Capstone Project

Mobile Price Range Prediction

(Classification Analysis)

Team members

Kajal Dhun

Navinkumar Sambari

Tanu Rajput

Problem statement

Mobile prices are an important reflection of the Humans and some ranges are of great interest for both buyers and sellers. In the competitive mobile phone market, companies want to understand sales data of mobile phones and factors which drive the prices. Price estimation and prediction is an important part of consumer strategy. Deciding on the correct price of a product is very important for the market success of a product. So the objective is to find out some relation between features of a mobile phone and its selling price.

This project will classify the price range of the mobile price indicating how high the price is.

Content

- Data description
- Data Wrangling -
    1) Talking about the mismatch columns and its handling.
    2) Outlier treatment
- Exploratory Data Analysis
- Correlation Analysis
- Models' List and their evaluation metrics
- Model selection
- Challenges faced
- Conclusion

Data Description

	Mobile_price_range_data	
Numerical Attributes	Categorical attributes	Target feature
battery_power	Blue	Price_range
clock_speed	Dual_sim	
fc	Four_g	
int_memory	Three_g	
m_dep	Touch_screen	
mobile_wt	Wifi	
n_cores		
pc		
px_height		
px_width		
ram		
sc_h		
sc_w		
talk_time		

Let's take a peek at the dataset

- This dataset contains 2000 rows and 21 columns
- As we can see no null values in any attribute
- It was observed that the categorical attributes are already encoded so we did not require any feature encoding process here.
- Our target feature is Price_range and rest features are the independent attributes.

Dataset Shape: (2000, 21)

	Name	dtypes	Missing	Uniques
0	battery_power	int64	0	1094
11	px_height	int64	0	1137
19	wifi	int64	0	2
18	touch_screen	int64	0	2
17	three_g	int64	0	2
16	talk_time	int64	0	19
15	sc_w	int64	0	19
14	sc_h	int64	0	15
13	ram	int64	0	1562
12	px_width	int64	0	1109
10	pc	int64	0	21
1	blue	int64	0	2
9	n_cores	int64	0	8
8	mobile_wt	int64	0	121
7	m_dep	float64	0	10
6	int_memory	int64	0	63
5	four_g	int64	0	2
4	fc	int64	0	20
3	dual_sim	int64	0	2
2	clock_speed	float64	0	26
20	price_range	int64	0	4

Data wrangling - Talking about the mismatch values in columns and its handling.

1. Checking number of observations in both attributes having values as 0.

```
[ ]  # Checking observations having px_hieght value as 0.
     print(f"number of observation having px_height as 0 = {data[data['px_height']==0].shape[0]}")
     # Checking observations having screen width value as 0.
     print(f"number of observation having screen width value as 0 = {data[data['sc_w']==0].shape[0]
```

```
number of observation having px_height as 0 = 2
number of observation having screen width value as 0 = 180
```

2. Handling the 'px_height' attribute which contains zero as values.

```
[ ]  # As there are only 2 observations having px_height=0. so just drop it.
     data.drop(data[data['px_height']==0].index, inplace=True)
```

3. Handling the 'sc_w' attribute which contains zero as values.

- It contains around 179 rows with zero value.
- So we decide to replace method then we used KNN imputation method.

- So after handling such mismatch errors, the shape of the dataset shows that 1998 rows and 21 columns

```
[ ]  # Replacing 0 with NAN so that we can implement KNN Imputer.
     data['sc_w']=data['sc_w'].replace(0,np.nan)
```

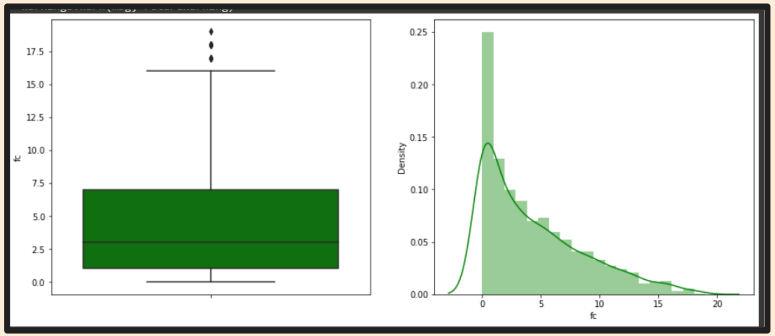
```
[ ]  # import KNN imputer from sklearn
     from sklearn.impute import KNNImputer
     impute_knn = KNNImputer(n_neighbors=1)
     data=pd.DataFrame(impute_knn.fit_transform(data),columns=data.columns)
```

