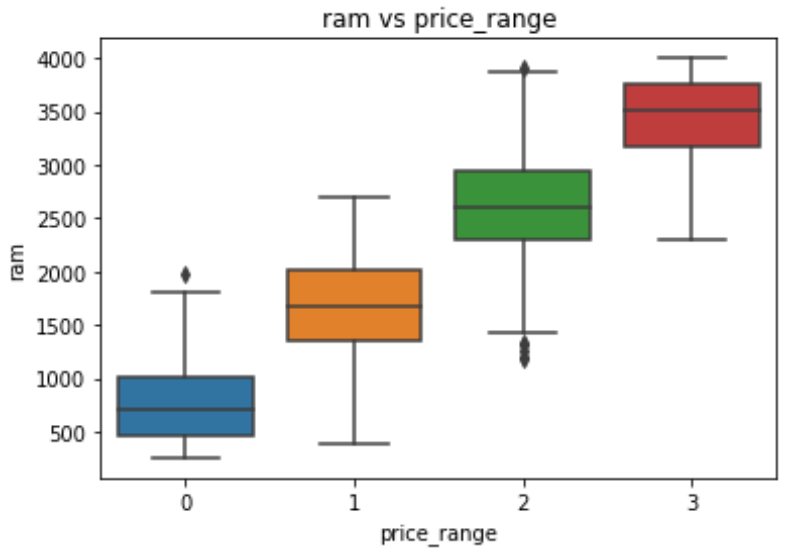


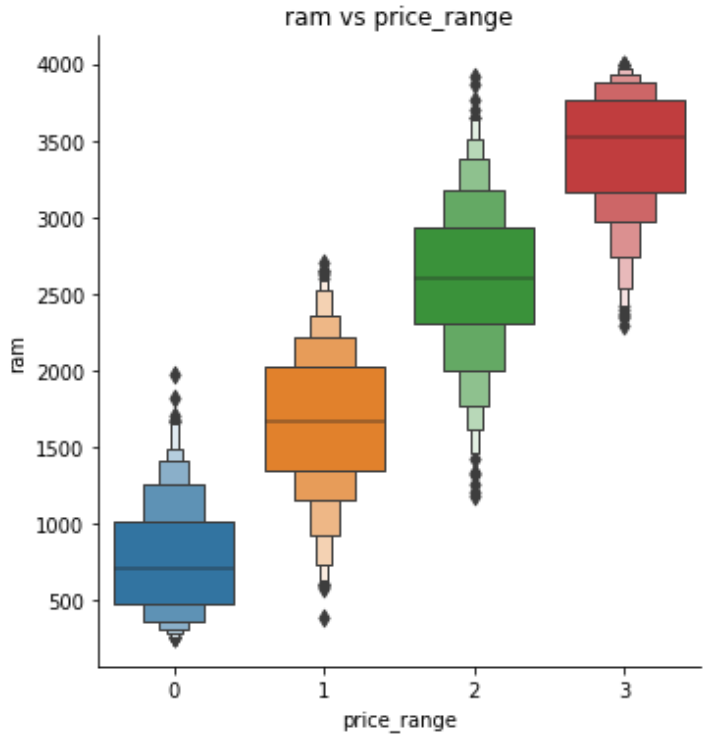


```
#_________groupby plot
sns.catplot(x="price_range", y="ram", kind="swarm", data=df)
plt.title('ram vs price_range')

sns.boxplot(x="price_range", y="ram", data = df)
plt.title('ram vs price_range')

sns.catplot(x="price_range", y="ram", kind="boxen", data=df)
plt.title('ram vs price_range')
```

```
In [218]: df.ram.groupby(df.price_range).mean()
Out[218]:
price_range
0     771.377143
1    1668.328571
2    2590.646724
3    3443.498567
Name: ram, dtype: float64
```

