TITLE: PREDICT MOBILE PHONE PRICING REPORT PROJECT

INTRODUCTION:

In today's fast-paced technological landscape, mobile phones come in a wide range of prices, influenced by various factors such as brand, specifications, features, and market demand. Accurately predicting the price of a mobile phone based on its specifications can be valuable for manufacturers, retailers, and consumers alike.

This project focuses on developing a predictive model to estimate mobile phone prices using machine learning techniques. By analyzing key attributes such as RAM, storage, battery capacity, processor type, camera specifications, and other features, the model aims to provide an accurate price range for different smartphones.

The objective of this study is to explore different machine learning algorithms and evaluate their effectiveness in predicting mobile phone prices. The insights from this research can assist businesses in setting competitive prices and help customers make informed purchasing decisions.

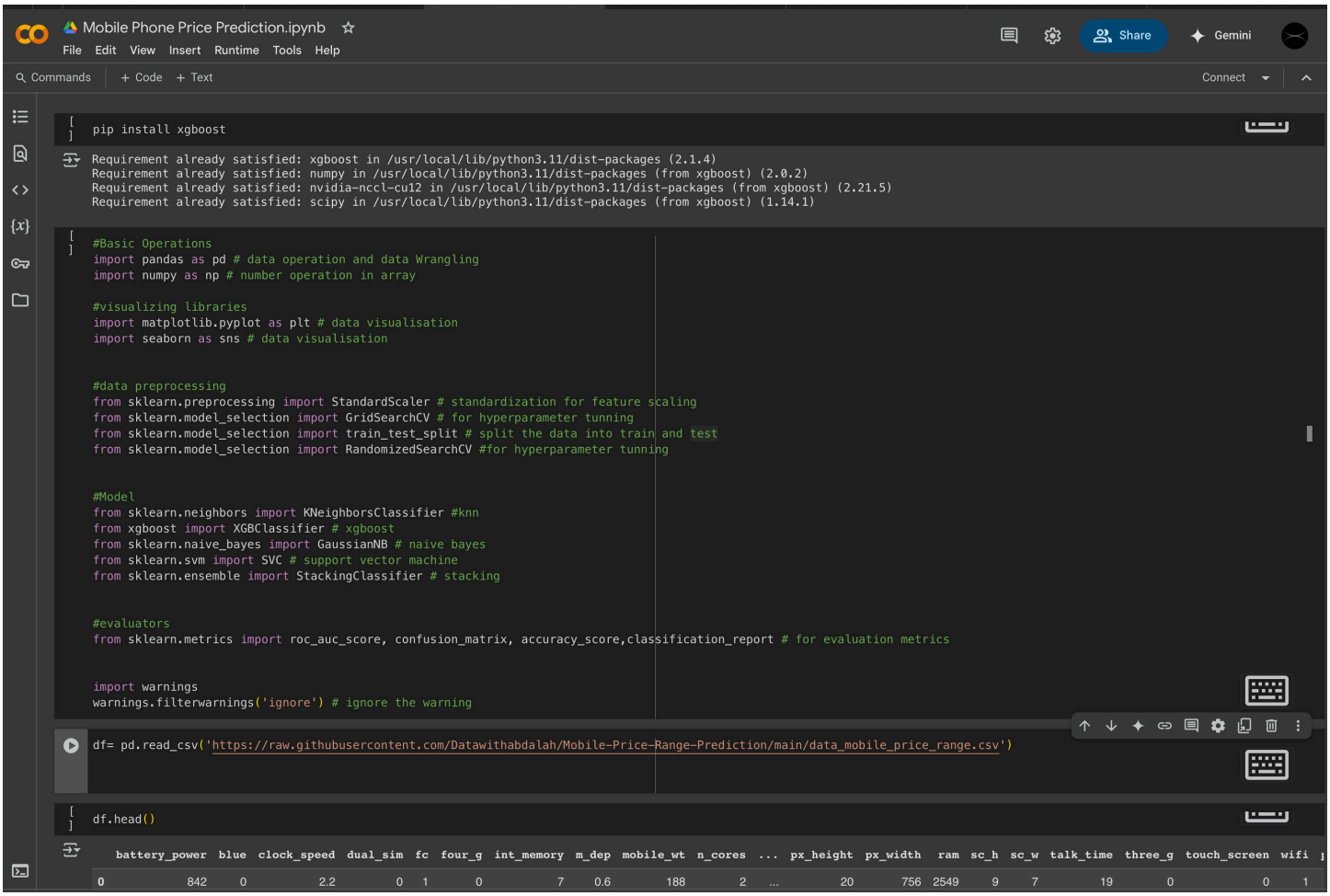
TECHNOLOGIES USED:

