A Classification Based approach for predicting Smartphone Price Categories

Sayan Das - B2430035 **Raihan Uddin** - B2430070

November 25, 2024

Outline

Introduction

Background

- The smartphone industry experiences continuous technological innovations, with manufacturers introducing advanced features.
- Multiple global players, such as Apple, Samsung, and Xiaomi, vie for market share, leading to frequent product launches and **pricing battles**.
- Consumers demand **value for money**, with preferences shifting toward devices offering high performance at competitive prices.

Motivation

We hope our model will help:

- Simplify pricing strategies for **manufacturers**
- Increase pricing transparency for **consumers**

Motivation

We hope our model will help:

- Simplify pricing strategies for **manufacturers**
- Increase pricing transparency for **consumers**

Objectives

- Develop a Robust Classification Model
- Determine the most influential factors behind smartphone pricing
- Compare algorithm performances
- Practical Applicability
- Methodological Contribution

Objectives

- Develop a Robust Classification Model
- Determine the most influential factors behind smartphone pricing
- Compare algorithm performances
- Practical Applicability
- Methodological Contribution

Objectives

- Develop a Robust Classification Model
- Determine the most influential factors behind smartphone pricing
- Compare algorithm performances
- Practical Applicability
- Methodological Contribution

Objectives

- Develop a Robust Classification Model
- Determine the most influential factors behind smartphone pricing
- Compare algorithm performances
- Practical Applicability
- Methodological Contribution

Objectives

- Develop a Robust Classification Model
- Determine the most influential factors behind smartphone pricing
- Compare algorithm performances
- Practical Applicability
- Methodological Contribution

Dataset Description

Source

kaggle

Link - https://www.kaggle.com/datasets/iabhishekofficial/smartphone-price-classification

The dataset is publicly available and contains **2000 smartphone entries** with **19 feature variables** and **1 target variable** representing `price_range`.

Features

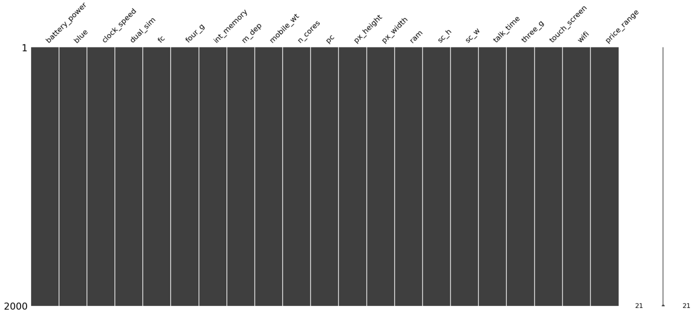
Feature Name	Description	Type
battery_power	battery capacity in mAh	Numerical
clock_speed	speed at which processor executes instructions	Numerical
fc	front Camera Megapixels	Numerical
pc	primary Camera Megapixels	Numerical
int_memory	internal Memory capacity	Numerical
m_dep	smartphone Depth in cm	Numerical
mobile_wt	weight of the smartphone	Numerical
n_cores	number of cores in processor	Numerical
px_height	pixel Resolution Height	Numerical
px_width	pixel Resolution Width	Numerical
ram	RAM in MB	Numerical
sc_h	screen Height in cm	Numerical
sc_w	screen Width in cm	Numerical
talk_time	longest time that a single battery charge will last over a call	Numerical
blue	has bluetooth or not	Categorical
dual_sim	has dual sim support or not	Categorical
four_g	has 4G or not	Categorical
three_g	has 3G or not	Categorical
wifi	has wifi or not	Categorical
touch_screen	has touch screen or not	Categorical

Target Variable

- The target variable `price_range` is **categorical** with **4 classes**.
  - 0 - Low Cost - **Budget** Smartphones
  - 1 - Medium Cost - **Mid-Range** Smartphones
  - 2 - High Cost - **High-End** Smartphones
  - 3 - Very High Cost - **Flagship** Smartphones

Data Preprocessing

Data Cleaning - Handling Missing Values I



Significance: Machine learning models often require complete data to function correctly. Missing values can lead to errors.

Observation: There are no missing values in the dataset.