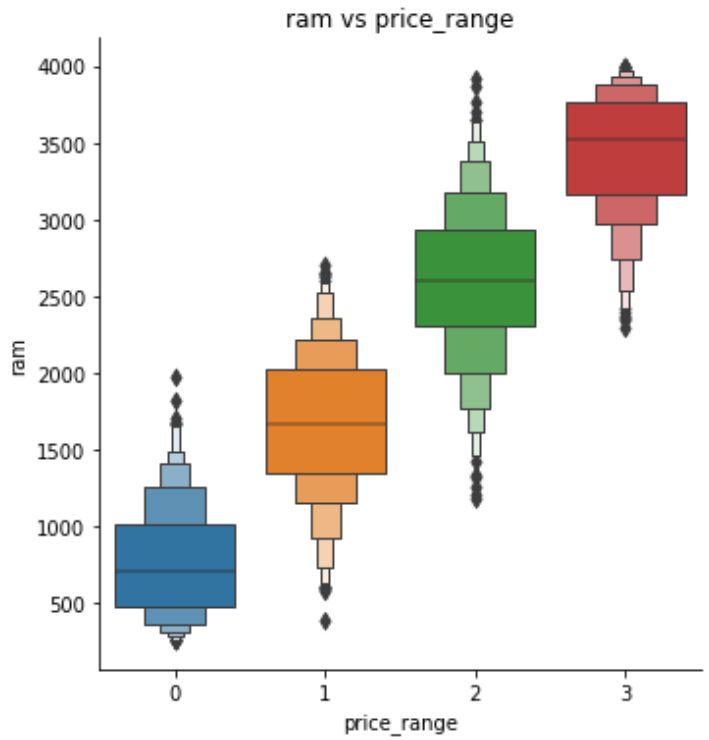






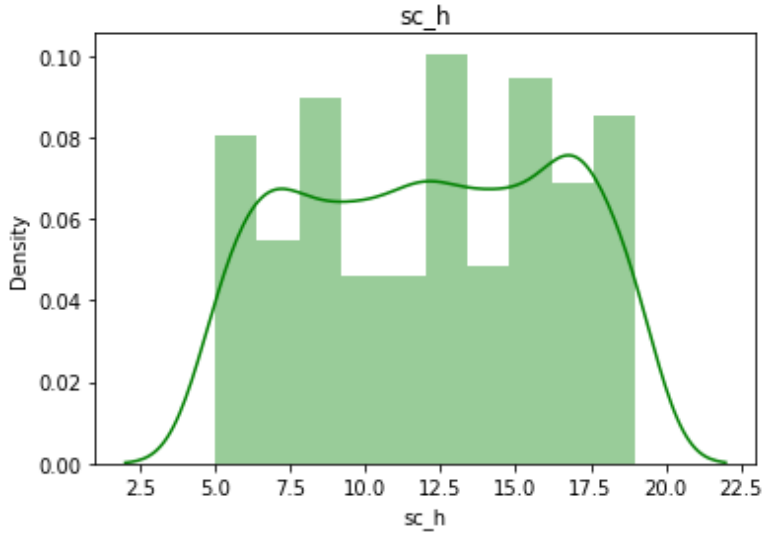
```
In [219]: import statsmodels.api as sm

In [220]: from statsmodels.formula.api import ols

In [221]: mod = ols('ram ~ price_range', data = df).fit()

In [222]: sm.stats.anova_lm(mod)
Out[222]:
                  df         sum_sq        mean_sq             F  PR(>F)
price_range      1.0  1.396717e+09  1.396717e+09  7313.090503     0.0
Residual      1398.0  2.670021e+08  1.909886e+05           NaN     NaN

In [223]: from statsmodels.stats.multicomp import pairwise_tukeyhsd

In [224]: rslt = pairwise_tukeyhsd(df.ram, df.price_range, alpha = 0.0

In [225]: print(rslt)
  Multiple Comparison of Means - Tukey HSD, FWER=0.05
====================================================
group1 group2  meandiff p-adj   lower     upper   reject
----------------------------------------------------
     0      1  896.9514 0.001  811.9701  981.9327   True
     0      2 1819.2696 0.001 1734.3488 1904.1904   True
     0      3 2672.1214 0.001 2587.0793 2757.1636   True
     1      2  922.3182 0.001  837.3974 1007.2389   True
     1      3   1775.17 0.001 1690.1278 1860.2122   True
     2      3  852.8518 0.001  767.8702  937.8335   True
----------------------------------------------------
```

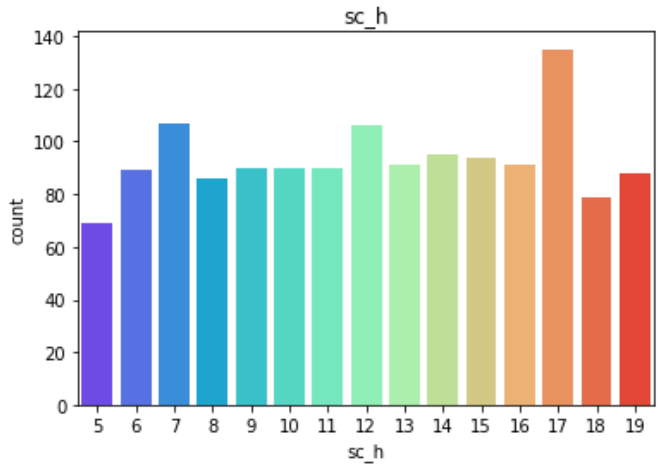
Good predictor!

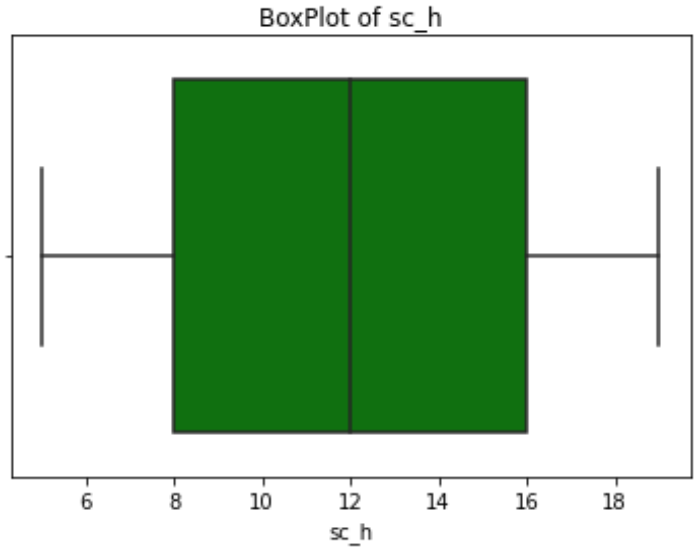


sc_h

```
df.sc_h.value_counts()

sns.countplot(df.sc_h, palette='rainbow')
plt.title('sc_h')

sns.distplot(df.sc_h, color='g')
plt.title('sc_h')

sns.boxplot(df.sc_h, color='g')
plt.title('BoxPlot of sc_h')
```