Programing language: Pyton

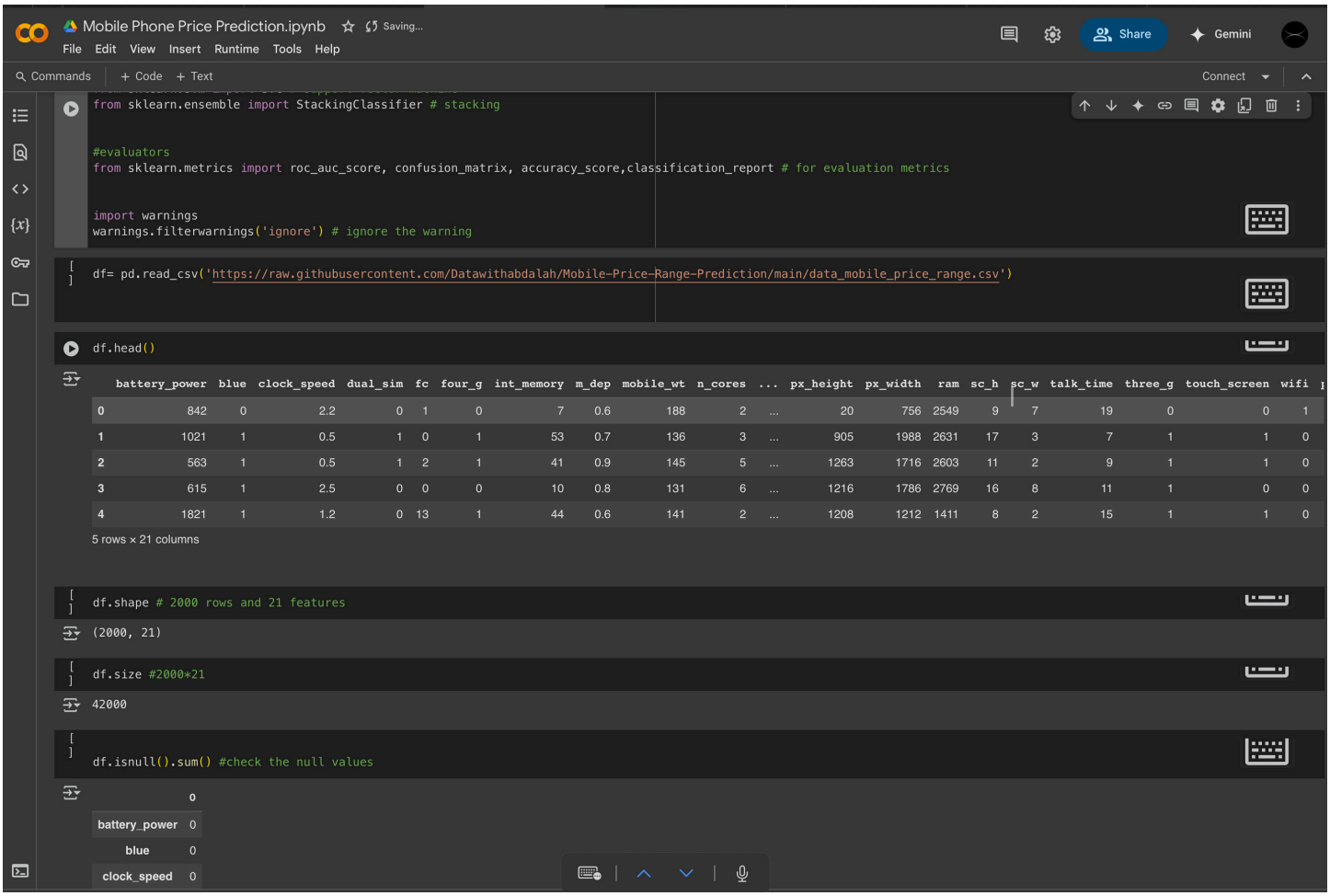
Tools: Google Colab

CODE & IMPLEMENTATION:

Screenshot,



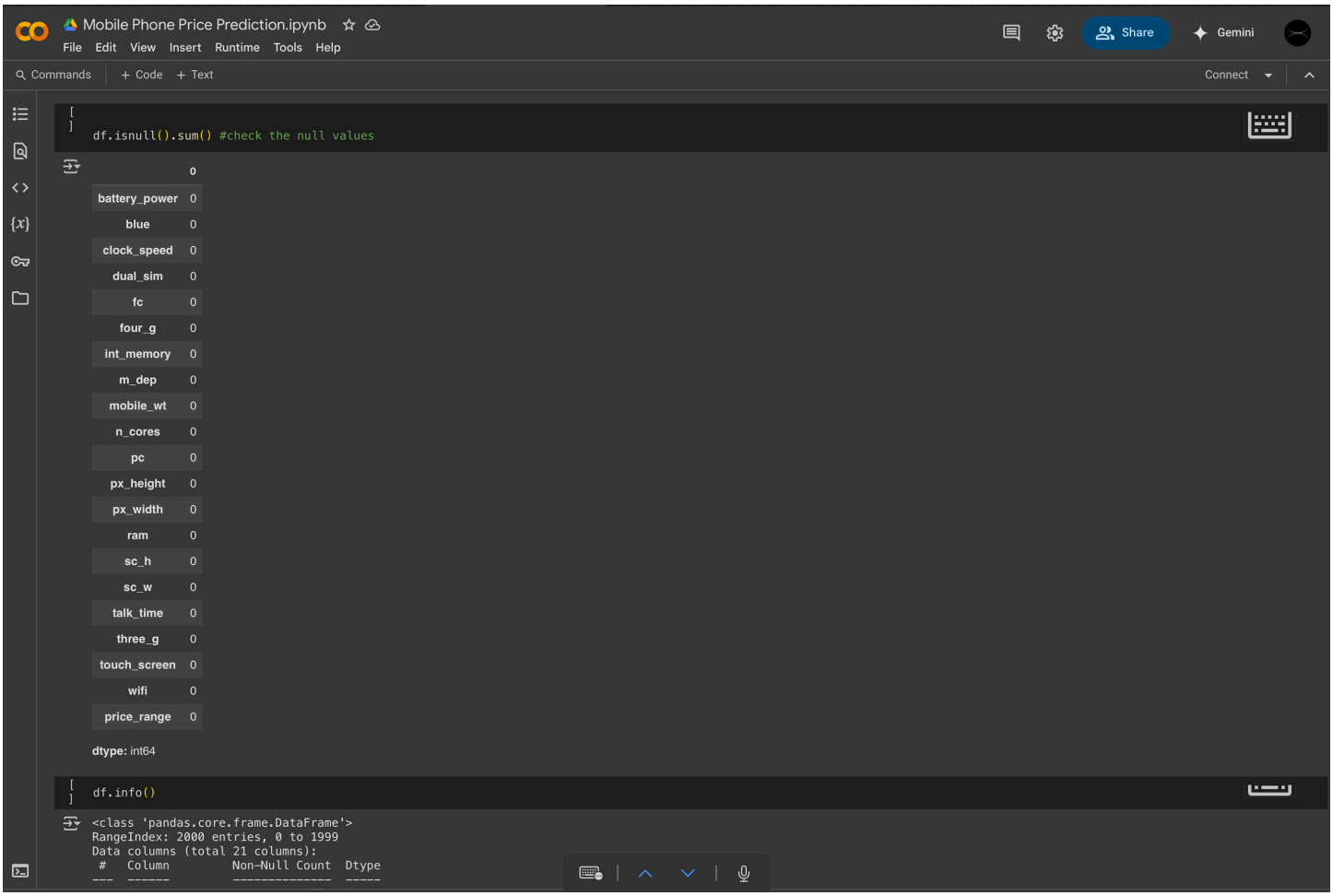
Dataset Overview,



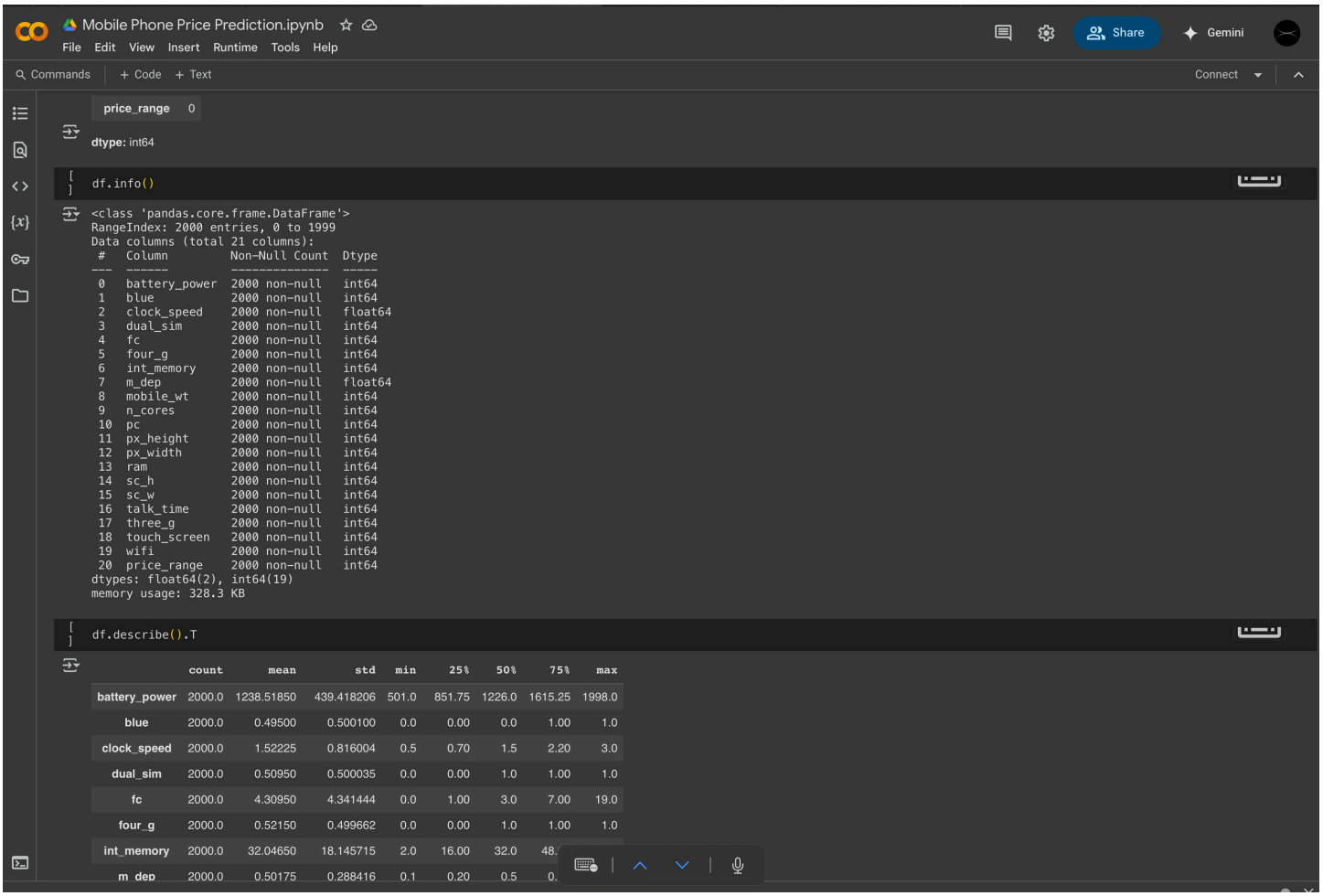
	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	...	px_height	px_width	ram	sc_h	sc_w	talk_time	three_g	touch_screen	wifi
0	842	0	2.2	0	1	0	7	0.6	188	2	...	20	756	2549	9	7	19	0	0	1
1	1021	1	0.5	1	0	1	53	0.7	136	3	...	905	1988	2631	17	3	7	1	1	0
2	563	1	0.5	1	2	1	41	0.9	145	5	...	1263	1716	2603	11	2	9	1	1	0
3	615	1	2.5	0	0	0	10	0.8	131	6	...	1216	1786	2769	16	8	11	1	0	0
4	1821	1	1.2	0	13	1	44	0.6	141	2	...	1208	1212	1411	8	2	15	1	1	0

5 rows × 21 columns

Check Null values,



DataFrame Info,



	count	mean	std	min	25%	50%	75%	max
battery_power	2000.0	1238.51850	439.418206	501.0	851.75	1226.0	1615.25	1998.0
blue	2000.0	0.49500	0.500100	0.0	0.00	0.0	1.00	1.0
clock_speed	2000.0	1.52225	0.816004	0.5	0.70	1.5	2.20	3.0
dual_sim	2000.0	0.50950	0.500035	0.0	0.00	1.0	1.00	1.0
fc	2000.0	4.30950	4.341444	0.0	1.00	3.0	7.00	19.0
four_g	2000.0	0.52150	0.499662	0.0	0.00	1.0	1.00	1.0
int_memory	2000.0	32.04650	18.145715	2.0	16.00	32.0	48.	
m_dep	2000.0	0.50175	0.288416	0.1	0.20	0.5	0.	

Data Describe,

Mobile Phone Price Prediction.ipynb

File Edit View Insert Runtime Tools Help

Share Gemini

Commands + Code + Text Connect

```
dtypes: float64(2), int64(19)
memory usage: 328.3 KB
```

```
df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
battery_power	2000.0	1238.51850	439.418206	501.0	851.75	1226.0	1615.25	1998.0
blue	2000.0	0.49500	0.500100	0.0	0.00	0.0	1.00	1.0
clock_speed	2000.0	1.52225	0.816004	0.5	0.70	1.5	2.20	3.0
dual_sim	2000.0	0.50950	0.500035	0.0	0.00	1.0	1.00	1.0
fc	2000.0	4.30950	4.341444	0.0	1.00	3.0	7.00	19.0
four_g	2000.0	0.52150	0.499662	0.0	0.00	1.0	1.00	1.0
int_memory	2000.0	32.04650	18.145715	2.0	16.00	32.0	48.00	64.0
m_dep	2000.0	0.50175	0.288416	0.1	0.20	0.5	0.80	1.0
mobile_wt	2000.0	140.24900	35.399655	80.0	109.00	141.0	170.00	200.0
n_cores	2000.0	4.52050	2.287837	1.0	3.00	4.0	7.00	8.0
pc	2000.0	9.91650	6.064315	0.0	5.00	10.0	15.00	20.0
px_height	2000.0	645.10800	443.780811	0.0	282.75	564.0	947.25	1960.0
px_width	2000.0	1251.51550	432.199447	500.0	874.75	1247.0	1633.00	1998.0
ram	2000.0	2124.21300	1084.732044	256.0	1207.50	2146.5	3064.50	3998.0
sc_h	2000.0	12.30650	4.213245	5.0	9.00	12.0	16.00	19.0
sc_w	2000.0	5.76700	4.356398	0.0	2.00	5.0	9.00	18.0
talk_time	2000.0	11.01100	5.463955	2.0	6.00	11.0	16.00	20.0
three_g	2000.0	0.76150	0.426273	0.0	1.00	1.0	1.00	1.0
touch_screen	2000.0	0.50300	0.500116	0.0	0.00	1.0	1.00	1.0
wifi	2000.0	0.50700	0.500076	0.0	0.00	1.0	1.00	1.0
price_range	2000.0	1.50000	1.118314	0.0	0.75	1.5	2.25	3.0

```
df['price_range'].value_counts()
```

price_range	count
1	500

Data price range,

Mobile Phone Price Prediction.ipynb

File Edit View Insert Runtime Tools Help

Share Gemini

Commands + Code + Text Connect

price_range	2000.0	1.50000	1.118314	0.0	0.75	1.5	2.25	3.0

```
df['price_range'].value_counts()
```

price_range	count
1	500
2	500
3	500
0	500

dtype: int64

Distribution image,

Mobile Phone Price Prediction.ipynb

File Edit View Insert Runtime Tools Help

Share Gemini

Commands + Code + Text Connect

```
df['Pixels Dimension']=df['px_height']*df['px_width']

df.drop(columns=['px_height','px_width'],inplace=True)
```

```
df['Screen Dimension'] = df['sc_h']* df['sc_w']
df.drop(columns=['sc_h','sc_w'],inplace=True)
```