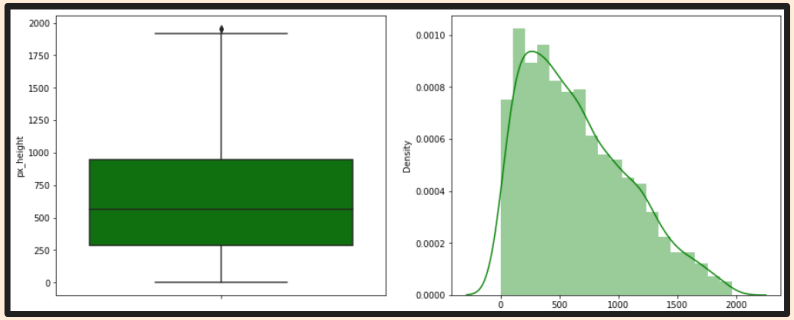
```
[ ]  # Checking shape again
     data.shape

     (1998, 21)
```

Data wrangling – Outlier Treatment

- In the given pictures, columns such as 'fc' and 'px_height' contain outliers.
  So we decided to remove outliers of these columns by using the Quantile method.

- Excluding these two column, rest of the attributes does not contains outliers

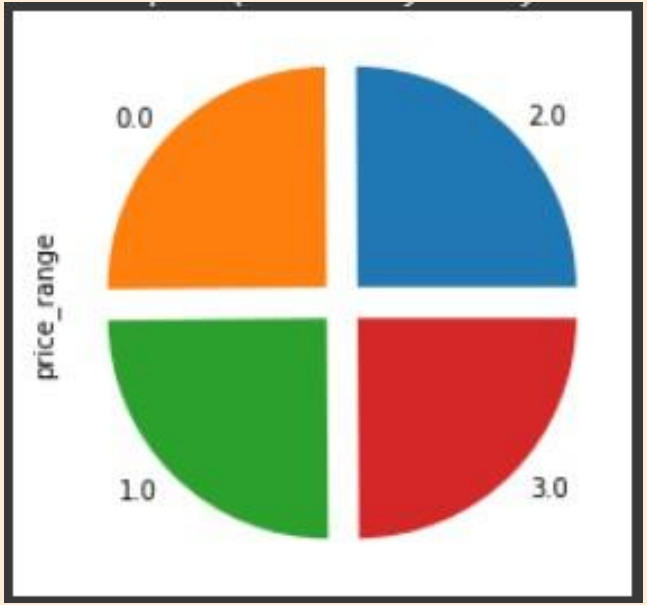




Exploratory Data Analysis

Target variable analysis

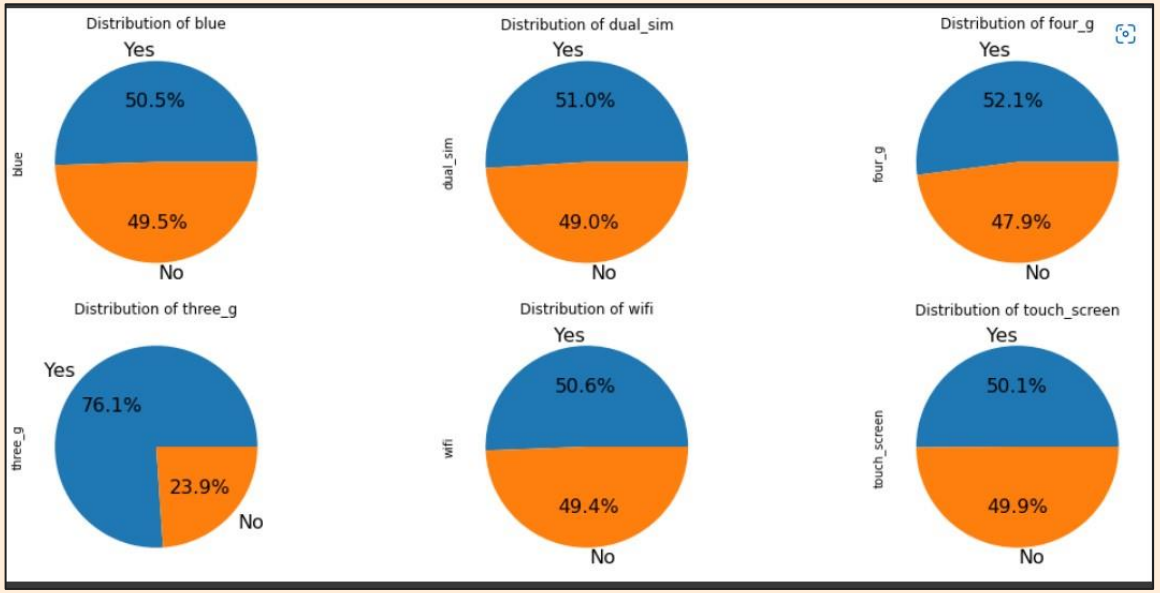
- **Target feature have almost equal number of observation. It is distributed equally.**
- **Percentage distribution of price range is 25% each.**