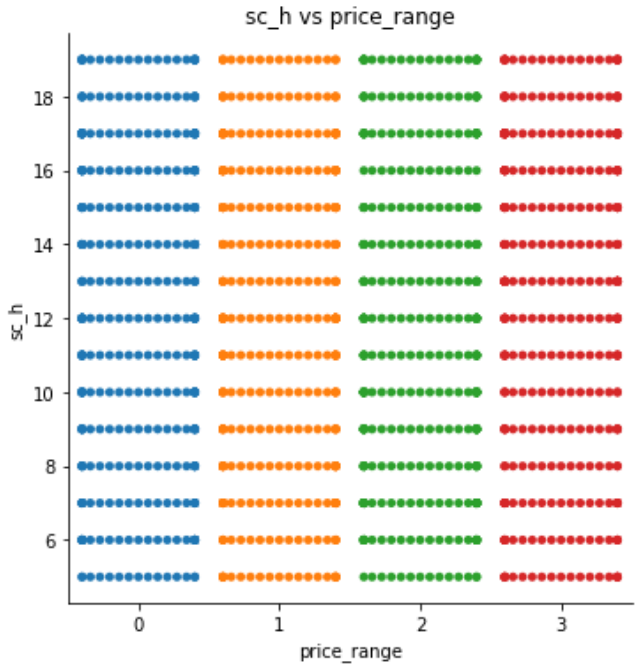


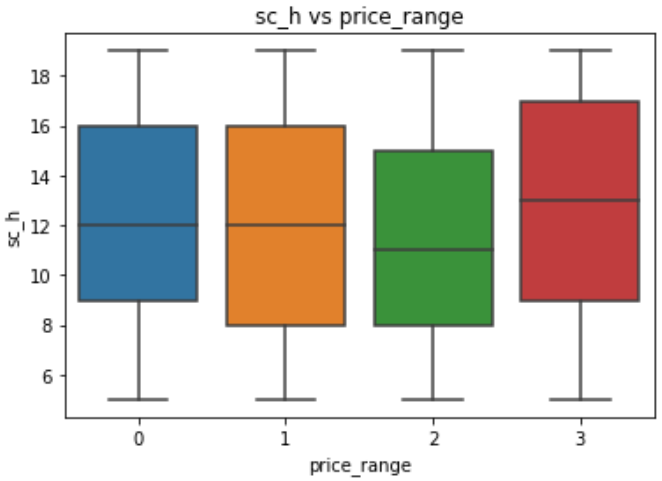


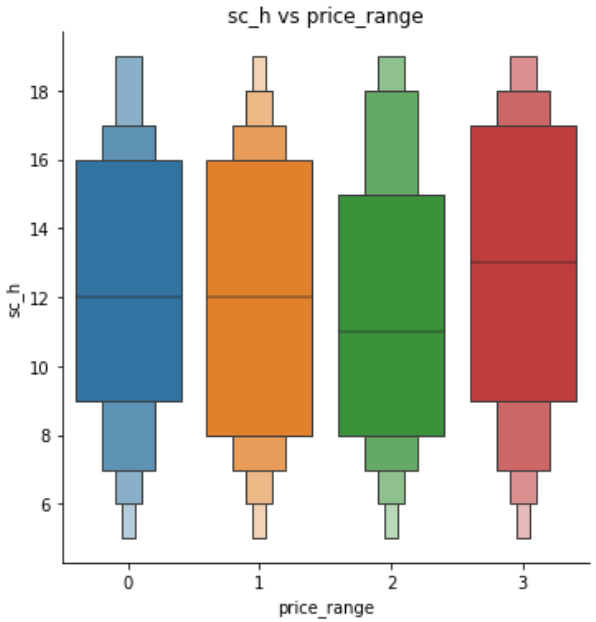


```
#_________groupby plot
sns.catplot(x="price_range", y="sc_h", kind="swarm", data=df)
plt.title('sc_h vs price_range')

sns.boxplot(x="price_range", y="sc_h", data = df)
plt.title('sc_h vs price_range')

sns.catplot(x="price_range", y="sc_h", kind="boxen", data=df)
plt.title('sc_h vs price_range')
```

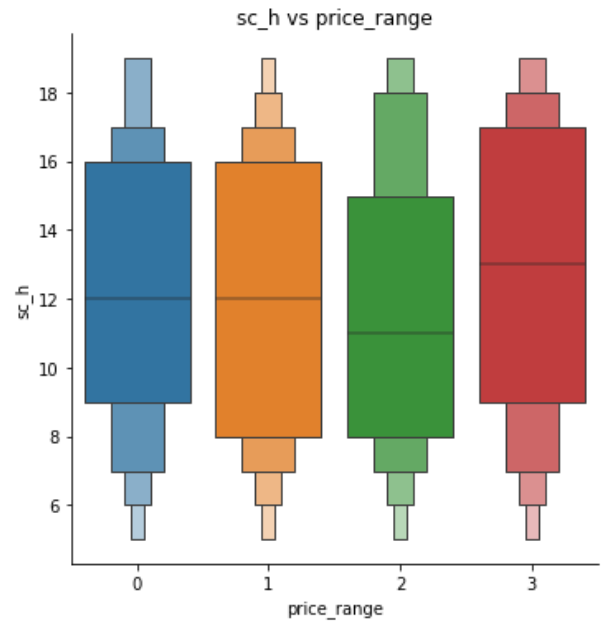
```
In [234]: df.sc_h.groupby(df.price_range).mean()
Out[234]:
price_range
0    12.268571
1    12.062857
2    11.769231
3    12.633238
Name: sc_h, dtype: float64
```





