

CONTENT

Chapter no	Chapter	Page No.
1	Chapter 1: Introduction	1
2	Chapter 2: Data Collection	2
3	Chapter 3: Data Feature Engineering	3-7
4	Chapter 4: Regression and ML model	8-14
5	Chapter 5: Predictor Web App	15-18
6	Chapter 6: Survey and Analysis	19-23
7	Chapter 7: Tableau: Ideal Product Sorting	24-26
8	Chapter 8: Tableau: Comparison Analysis	27-32
9	Chapter 9: Conclusion	33
	Reference	33

CHAPTER 1

INTRODUCTION

Today more people live digitally, it stands to reason that, smartphones became more important in day-to-day life. This made the growth in smartphone ownership tremendously over the past decade, men and women tend to own smartphones in equal numbers. The actual use of the phone was calling and texting, after the smartphone enters, it could replace larger devices to some extent. Aside from that, people can also use smartphones to pay bills, check bank account balances, send money to friends and family, shop online at their favorite stores, live streaming, content making, and gaming. The typical person checks their phone every five minutes. In this project, going to find out the best flagship-grade smartphones or smartphones that have features close to ideal features from 2020 to January 2022.

‘Ideal Product Detection’, ideal product or ideal flagship smartphone indicates, a smartphone that has the best features according to the customer or the features that a consumer wants to be in a flagship-grade smartphone. In this project, this flagship smartphone is divided into two types, Gaming Flagship Smartphone and Non-gaming Flagship Smartphone. To find the best products of these types, first of all, had to collect data regarding the smartphones that came with the flagship chipset between 2020 and 2022. Then created a regression model for machine learning, an app for product prediction, did a survey to find out the features that should be in a flagship-grade smartphone according to customers, have done the prediction of the ideal product for each company, sorted the devices from the collected data in accordance with the predicted devices, and finally done a comparison analysis of sorted products with predicted devices. All the above tasks are performed using Python, Tableau, and Excel.

CHAPTER 2

DATA COLLECTION

Data collection is the preliminary part of this project. Data for this project is collected from authentic websites like 91mobiles, GadgetNDTV, GSMArena, these websites have the complete details regarding all types of mobiles and other gadgets. The processor used in a smartphone was the first criteria for collecting data. Companies like Qualcomm and MediaTek have a major contribution in android smartphones in terms of flagship-grade chipsets, therefore, smartphones Qualcomm Snapdragon 800 series (above 855), MediaTek Dimensity series (above Dimensity1000) are considered as Flagship smartphones. All the Apple iPhone models since 2020 were also considered in the list because Apple iPhones are always flagship devices. In this data smartphones since 2020 were only considered, it is because 5G was the next criteria. The reason why 5G was one of the criteria is, nowadays people buy their smartphones to use them for a long time, so considering the longevity of smartphones up to 4 years and the Indian launch of 5G in 2022, 5G devices are the best choice for future use.

After considering the criteria, the collected data has a bunch of smartphones of both Gaming and Non-gaming types. Within the Non-Gaming type, there were different categories such as Budget Flagships, Balanced Flagships, Flagships, Overall Flagships, Mini Flagships, Fold Flagships, and Ultra Flagships.

These are the 21 fields that the collected data was having:

Release Year, Phone category, Secondary Display(inch), Display Features, Refresh Rate, Touch Sample Rate (Hz), Processor, Code/CPU & max clock/GPU/Size/Modem, Storage RAM & ROM, Number Cameras, Rear Camera, Front Camera-Sensor (MP), Camera-By, Battery(mAh), Wired charger(w), Wireless Charging(w), Back Body/Chassis/Depth, Weight, Security Sensor, Price – out of which 13 are object type, 7 are integer type, 1 float type

Apple, Samsung, OnePlus, Xiaomi, Poco, Realme, Vivo, Asus, IQOO, Lenovo, Motorola, and Oppo are the brands present in the data.

CHAPTER 3

DATA FEATURE ENGINEERING

The feature engineering of the collected data (Flagship 2020-2022) has been done by using python, in Google Colab. In this section, I have to find and remove the null values if any, (there were no null values in the data). In the data, there are some columns with more than one specific value, each value has an independent correlation with the price of the product so, I had to split each and every field in a column if that contains more than one independent value. After that by analyzing the datatype of the data, I had to convert some object datatypes to integer or float datatypes, why because after splitting the numerical values are remained as an object type. For example, in the raw data, the camera column has a value like '48+16+8', this is an object type, after splitting this into three different numeric values they remained as an object type so, had to convert the object type into the integer type. In this part splitting functions played a major role, also to understand the data, I had performed some visualizations on data.

After the feature engineering part, I have downloaded the newly cleaned data for further purposes. 'Pythonsplit Flagship 2020-22' is the name of the newly cleaned downloaded data.

PYTHON SECTION FOR DATA FEATURE ENGINEERING:

- By importing the Pandas, created the DataFrame and thereby performed the feature engineering on that DataFrame also using Numpy.

```
import pandas as pd
import numpy as np

[ ] from google.colab import files
    files.upload()
```

Choose Files No file chosen Upload widget is only available when
Saving FLAGSHIP 20-22 Data.xlsx to FLAGSHIP 20-22 Data.xlsx
{'FLAGSHIP 20-22 Data.xlsx': b'PK\x03\x04\x14\x00\x06\x00\x08\x00'}

- After importing files, able to upload data from the system.

```
[ ] #fd=flagshipdata
    fd=pd.read_excel("FLAGSHIP 20-22 Data.xlsx")
    fd
```

- The below shows the data-types of each column present in the Flagship data(fd).

```
[ ] fd.dtypes
```

```
Release Year          int64
Phone                 object
Category              object
Secondary Display(inch) float64
Display Features      object
Refresh Rate          object
Touch Sample Rate(Hz) int64
Processor             object
Code,CPU & max clock,GPU,Size,Modem object
Storage RAM & ROM      object
Number Cameras        int64
Rear Camera           object
Front Camera Sensor(MP) object
Camera By             object
Battery (mAh)         int64
Wired Charger(w)      int64
Wireless Charging(w)  int64
Back Body, Chasis & Depth object
Weight               object
Security sensor       object
Price                int64
dtype: object
```

- fd.isnull().sum() is used to find out whether there is any null value or not. By using this code, it is clear this data had no null values.

```
[ ] fd.isnull().sum()
```

```
Release Year          0
Phone                0
Category             0
Secondary Display(inch) 0
Display Features      0
Refresh Rate          0
Touch Sample Rate(Hz) 0
Processor            0
Code,CPU & max clock,GPU,Size,Modem 0
Storage RAM & ROM      0
Number Cameras        0
Rear Camera           0
Front Camera Sensor(MP) 0
Camera By            0
Battery (mAh)         0
Wired Charger(w)      0
Wireless Charging(w)  0
Back Body, Chasis & Depth 0
Weight               0
Security sensor       0
Price                0
dtype: int64
```

- First of all, in the feature engineering part I had replaced certain terms from some of the columns like 'Rear Camera', 'Storage Ram & Rom', 'Weight', to make it meaningful in terms of datatype.

```
[ ] # Removed the text 'MP' to make it easy while splitting and convert to int type
fd['Rear Camera']=fd['Rear Camera'].str.replace("MP","")
fd['Rear Camera']
```

```
0      108 + 13 + 5
1       64 + 13 + 5
2     108 + 13 + 2 + 2
3     50 + 48 + 32 + 8
4      48 + 13 + 13
...
64      12 + 12
65     12 + 12 + 8
66     50 + 50 + 2
67     50 + 8 + 2
68     50 + 16 + 2
Name: Rear Camera, Length: 69, dtype: object
```

As the sample shown above, 'MP' from 'Rear Camera', 'GB' from 'Storage RAM & ROM', 'Kg' from 'Weight' are removed.

- After replacing the extra terms from 3 columns, I had performed splitting on each of the columns that contain more than one independent values.

For example, the column named Display Features had 5 different values in 1 as shown on the right.

**Display
Features**

6.67
FullHD+
IPS-LCD
2.5D-
Curved
1080x2400

```
[ ] # splitting of display features will get us core feature that influence in pricing
dis=fd['Display Features'].str.split(" ",n=20,expand=True)
dis
```

	0	1	2	3	4	5
0	6.67	FullHD+	IPS-LCD	2.5D-Curved	1080x2400	None
1	6.67	FullHD+	IPS-LCD	2.5D-Curved	1080x2400	None
2	6.67	FullHD+	Super-AMOLED	3D-Curved	1080x2340	None
3	6.56	QHD+	Super-AMOLED	3D-Curved	1080x2376	None
4	6.56	FullHD+	AMOLED	3D-Curved	1080x2376	None
...
64	6.1	FullHD+	XDR-OLED	Flat	1170x2532	None
65	6.4	FullHD+	Dynamic-AMOLED-2X	2.5D-Curved	1080x2400	None
66	6.67	QHD+	E5-AMOLED	2.5D-Curved	1440x3216	None
67	6.62	FullHD+	E4-AMOLED	2.5D-Curved	1440x2400	None
68	6.56	FullHD+	Fluid-AMOLED	2.5D-Curved	1080x2400	None

By using splitting function as shown above, got separated the 5 different values into each column, then took each needed values and added to each new columns in the fd-data.

- Before adding the fields into a new column, we have to verify the separated values are did split correctly.

To verify the split fields, took the unique values for each split column as shown below,

```
[ ] print(dis[3].unique(),dis[4].unique(),dis[5].unique())

['2.5D-Curved' '3D-Curved' 'Plus' 'Flat'] ['1080x2400' '1080x2340' '1080x2376' '1768x2208' '1080x2636' '1440x3220'
'1440x3200' 'Flat' '1440x3168' '1284x2778' '1170x2532' '1180x2400'
'2208x1768' '1080x2640' '1440x3216' '1080x2460' '1080x2448' '1440x2400'] [None '1080x2400']
```

By taking the unique value for each column, it is able to found the errors or mismatch splitting happened in the part that is, the 3rd, 4th and the 5th positioned columns had some alternate values from the consecutive columns. It is because the values in the 2nd column had extra space in between values also the splitting is made based on space that is why it happened. This was simple to make it correct, by replacing the value by the wrong values from the next consecutive column as shown below.

```
dis[3]=dis[3].str.replace("Plus", "Flat")
dis[4]=dis[4].str.replace("Flat", "1080x2400")
dis.drop(columns=5,inplace=True)
```

- After correcting the fields again verified and then added each necessary fields into the data as new columns

```
[ ] print(dis[3].unique(),dis[4].unique())

['2.5D-Curved' '3D-Curved' 'Flat'] ['1080x2400' '1080x2340' '1080x2376' '1768x2208' '1080x2636' '1440x3220'
'1440x3200' '1440x3168' '1284x2778' '1170x2532' '1180x2400' '2208x1768'
'1080x2640' '1440x3216' '1080x2460' '1080x2448' '1440x2400']
```

```
[ ] fd["Screen Size(inch)"]=dis[0]
fd["Screen Resolution"]=dis[4]
fd["Display Resolution Name"]=dis[1]
fd["Display Type"]=dis[2]
fd["Display Dimension"]=dis[3]
fd.head()
```

Similarly, as like 'Display Features' the splitting has used for columns such as 'Phone', 'Refresh Rate', 'Processor', 'Code, CPU & max clock, GPU, Size, Modem', 'Storage RAM & ROM', 'Rear Camera', 'Camera By', 'Back Body, Chasis & Depth',

Brand, Model, Screen Size(inch), Screen Resolution, Display Resolution Name, Display Type, Display Dimension, Refresh Rate (Hz), Variable Refresh Rate, Processor Brand, Chipset, Primary Clock Speed(GHz), Chip Architecture Size(nm), RAM, Internal Storage, Rear Primary Camera,

Rear Secondary Camera, Rear Tertiary Camera, Rear Quaternary Camera, Camera Sensor By, Camera Tuning By, Back Body, Chasis, Thickness, PPI: These are the new columns added to the fd-data after performing the split function.

- PPI is a different column made based on the resolution and screen size of a device, PPI has a definite correlation with the price of a device.