Data Cleaning - Handling Duplicate Values

```
df.duplicated().sum()
```

```
np.int64(0)
```

Significance: Duplicate entries can distort the true representation of the data, leading to **bias**.

Observation: There are no duplicate values in the dataset.

Data Cleaning - Handling Invalid Values I

```
negative_counts = df.apply(lambda x: (x < 0).sum())
print(negative_counts)
```

```
battery_power    0
blue             0
clock_speed      0
dual_sim         0
fc               0
four_g           0
int_memory       0
m_dep            0
mobile_wt        0
n_cores          0
pc               0
px_height        0
px_width         0
ram              0
sc_h             0
sc_w             0
talk_time        0
three_g          0
touch_screen     0
wifi             0
price_range      0
dtype: int64
```

Significance: None of the features can have negative values.

Observation: There are no negative values in the dataset.

Data Cleaning - Handling Invalid Values II

```python
zero_counts = df.apply(lambda x: (x == 0).sum())
print(zero_counts)
```

```
battery_power       0
blue             1010
clock_speed         0
dual_sim          981
fc                474
four_g            957
int_memory          0
m_dep               0
mobile_wt           0
n_cores             0
pc                101
px_height           2
px_width            0
ram                 0
sc_h                0
sc_w              180
talk_time           0
three_g           477
touch_screen      994
wifi              986
price_range       500
dtype: int64
```

Significance: Most of the numerical features can not be zero except `fc` and `pc`. These two being zero means the phone does not have a front or primary camera.

Observation: `px_height` and `sc_w` are have 2 and 180 zero values respectively.

Action: We replaced these zero values with the mean of the respective features.

Data Cleaning - Handling Invalid Values III

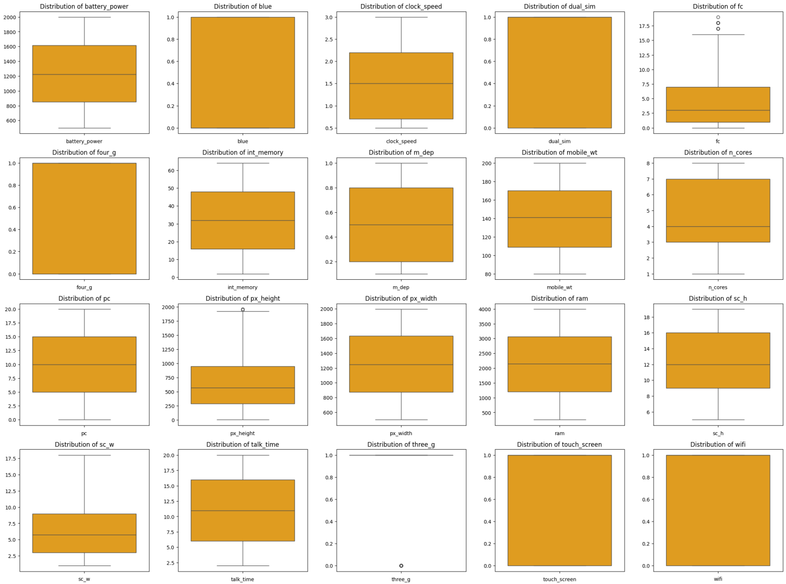
```
to_replace_with_mean = ['sc_w', 'px_height']

for feature in to_replace_with_mean:
  df[feature] = df[feature].replace(0, df[feature].mean())
```

Data Cleaning - Outlier Handling I

Significance:

- Outliers can distort the true representation of data.
- Machine learning models can be sensitive to outliers.

Data Cleaning - Outlier Handling II



Data Cleaning - Outlier Handling (cont.)

Observation: The box plots revealed outliers in two features, `fc` and `px_height`.