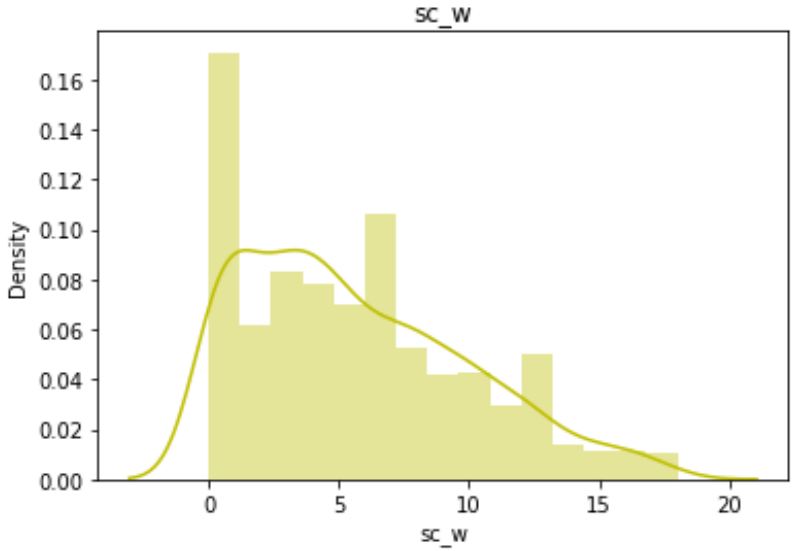


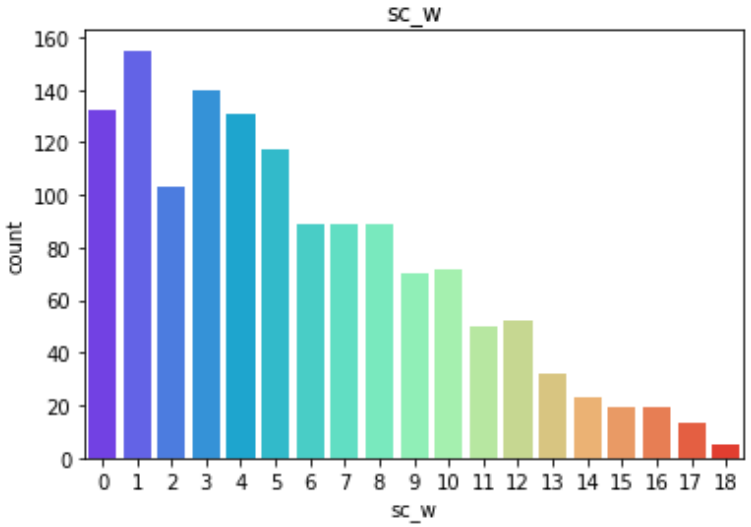
```
In [235]: import statsmodels.api as sm

In [236]: from statsmodels.formula.api import ols

In [237]: mod = ols('sc_h ~ price_range', data = df).fit()

In [238]: sm.stats.anova_lm(mod)
Out[238]:
                 df         sum_sq    mean_sq         F    PR(>F)
price_range     1.0      11.082309  11.082309  0.61622  0.432588
Residual     1398.0   25142.106263  17.984339       NaN       NaN

In [239]: from statsmodels.stats.multicomp import pairwise_tukeyhsd

In [240]: rslt = pairwise_tukeyhsd(df.sc_h, df.price_range, alpha = 0.05)

In [241]: print(rslt)
Multiple Comparison of Means - Tukey HSD, FWER=0.05
===================================================
group1 group2 meandiff p-adj   lower  upper  reject
---------------------------------------------------
     0      1  -0.2057    0.9 -1.0288 0.6173  False
     0      2  -0.4993 0.4024 -1.3218 0.3231  False
     0      3   0.3647 0.6461  -0.459 1.1883  False
     1      2  -0.2936 0.7698 -1.1161 0.5288  False
     1      3   0.5704  0.283 -0.2533  1.394  False
     2      3    0.864 0.0353  0.0409 1.6871   True
---------------------------------------------------
```

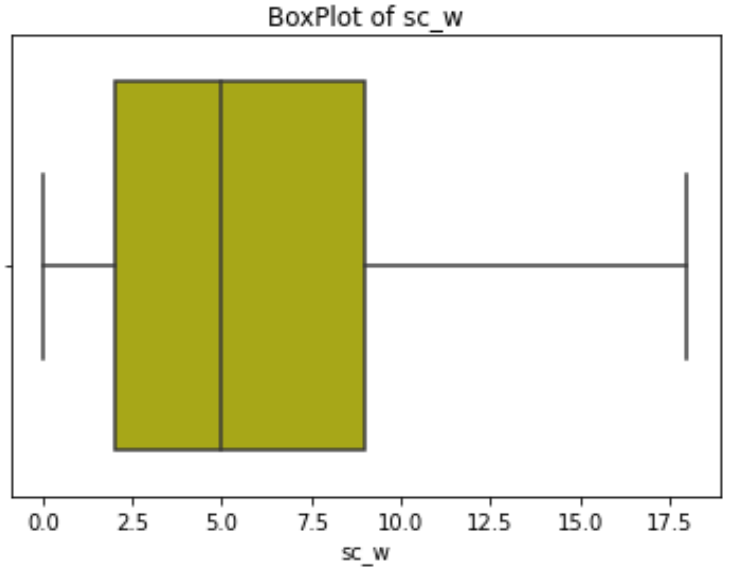


SC_W

```
df.sc_w.value_counts()

sns.countplot(df.sc_w, palette='rainbow')
plt.title('sc_w')

sns.distplot(df.sc_w, color='y')
plt.title('sc_w')

sns.boxplot(df.sc_w, color='y')
plt.title('BoxPlot of sc_w')
```