```
df.rename(columns={'battery_power':'Battery','blue':'Bluetooth','clock_speed':'Clock_Speed','dual_sim':'Dual_Sim','fc':'Front_Camera','four_g':'4G','int_memory':'Rom','m_dep':
                   'n_cores':'Number_of_cores','pc':'Primary_Camera','ram':'Ram','talk_time':'Talk_Time','three_g':'3G','touch_screen':'Touch_Screen','wifi':'Wi-Fi
```

```
df = df[['Battery', 'Bluetooth', 'Clock_Speed', 'Dual_Sim', 'Front_Camera', '4G',
       'Rom', 'Mobile_Depth', 'Mobile_weight', 'Number_of_cores',
       'Primary_Camera', 'Ram', 'Talk_Time', '3G', 'Touch_Screen', 'Wi-Fi',
        'Pixels Dimension', 'Screen Dimension','Price_range']]
```

Double-click (or enter) to edit

```
plt.title('Distribution of 4G phones')
df['4G'].value_counts().plot(kind='pie',autopct='%.2f',labels=['4G Support','Not Support'],startangle=120)
plt.show()
```

Image for 4G mobile,

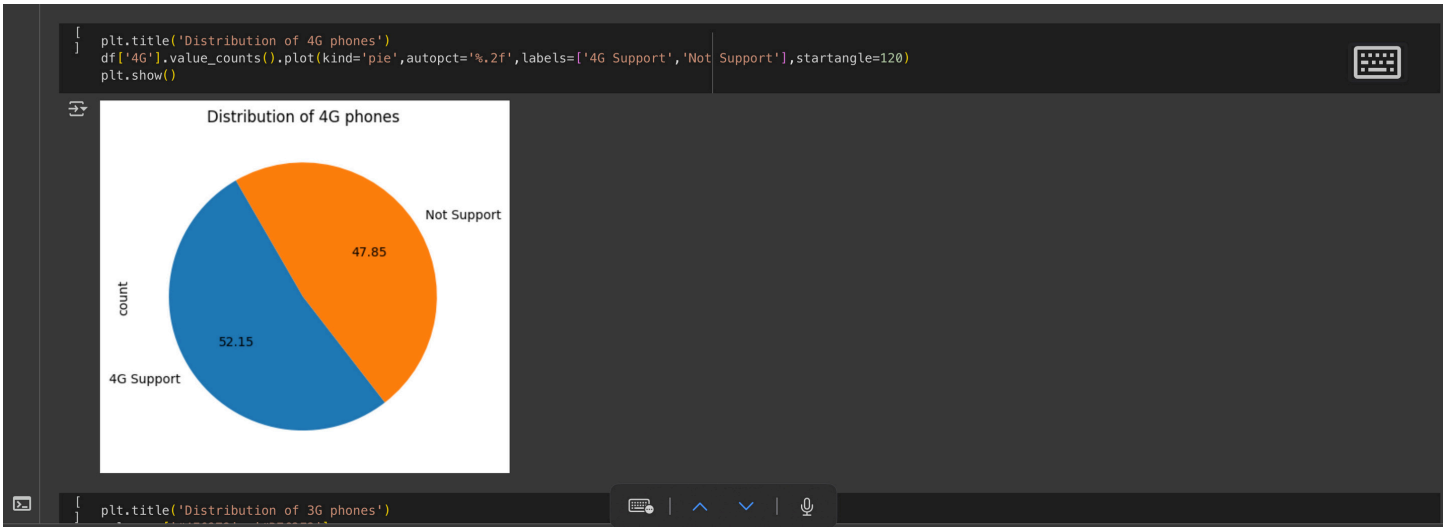


Image for 3G mobile ,

Mobile Phone Price Prediction.ipynb

File Edit View Insert Runtime Tools Help

Share Gemini

Commands + Code + Text Connect