This is how the Pixels Per Inch (PPI) is calculated

```
[ ] fd['PPI'] = (((ppi[0]**2) + (ppi[1]**2))**0.5/fd['Screen Size(inch)']).astype('float')
fd.head(2)
```

- After adding the necessary fields as new columns, the unwanted columns were dropped from the data as the example shows below.

```
[ ] fd.drop(columns=["Phone","Display Features"],inplace=True)
fd.head()
```

- After feature engineering, the data is then downloaded and saved for the Modelling part.

```
[ ] fd.to_csv('Pythonsplit FLAGSHIP 20-22 Data.csv',index=False)
files.download('Pythonsplit FLAGSHIP 20-22 Data.csv')
```


CHAPTER 4

REGRESSION AND ML MODEL

After data feature engineering, performed the Regression and Machine Learning using the 'Pythonsplit Flagship 2020-22 Data' data. First of all, plotted the correlation of each field with respect to the price of each product. From the correlation, I have chosen a highly positively correlated and highly negatively correlated numerical field-they are independently correlated to the price-for the regression model. Other than numerical values, there are string values in which some of the fields are correlated to the price. To find out the relation, I have plotted a barplot of each and every column with the price, that has string values as well as has the chance of high correlation with price. Matplotlib and seaborn are the libraries used for plotting.

Then, removed the less correlated columns from the data and done label encoding for the selected columns that have string values. Using the data that has encoded values, made the regression model and then did split it into train and test models for the machine learning purpose. From the sklearn library imported train_test_split, onehotencoder, columntransformer, pipeline, r2_score, mean_absolute_error, Linear regression, randomforest, votingregressor, and such. Out of which, voting-regressor had more accuracy so, for the further development voting-regressor model is considered. After creating the model, using the pickle library the model is exported and saved in local file storage.

For Regression and Machine Learning, 21 columns are sorted from the cleaned data. Out of those 21 columns Category, Security sensor, Brand, Display Type, Variable Refresh Rate, Processor Brand, Chipset, Camera Sensor By, Camera Tuning By, Back Body, Chasis are the object type fields, and Wireless Charging(w), Chip Architecture Size(nm), RAM, Internal Storage, Rear Secondary Camera, Rear Tertiary Camera are the integer type fields and ultimately Weight(gm), Screen Size(inch), Thickness, PPI are the float type fields.

PYTHON SECTION FOR REGRESSION AND ML MODEL:

Using Pandas, Numpy and files from google.colab uploaded the data and created the data frame that needed for ML Model. Here, the data frame is named as vd (Visualization Data) because, before performing the regression on the data, data need to be reduced by finding the highly correlated independent fields with respect to the dependent field, there are object, int and float type data are present, so had to find the variation of object types data, for that visualization is the best way pull the relationship between two variables.

- From the data, Price is the target field so, run a code to get the correlation of all the numeric fields with respect to the price

```
[ ] vd.corr()['Price INR']
```

```
Release Year          -0.187014
Secondary Display(inch)  0.512664
Touch Sample Rate(Hz)  -0.306170
Number Cameras        -0.250739
Front Camera Sensor(MP) -0.229517
Battery (mAh)         -0.334664
Wired Charger(w)      -0.552173
Wireless Charging(w)   0.412507
Weight(gm)            0.348731
Price INR             1.000000
Screen Size(inch)     0.208907
Refresh Rate(Hz)      -0.191111
Primary Clock Speed(GHz) 0.155217
Chip Architecture Size(nm) -0.300308
RAM                   0.391935
Internal Storage       0.228298
Rear Primary Camera   -0.476563
Rear Secondary Camera  0.207427
Rear Tertiary Camera   0.282909
Rear Quaternary Camera 0.045029
Thickness             -0.413856
PPI                   0.340955
Name: Price INR, dtype: float64
```

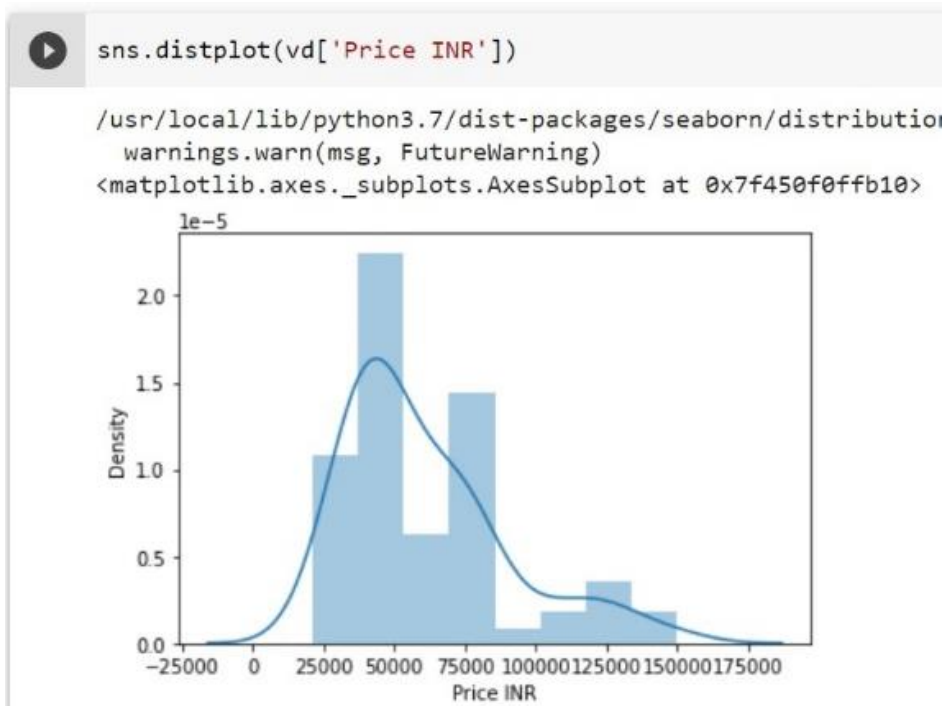
Even there are some highly negative and highly positive correlations, all cannot be considered as a proper correlation, some of them are wrong correlations

By analyzing the data, PPI, Thickness, Rear Secondary Camera, Rear Tertiary Camera, Ram, Internal Storage, Primary Clock Speed (GHz), Chip Architecture Size, Battery, Weight and Wireless Charging(w) have shown the correct correlation with Price.

- For the visualization part of data in this section, Matplotlib and Seaborn libraries are Used.

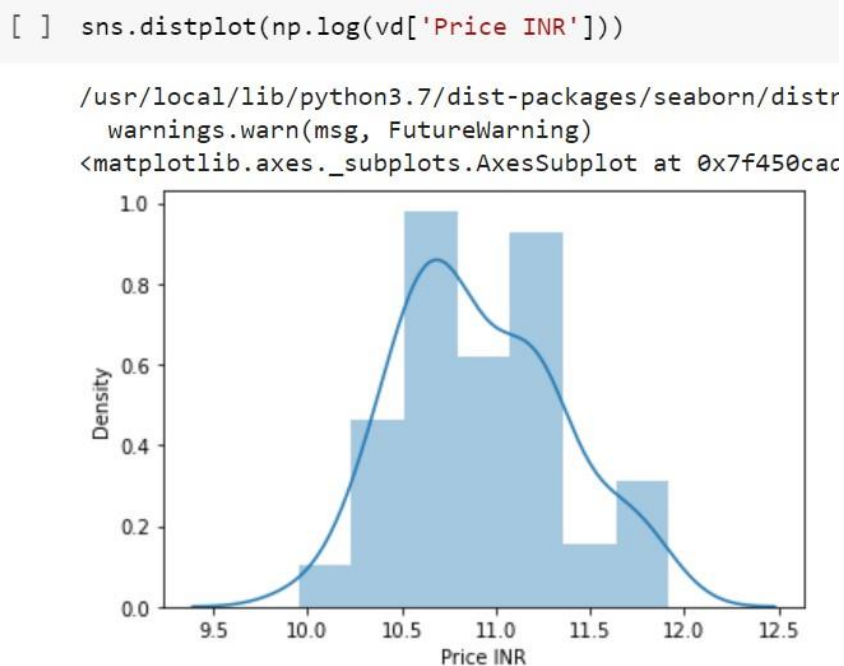
```
[ ] import matplotlib.pyplot as plt
    import seaborn as sns
```

- Here the Price distribution is plotted using seaborn (sns). From the visualization it is evident that, price of most numbers of flagship devices, within the data, is sits between Rupees 25000 and 85000.

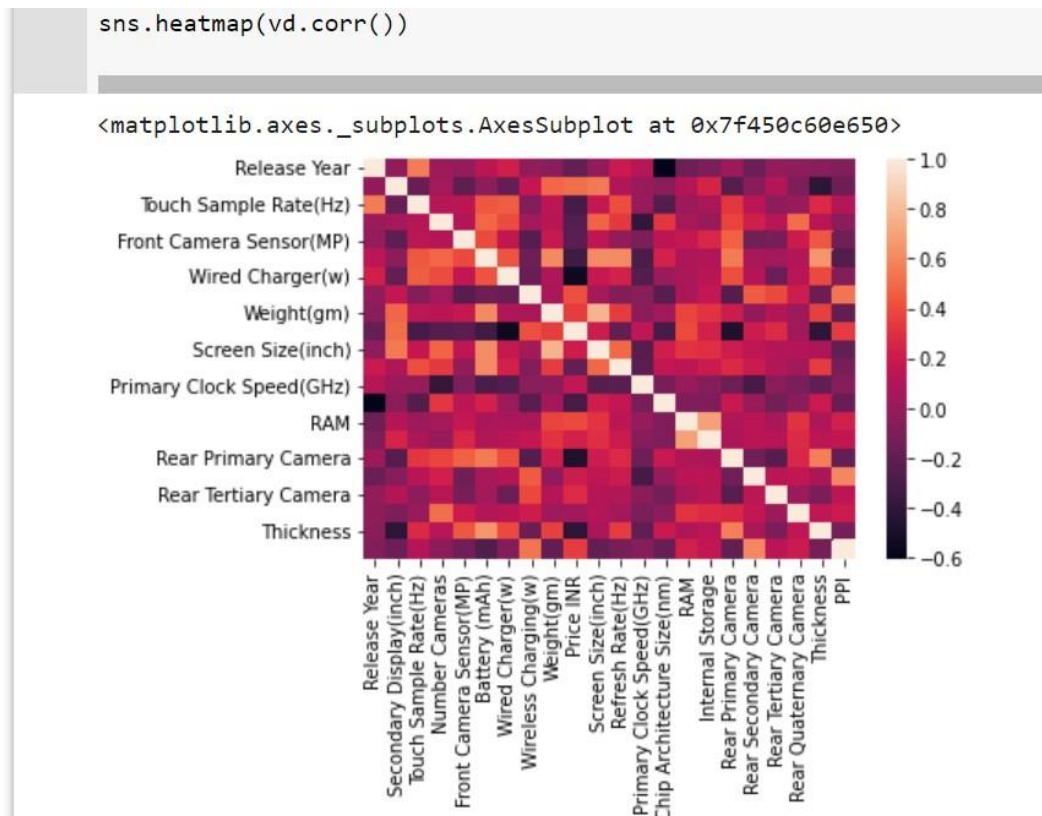


- Price in log format is more meaningful, better in terms of statistics and better for performing regression model,

Therefore, the price distribution in log format is also plotted.



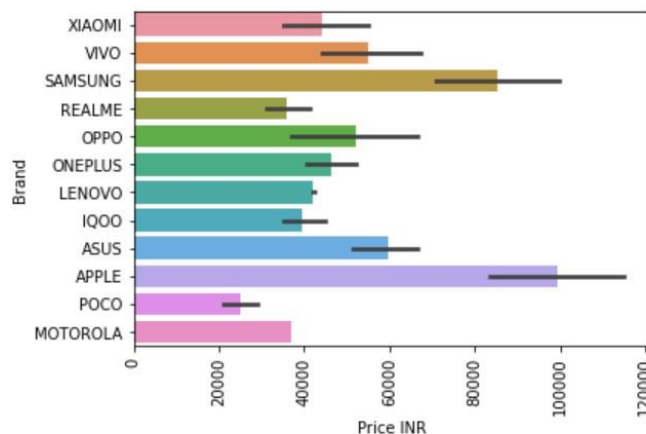
- The visual representation of the correlation for numerical fields with respect to each other is visualized using heatmap in seaborn



- For most cases Seaborn is used to find out how strong is the relation between object type variables and Price.

