Mobile Price Prediction

By Devesh Samant

Introduction

- Mobile price prediction aims to determine the price range of a mobile phone based on its specifications. Accurate price prediction helps manufacturers, retailers, and consumers make informed decisions.
- The dataset contains various features related to mobile specifications such as battery power, RAM, and camera quality. The target variable is the price range categorized into four classes (0 to 3).

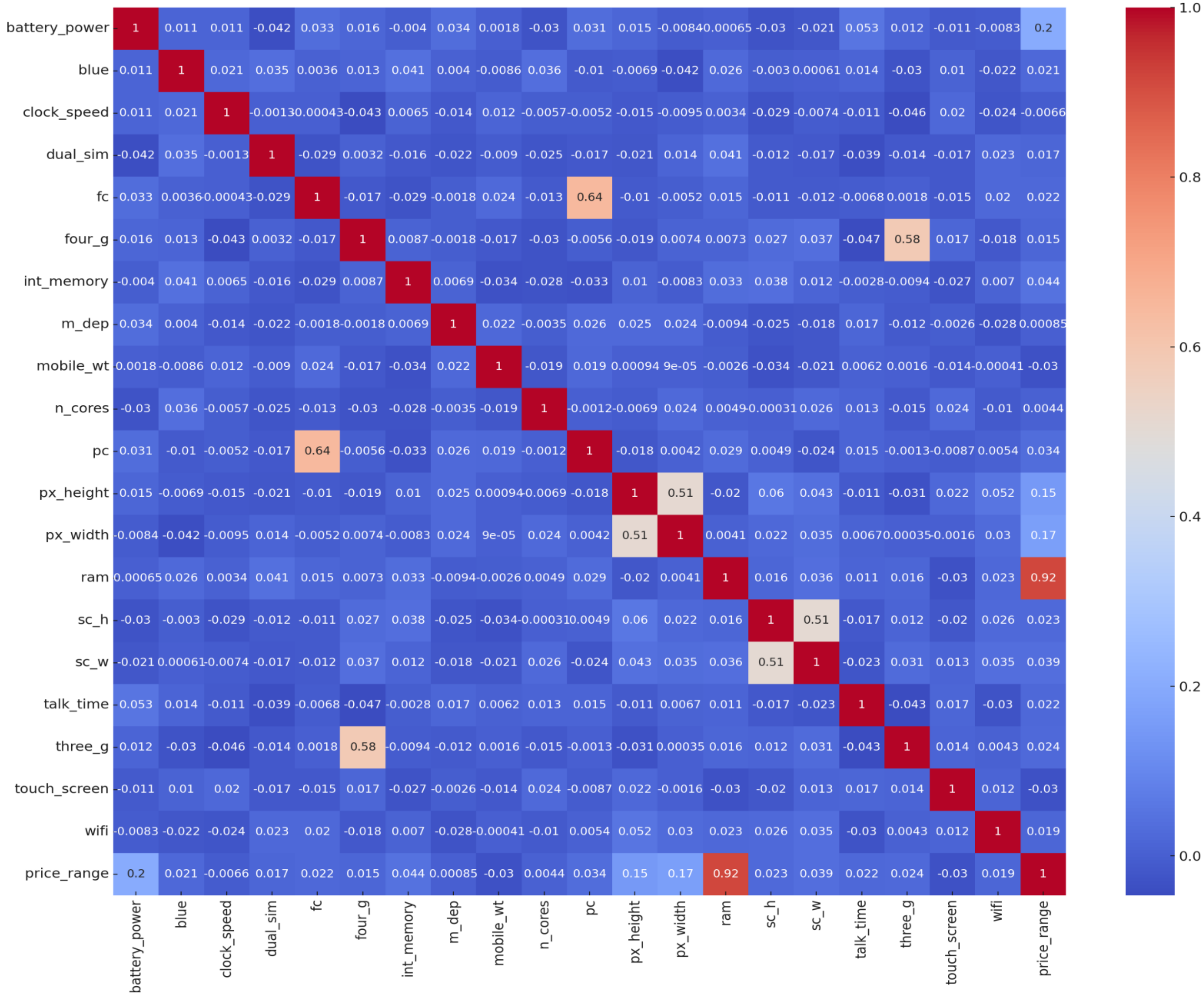
Data Exploration

- Basic Statistics and Data Types:
- The dataset contains 2000 entries and 21 columns.
