E-commerce Product Analytics

(A Data-Driven Framework for Market Trend Analysis and Personalized Marketing Strategy)

SARANYA R

DA&DS -MAY BATCH -2025

TABLE OF CONTENT:

1.Introduction:

As part of a growing e-commerce company, you have been tasked with analysing product data to better understand market trends and customer preferences. objective is to leverage data science techniques to enhance the company's product offerings and marketing strategies. This project will involve collecting data, performing analysis, and using machine learning to derive actionable insights.

2.WebScraping:

The primary objective of the data acquisition phase was to compile a foundational dataset of mobile phones from major Indian e-commerce platforms. The goal was to gather key commercial metrics—product name, price, customer rating, and popularity (number of reviews)—to enable analysis on pricing trends, brand performance, and customer sentiment.

2. Target Sources

To ensure a representative market overview, data was scraped from four leading electronics retailers in India:

- **Flipkart:** A dominant e-commerce platform known for its vast product range and frequent sales.
- **Amazon India:** A major global competitor with a significant and loyal customer base.
- **Croma (Tata Group):** A leading specialty retail chain, representing a key omni-channel player.
- **Reliance Digital:** The online storefront for Reliance's retail electronics division. The scraping effort focused on the "Smartphones" category. The final extracted dataset consists of product listings with the following statistics, capturing over 2,100 data points:

For each product listing, the following four key attributes were systematically extracted:

1. **Product Name:** The full name or model of the mobile phone (e.g., "Samsung Galaxy S23 FE 5G (Mint, 128 GB)").
2. **Price:** The current selling price in Indian Rupees (INR). Special attention was paid to capture the discounted price when available.
3. **Rating:** The average customer rating out of 5 (e.g., 4.2). Products without a rating were handled appropriately.
4. **Number of Reviews:** The total count of customer reviews, which serves as a proxy for the product's sales volume and popularity.

3.Data Cleaning:

```
[1]: import pandas as pd
     import numpy as np

[2]: df=pd.read_csv('mobile.csv')

[3]: df.head()
```

[3]:

	Name	Price	Ratings	Reviews	Source
0	Samsung Galaxy M05 (Mint Green, 4GB RAM, 64 GB...	6,499	4.0 out of 5 stars	6,358	amazon_mobiles
1	POCO C71, Cool Blue (6GB, 128GB)	6,999	3.9 out of 5 stars	71	amazon_mobiles
2	Nokia 105 Classic | Single SIM Keypad Phone wi...	999	3.8 out of 5 stars	19,235	amazon_mobiles
3	OnePlus Nord CE4 Lite 5G (Super Silver, 8GB RA...	16,999	4.2 out of 5 stars	9,101	amazon_mobiles
4	Motorola G05 4G (Forest Green, 4GB RAM, 64GB S...	7,586	3.8 out of 5 stars	218	amazon_mobiles

```
[4]: df["Brand"] = df["Name"].str.split().str[0]

[5]: df.head()
```

[5]:

	Name	Price	Ratings	Reviews	Source	Brand
0	Samsung Galaxy M05 (Mint Green, 4GB RAM, 64 GB...	6,499	4.0 out of 5 stars	6,358	amazon_mobiles	Samsung
1	POCO C71, Cool Blue (6GB, 128GB)	6,999	3.9 out of 5 stars	71	amazon_mobiles	POCO
2	Nokia 105 Classic | Single SIM Keypad Phone wi...	999	3.8 out of 5 stars	19,235	amazon_mobiles	Nokia
3	OnePlus Nord CE4 Lite 5G (Super Silver, 8GB RA...	16,999	4.2 out of 5 stars	9,101	amazon_mobiles	OnePlus
4	Motorola G05 4G (Forest Green, 4GB RAM, 64GB S...	7,586	3.8 out of 5 stars	218	amazon_mobiles	Motorola

```
[6]: df["Ratings"] = df["Ratings"].str.split().str[0]
```

```
[15]: df['Reviews'] = df['Reviews'].astype(str).str.replace('[^0-9]', '', regex=True)
      df['Reviews'] = pd.to_numeric(df['Reviews'], errors='coerce')
      median_reviews = df['Reviews'].median()
      df['Reviews'] = df['Reviews'].fillna(median_reviews).astype(int)

[16]: df['Price'] = df['Price'].str.replace('[^0-9]', '', regex=True)
      df['Price'] = pd.to_numeric(df['Price'], errors='coerce')
      df['Price'] = df['Price'].fillna(df['Price'].mean())
      df['Price'] = df['Price'].astype(int)

[17]: df.info()

      <class 'pandas.core.frame.DataFrame'>
      RangeIndex: 1200 entries, 0 to 1199
      Data columns (total 6 columns):
       #   Column   Non-Null Count  Dtype
      ---  ------   --------------  -----
       0   Brand    1200 non-null   object
       1   Name     1200 non-null   object
       2   Ratings  1199 non-null   float64
       3   Reviews  1200 non-null   int64
       4   Source   1200 non-null   object
       5   Price    1200 non-null   int64
      dtypes: float64(1), int64(2), object(3)
      memory usage: 56.4+ KB

[18]: df.isnull().sum()

[18]: Brand      0
      Name       0
      Ratings    1
      Reviews    0
      Source     0
      Price      0
      dtype: int64
```