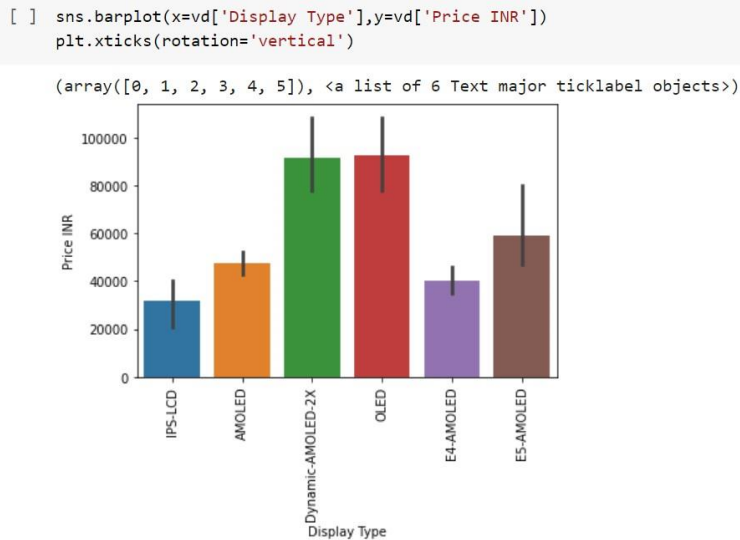
For example, as shown below barplot in seaborn is used to find which company has flagship smartphone at highest price.

```
[ ] sns.barplot(y=vd['Brand'],x=vd['Price INR'])
plt.xticks(rotation='vertical')
plt.show()
```



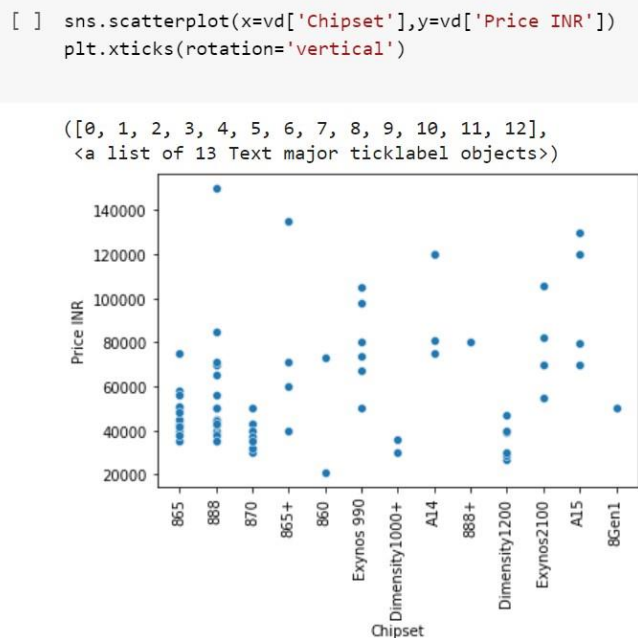
So, from this plot it is clear Apple has the most expensive device which is commonly ranging from above 80000 to near 120000 Rupees. Then followed by Samsung.

Similar barplot is used for rest columns with object data type to find out the best variable fit for Machine Learning.



Here the Display Type has a strong relationship with Price. So, definitely, it can be used for Machine Learning. Similarly found the object type fields that have strong relationships with the price.

- Scatterplot is Performed on Chipset with respect to price. At a glance, some of the points are like outliers but actually, the differences are real, which are based on the brand value that is, if there are same specs under different brands the price definitely varies in accordance with the brand. Therefore, the variations below shown are highly based on brand value and cannot be removed.



- Later finding the fields that are best for machine learning, divided the data into independent and dependent as represented below.

```
[ ] x = vd.drop(columns=['Price INR','Model','Secondary Display(inch)','Display Dimension','Primary Clock Speed(GHz)',
                        'Front Camera Sensor(MP)','Number Cameras','Screen Resolution','Refresh Rate(Hz)',
                        'Touch Sample Rate(Hz)','Wired Charger(w)',
                        'Rear Primary Camera','Rear Quaternary Camera','Battery (mAh)','Release Year'])
y = np.log(vd['Price INR'])
```

Price is the dependent variable so it is taken into 'y' and the independent fields with strong correlations are added to 'x'. Using this x and y, created the train and test data.

- Here using sci-kit learn to import train_test_split to make the x_train, x_test and y_train, y_test data and also here 20 percentage of the data is considered for test data.

After creating the train and test data, imported certain libraries such as LinearRegression, RandomForestRegressor, GradientBoostingRegressor, ExtraTreesRegressor and VotingRegressor to perform Regression. ColumnTransformer and OneHotEncoder is imported for converting object type data into numerical data.

```
[ ] from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=2)
```

```
[ ] from sklearn.linear_model import LinearRegression
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import OneHotEncoder
from sklearn.metrics import r2_score,mean_absolute_error
```

```
[ ] from sklearn.ensemble import RandomForestRegressor
```

- After performed regression using all the imported regressor, the one model with best accuracy is considered to export for making prediction web page.

Out of all those regressors, VotingRegressor has shown more accuracy score than rest so, that ML model using VotingRegressor is considered as the best model.


```

from sklearn.ensemble import GradientBoostingRegressor, ExtraTreesRegressor
from sklearn.ensemble import VotingRegressor, StackingRegressor
#from xgboost import XGBRegressor

step1 = ColumnTransformer(transformers=[
    ('col_tnf', OneHotEncoder(sparse=False, drop='first'), [0,3,4,6,7,8,9,15,16,17,18])
], remainder='passthrough')

rf = RandomForestRegressor(n_estimators=350, random_state=3, max_samples=None, max_features=0.75, max_depth=15)
gbdt = GradientBoostingRegressor(n_estimators=100, max_features=0.5)
#xgb = XGBRegressor(n_estimators=25, learning_rate=0.3, max_depth=5)
et = ExtraTreesRegressor(n_estimators=100, random_state=3, max_samples=None, max_features=0.75, max_depth=10)

#step2 = VotingRegressor([('rf', rf), ('gbdt', gbdt), ('xgb', xgb), ('et', et)], weights=[5,1,1,1])
step2 = VotingRegressor([('rf', rf), ('gbdt', gbdt), ('et', et)], weights=[5,1,1])

pipe = Pipeline([
    ('step1', step1),
    ('step2', step2)
])

pipe = Pipeline([
    ('step1', step1),
    ('step2', step2)
])

pipe.fit(x_train, y_train)

y_pred = pipe.predict(x_test)

print('R2 score', r2_score(y_test, y_pred))
print('MAE', mean_absolute_error(y_test, y_pred))

R2 score 0.7747625411747743
MAE 0.16280962658956025

```

- After creating the best ML Model, the model is then exported using Pickle library as shown below. By exporting the model using pickle, we could add the ML model independently to any python tool to perform further coding. Because, here the model is going to added into pycharm tool for performing stramlit actions.

```

[ ] import pickle

pickle.dump(vd, open('vd.pkl', 'wb'))
pickle.dump(pipe, open('pipe.pkl', 'wb'))

```

CHAPTER 5

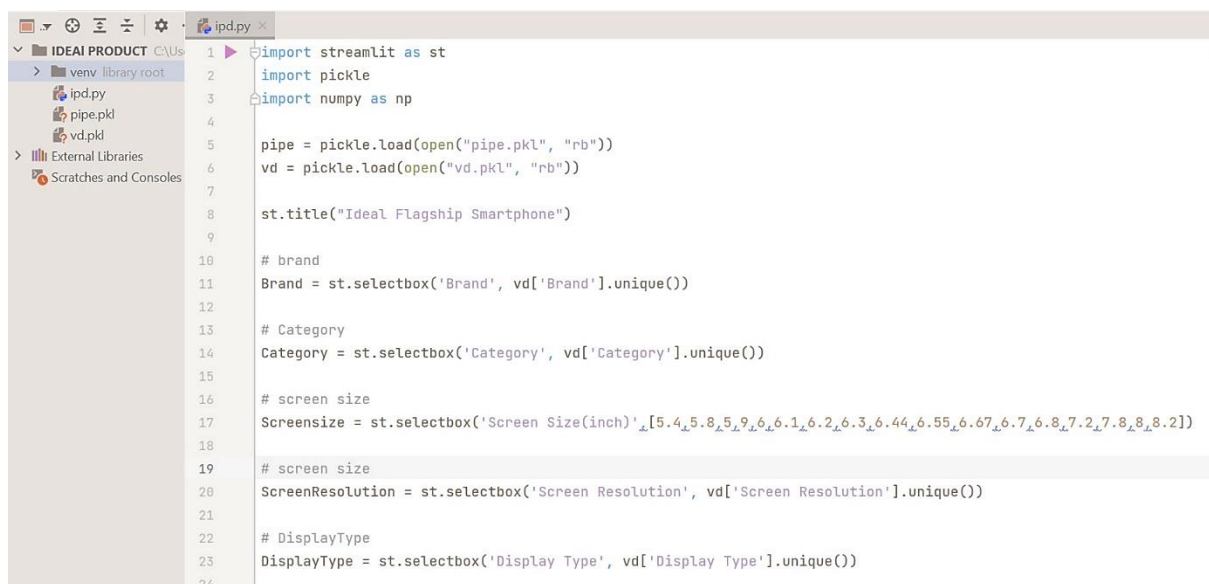
PREDICTOR WEB APP

The Predictor web page is created using streamlit library for which pycharm is used for coding. After exporting the machine learning models from colab, import those files into pycharm software. Within pycharm, had to install libraries like streamlit, pickle, NumPy, sklearn. Using streamlit, applied selector box, typing box, and such for the web page. After creating the entire web page elements, run streamlit using the name of python file, ipd.py, and will direct towards the web page. From the web page, we had to give the features to predict the price, the predicted price is definitely based on the specs because the price is the dependent variable to all the fields in the data.

The objective of the web page is to find out the price of the flagship phones of each brand with ideal specs which are collected through a survey.

PYTHON SECTION FOR PREDICTOR WEB APP:

- In pycharm, the added files named 'pipe.pkl' and 'vd.pkl' were the exported model and data from the ML learning section. Here the python file is named as 'ipd.py'. which represents 'ideal product detection' Using Streamlit, here created box layouts of the predictor web page of each spec field that is necessary for predicting the price of a flagship smartphone. Below shown is the method used to create the web page



```
1 import streamlit as st
2 import pickle
3 import numpy as np
4
5 pipe = pickle.load(open("pipe.pkl", "rb"))
6 vd = pickle.load(open("vd.pkl", "rb"))
7
8 st.title("Ideal Flagship Smartphone")
9
10 # brand
11 Brand = st.selectbox('Brand', vd['Brand'].unique())
12
13 # Category
14 Category = st.selectbox('Category', vd['Category'].unique())
15
16 # screen size
17 Screensize = st.selectbox('Screen Size(inch)', [5.4, 5.8, 5.9, 6.1, 6.2, 6.3, 6.4, 6.5, 6.6, 6.7, 6.8, 7.2, 7.8, 8.2])
18
19 # screen size
20 ScreenResolution = st.selectbox('Screen Resolution', vd['Screen Resolution'].unique())
21
22 # DisplayType
23 DisplayType = st.selectbox('Display Type', vd['Display Type'].unique())
24
```



```

ipd.py ×
C:\Users\SreeHari\PycharmProjects\IDEAL PRODUCT\ipd.py
26 DisplayDimension = st.selectbox('Display Dimension', vd['Display Dimension'].unique())
27
28 # RefreshRate
29 RefreshRate = st.selectbox('Refresh Rate(Hz)', vd['Refresh Rate(Hz)'].unique())
30
31 # TouchSampleRate
32 TouchSampleRate = st.selectbox('Touch Sample Rate(Hz)', [180,240,300,360,480,600,1000])
33
34 # VariableRefreshRate
35 VariableRefreshRate = st.selectbox('Variable Refresh Rate', vd['Variable Refresh Rate'].unique())
36
37 # ProcessorBrand
38 ProcessorBrand = st.selectbox('Processor Brand', vd['Processor Brand'].unique())
39
40 # Chipset
41 Chipset = st.selectbox('Chipset', vd['Chipset'].unique())
42
43 # ChipSize
44 ChipSize = st.selectbox('Chip Architecture Size(nm)', [7,6,5,4])
45
46 # RAM
47 RAM = st.selectbox('RAM(GB)', [4,6,8,12,16])
48
49 # InternalStorage
50 InternalStorage = st.selectbox('Internal Storage(GB)', [64,128,256,512])
51
52 # RearPrimaryCamera
53 RearPrimaryCamera = st.selectbox('Rear Primary Camera(MP)', [12,48,50,64,108])
54
55 # RearSecondaryCamera
56 RearSecondaryCamera = st.selectbox('Rear Secondary Camera(MP)', [0,5,8,10,12,13,16,20,24,32,48,50,64])
57
58 # RearTertiaryCamera
59 RearTertiaryCamera = st.selectbox('Rear Tertiary Camera(MP)', [0,5,8,10,12,16,20,24,32,48,50,64])
60
61 # RearQuaternaryCamera
62 RearQuaternaryCamera = st.selectbox('Rear Quaternary Camera(MP)', [0,5,8,10,12,16,20,24,32,48,50,64])
63
64 # CameraSensorBy
65 CameraSensorBy = st.selectbox('Camera Sensor By', vd['Camera Sensor By'].unique())
66
67 # CameraTuningBy
68 CameraTuningBy = st.selectbox('Camera Tuning By', vd['Camera Tuning By'].unique())
69
70 # Battery
71 Battery = st.number_input('Battery (mAh)')