Binary Categorical variables' analysis

- Percentage distribution of mobiles having 'bluetooth', 'dual_sim', 'wifi' and 'touch screen' are almost 50%.

- Speaking about the 'three_g' attribute, around 76% of the mobiles have the 3G feature and around 23.9% of the mobiles does not have this feature.

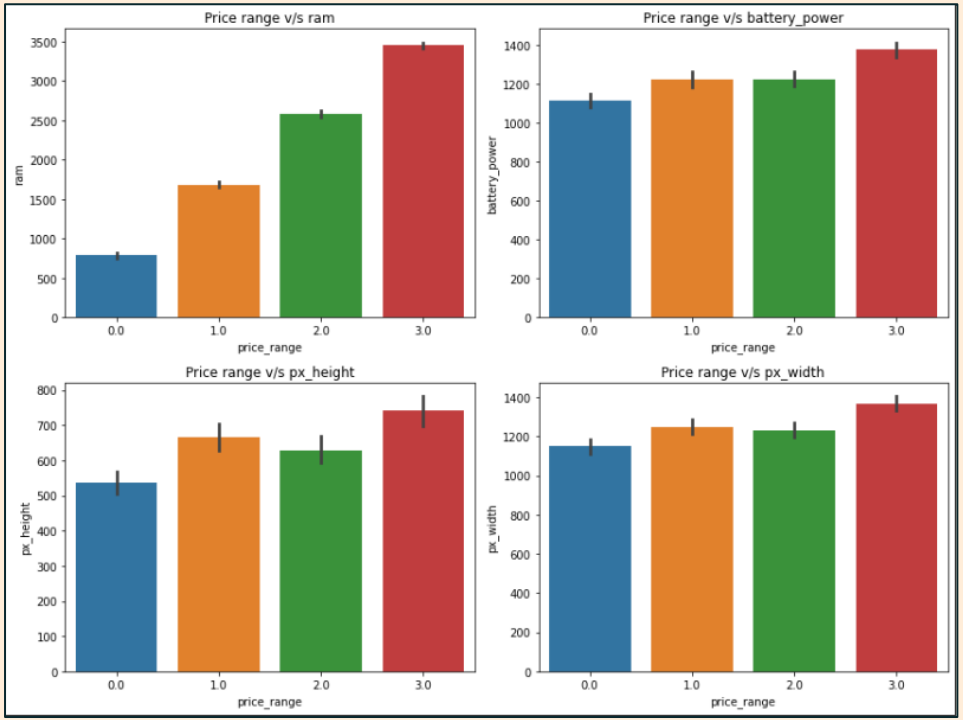


Price_range vs ram, battery power, px_height, px_width

1. Mobiles having RAM more than 3GB comes under 3.0 price_range category, I.e, very high cost category.

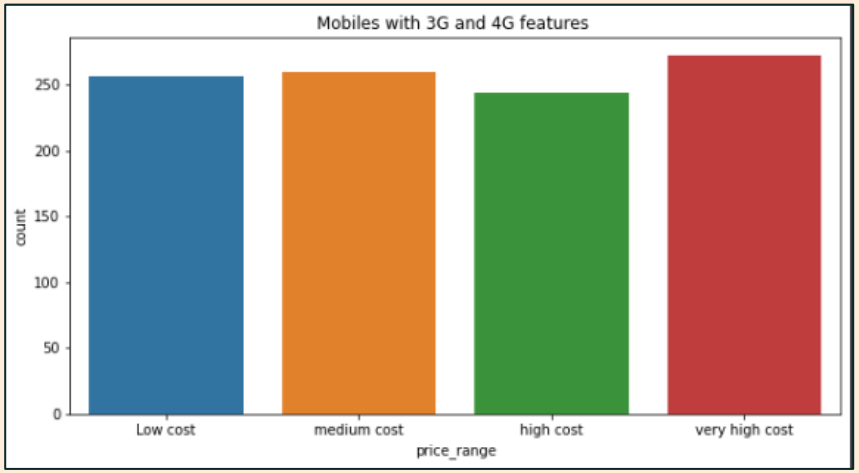
Mobile with battery power more than 1300 mAh comes under very high cost price_range category.