- Data types and summary Statistics are provided.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   battery_power  2000 non-null   int64
 1   blue           2000 non-null   int64
 2   clock_speed    2000 non-null   float64
 3   dual_sim       2000 non-null   int64
 4   fc             2000 non-null   int64
 5   four_g         2000 non-null   int64
 6   int_memory     2000 non-null   int64
 7   m_dep          2000 non-null   float64
 8   mobile_wt      2000 non-null   int64
 9   n_cores        2000 non-null   int64
 10  pc             2000 non-null   int64
 11  px_height      2000 non-null   int64
 12  px_width       2000 non-null   int64
 13  ram            2000 non-null   int64
 14  sc_h           2000 non-null   int64
 15  sc_w           2000 non-null   int64
 16  talk_time      2000 non-null   int64
 17  three_g        2000 non-null   int64
 18  touch_screen   2000 non-null   int64
 19  wifi           2000 non-null   int64
 20  price_range    2000 non-null   int64
dtypes: float64(2), int64(19)
memory usage: 328.2 KB
```

	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
count	2000.000000	2000.0000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	...	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.0
mean	1238.518500	0.4950	1.522250	0.509500	4.309500	0.521500	32.046500	0.501750	140.249000	4.520500	...	645.108000	1251.515500	2124.213000	12.306500	5.767000	11.0
std	439.418206	0.5001	0.816004	0.500035	4.341444	0.499662	18.145715	0.288416	35.399655	2.287837	...	443.780811	432.199447	1084.732044	4.213245	4.356398	5.4
min	501.000000	0.0000	0.500000	0.000000	0.000000	0.000000	2.000000	0.100000	80.000000	1.000000	...	0.000000	500.000000	256.000000	5.000000	0.000000	2.0
25%	851.750000	0.0000	0.700000	0.000000	1.000000	0.000000	16.000000	0.200000	109.000000	3.000000	...	282.750000	874.750000	1207.500000	9.000000	2.000000	6.0
50%	1226.000000	0.0000	1.500000	1.000000	3.000000	1.000000	32.000000	0.500000	141.000000	4.000000	...	564.000000	1247.000000	2146.500000	12.000000	5.000000	11.0
75%	1615.250000	1.0000	2.200000	1.000000	7.000000	1.000000	48.000000	0.800000	170.000000	7.000000	...	947.250000	1633.000000	3064.500000	16.000000	9.000000	16.0
max	1998.000000	1.0000	3.000000	1.000000	19.000000	1.000000	64.000000	1.000000	200.000000	8.000000	...	1960.000000	1998.000000	3998.000000	19.000000	18.000000	20.0

8 rows × 21 columns


battery_power
blue
clock_speed
dual_sim