```

```

73 # WiredCharger
74 WiredCharger = st.selectbox('Wired Charger(w)', [5,10,15,20,25,30,33,40,45,50,55,65,66,
75          67,80,90,100,120,125])
76
77 # WirelessCharging
78 WirelessCharging = st.selectbox('Wireless Charging(w)', [0,12,15,20,25,30,40,45,50,55,65,66,80,90])
79
80 # SecuritySensor
81 SecuritySensor = st.selectbox('Security sensor', vd['Security sensor'].unique())
82
83 # BackBody
84 BackBody = st.selectbox('Back Body', vd['Back Body'].unique())
85
86 # Chassis
87 Chassis = st.selectbox('Chassis', vd['Chasis'].unique())
88
89 # Thickness
90 Thickness = st.number_input('Thickness(mm)')
91
92
93 # Weight
94 Weight = st.number_input('Weight(gm)')
95
96 if st.button('Predict Product Price'):
97     X_res = int(ScreenResolution.split('x')[0])
98     Y_res = int(ScreenResolution.split('x')[1])
99     PPI = ((X_res**2) + (Y_res**2))**0.5/Screensize
100     query = np.array([Category,WirelessCharging,Weight,SecuritySensor,Brand,Screensize,
101         DisplayType,VariableRefreshRate,ProcessorBrand,Chipset,ChipSize,
102         RAM,InternalStorage,RearSecondaryCamera,RearTertiaryCamera,
103         CameraSensorBy,CameraTuningBy,BackBody,Chassis,Thickness,PPI])
104     query = query.reshape(1,21)
105     st.title(int(np.exp(pipe.predict(query))))
106

```

After creating web page elements, the web page is accessed by running the streamlit as shown below,

```

Terminal: Local (2) × Local × + v
Windows PowerShell

Try the new cross-platform PowerShell https://aka.ms/pscore6

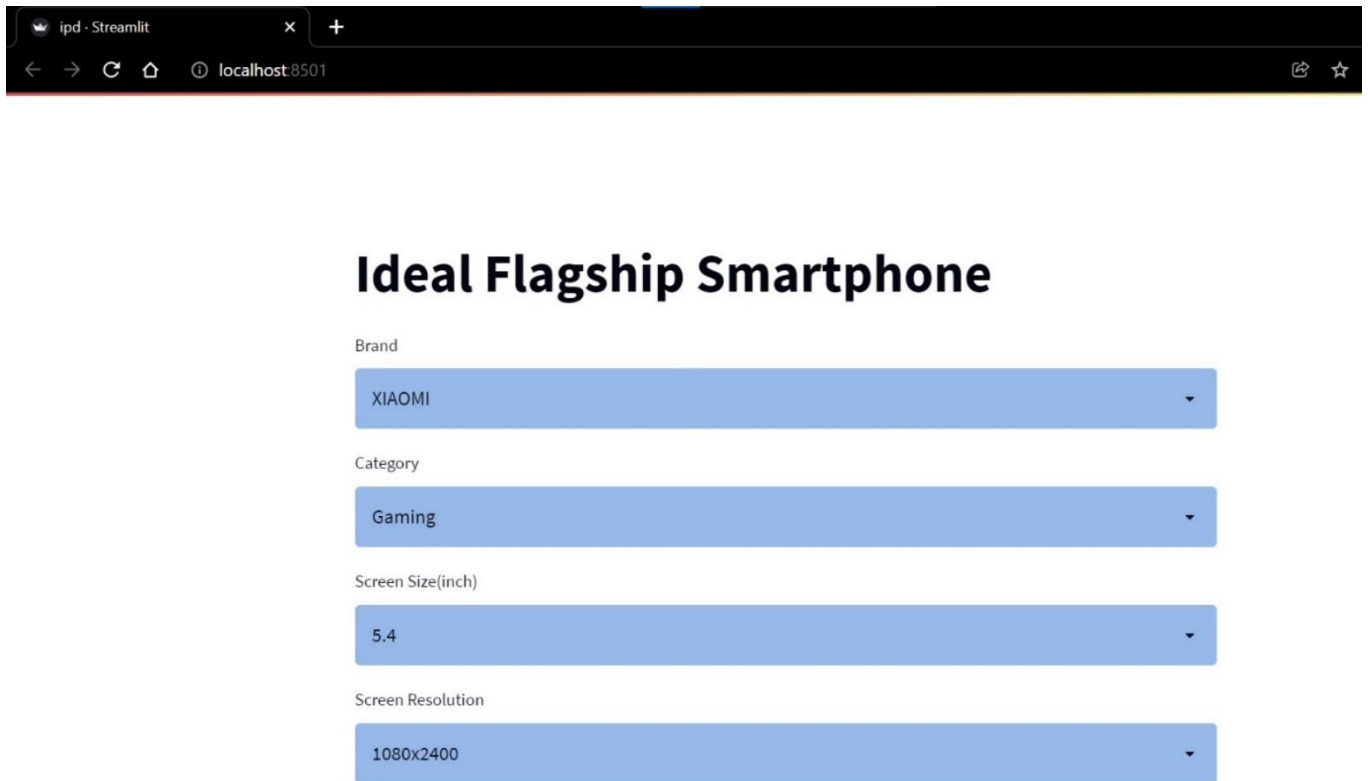
PS C:\Users\SreeHari\PycharmProjects\IDEAL PRODUCT> streamlit run ipd.py

You can now view your Streamlit app in your browser.

Local URL: http://localhost:8501
Network URL: http://192.168.1.35:8501

```

- After running the code it will automatically direct towards the web page using the default browser of the system and which is look like as shown below,



The screenshot shows a web browser window with a single tab titled 'ipd · Streamlit'. The address bar displays 'localhost:8501'. The main content area features a title 'Ideal Flagship Smartphone' in a large, bold, black font. Below the title, there are four dropdown menus, each with a light blue background and a dark blue border. The first dropdown is labeled 'Brand' and shows 'XIAOMI'. The second is labeled 'Category' and shows 'Gaming'. The third is labeled 'Screen Size(inch)' and shows '5.4'. The fourth is labeled 'Screen Resolution' and shows '1080X2400'. Each dropdown menu has a small downward-pointing arrow on its right side.

Ideal Flagship Smartphone

Brand

XIAOMI

Category

Gaming

Screen Size(inch)

5.4

Screen Resolution

1080X2400

▼ Second Analysis is based on Normal Flagship Devices

```
[ ] files.upload()
```

Choose Files No file chosen

Upload widget is only available when the cell has been

Saving Normal Flagship Survey Data.xlsx to Normal Flagship Survey Data.xlsx

```
{'Normal Flagship Survey Data.xlsx': b'PK\x03\x04\x14\x00\x06\x00\x08\x00\x00'}
```

```
[ ] nd=pd.read_excel('Normal Flagship Survey Data.xlsx')
nd.head()
```

Both the gd and nd data has same type of fields.

```
[ ] gd.dtypes
```

```
Brand You Choose          object
Type of smartphone you need  object
Screen Size you prefer    float64
Screen Resolution         object
Type                      object
Type of display you prefer  object
Refresh rate to choose (Hz) int64
Touch sample/ response rate (Hz) int64
Primary Camera (MP) you want int64
Ultra-wide camera (MP)     object
Tertiary Camera_telephoto or macro_camera (MP) object
sensor selected for 1st image object
sensor selected for 2nd image object
sensor selected for 3rd image object
sensor selected for 4th image object
Side frame                object
Back Panel                object
Battery You Prefer (mAh)   int64
Weight of smartphone you prefer (g) int64
Wired charger capacity (w) int64
Wireless Charger (w)      object
Thickness you prefer (mm)  float64
Processor you prefer       object
dtype: object
```

```
[ ] nd.dtypes
```

```
Brand You Choose          object
Type of smartphone you need  object
Screen Size you prefer    float64
Screen Resolution         object
Type                      object
Type of display you prefer  object
Refresh rate to choose (Hz) int64
Touch sample/ response rate (Hz) object
Primary Camera (MP)       int64
Ultra-wide camera (MP)     object
Tertiary Camera_telephoto or macro_camera (MP) object
Sensor selected from the 1st image object
Sensor selected from the 2nd image object
Sensor selected from the 3rd image object
Sensor selected from the 4th image object
Side frame                object
Back Panel                object
Battery You Prefer (mAh)   int64
Weight of smartphone you prefer (g) int64
Wired charger capacity (w) int64
Wireless Charger (w)      object
Thickness you prefer (mm)  float64
Processor you prefer       object
dtype: object
```

- Matplotlib is used for all the visualizations performed in this section to find out the most selected feature from the data.

```
[ ] import matplotlib.pyplot as plt
import seaborn as sns
```

Throughout the visualization process, countplot is used for finding the most selected feature. For that countplot, I defined a function cp(), cp stands for countplot, to plot countplot seamlessly. The defined function is shown below,

```
[ ] #cp stands for countplot
def cp(df):
    colors=[]
    column_name=input("column Name ")
    length=len(df[column_name].unique())
    print('length :',length )
    figx=float(input('x_figsize : '))
    figy=float(input('y_figsize : '))
    clr_list=['b','y','m','g','r','c','dimgray','darkorange','limegreen','lightsteelblue','rosybrown','darkgoldenrod','lightseagreen',
             'mediumslateblue','indianred','tan','olivedrab','teal','mediumseagreen','crimson','orchid','midnightblue','maroon','palegreen',
             'peru','orange','darkseagreen','slategray','dodgerblue','mediumvioletred','mediumslateblue','mediumturquoise','salmon','cornflowerblue',
             'thistle','steelblue','yellowgreen','khaki','orangered','greenyellow','chocolate','burlywood','powderblue','lavender','indigo',
             'firebrick','oldlace','coral','deeppink','rebeccapurple','navy','deepskyblue','forestgreen','mediumaquamarine','papayawhip','tomato',
             'silver']
    for i in range(1,length+1):
        print("color",i)
        colors.append(clr_list[int(input())])
    plt.subplots(figsize=(figx,figy))
    print(df[column_name].value_counts(),df[column_name].value_counts().plot.bar(color=colors))
    plt.savefig(input('enter image name :'))
```

- Working of cp() function is as shown below,

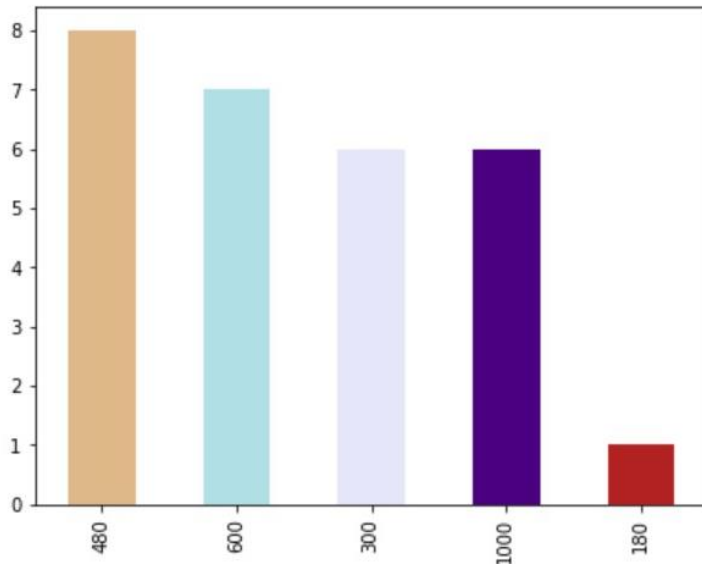
```
[ ] #Touch sample/ response rate (Hz)
cp(gd)
```

```
column Name Touch sample/ response rate (Hz)
length : 5
x_figsize : 6.6
y_figsize : 5
color 1
41
color 2
42
color 3
43
color 4
44
color 5
45
480      8
600      7
300      6
1000     6
180      1
Name: Touch sample/ response rate (Hz), dtype: int64 AxesSubplot(0.125,0.125;0.775x0.755)
enter image name :Touch sample rate.jpg
```

Inside the cp() , we have to mention the name of the data frame on which we need to visualize countplot. For example, here considered gaming data that is gd so, cp(gd)

After running cp(gd), the code will ask for the column name that needed to plotted then it will show the unique length of that particular column, then the code ask for figure size of both axes. Then after that code will ask for colours in numbers, there are 57 colours in the cp() function so we need to enter colours between 0 and 56 according to the length of the column.