Action: To identify these outliers, we will use the IQR (Interquartile Range) method and remove extreme values.

$$\text{IQR} = Q3 - Q1$$

Once the IQR is calculated, outliers are identified using the following bounds:

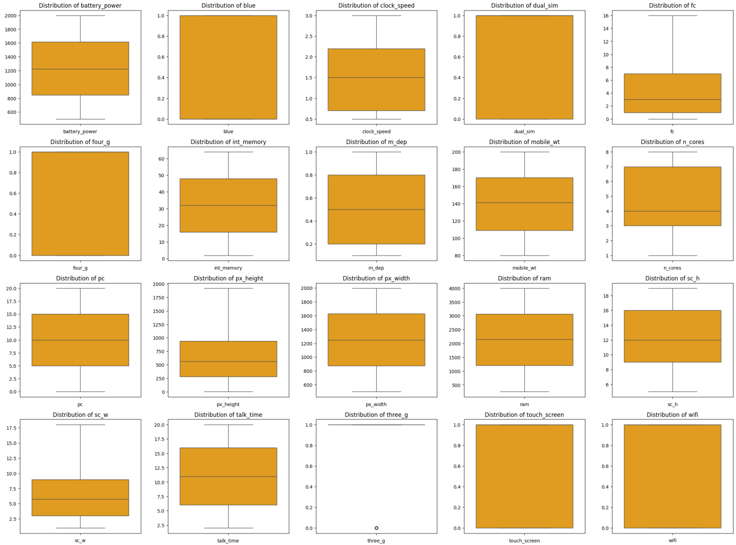
$$\text{Lower Bound} = Q1 - 1.5 \times \text{IQR} \qquad \text{Upper Bound} = Q3 + 1.5 \times \text{IQR}$$

Any data point that falls below the lower bound or above the upper bound is considered an outlier.

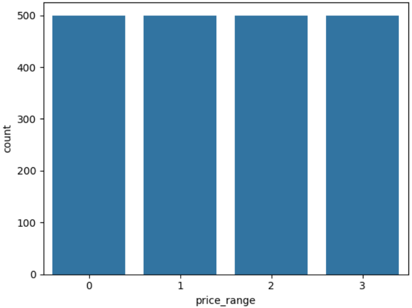
Data Cleaning - Outlier Handling (cont.)

```python
def remove_outliers_iqr(data, column):
  Q1 = data[column].quantile(0.25)
  Q3 = data[column].quantile(0.75)
  IQR = Q3 - Q1

  lower_bound = Q1 - 1.5 * IQR
  upper_bound = Q3 + 1.5 * IQR

  filtered_data = data[(data[column] >= lower_bound) &
                       (data[column] <= upper_bound)]
  return filtered_data

df = remove_outliers_iqr(df, 'fc')
df = remove_outliers_iqr(df, 'px_height')
```

Data Cleaning - Outlier Handling (cont.)



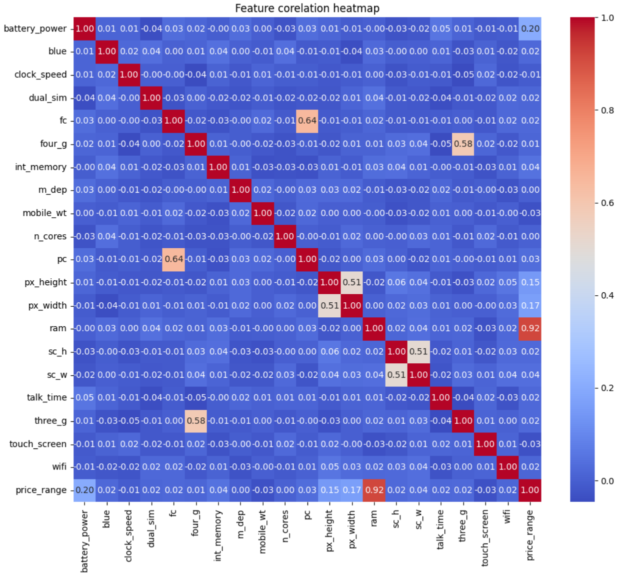
Data Cleaning - Checking for Class Imbalance



Significance: Imbalanced dataset makes the model biased towards the majority class.

Observation: There are no class imbalance in the dataset.

Data Cleaning - Correlation Analysis I



Significance:
Correlation analysis helps identify relationships between features. It can help in feature engineering.