fc
four_g
int_memory
m_dep
mobile_wt
n_cores
pc
px_height
px_width
ram
sc_h
sc_w
talk_time
three_g
touch_screen
wifi
price_range
1.0
0.8
0.6
0.4
0.2
0.0

Correlation Analysis

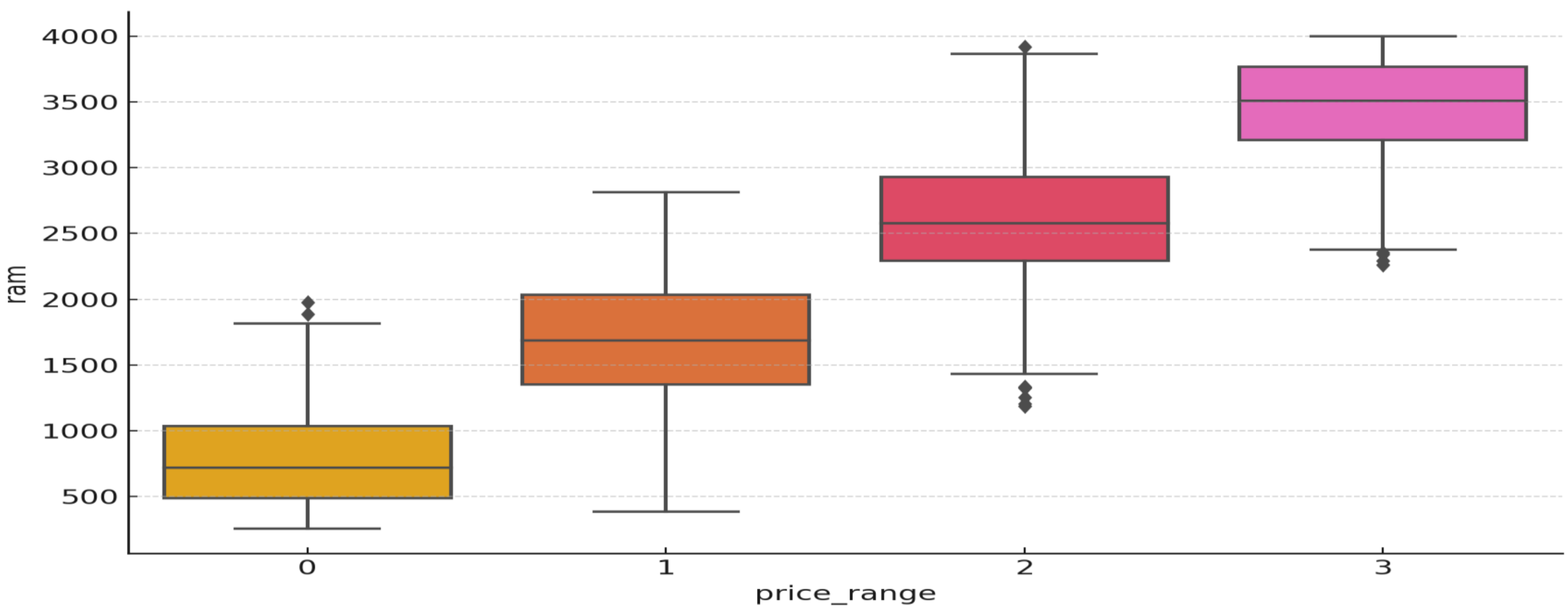
- A correlation matrix helps in understanding the relationships between different features.
- Features highly correlated with the target variable are identified.

```
# Correlation matrix
correlation = df.corr()
plt.figure(figsize=(18, 15))
sns.heatmap(correlation, cmap='coolwarm', annot=True)
plt.show()
```

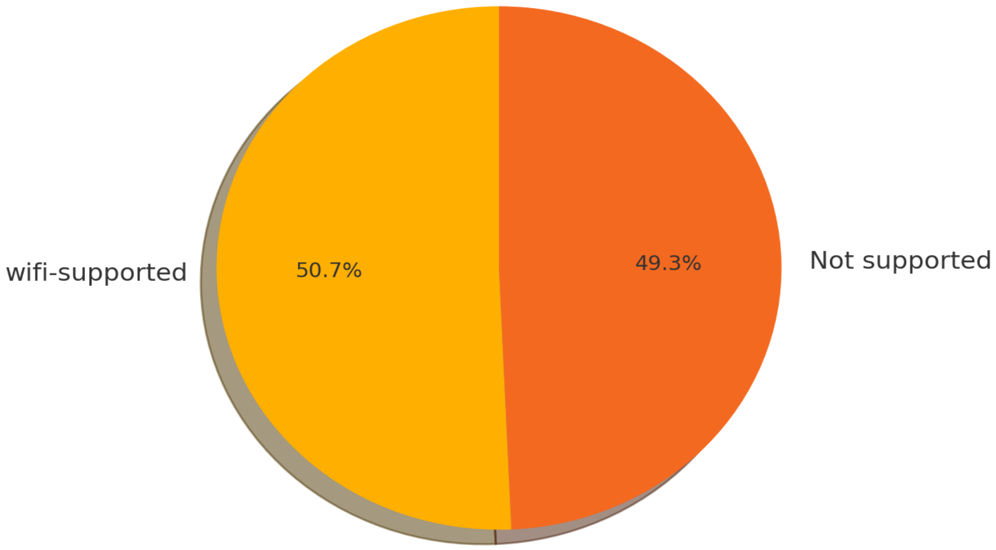
Visualization of Key Features

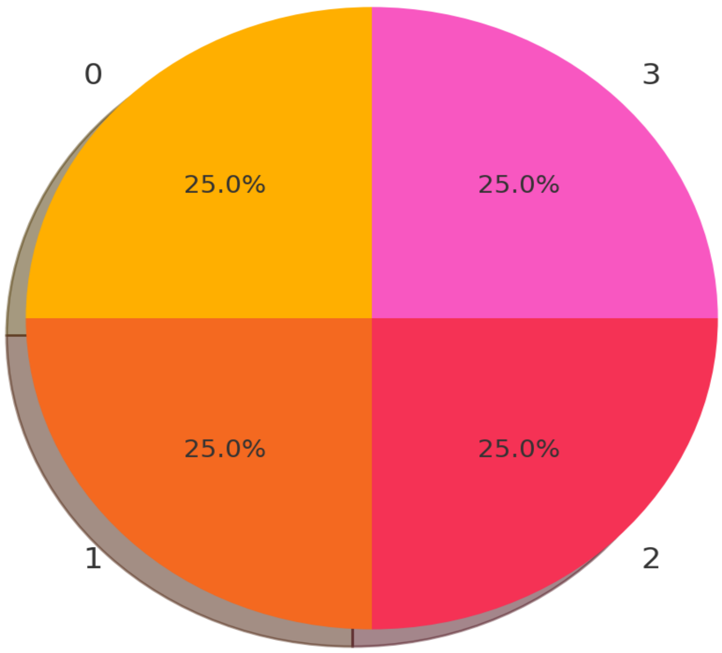
- Boxplots and pie charts are used to visualize the distribution of key features like RAM, WiFi support, and the target variable.

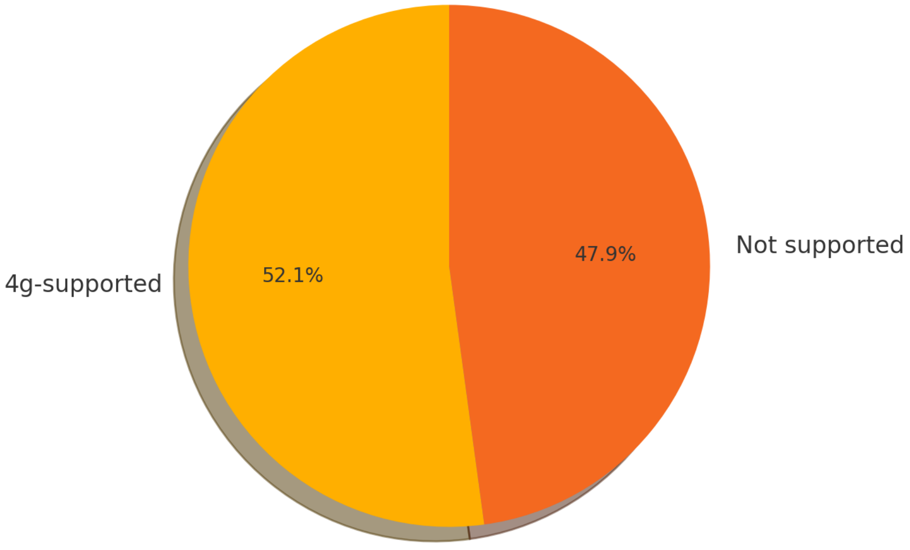
Boxplot



Pie Charts


wifi-supported
50.7%
49.3%
Not supported


0
3
25.0%
25.0%
25.0%
25.0%
1
2


4g-supported
52.1%
47.9%
Not supported

Feature Engineering and Selection

- Feature engineering involves creating new features or modifying existing ones to improve the model's performance.
- Feature selection is the process of choosing the most relevant features for training the model.