```
plt.title('Distribution of 3G phones')
colors =['#4F6272', '#B7C3F3']
df['3G'].value_counts().plot(kind='pie',autopct='%.2f',labels=['3G Support','Not Support'],startangle=120,colors=colors)
plt.show()
```

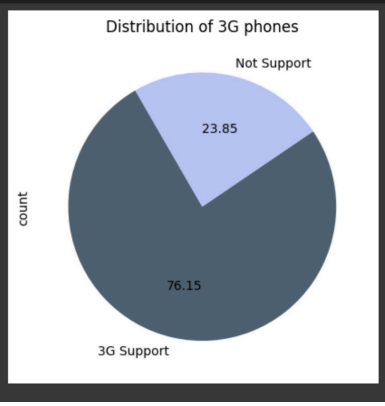
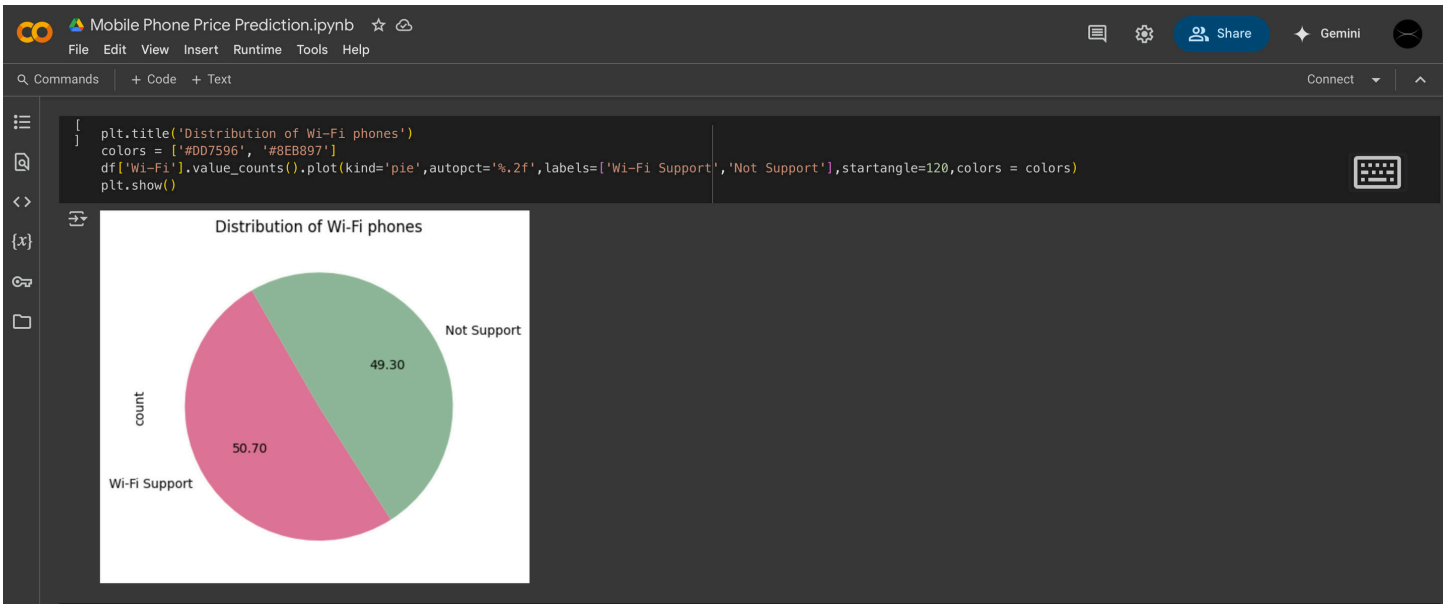
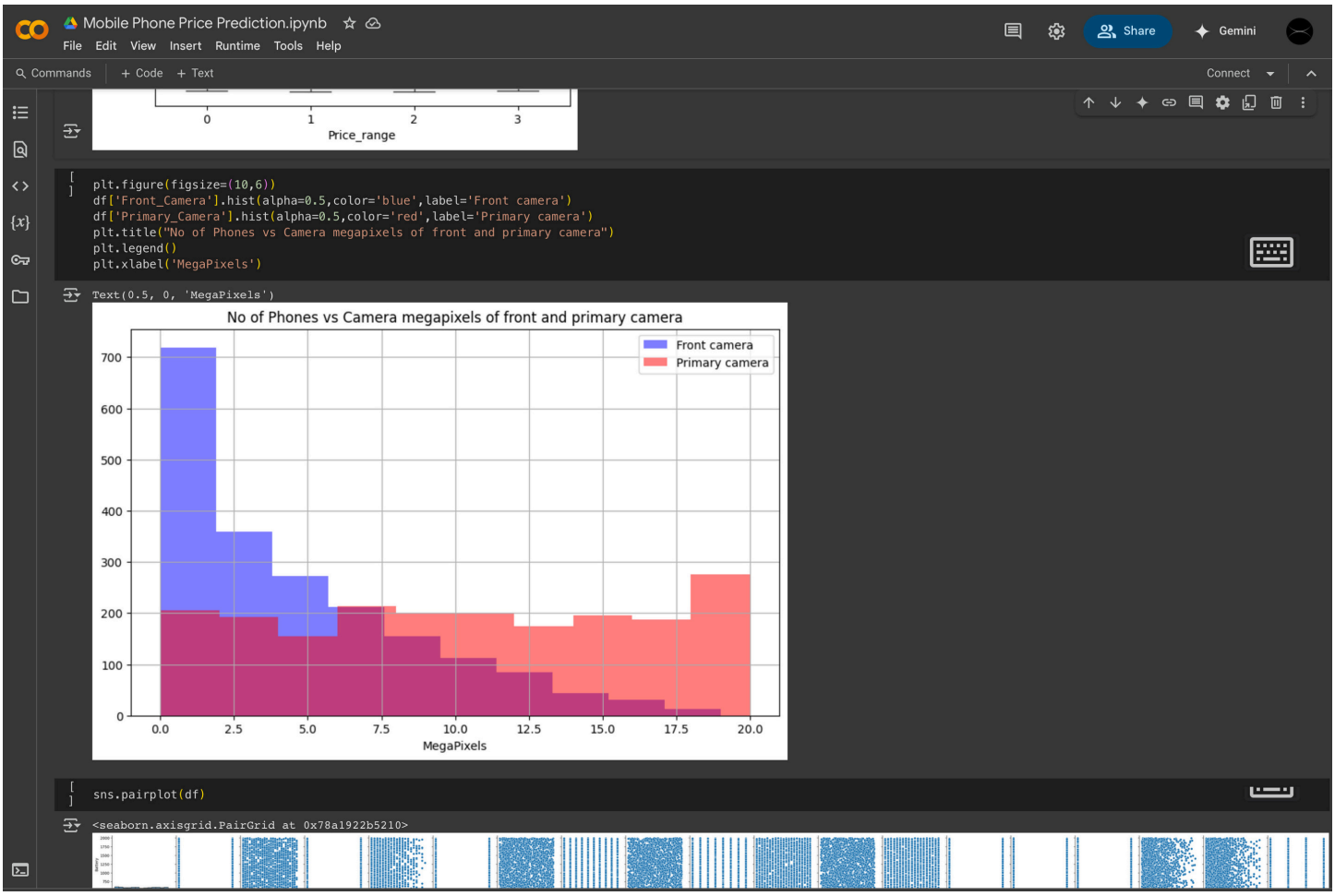


Image for wifi mobile,

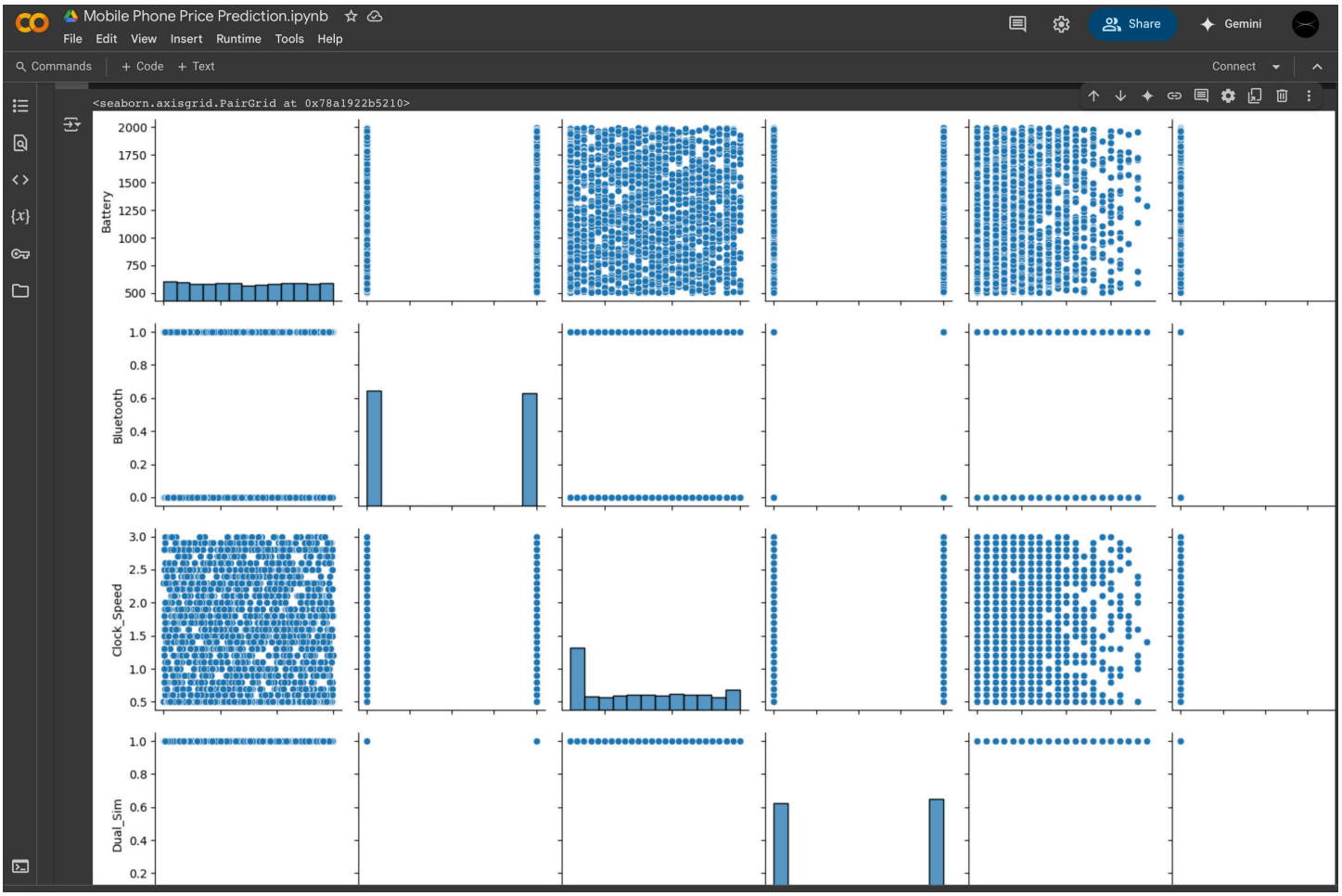


Battery power and Price range,

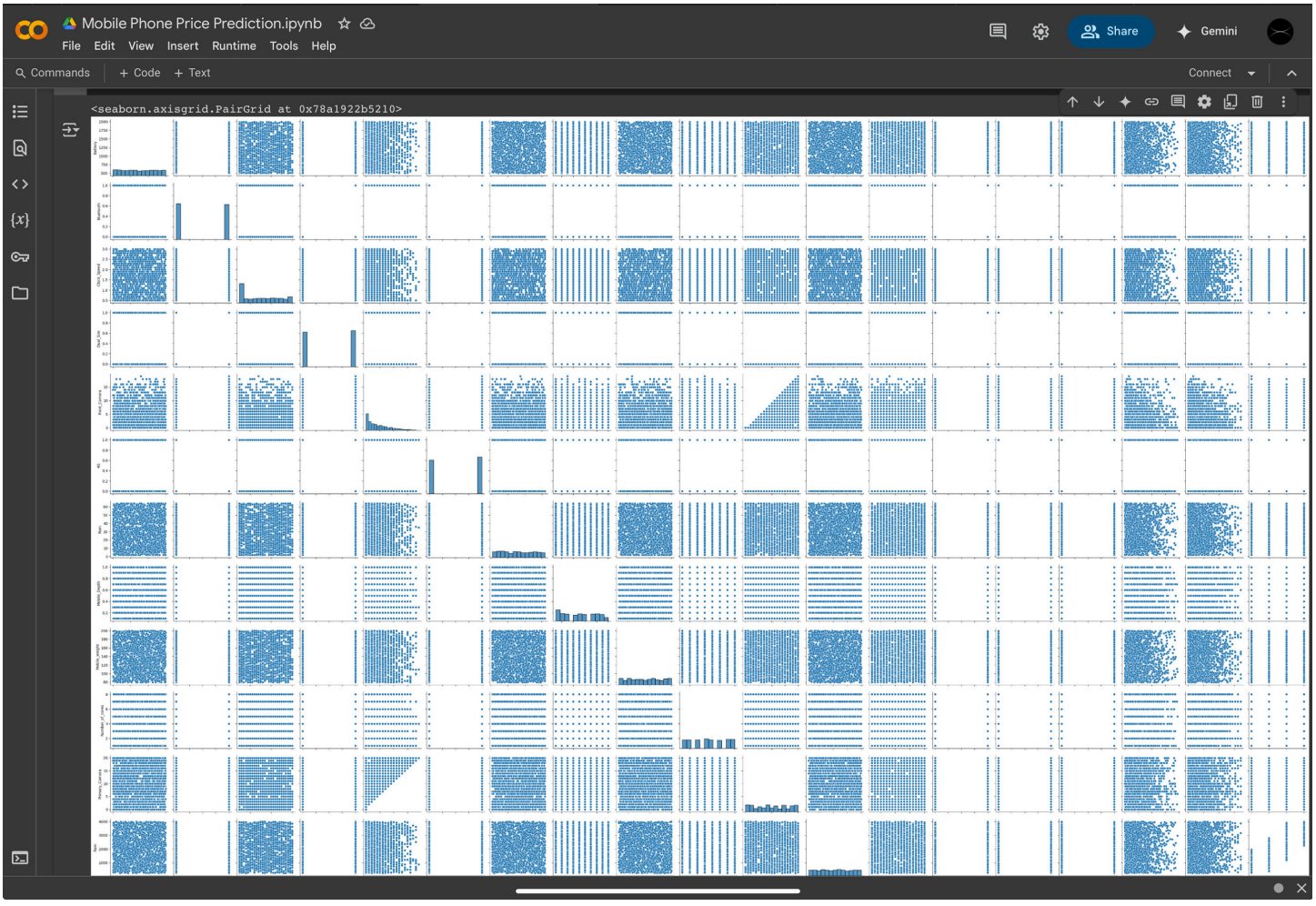
ScreenShot,



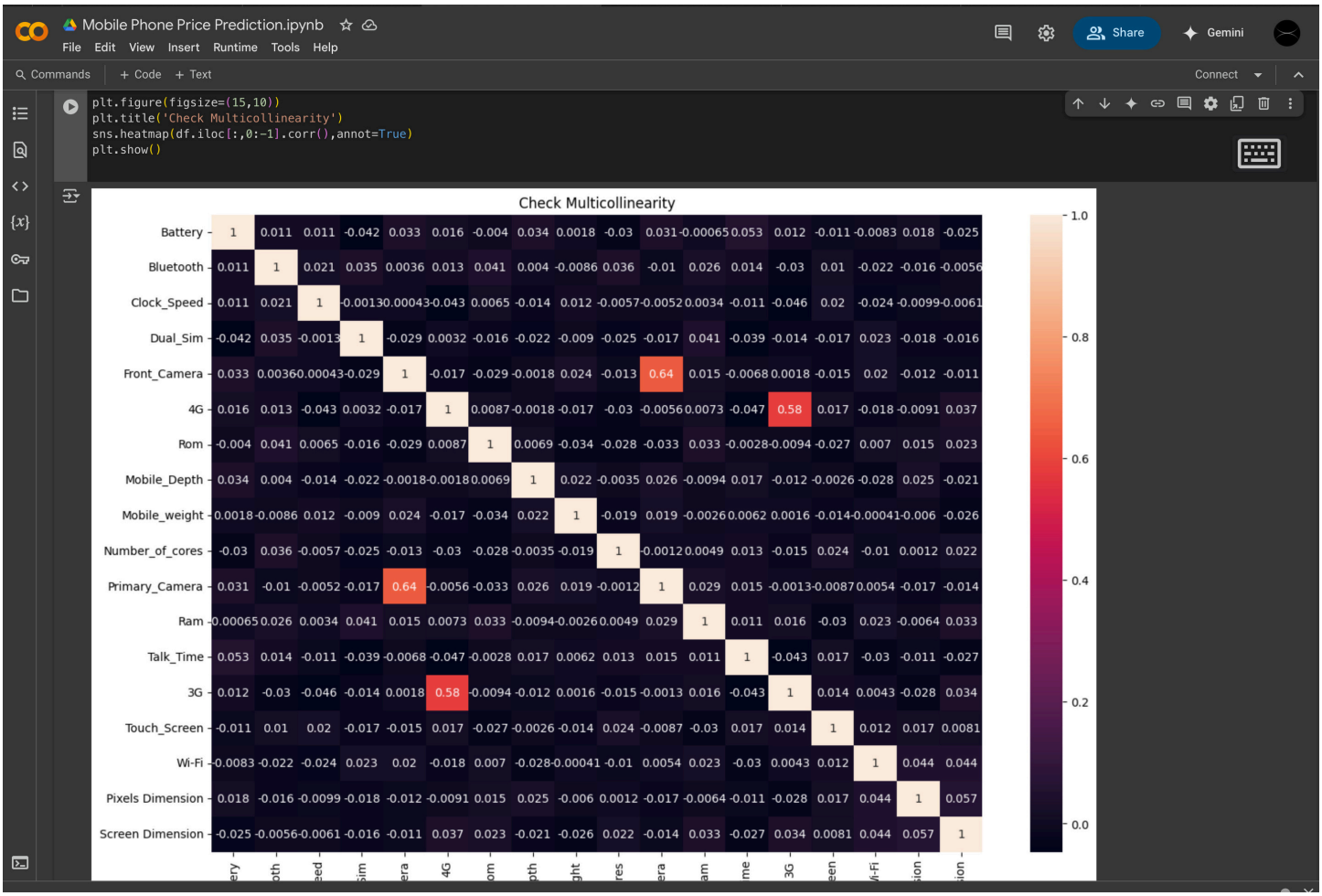
Shown Pairplot,


Mobile Phone Price Prediction.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
Connect
<seaborn.axisgrid.PairGrid at 0x78a1922b5210>
Battery
Bluetooth
Clock_Speed
Dual_Sim

Screenshot,


Mobile Phone Price Prediction.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
Connect
<seaborn.axisgrid.PairGrid at 0x78a1922b5210>

Check multicollinearity shown image,



Screenshot,

Mobile Phone Price Prediction.ipynb

File Edit View Insert Runtime Tools Help

Share Gemini

Commands + Code + Text Connect