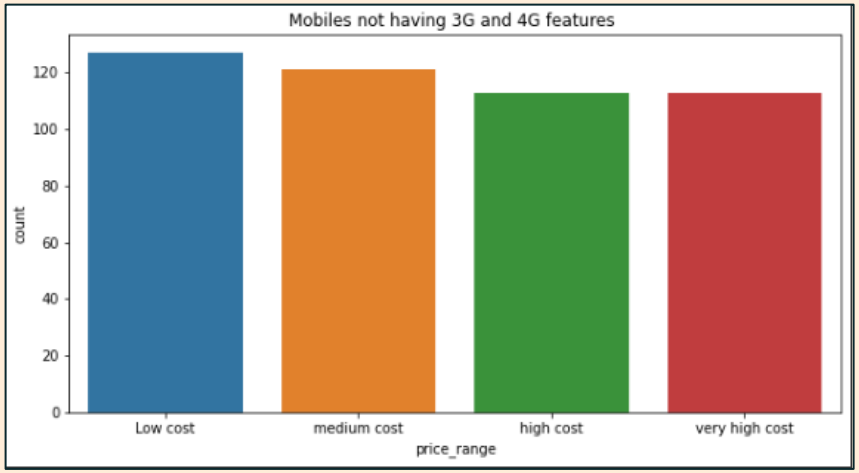
Mobiles with more than 700 pixel heigh and width than 1300 has very high cost.



Count of mobiles contain 3G and 4G features.

In the first pic, the number of mobiles which contains 3G and 4G features is high which falls under very high cost price_range category.