```
data=df.copy()
data.drop(['n_cores', 'm_dep','four_g','three_g','blue','clock_speed','sc_w','sc_h'], axis=1, inplace=True)
X = data.drop('price_range', axis=1)
Y = data['price_range']
```

Data Preprocessing

- Data preprocessing steps include handling missing values, encoding categorical variables, and scaling numerical features.

Model Training

- Random Forest classifier is chosen for training due to its robustness and accuracy.
- The model is trained and evaluated using various metrics.

```
Accuracy of RF: 0.9065217391304348
Precision      14
Accuracy       14
Train_Score    12
Test_Score     14
dtype: int64
```

Hyperparameter Tuning

Hyperparameter tuning is a crucial step in optimizing the performance of a machine learning model. It involves adjusting the parameters that govern the training process to find the optimal set of values that yield the best model performance.

Importance of Hyperparameter Tuning:

- **Enhances Model Accuracy**: Proper tuning can significantly improve the model's accuracy and predictive power.
- **Prevents Overfitting**: By finding the right balance, hyperparameter tuning helps in preventing the model from overfitting to the training data.
- **Optimizes Performance**: It ensures the model is both efficient and effective in its predictions.

Method Used: Custom Tuning with Random Forest

We implemented a custom hyperparameter tuning method for the Random Forest classifier. The code iteratively tests different values for the max_depth parameter, which controls the maximum depth of each tree in the forest.

Process:

1. **Parameter Initialization**: We initialized the Random Forest classifier with a fixed random state to ensure reproducibility.
2. **Train-Test Split**: The data was split into training and testing sets with a test size of 23%.
3. **Iterative Training and Evaluation**: For each value of max_depth from 1 to 30,the model was trained and evaluated using precision, accuracy, training score, and testing score.
4. **Result Collection**: The performance metrics for each max_depth value were collected and analyzed.

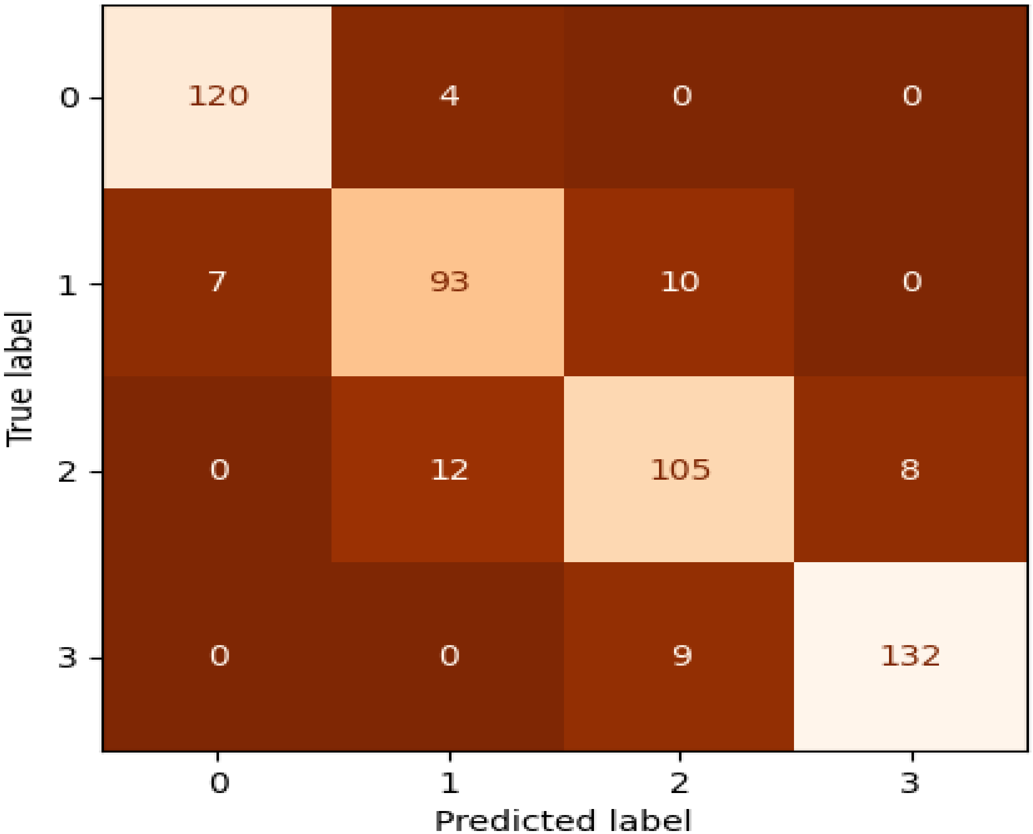
Model Evaluation Metrics

- The accuracy score, confusion matrix, and other relevant metrics are used to evaluate the model's performance.