Then, it will show a column to enter a name to save the figure which is going to plot, after entering the name, the countplot will be plotted as shown below,



This countplot is based on the 'Touch sample Rate' of device. From the plot it is clear that most numbers of users selected 480Hz as the best touch sample rate for a gaming flagship device.

A similar count plot is plotted for the rest of the fields as well as for the Normal flagship device data that is nd. Thereby it was easy to find the features that most users were selected.

- In the survey there was 4 question to choose the sensor by observing the given images so, to find out which sensor was selected most, the method shown below is used for both gd and nd data.

```
[ ] gso1=len(gd[gd['sensor selected for 1st image']=='SONY'])
    gso2=len(gd[gd['sensor selected for 2nd image']=='SONY'])
    gso3=len(gd[gd['sensor selected for 3rd image']=='SONY'])
    gso4=len(gd[gd['sensor selected for 4th image']=='SONY'])

    gsa1=len(gd[gd['sensor selected for 1st image']=='SAMSUNG'])
    gsa2=len(gd[gd['sensor selected for 2nd image']=='SAMSUNG'])
    gsa3=len(gd[gd['sensor selected for 3rd image']=='SAMSUNG'])
    gsa4=len(gd[gd['sensor selected for 4th image']=='SAMSUNG'])

    print('For Gaming Flagships','\nSony total selections =',gso1+gso2+gso3+gso4,'\nSamsung total selections = ', gsa1+gsa2+gsa3+gsa4)
    #here samsung has the overall amximum selections out of the 4 image for gaming flagships
```

```
For Gaming Flagships
Sony total selections = 49
Samsung total selections = 63
```

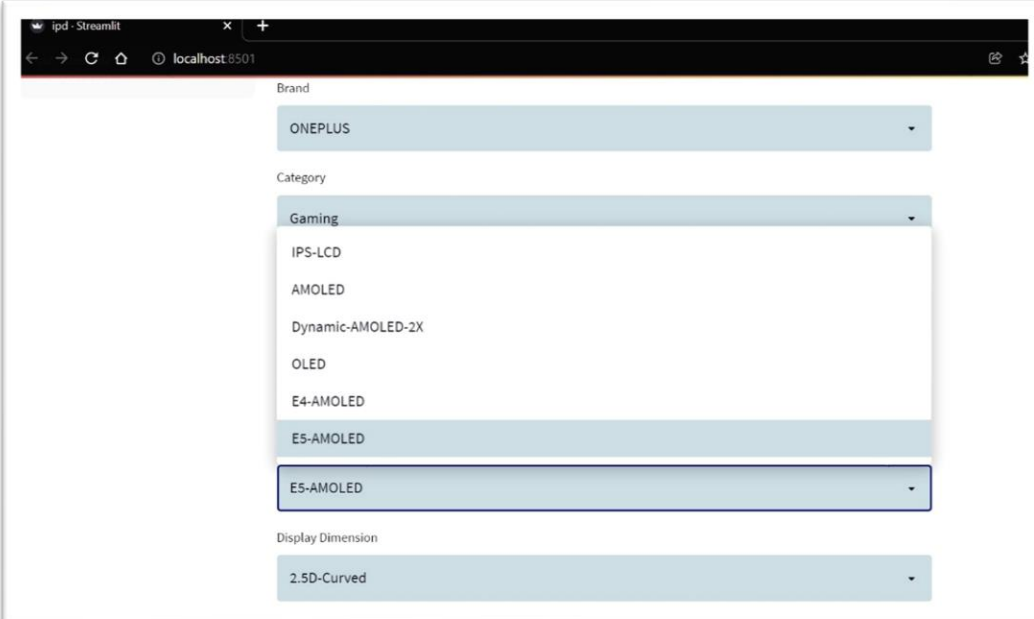
```
[ ] nso1=len(nd[nd['Sensor selected from the 1st image']=='SONY'])
    nso2=len(nd[nd['Sensor selected from the 2nd image']=='SONY'])
    nso3=len(nd[nd['Sensor selected from the 3rd image']=='SONY'])
    nso4=len(nd[nd['Sensor selected from the 4th image']=='SONY'])

    nsa1=len(nd[nd['Sensor selected from the 1st image']=='SAMSUNG'])
    nsa2=len(nd[nd['Sensor selected from the 2nd image']=='SAMSUNG'])
    nsa3=len(nd[nd['Sensor selected from the 3rd image']=='SAMSUNG'])
    nsa4=len(nd[nd['Sensor selected from the 4th image']=='SAMSUNG'])

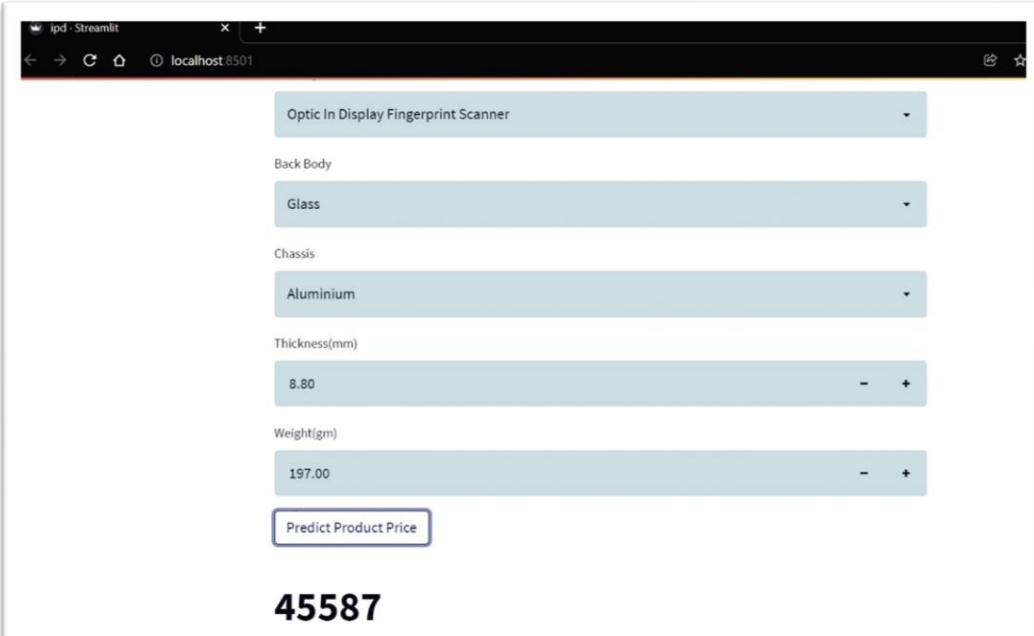
    print('For Normal Flagship','\nSony total selections =',nso1+nso2+nso3+nso4,'\nSamsung total selections = ', nsa1+nsa2+nsa3+nsa4)
    ##here samsung has the overall amximum selections out of the 4 image for normal flagships**
```

```
For Normal Flagship
Sony total selections = 36
Samsung total selections = 72
```

Then the most selected features are considered as the ideal features and those features were chosen in the select-box of the web page, that created using streamlit in python, as shown below,



The screenshot shows a web browser window with the address bar displaying 'localhost:8501'. The application interface includes three dropdown menus: 'Brand' with 'ONEPLUS' selected, 'Category' with 'E5-AMOLED' selected (the dropdown menu is open showing options like IPS-LCD, AMOLED, Dynamic-AMOLED-2X, OLED, E4-AMOLED, and E5-AMOLED), and 'Display Dimension' with '2.5D-Curved' selected.



The screenshot shows the same web application with additional features selected: 'Optic In Display Fingerprint Scanner' for the fingerprint scanner, 'Glass' for the back body, and 'Aluminium' for the chassis. The 'Thickness(mm)' is set to 8.80 and 'Weight(gm)' is set to 197.00, both using slider controls. A 'Predict Product Price' button is visible, and the predicted price '45587' is displayed in large bold text at the bottom.

After selecting the features from the boxes, the predicted price based on the selected features will be shown as above by clicking the button named 'Predict Product Price'.

This prediction is made for all the brands with same ideal features in both gaming and non-gaming categories. Finally, these ideal predicted devices were added along with very first collected data to form a new data for further analysis.

CHAPTER 7

TABLEAU: IDEAL PRODUCT SORTING

In this part, using the new data, which has the predicted devices in, performed visualizations in tableau to find out the devices that are very close, in terms of features, towards the ideal devices that I predicted. From this section, I understood no smartphones have all the ideal features and value as the user expected. Suppose the user selected 10 ideal features but there may only have a maximum of 9 out of 10 ideal features in a device that is, maybe the device has a different chipset or camera or battery when compared to ideal features after considering the ideal price. Also, there are devices that have 5 to 10 more features than the 10 ideal features but they definitely cost more than 2 times of an ideal product.

Using the tableau, after comparing and analyzing the features of the existing devices between 2020 and 2022(till Jan 2nd week) with the ideal features for both gaming and non-gaming smartphones, I detected 5 Gaming-Flagship devices from four brands and 11 Normal-Flagship devices from six different brands, these devices are nearer to the ideal features in both categories.

TABLEAU SECTION FOR IDEAL PRODUCT DETECTION:

Using interactive dashboards in Tableau, found the best gaming and non-gaming devices which are feature-wise close to the ideal features of both ideal gaming and ideal non-gaming devices.

- There are 15 sheets created for both gaming and non-gaming devices. First in each sheet added each feature and then made a filter based on the ideal feature for the selected feature. For example, in the below image, the screen size of all the gaming flagship from 2020-22 is selected, and also a filter is made which is based on the ideal screen size, here we considered screen size which is more closes to the ideal screen size that most users selected from the survey. A similar way is used for the rest of the 14 features in both gaming and non-gaming flagship devices.

File Data Worksheet Dashboard Story Analysis Map Format Window Help

Standard

Data Analytics Pages

Sheet1 (FLAGSHIP 20-22 ...)

Search

Tables

- Back Body
- Brand
- Camera Sensor By
- Camera Tuning By
- Category
- Chassis
- Chipset
- Display Dimension
- Display Resolution Name
- Display Type
- Model
- Processor Brand
- Release Year
- Screen Resolution

Filters

Category: Gaming

Screen Size(inch)

Marks

Automatic

Color Size Text

Detail Tooltip

SUM(Price INR)

Columns

Rows

Brand Model SUM(Screen Size(inch...))