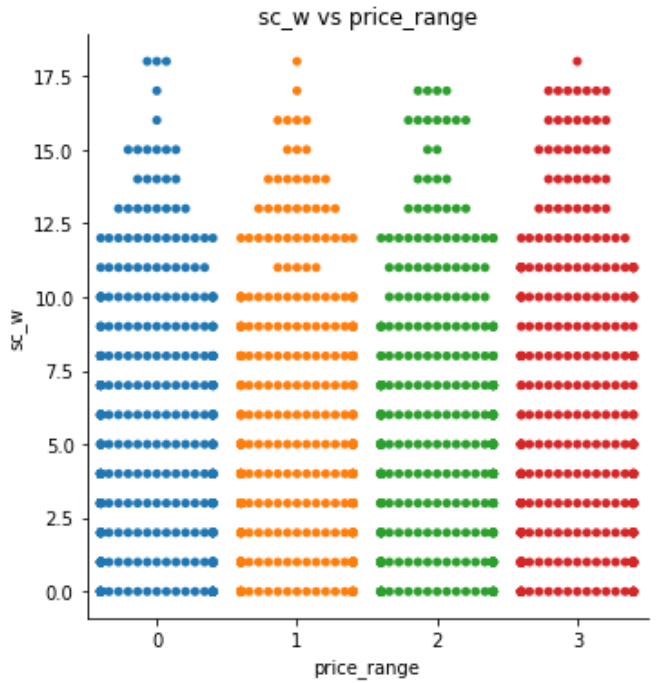


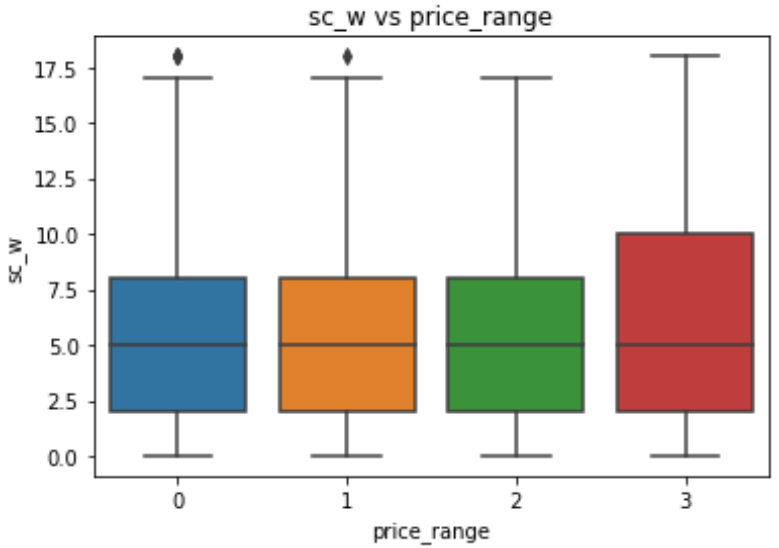


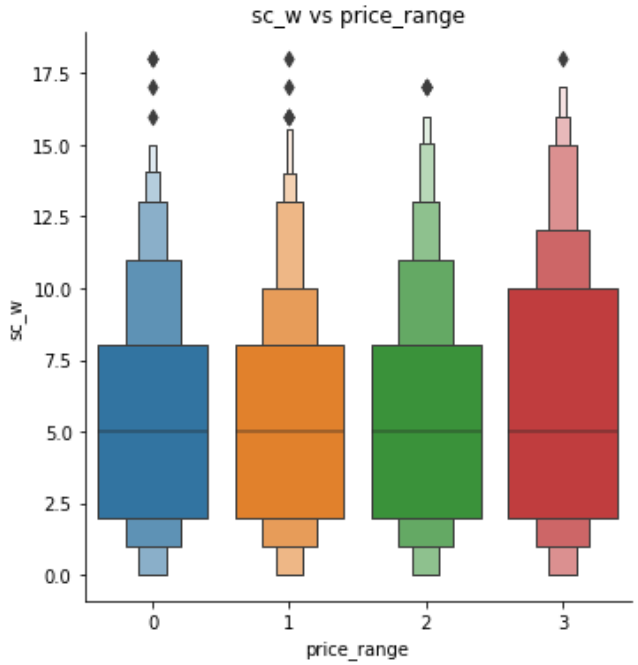


```
#_________groupby plot
sns.catplot(x="price_range", y="sc_w", kind="swarm", data=df)
plt.title('sc_w vs price_range')

sns.boxplot(x="price_range", y="sc_w", data = df)
plt.title('sc_w vs price_range')

sns.catplot(x="price_range", y="sc_w", kind="boxen", data=df)
plt.title('sc_w vs price_range')
```

```
In [251]: df.sc_w.groupby(df.price_range).mean()
Out[251]:
price_range
0    5.594286
1    5.468571
2    5.564103
3    6.005731
Name: sc_w, dtype: float64
```

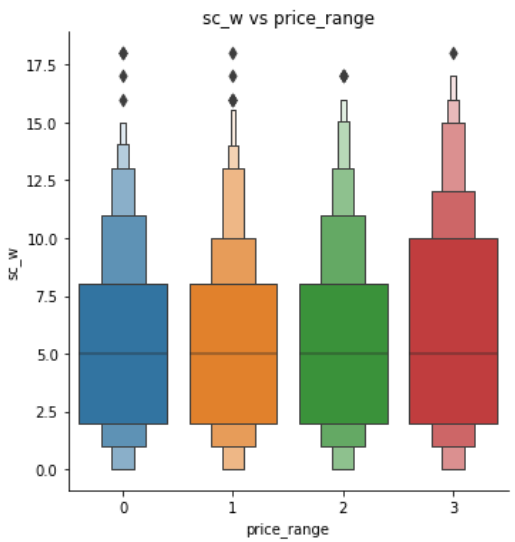






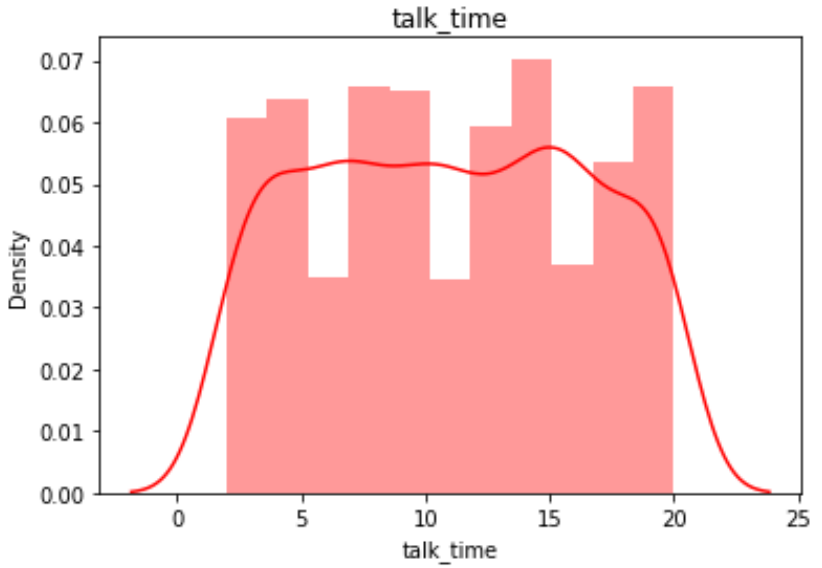
```
In [252]: import statsmodels.api as sm

In [253]: from statsmodels.formula.api import ols

In [254]: mod = ols('sc_w ~ price_range', data = df).fit()

In [255]: sm.stats.anova_lm(mod)
Out[255]:
                  df         sum_sq    mean_sq          F    PR(>F)
price_range      1.0      30.833691  30.833691  1.651407  0.19898
Residual      1398.0   26102.279880  18.671159       NaN       NaN

In [256]: from statsmodels.stats.multicomp import pairwise_tukeyhsd

In [257]: rslt = pairwise_tukeyhsd(df.sc_w, df.price_range, alpha = 0.05)

In [258]: print(rslt)
Multiple Comparison of Means - Tukey HSD, FWER=0.05
=================================================
group1 group2 meandiff p-adj   lower  upper  reject
-------------------------------------------------
     0      1  -0.1257    0.9  -0.966 0.7146  False
     0      2  -0.0302    0.9 -0.8699 0.8095  False
     0      3   0.4114 0.5789 -0.4295 1.2523  False
     1      2   0.0955    0.9 -0.7442 0.9352  False
     1      3   0.5372 0.3553 -0.3037 1.3781  False
     2      3   0.4416 0.5265 -0.3987 1.2819  False
-------------------------------------------------
```

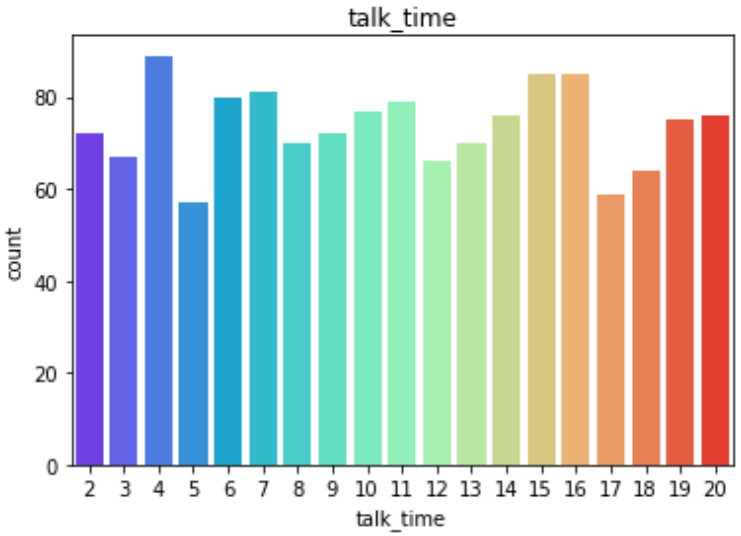
Not Good predictor!