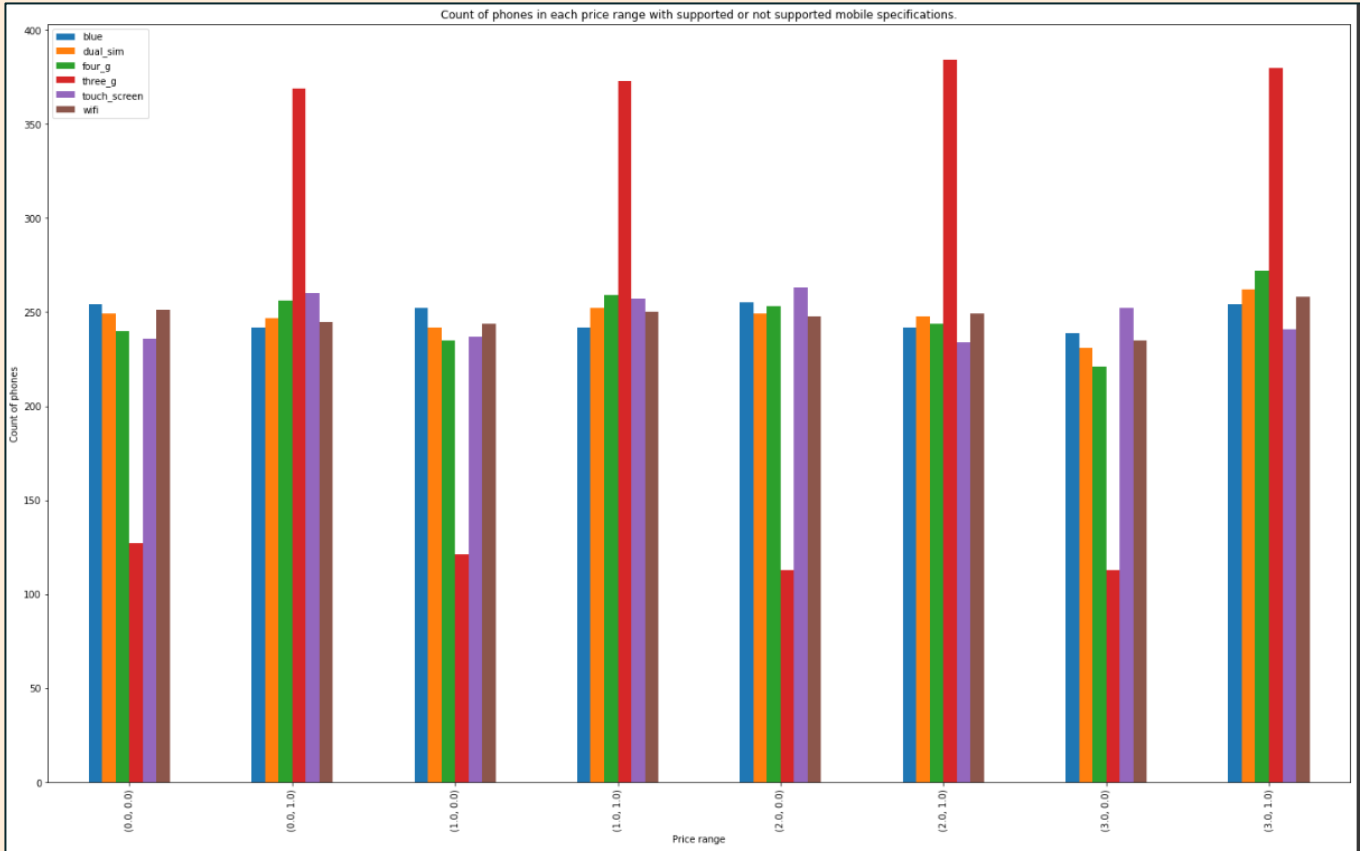




In the second pic, the number of mobiles which does not contains 3G and 4G features is high which falls under very low cost price_range category.

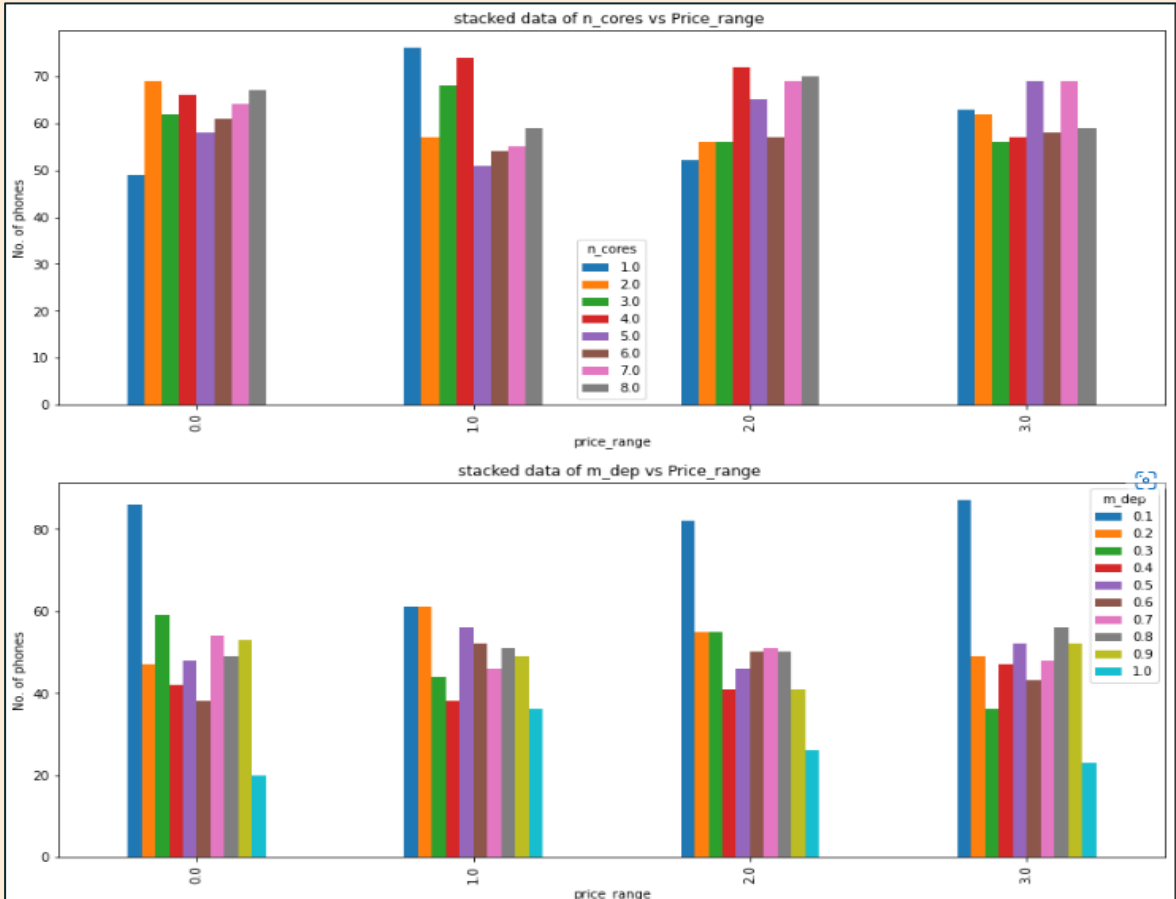
Count of mobiles in each price range with supporting or not supporting specifications.