IGP-ScreenSize

Brand	Model	Screen Size(inch)	Price INR
ASUS	ROG 5	6.78	55,999
IQOO	7	6.62	34,990
	7 LEGEND	6.62	44,990
LENOVO	LEGION DUEL	6.65	41,990
POCO	F3 GT	6.67	28,999
	X3 PRO	6.67	20,999
REALME	GT 2	6.62	34,999
	GT NEO 2	6.62	31,999
XIAOMI	Mi 10T	6.67	34,999
	Mi 10T PRO	6.67	39,999

- After creating all the 15 sheets for both gaming and non-gaming flagship devices, the dashboards were created based on the selected devices. From those selected devices, devices that came under more than 8 features out of the 15 features will consider as a real best product that has specifications more comparable to ideal specifications of the gaming and non-gaming flagship smartphones.

The dashboards were created as shown below,

Brand			
Brand	Model		
ASUS	ROG 3		59,999
	ROG 5		55,999
IQOO	7		34,990
	7 LEGEND		44,990
LENOVO	LEGION DUEL		41,990
	LEGION DUEL 2		42,190
ONEPLUS	9RT		42,999
POCO	F3 GT		28,999
	X3 PRO		20,999
REALME	GT		37,999
	GT 2		34,999
	GT NEO 2		31,999
XIAOMI	Mi 10T		34,999
	Mi 10T PRO		39,999

IGP-ScreenSize			
Brand	Model	Screen Size(inch)	Price INR
ASUS	ROG 5	6.78	55,999
IQOO	7	6.62	34,990
	7 LEGEND	6.62	44,990
LENOVO	LEGION DUEL	6.65	41,990
POCO	F3 GT	6.67	28,999
	X3 PRO	6.67	20,999
REALME	GT 2	6.62	34,999
	GT NEO 2	6.62	31,999
XIAOMI	Mi 10T	6.67	34,999
	Mi 10T PRO	6.67	39,999

Display Technology			
Brand	Model	Display Type	Price INR
ASUS	ROG 3	AMOLED	59,999
	ROG 5	E4-AMOLED	55,999
IQOO	7	AMOLED	34,990
	7 LEGEND	E4-AMOLED	44,990
LENOVO	LEGION DUEL	AMOLED	41,990
	LEGION DUEL 2	E4-AMOLED	42,190
ONEPLUS	9RT	Fluid-AMOLED	42,999
POCO	F3 GT	E4-AMOLED	28,999
REALME	GT	Super-AMOLED	37,999
	GT 2	E4-AMOLED	34,999
	GT NEO 2	E4-AMOLED	31,999

Min 480Hz			
Brand	Model	Touch Sample Rate(Hz)	Refresh Rate(Hz)
LENOVO	LEGION DUEL 2	720	144
ONEPLUS	9RT	600	120
POCO	F3 GT	480	120
REALME	GT 2	600	120
	GT NEO 2	600	120

RR 144Hz			
Brand	Model	Refresh Rate(Hz)	Touch Sam..
ASUS	ROG 3	144	270
	ROG 5	144	300
LENOVO	LEGION D..	144	240
	LEGION D..	144	720
XIAOMI	Mi 10T	144	144
	Mi 10T P..	144	144

Ideal FullHD+ screen			
Brand	Releas..	Model	Display..
ASUS	2020	ROG 3	FullHD+
	2021	ROG 5	FullHD+
IQOO	2021	7	FullHD+
		7 LEGEND	FullHD+
LENOVO	2020	LEGION DUEL	FullHD+
	2021	LEGION DUEL 2	FullHD+
ONEPLUS	2022	9RT	FullHD+
POCO	2021	F3 GT	FullHD+
		X3 PRO	FullHD+
REALME	2021	GT	FullHD+
		GT NEO 2	FullHD+
	2022	GT 2	FullHD+
XIAOMI	2020	Mi 10T	FullHD+
		Mi 10T PRO	FullHD+

Processor					Battery			
Brand	Model	Processor Brand	Chipset		Brand	Model	Battery (mAh)	Wired Charge..
ASUS	ROG 5	Snapdragon	888	55,999	ASUS	ROG 3	6,000	30
IQOO	7 LEGEND	Snapdragon	888	44,999	ASUS	ROG 5	6,000	65
LENOVO	LEGION DUEL 2	Snapdragon	888	42,199	LENOVO	LEGION DUEL	5,000	65
ONEPLUS	9RT	Snapdragon	888	42,999	LENOVO	LEGION DUEL 2	5,500	90
REALME	GT	Snapdragon	888	37,999	POCO	F3 GT	5,065	67
	GT 2	Snapdragon	888	34,999		X3 PRO	5,160	33
					REALME	GT 2	5,000	65
						GT NEO 2	5,000	65
					XIAOMI	Mi 10T	5,000	33
						Mi 10T PRO	5,000	33

Charger			
Brand	Model	Wired Charger(w)	
LENOVO	LEGION DUEL 2 90		42,199

Sensor					
Brand	Model	Camera Sensor	Rear Primary Camera	Rear Secondary Camera	Rear Tertiary Camera
LENOVO	LEGION DUEL	SAMSUNG	64	16	0
	LEGION DU..	SAMSUNG	64	16	0
XIAOMI	Mi 10T PRO	SAMSUNG	108	13	5

Camera					
Brand	Model	Rear Primary Camera	Rear Secondary Camera	Rear Tertiary Camera	Camera S..
ASUS	ROG 5	64	13	5	Sony
	ROG 3	64	13	5	Sony
LENOVO	LEGION D..	64	16	0	SAMSUNG
	LEGION D..	64	16	0	SAMSUNG
ONEPL..	9RT	50	16	2	Sony
XIAOMI	Mi 10T P..	108	13	5	SAMSUNG
	Mi 10T	64	13	5	Sony

Security Sensor			
Brand	Model	Security sensor	
ASUS	ROG 3	Optic In Display Fingerprint Scanner	59,999
	ROG 5	Optic In Display Fingerprint Scanner	55,999
IQOO	7	Optic In Display Fingerprint Scanner	54,999
	7 LEGEND	Optic In Display Fingerprint Scanner	44,999
LENOVO	LEGION D..	Optic In Display Fingerprint Scanner	41,999
	LEGION D..	Optic In Display Fingerprint Scanner	42,199
ONEPL..	9RT	Optic In Display Fingerprint Scanner	42,999
REALME	GT	Optic In Display Fingerprint Scanner	37,999
	GT 2	Optic In Display Fingerprint Scanner	34,999
	GT NEO 2	Optic In Display Fingerprint Scanner	31,999

weight			
Brand	Model	Weight(gm)	
IQOO	7	196	34,999
ONEPLUS	9RT	198.5	42,999
REALME	GT	186	37,999
	GT 2	199.8	34,999
	GT NEO 2	199.8	31,999

back body and frame				
Brand	Model	Back Body	Chasis	
ASUS	ROG 3	Glass	Aluminium	59,999
	ROG 5	Glass	Aluminium	55,999
IQOO	7 LEGEND	Glass	Aluminium	44,999
LENOVO	LEGION DUEL	Glass	Aluminium	41,999
	LEGION DUEL 2	Glass	Aluminium	42,199
ONEPLUS	9RT	Glass	Aluminium	42,999
POCO	F3 GT	Glass	Aluminium	28,999
REALME	GT 2	Glass	Aluminium	34,999
XIAOMI	Mi 10T	Glass	Aluminium	34,999
	Mi 10T PRO	Glass	Aluminium	39,999

thickness			
Brand	Model	Thickness	
IQOO	7	8.4	34,999
	7 LEGEND	8.7	44,999
ONEPLUS	9RT	8.3	42,999
POCO	F3 GT	8.3	28,999
REALME	GT	8.4	37,999
	GT 2	8.6	34,999
	GT NEO 2	9	31,999
XIAOMI	Mi 10T	9.3	34,999
	Mi 10T PRO	9.3	39,999

By selecting any of the brands from the Brand table in the first dashboard, it will highlight their model under each feature if the model is present under that particular feature. That is, from the above dashboard ONEPLUS 9RT is considered an ideal gaming device. However, ONEPLUS 9RT is lacking some features that are not near to ideal features like display refresh rate, battery, and such. A similar method is applied for all the devices and detected 5 best real gaming flagship devices and 11 best real normal flagship devices.

CHAPTER 8

TABLEAU: COMPARISON ANALYSIS

In this section, a comparative visual analysis is made in between the detected devices and their respective ideal product using tableau. In the tableau, this comparative analysis is created as an interactive dashboard. There are two different dashboards for both Gaming and Normal Flagship devices.

Using the dashboard, we can easily compare the difference between the features of the selected device of a selected brand and its respective ideal product. So, if the user expected to buy a gaming or a non-gaming flagship device below Rs.50000 then this analysis would help the user to find out the best device to go for, that is the major objective of this analysis.

TABLEAU SECTION FOR COMPARISON VISUALIZATION:

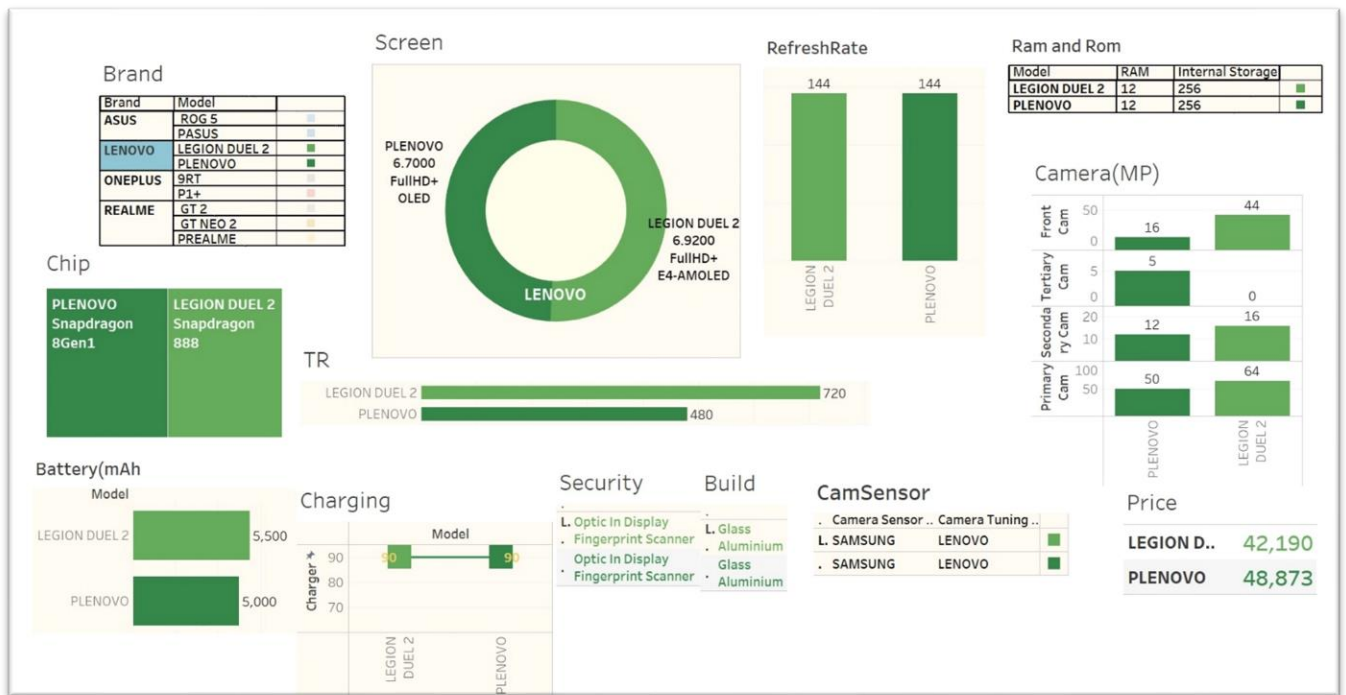
In this section, 13 sheets are created, based on the features, for both gaming and normal flagship device category to create interactive dashboard in each category to drive a comparative analysis between the detected models of respective brands with their respective ideal models. This comparison will help to understand how much the detected real best products (or the ideal products in real life) deviates from the predicted ideal products in terms of specifications.