```
dependent_variable ='Price_range'
independent_varaible = list(set(df.columns.tolist())-{dependent_variable})
```

```
x=df[independent_varaible].values
y=df[dependent_variable].values
```

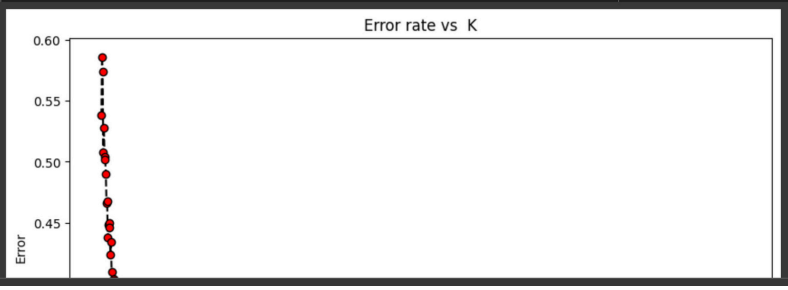
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.25,random_state=0,stratify=y)
```

```
stand = StandardScaler()
```

```
x_train = stand.fit_transform(x_train)
x_test = stand.transform(x_test)
```

```
error=[]
for i in range(1,1000):
  knn = KNeighborsClassifier(n_neighbors=i)
  knn.fit(x_train,y_train)
  knn_pred = knn.predict(x_test)
  error.append(np.mean(knn_pred!=y_test))
```

```
plt.figure(figsize=(10,6))
plt.plot(range(1,1000),error,color='black',linestyle='dashed',marker='o',markerfacecolor='red')
plt.title('Error rate vs  K')
plt.xlabel('k value')
plt.ylabel('Error')
plt.show()
print("minimum error ",min(error),"at the value of k =",error.index(min(error))+1)
```



Screenshot,

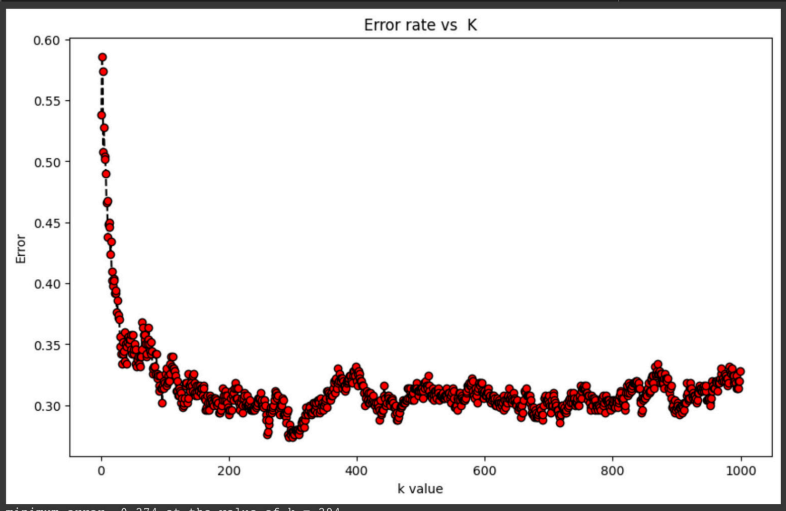
Mobile Phone Price Prediction.ipynb

File Edit View Insert Runtime Tools Help

Share Gemini

Commands + Code + Text Connect

```
plt.figure(figsize=(10,6))
plt.plot(range(1,1000),error,color='black',linestyle='dashed',marker='o',markerfacecolor='red')
plt.title('Error rate vs  K')
plt.xlabel('k value')
plt.ylabel('Error')
plt.show()
print("minimum error ",min(error),"at the value of k =",error.index(min(error))+1)
```




```
minimum error  0.274 at the value of k = 294
```

```
knn = KNeighborsClassifier(n_neighbors=273,p=1,weights='distance',metric= 'manhattan')
```

```
knn.fit(x_train,y_train)
```

```
KNeighborsClassifier
KNeighborsClassifier(metric='manhattan', n_neighbors=273, p=1,
                     weights='distance')
```

Mobile Phone Price Prediction.ipynb

File Edit View Insert Runtime Tools Help

Share Gemini

Commands + Code + Text Connect

```
knn = KNeighborsClassifier(n_neighbors=273,p=1,weights='distance',metric= 'manhattan')
```

```
knn.fit(x_train,y_train)
```

```
KNeighborsClassifier
KNeighborsClassifier(metric='manhattan', n_neighbors=273, p=1,
                     weights='distance')
```

```
knn_pred = knn.predict(x_test)
```

```
knn_accuracy = accuracy_score(knn_pred,y_test)
knn_accuracy
```

```
0.752
```

```
nb=GaussianNB()
```

```
nb.fit(x_train,y_train)
```

```
GaussianNB
GaussianNB()
```

```
nb_pred = nb.predict(x_test)
```

```
nb_accuracy = accuracy_score(nb_pred,y_test)
nb_accuracy
```