Each price range category has equal number of mobiles having both supporting and non supporting specifications.

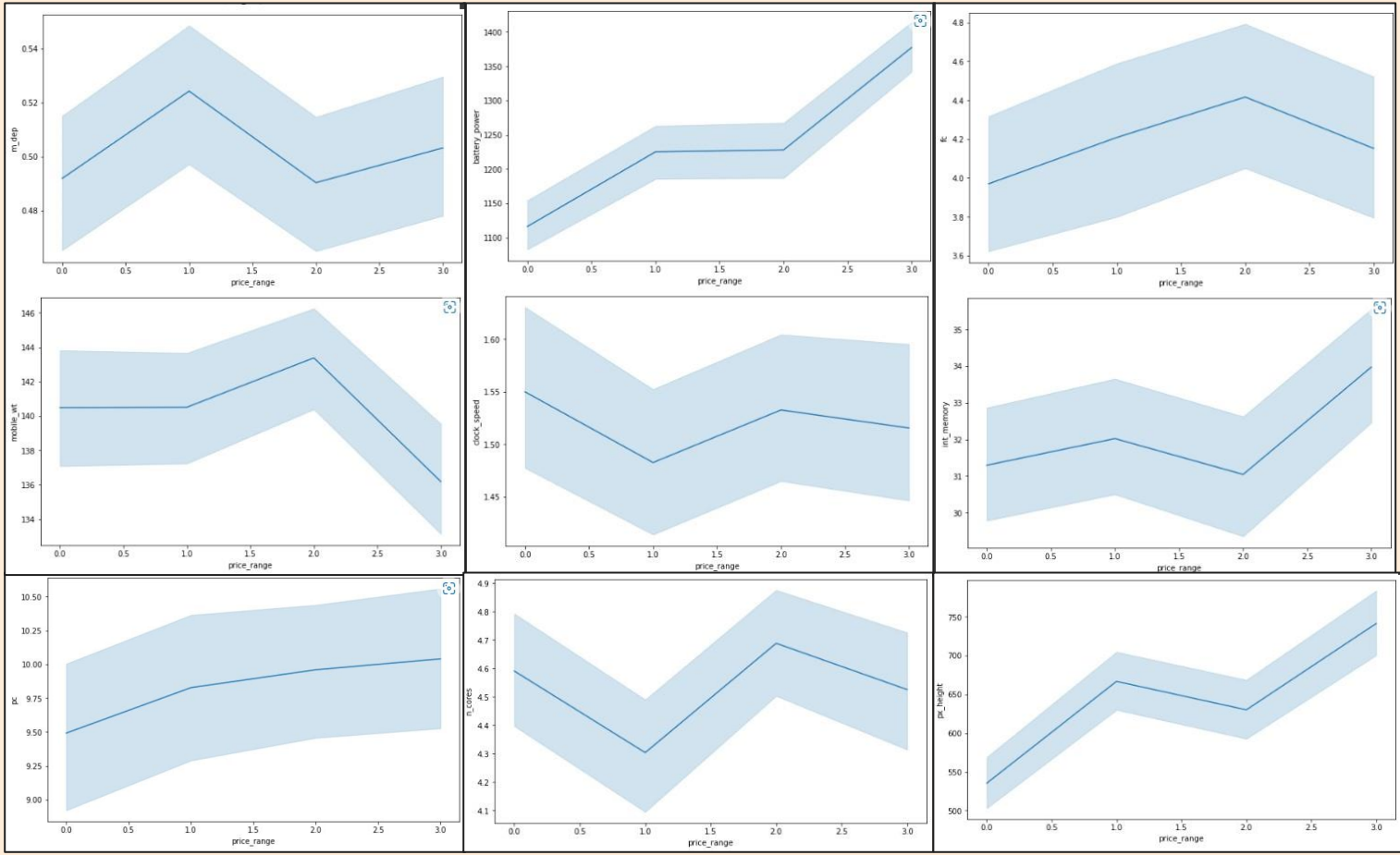


Count of mobiles for each n_cores and m_dep wrt Price_range

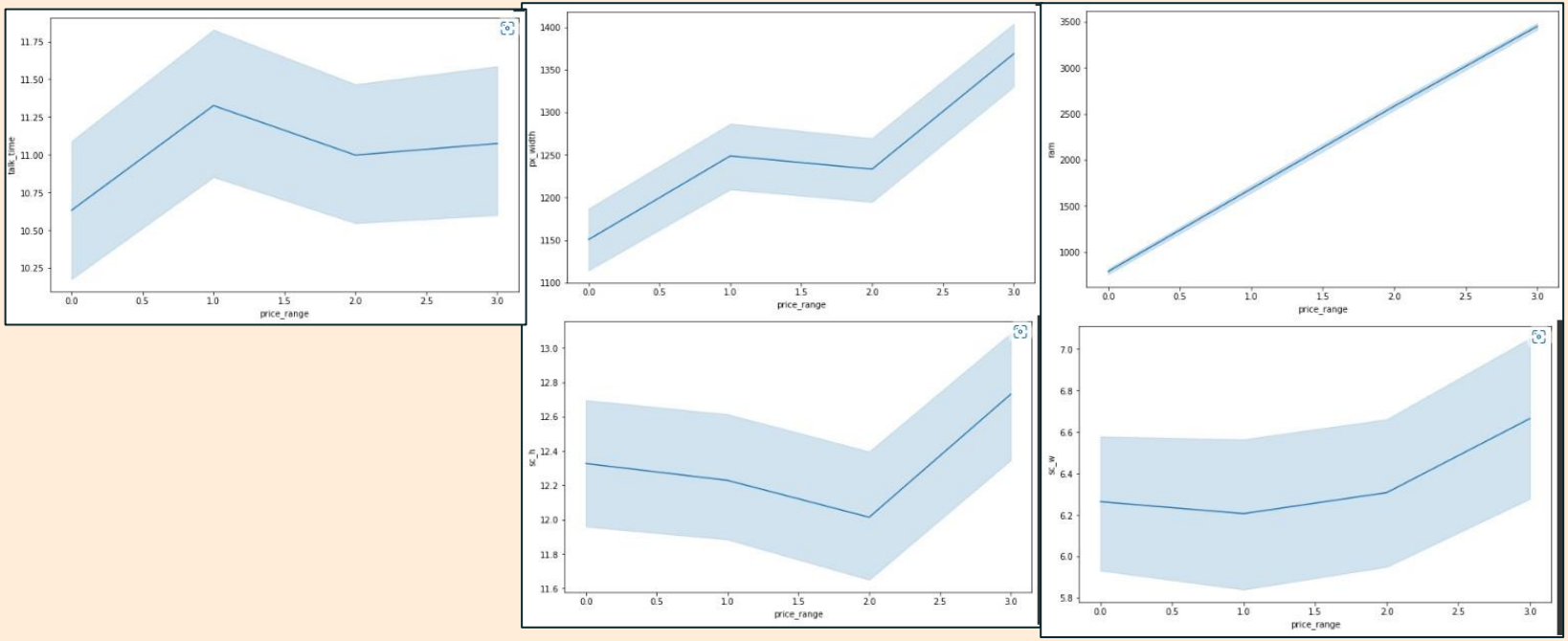
- We can see that there are few mobiles in price range 0 and 1 with lesser number of cores.
- The count of mobiles contains cores 5.0, 7.0 and 8.0 is high.
- Number of mobiles with less thickness is high wrt price range.