Comparison dashboards for Gaming Flagships:

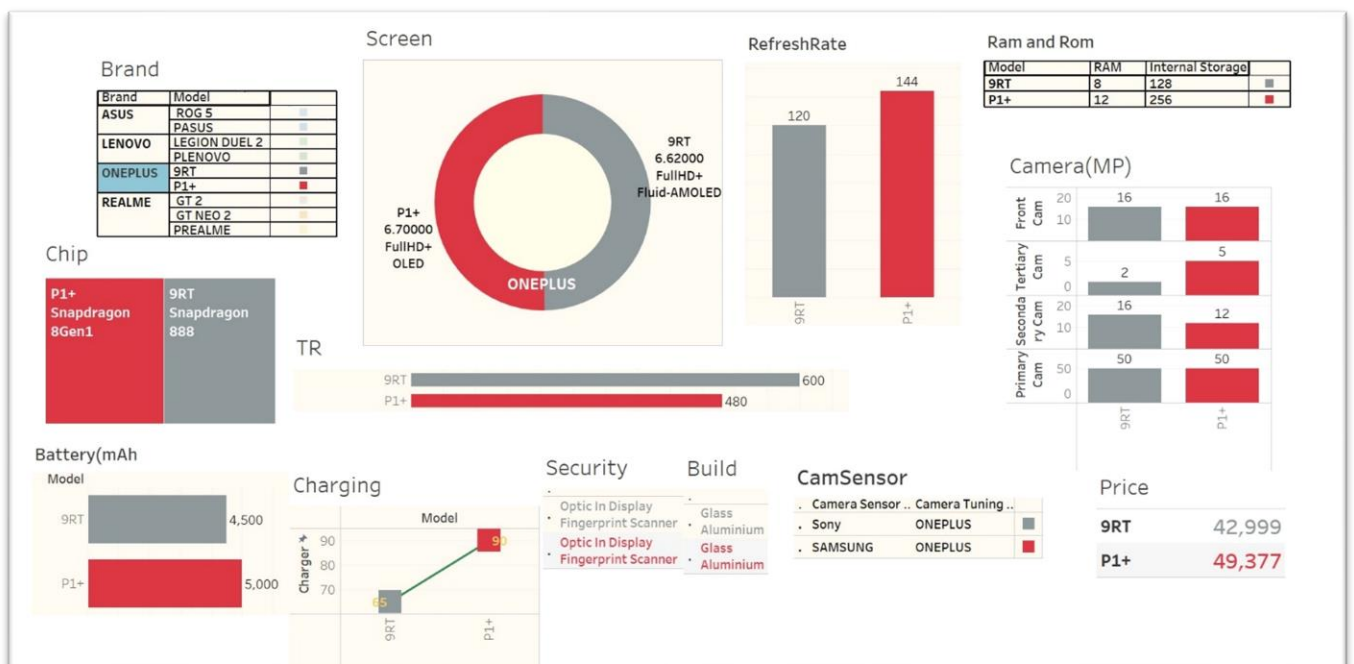
ASUS ROG 5 vs PASUS (P stands for predicted)



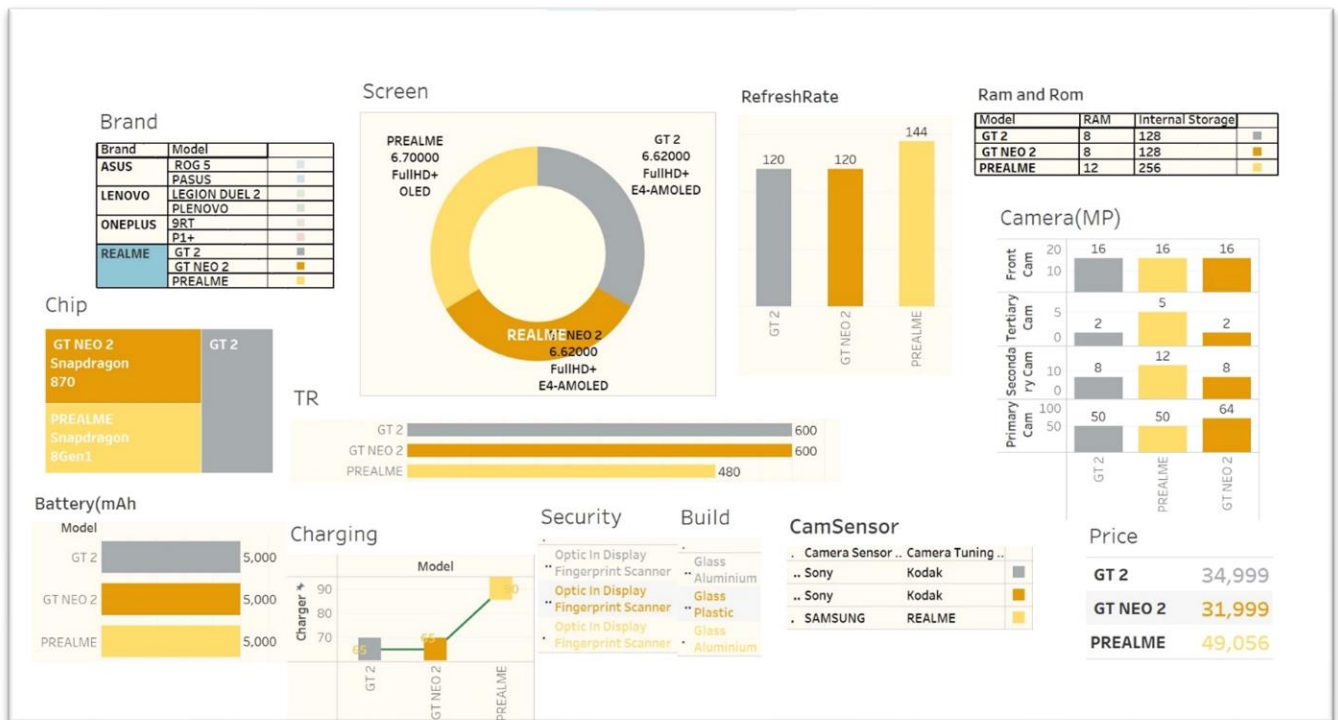
LENOVO LEGION DUEL 2 vs PLENOVO



ONEPLUS 9RT vs P1+

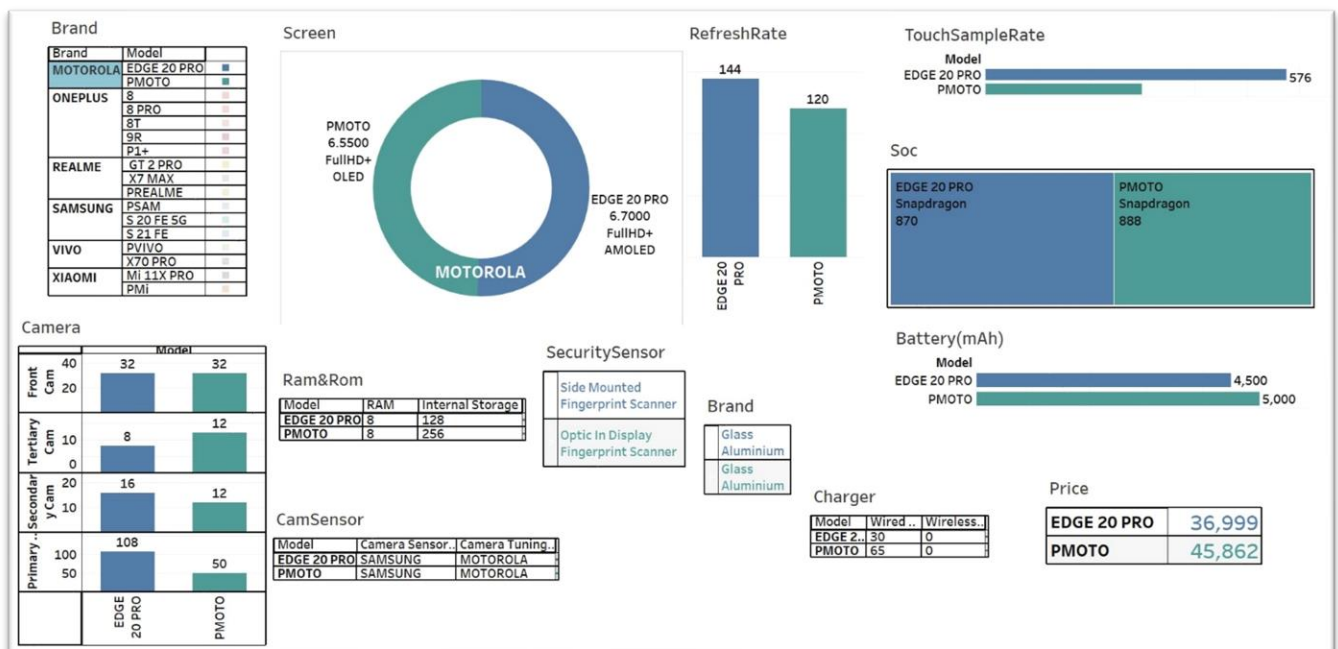


REALME GT 2 and GT NEO 2 vs PREALME

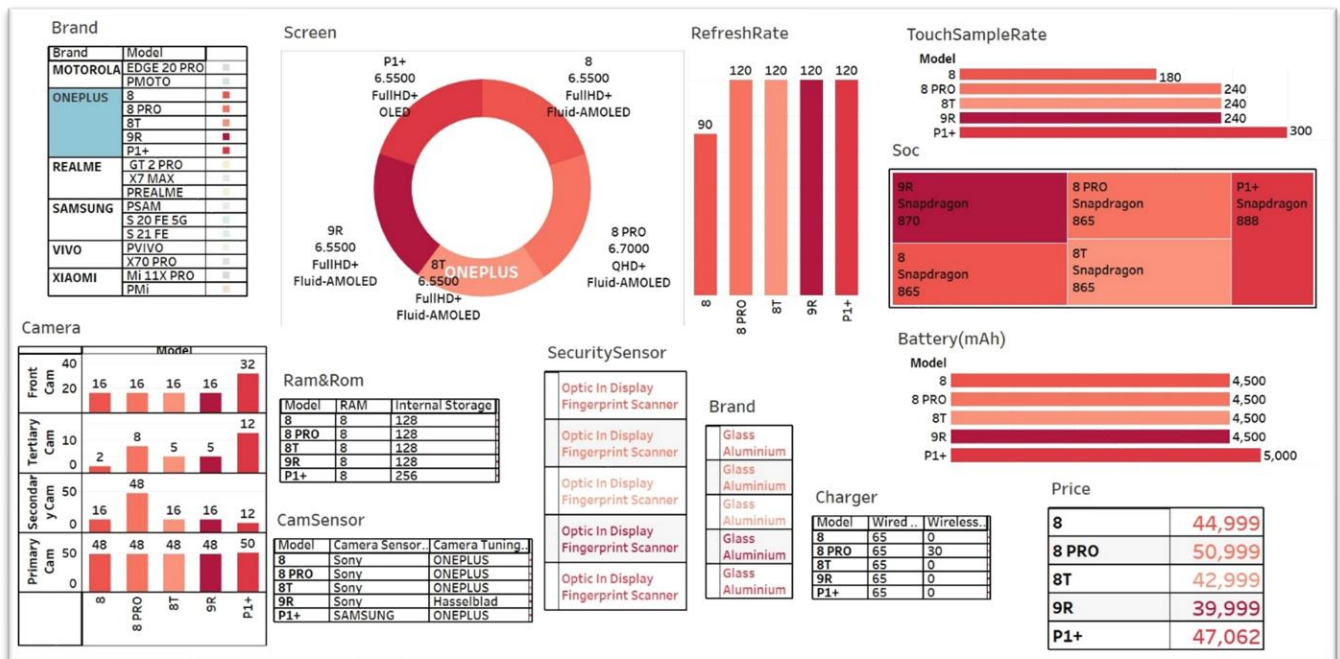


Comparison dashboards for Normal Flagships:

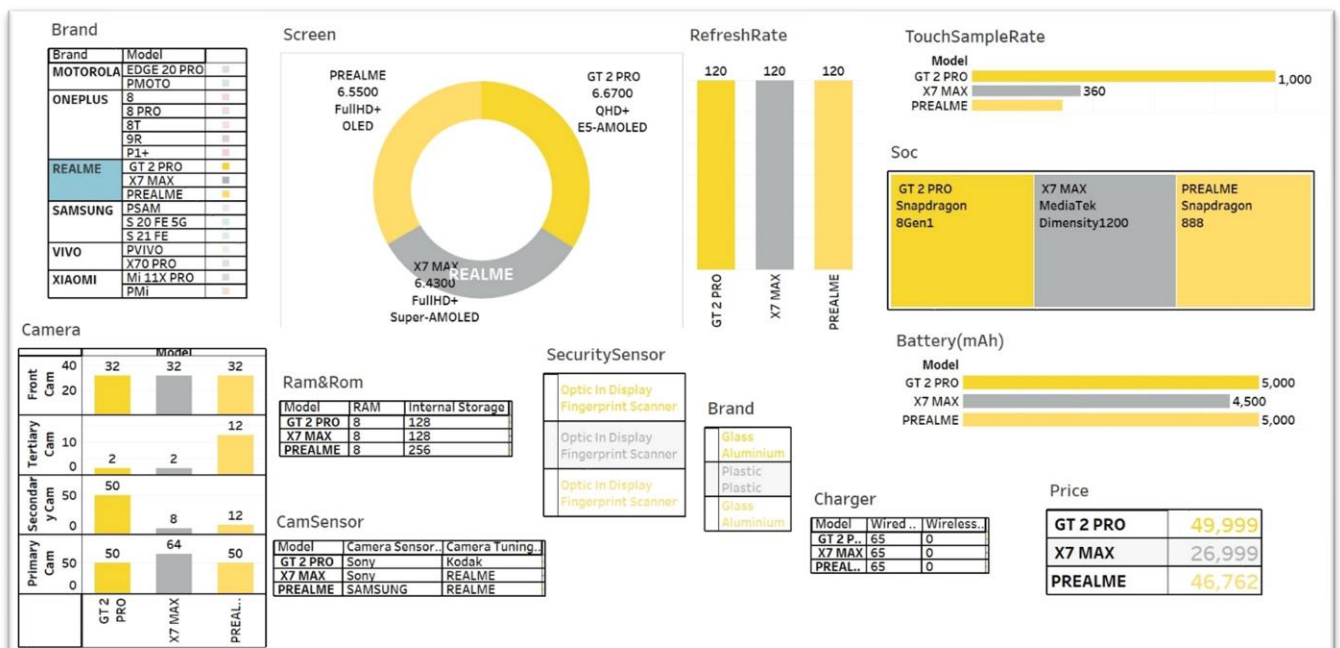
MOTOROLA EDGE 20 PRO vs PMOTOROLA



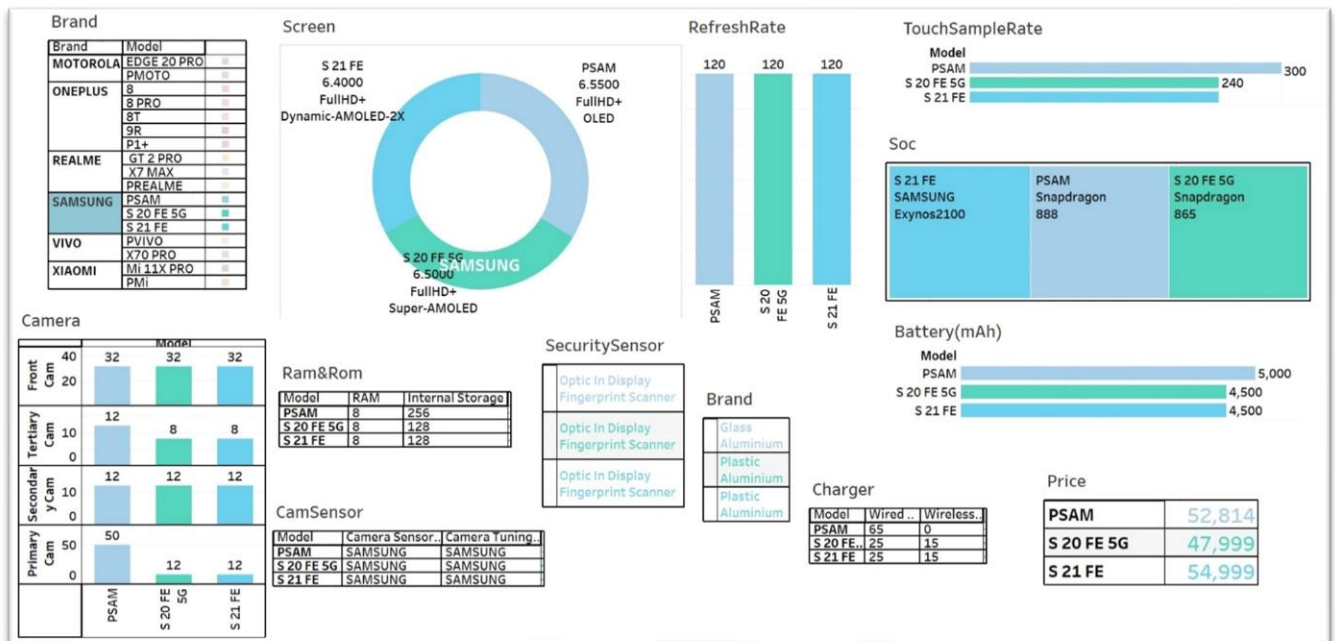
ONEPLUS 8, 8 PRO, 8T and 9R vs P1+



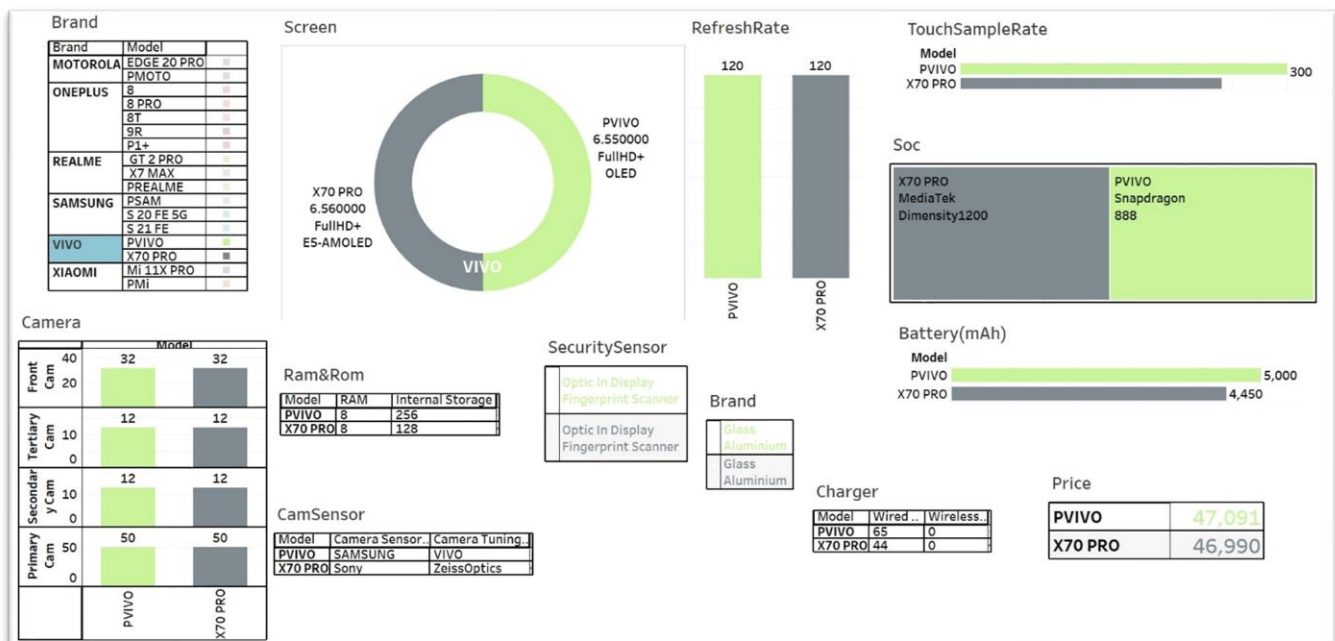
REALME GT 2 PRO and X7 MAX vs PREALME



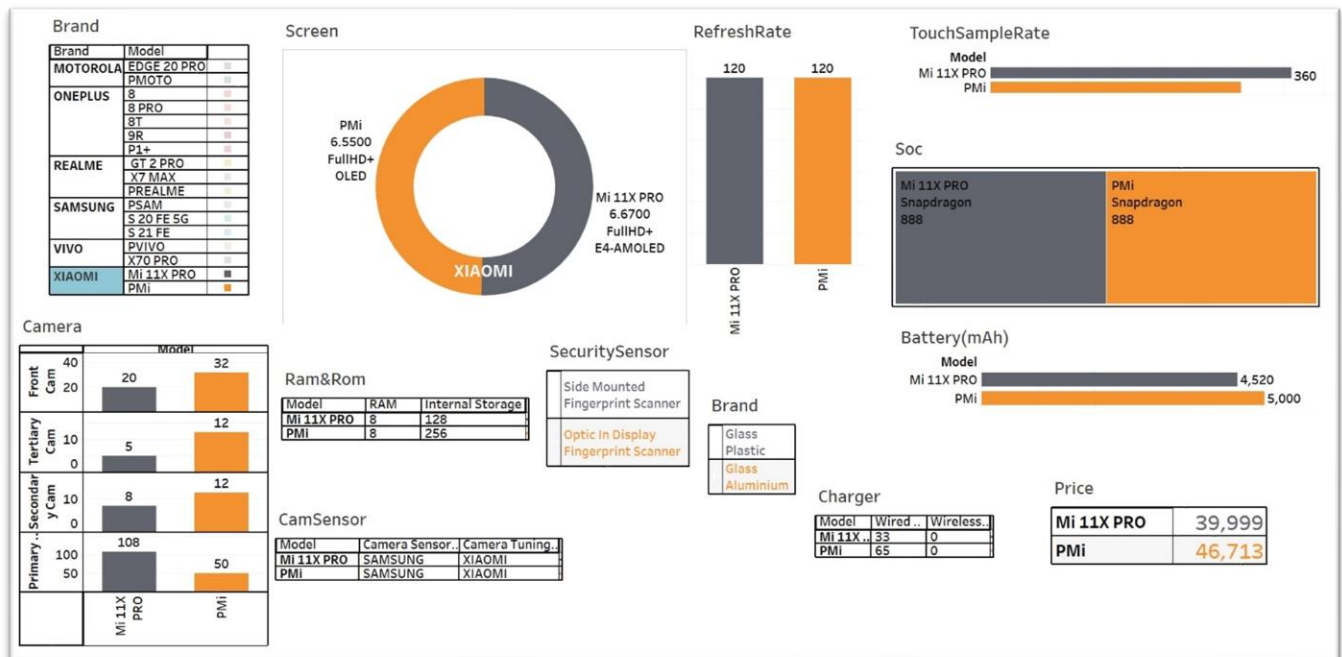
SAMSUNG S20 FE 5G and S21 FE vs PSAMSUNG



VIVO X70 PRO vs PVIVO



XIAOMI MI 11X PRO vs PMI



For both dashboards of Gaming and Normal Flagship products, interaction on dashboard is made by clicking any of the brand from the brand table, it will provide the comparison based which brand was chosen.

CHAPTER 9

CONCLUSION

The objective of this project is to find out or detect the flagship product that is so much of an ideal product type. For this project, only considered flagship devices since 2020, and get 69 different products. By considering these 69 products and their core features, I was made a Machine Learning model for the price as the independent variable. To identify the ideal features for a product, I have made a survey and from that survey I was able to identify the ideal features for both gaming and non-gaming smartphones. Then using those specs, predicted the ideal products of each company for different categories. By comparing the ideal product with the existing product from the collected data, detected the products which are close to ideal products in terms of features and made an interactive dashboard in tableau for a comparison. The detected devices are the best in this segment. Here the Ideal Product Detection is used only for flagship devices, similarly, it is able to find the best product for each category from entry-level to ultra-flagship by adding all kinds of devices into the data. This project demanded Flagship devices as Ideal Products, and in that way, this project achieved the objectives.

REFERENCE:

The data used for this project was collected from the following websites below,

<https://gadgets360.com/mobiles>

<https://www.91mobiles.com/>

<https://www.gsmarena.com/>

All the 3 websites above are legit and provide actual features regarding devices.