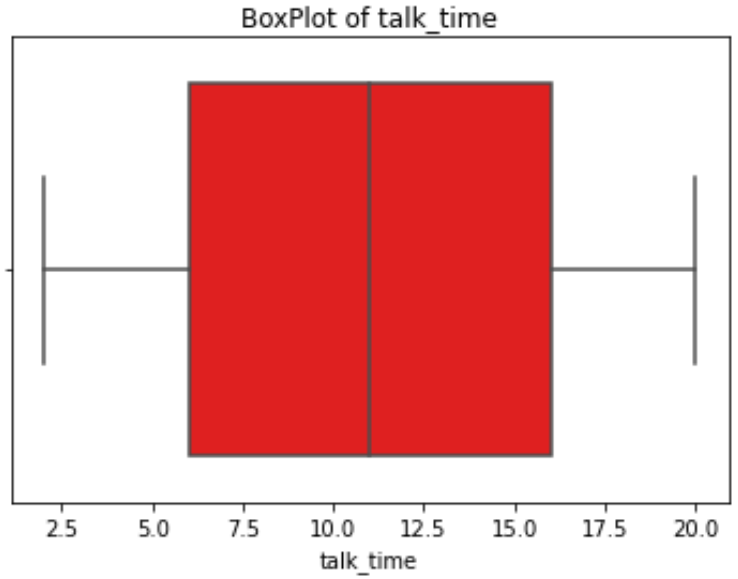


talk_time

```
df.talk_time.value_counts()

sns.countplot(df.talk_time, palette='rainbow')
plt.title('talk_time')

sns.distplot(df.talk_time, color='r')
plt.title('talk_time')

sns.boxplot(df.talk_time, color='r')
plt.title('BoxPlot of talk_time')
```

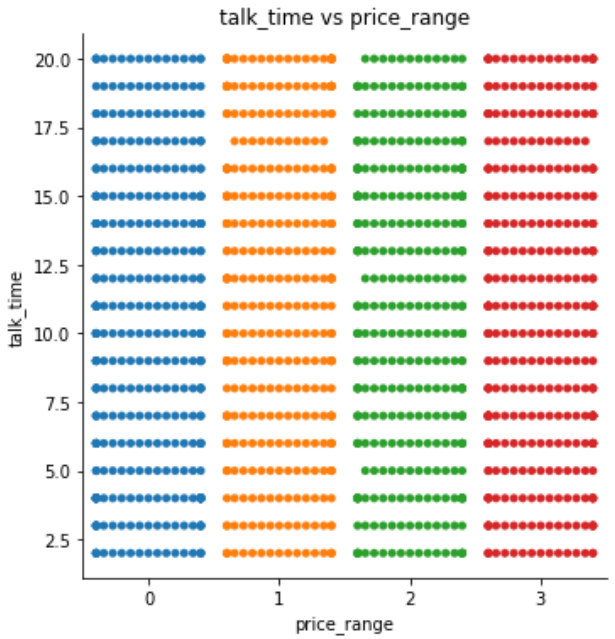


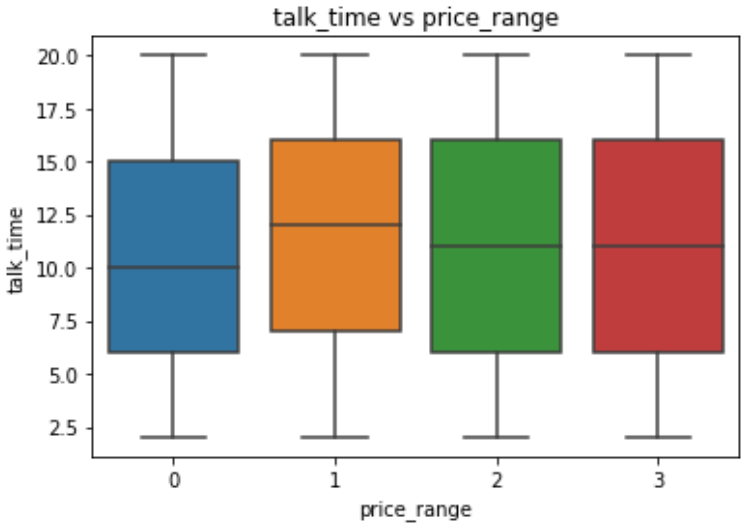


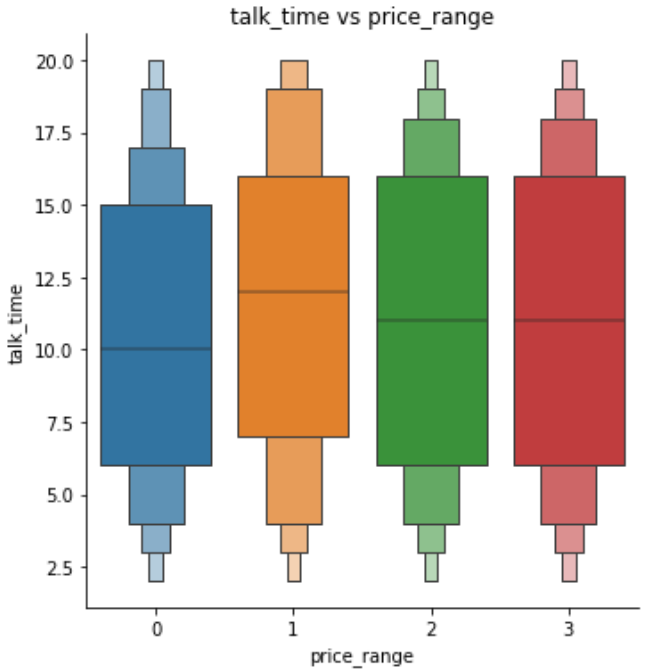


```
#_________groupby plot
sns.catplot(x="price_range", y="talk_time", kind="swarm", data=df)
plt.title('talk_time vs price_range')

sns.boxplot(x="price_range", y="talk_time", data = df)
plt.title('talk_time vs price_range')

sns.catplot(x="price_range", y="talk_time", kind="boxen", data=df)
plt.title('talk_time vs price_range')
```