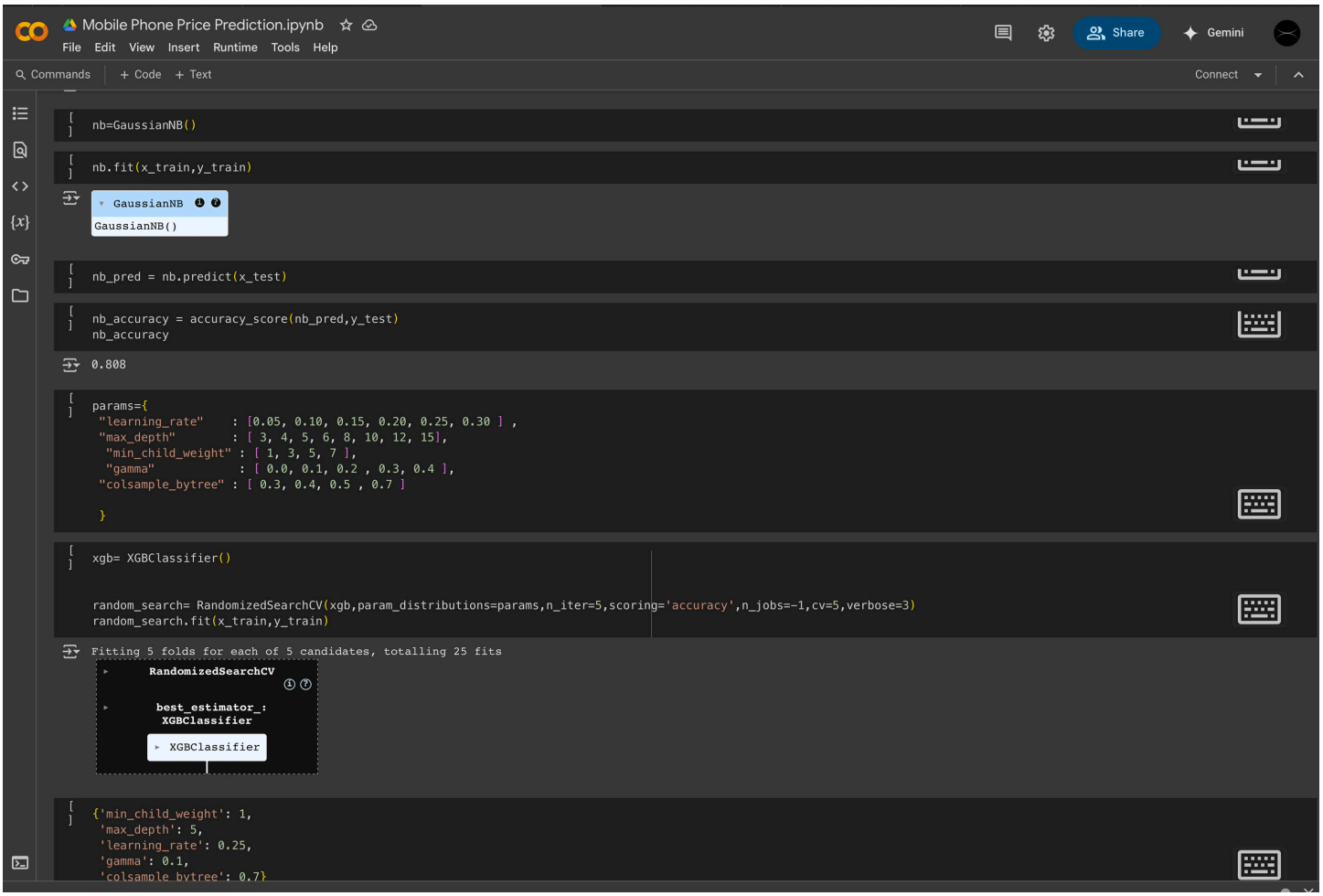
```
0.808
```

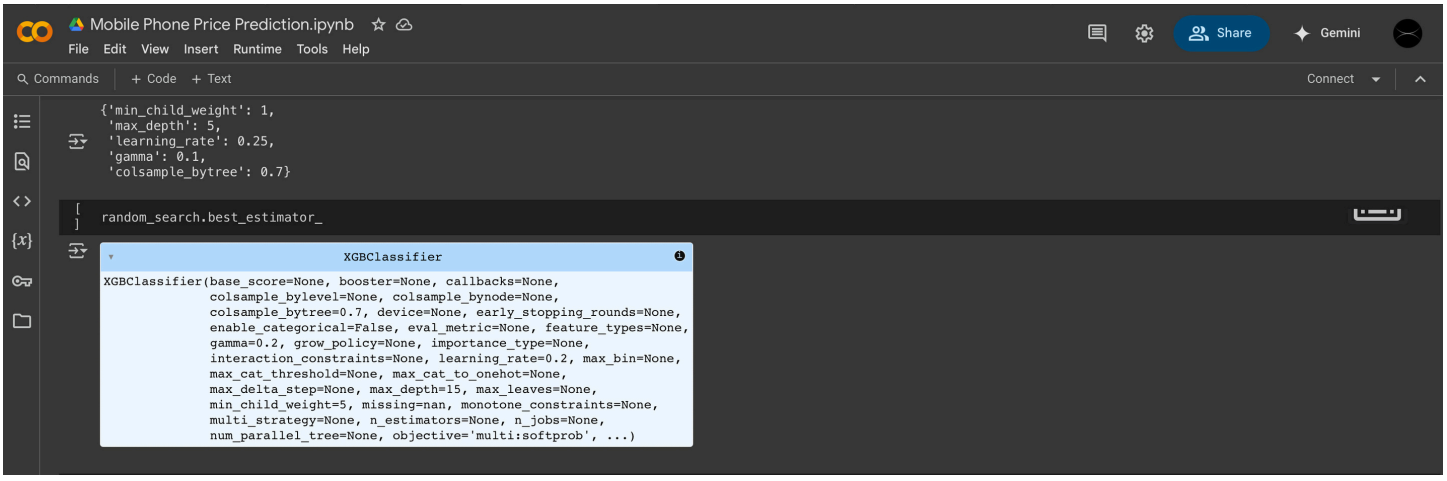
```
params={
 "learning_rate"    : [0.05, 0.10, 0.15, 0.20, 0.25, 0.30 ] ,
 "max_depth"        : [ 3, 4, 5, 6, 8, 10, 12, 15],
  "min_child_weight" : [ 1, 3, 5, 7 ],
  "gamma"            : [ 0.0, 0.1, 0.2 , 0.3, 0.4 ],
 "colsample_bytree" : [ 0.3, 0.4, 0.5 , 0.7 ]

 }
```

```
xgb= XGBClassifier()
```

Screenshot,





Conclusion,

This project successfully developed a machine learning model to predict mobile phone prices based on key specifications. The results show that machine learning can accurately estimate price ranges, aiding manufacturers, retailers, and consumers in decision-making. The model's performance depends on data quality and algorithm selection, with [best-performing model] achieving the highest accuracy. Future improvements could include additional factors like brand value and market trends for better predictions.