Data Cleaning - Correlation Analysis II

Observation:

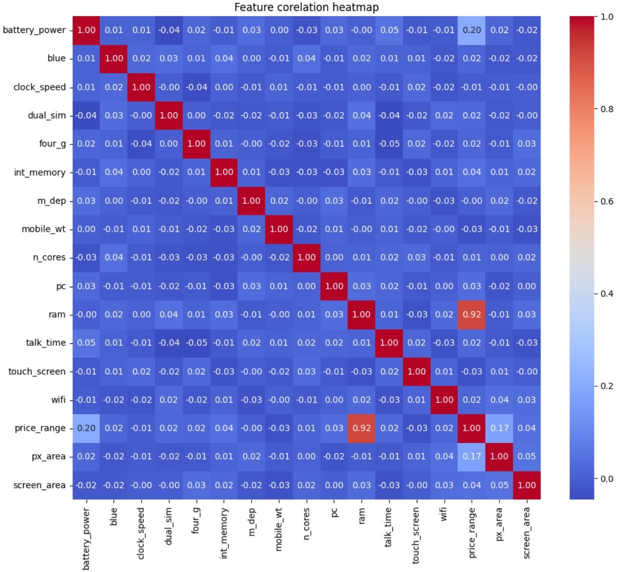
1. **ram and price_range :** Higher RAM capacity leads to higher price range.
2. **three_g and four_g :** A high correlation here suggests that devices with 4G almost always support 3G, making one of these features redundant.
3. **fc and pc :** These features are correlated, as better primary cameras often accompany better front cameras.
4. **px_height and px_width :** These are components of screen resolution and are naturally correlated.
5. **sc_h and sc_w :** These are also naturally correlated.

Data Cleaning - Correlation Analysis III

Action:

1. **three_g and four_g :** `three_g` was removed from the dataset.
2. **fc and pc :** `fc` was removed from the dataset.
3. **px_height and px_width :** They were combined to form a new feature `px_area = px_height * px_width`.
4. **sc_h and sc_w :** They were combined to form a new feature `screen_area = sc_h * sc_w`.

Data Cleaning - Correlation Analysis IV



Correlation Matrix after feature engineering

Feature Selection I

Significance: By selecting the most relevant features, the model can focus on the **most important** information. Moreover, including irrelevant or redundant features can cause the model to **overfit** the training data. Also, fewer features mean **less data to process**.

Action: In our project, we used the ANOVA F-test (Analysis of Variance) method to evaluate each feature's relationship with the target variable, price_range, and select features that are statistically significant. The threshold for selection is a p-value of less than 0.1 i.e. a 90% confidence level.

Feature Selection II

```
y = df.pop('price_range')
X = df
feature_selector = SelectKBest(f_classif, k='all')
X_selected = feature_selector.fit_transform(X, y)
p_values = feature_selector.pvalues_
f_scores = feature_selector.scores_

selected_features = X.columns[p_values < 0.1]
```

Outcome: The selected features are:
'battery_power', 'int_memory', 'mobile_wt', 'n_cores', 'ram', 'px_area', 'screen_area'

Normalizing Features I

Significance: Normalizing features ensures that all features contribute equally to the model training process and prevents any feature from dominating the others.

Action: We used the `PowerTransformer` from `sklearn.preprocessing` to normalize the features `px_area` and `screen_area`.

Outcome: The features `px_area` and `screen_area` were successfully normalized, resulting in a more balanced dataset for model training.

Train-Test Split I

Significance: Splitting the dataset into training and testing sets allows us to evaluate model performance and ensure that the model generalizes well to unseen data.

Action: We used `train_test_split` from `sklearn.model_selection` to split the dataset into 80% training and 20% testing sets.

Outcome: The train-test split resulted in 1584 samples with 7 features for training and 396 samples with 7 features for testing, with corresponding target arrays of 1584 and 396 elements, respectively.

Scaling I

Significance: Scaling ensures that all feature values are normalized, which improves the performance and convergence of many machine learning algorithms.

Action: We used the `StandardScaler` from `sklearn.preprocessing` to scale the feature values.

Outcome: The dataset was successfully scaled, resulting in normalized feature values that contribute equally to the model training process.

Acknowledgement

References

Thank You!