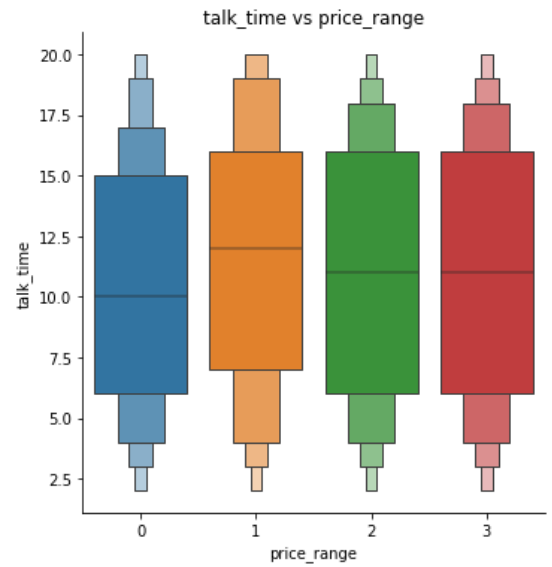
```
In [267]: df.talk_time.groupby(df.price_range).mean()
Out[267]:
price_range
0    10.468571
1    11.562857
2    11.054131
3    10.859599
Name: talk_time, dtype: float64
```







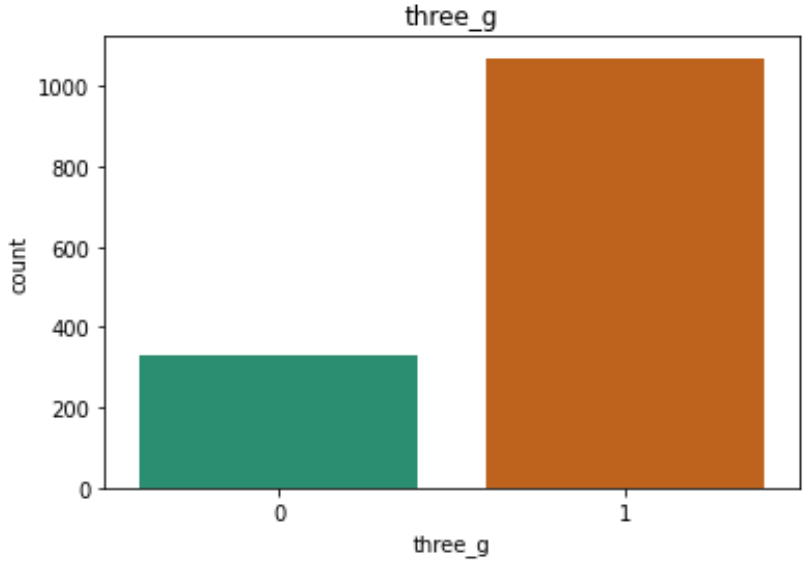
```
In [268]: mod = ols('talk_time ~ price_range', data = df).fit()

In [269]: sm.stats.anova_lm(mod)
Out[269]:
                df         sum_sq    mean_sq          F    PR(>F)
price_range    1.0       7.762639   7.762639   0.261202  0.609376
Residual    1398.0   41546.979504  29.718869        NaN       NaN

In [270]: from statsmodels.stats.multicomp import pairwise_tukeyhsd

In [271]: rslt = pairwise_tukeyhsd(df.talk_time, df.price_range, alpha = 0.05)

In [272]: print(rslt)
Multiple Comparison of Means - Tukey HSD, FWER=0.05
=================================================
group1 group2 meandiff p-adj   lower   upper  reject
-------------------------------------------------
     0      1   1.0943 0.0395  0.0362 2.1523    True
     0      2   0.5856 0.4847 -0.4717 1.6429   False
     0      3    0.391  0.752 -0.6678 1.4498   False
     1      2  -0.5087 0.5906  -1.566 0.5486   False
     1      3  -0.7033 0.3197 -1.7621 0.3555   False
     2      3  -0.1945    0.9 -1.2526 0.8635   False
-------------------------------------------------
```



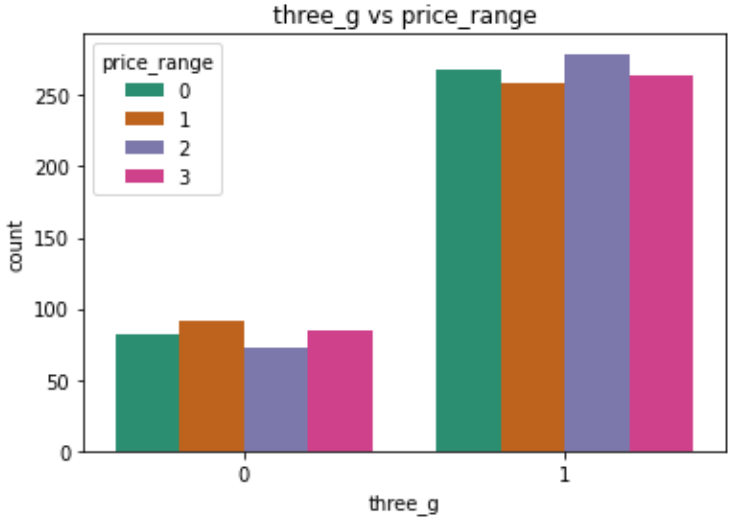
three_g

```
In [274]: df.three_g.value_counts()
Out[274]:
1    1070
0     330
Name: three_g, dtype: int64
```

```
sns.countplot(df.three_g, palette='Dark2')
plt.title('three_g')
```




```
sns.countplot(df.three_g, hue=df.price_range, palette='Dark2')
plt.title('three_g vs price_range')
```