- Metrics such as precision, recall, and F1 score are considered.

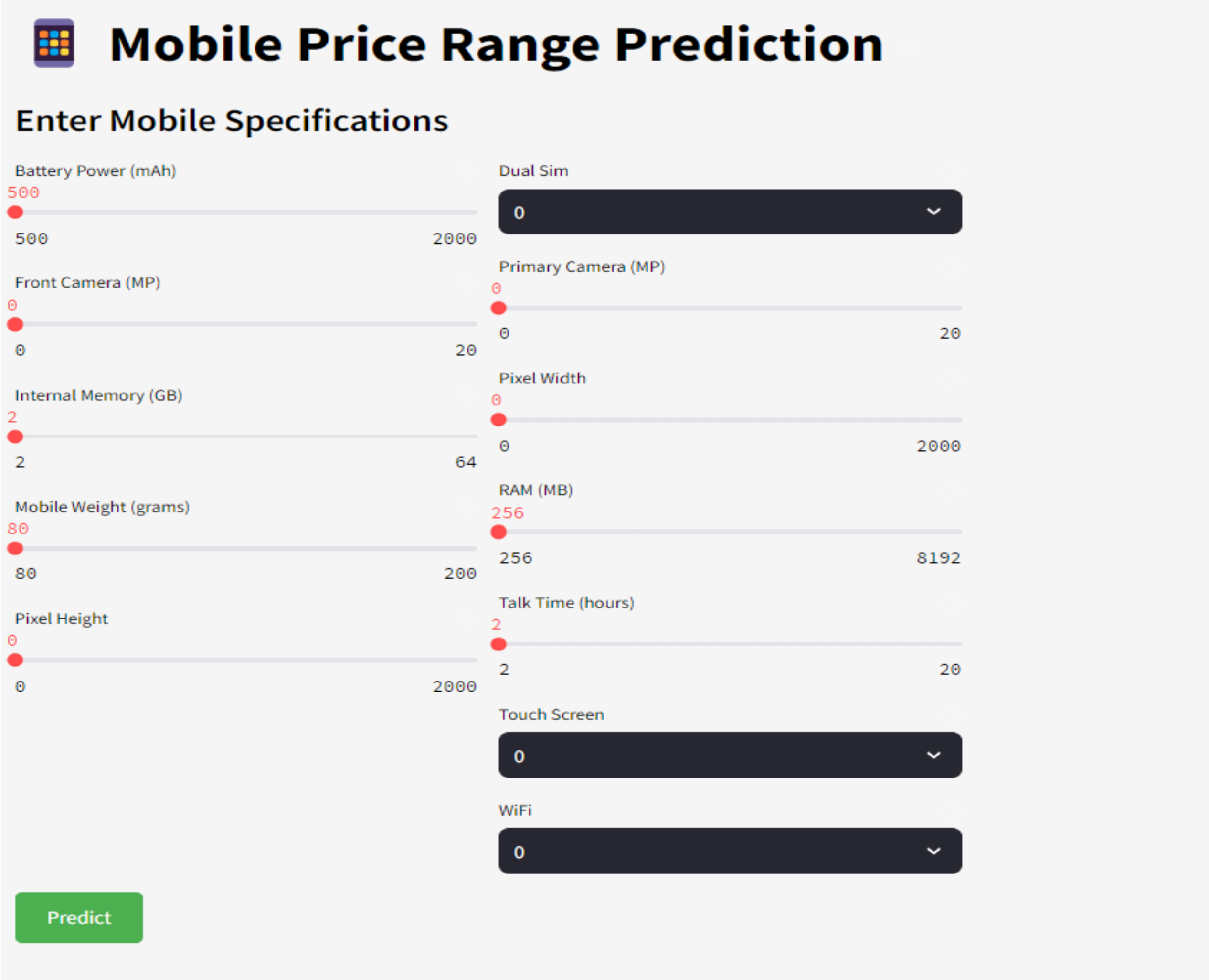
```
Mean Cross-Validation Accuracy: 88.80000000000001
Cross-Validation Score Standard Deviation: 0.005811865258054214 5
Mean Cross-Validation Precision: 88.93276123278646
Cross-Validation Precision Standard Deviation: 0.005565916673537844
Mean Cross-Validation Recall: 88.80000000000001
Cross-Validation Recall Standard Deviation: 0.0058118652580542145
Mean Cross-Validation F1 Score: 88.80555545675986
Cross-Validation F1 Score Standard Deviation: 0.0055985549258987 4
```

Confusion Matrix

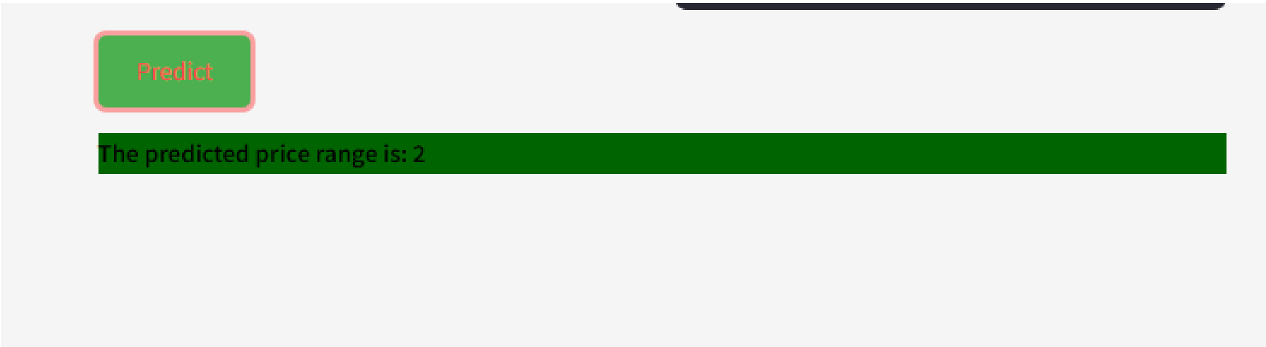
- The confusion matrix provides insights into the model's performance in predicting each class.



UI


Mobile Price Range Prediction
Enter Mobile Specifications
Battery Power (mAh)
500
500
2000
Front Camera (MP)
0
0
20
Internal Memory (GB)
2
2
64
Mobile Weight (grams)
80
80
200
Pixel Height
0
0
2000
Dual Sim
0
Primary Camera (MP)
0
0
20
Pixel Width
0
0
2000
RAM (MB)
256
256
8192
Talk Time (hours)
2
2
20
Touch Screen
0
WiFi
0
Predict

Result


Predict
The predicted price range is: 2

Conclusion

This study aimed to develop a robust model to predict mobile price ranges based on various features. Through comprehensive data exploration, feature engineering, and rigorous model training and evaluation, we achieved significant insights and results.

Key Findings:

- **Data Exploration**: We identified key features such as RAM, battery power, and pixel resolution that are highly correlated with the price range.
- **Visualization**: Boxplots and pie charts provided a clear understanding of feature distributions and their impact on the target variable.
- **Model Training**: A Random Forest classifier was used, achieving high accuracy and robust performance metrics.
- **Evaluation**: The model was evaluated using accuracy, precision, recall, and F1 score, confirming its reliability.

Future Work:

- **Feature Enhancement**: Including more advanced features such as user reviews and market trends could improve the model.
- **Model Optimization**: Experimenting with other machine learning algorithms and deep learning techniques could enhance prediction accuracy.
- **Deployment**: Developing a user-friendly application for real-time price prediction could provide practical benefits for consumers and manufacturers.