Independent features v/s Target feature



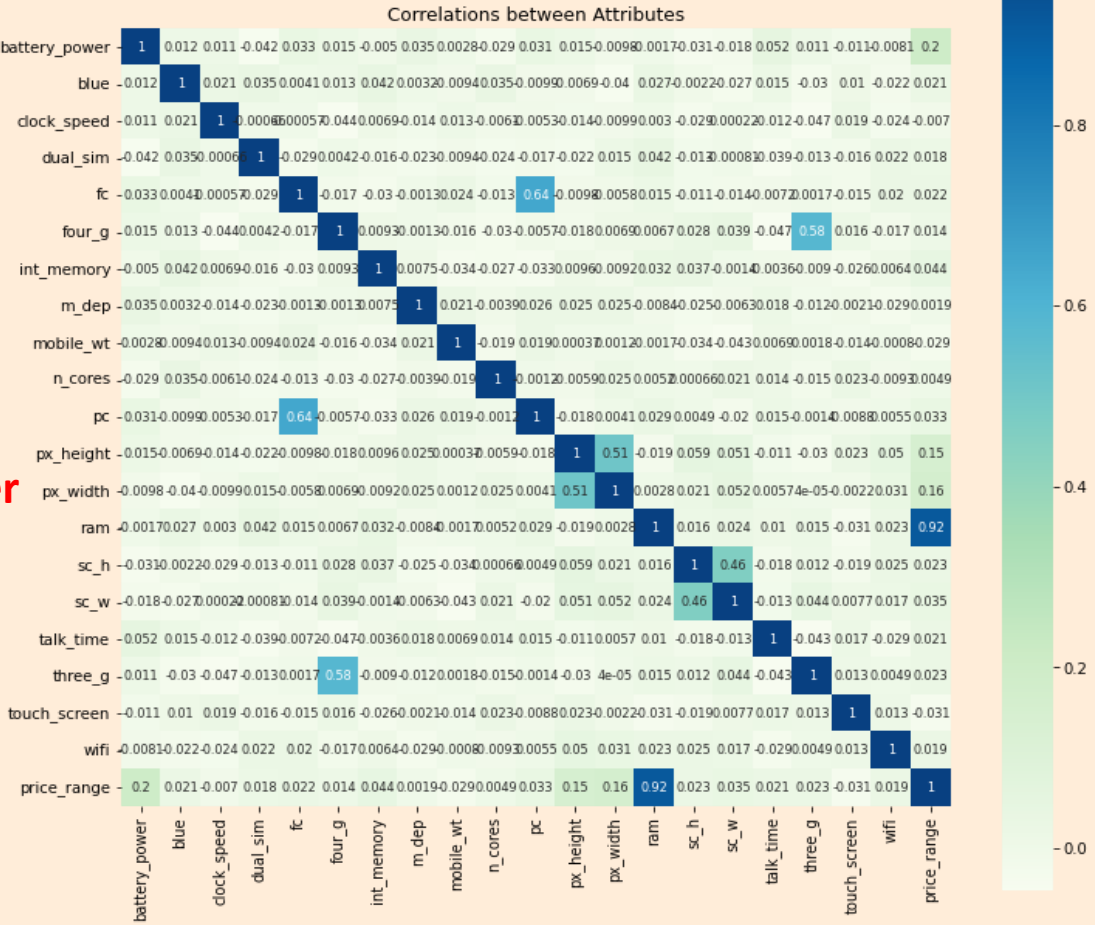
Independent features v/s Target feature



Correlation analysis

Let's talk about relations between some of the attributes....

- price_range and RAM
- pc and fc
- pc_height and pc_width
- price_range and battery_power



Modeling

List of algorithms used:

- Logistic Regression
- Super Vector Machine
- Random Forest
- Decision Tree Classifier
- K Nearest Neighbour

Evaluation metrics with hyperparameter tuning

```
[ ]  metrics_with_hp
```

	model	Train accuracy score	Test accuracy score	Train precision score	Test precision score	Train recall score	Test recall score	Train f1 score	Test f1 score
0	Logistic Regression	0.985859	0.961616	0.985859	0.961616	0.985859	0.961616	0.985859	0.961616
1	superVectormachine	0.986532	0.957576	0.986532	0.957576	0.986532	0.957576	0.986532	0.957576
2	Random Forest	0.999327	0.856566	0.999327	0.856566	0.999327	0.856566	0.999327	0.856566
3	DecisionTreeClassifier	0.956229	0.840404	0.956229	0.840404	0.956229	0.840404	0.956229	0.840404
4	K Nearest Neighbour	0.674074	0.628283	0.674074	0.628283	0.674074	0.628283	0.674074	0.628283

Model selection:

1) According to Model Evaluation metrics dataframe, the test score of the RandomForest and DecisionTreeClassifier are nearly same.
2) So, the best results that we getting from Logistic Regression and SuperVectorMachine.
3) But we are selecting the SuperVectorMachine for model selection and prediction.

Challenges we faced:

- Faced some challenge while handling mismatch values in few columns
- We felt little challenging when we start working on
- different algorithms and its metrics and also choosing quite number of algorithms to work upon.
- Deciding about the best model for prediction.

Conclusion:

- With data exploration we found out that the price range is mostly affected by RAM, px_height, px_width, fc, pc, batter_power, four_g.
- As higher the RAM, higher the PRICE.
- Performed modelling using various algorithms and also evaluated metrics such as training score and testing score.
- So after hyperparameter tuning, we got to know that the best predictive model is SuperVectorMachine with training score = 0.98 and testing score = 0.95. So we decided to choose SuperVectorMachine as our predictive model.

THANK

YOU