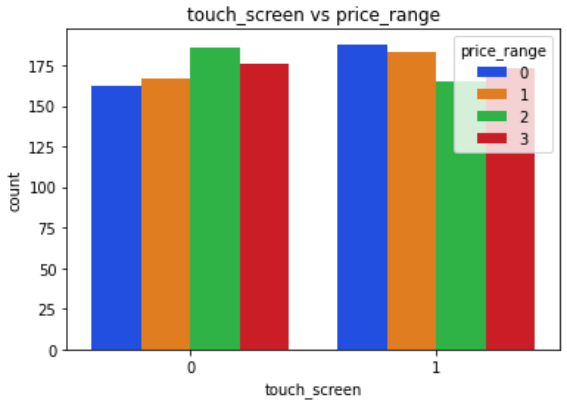



```
In [277]: from scipy.stats import chi2, chi2_contingency

In [278]: ct_three_g = pd.crosstab(df.three_g, df.price_range)

In [279]: ct_three_g
Out[279]:
price_range    0    1    2    3
three_g
0             82   91   72   85
1            268  259  279  264

In [280]: chi2_contingency(ct_three_g, correction=False)
Out[280]:
(3.0915421687453892,
 0.37772539284899026,
 3,
 array([[ 82.5       ,  82.5       ,  82.73571429,  82.26428571],
        [267.5       , 267.5       , 268.26428571, 266.73571429]]))
```

Not Good predictor!

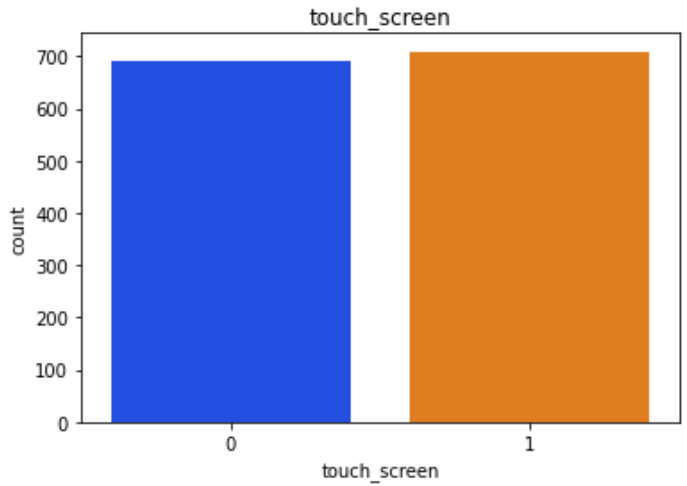
touch_screen

```
sns.countplot(df.touch_screen, hue=df.price_range, palette='bright')
plt.title('touch_screen vs price_range')
```




```
In [284]: df.touch_screen.value_counts()
Out[284]:
1    709
0    691
Name: touch_screen, dtype: int64
```

```
sns.countplot(df.touch_screen, palette='bright')
plt.title('touch_screen')
```



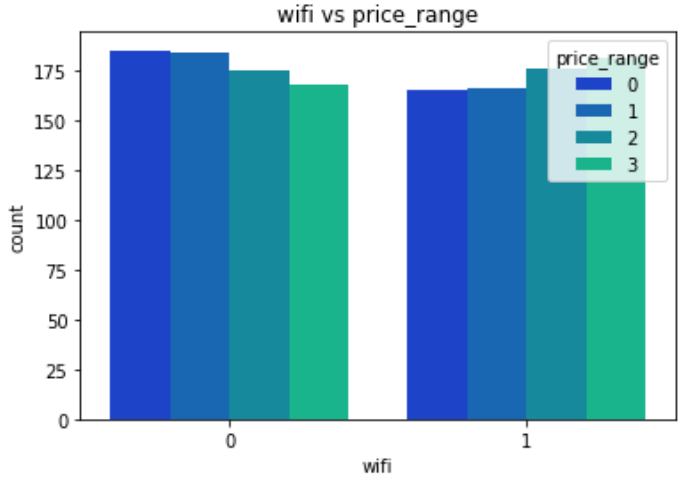

```
In [286]: from scipy.stats import chi2, chi2_contingency

In [287]: ct_touch_screen = pd.crosstab(df.touch_screen, df.price_range)

In [288]: ct_touch_screen
Out[288]:
price_range     0    1    2    3
touch_screen
0             162  167  186  176
1             188  183  165  173

In [289]: chi2_contingency(ct_touch_screen, correction=False)
Out[289]:
(3.7142407801960866,
 0.2940201872038134,
 3,
 array([[172.75      , 172.75      , 173.24357143, 172.25642857],
        [177.25      , 177.25      , 177.75642857, 176.74357143]]))
```

Not Good predictor!

Dr Vinod on EDA 8971073111 vinodanalytics@gmail.com

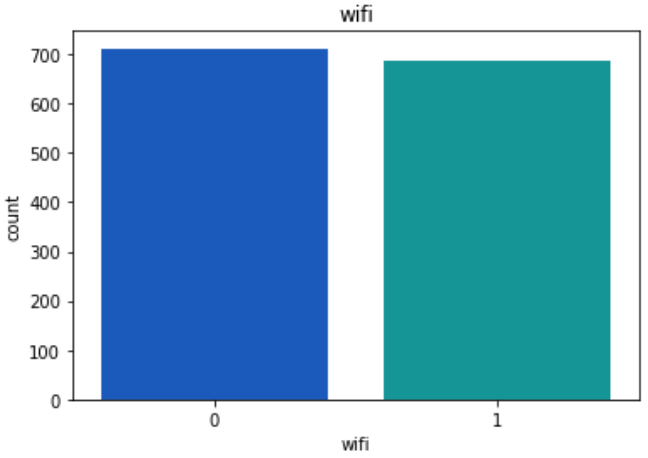
wifi

```
sns.countplot(df.wifi, hue=df.price_range, palette='winter')
plt.title('wifi vs price_range')
```




```
In [291]: df.wifi.value_counts()
Out[291]:
0    712
1    688
Name: wifi, dtype: int64
```

```
sns.countplot(df.wifi, palette='winter')
plt.title('wifi')
```




```
In [294]: from scipy.stats import chi2, chi2_contingency

In [295]: ct_wifi = pd.crosstab(df.wifi, df.price_range)

In [296]: ct_wifi
Out[296]:
price_range    0    1    2    3
wifi
0            185  184  175  168
1            165  166  176  181

In [297]: chi2_contingency(ct_wifi, correction=False)
Out[297]:
(2.1448628747198404,
 0.5428901494541059,
 3,
 array([[178.        , 178.        , 178.50857143, 177.49142857],
        [172.        , 172.        , 172.49142857, 171.50857143]]))
```

Not Good predictor!


HAPPINESS IS ...
... learning new skills