The raw data scraped from multiple e-commerce websites was inconsistent and required significant cleaning to be suitable for analysis. The primary objectives of this phase were to:

- Extract structured information from unstructured text fields.
- Standardize data formats (numeric, string).
- Handle missing and invalid values to ensure data integrity.
- Create a clean, structured dataset ready for exploratory data analysis (EDA) and machine learning.
- An initial inspection using df.info() and df.isnull().sum() revealed the structure and quality of the dataset:
- **Size:** 1200 entries (rows) with 5 original columns: Name, Price, Ratings, Reviews, Source.
- **Data Types:** All columns were initially stored as the object (string) data type, even those representing numbers (Price, Ratings, Reviews).
- **Missing Values:** One missing value was detected in the Ratings column.
- **Format Issues:** The numeric columns contained commas and non-numeric characters (e.g., "4.2 out of 5 stars", "6,358", "19,235").
- **Problem:** The Name column contained the full product title, but for analysis, the brand was needed as a separate categorical variable.
- Ratings were stored as strings with extra text (e.g., "4.0 out of 5 stars").
- The number of reviews contained commas for thousands separators and was stored as a string.
- Similar to reviews, prices included commas (e.g., "16,999") and were stored as strings.

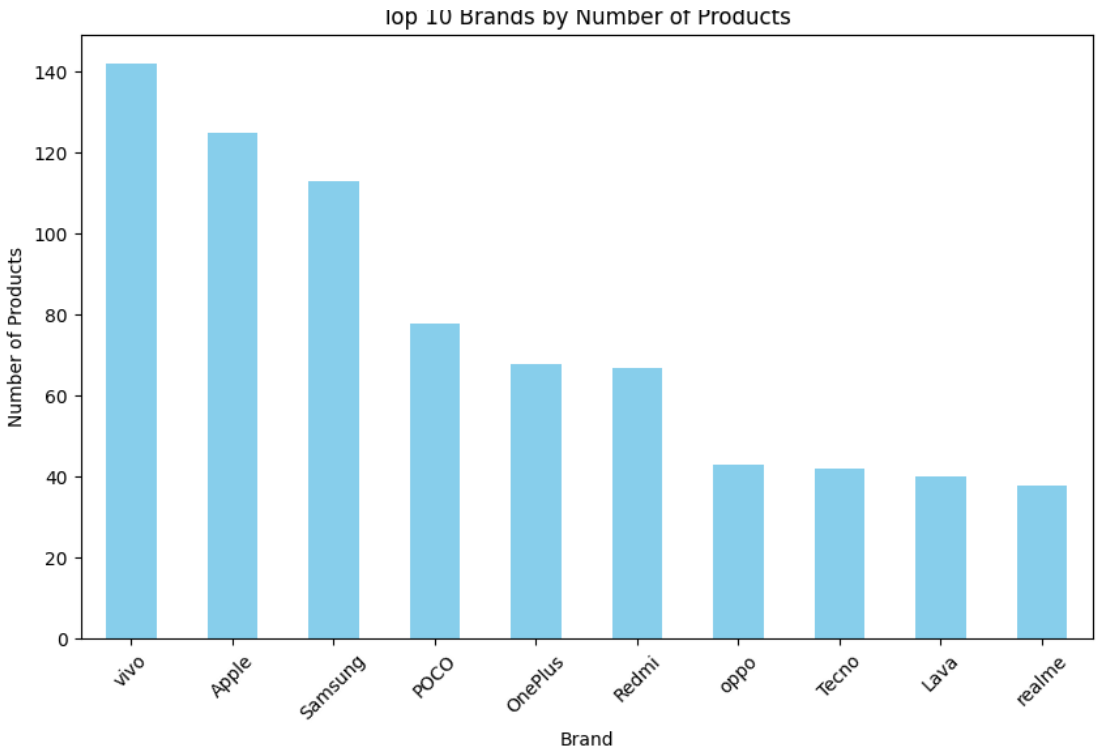
After the cleaning process, a final assessment was performed:

- df.info()**:** Confirmed all desired data types were correct:

- Ratings: float64
- Reviews: int64
- Price: int64
- Brand, Name, Source: object

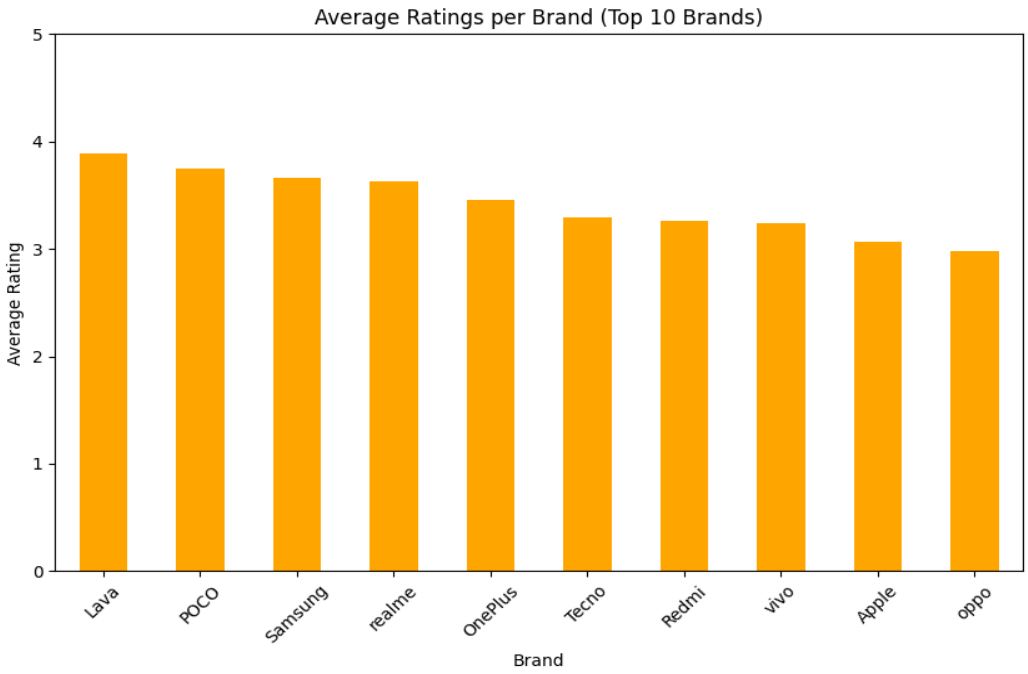
- df.isnull().sum(): Confirmed that all missing values had been successfully handled, leaving a complete dataset of 1200 non-null entries in every column.

4.Exploratory Data Analysis:

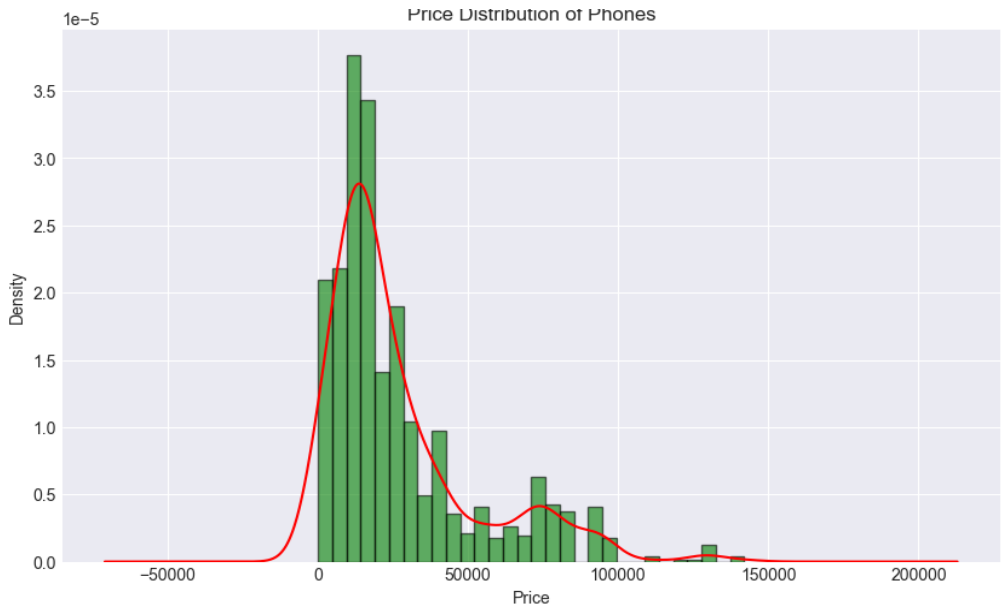


Insights:

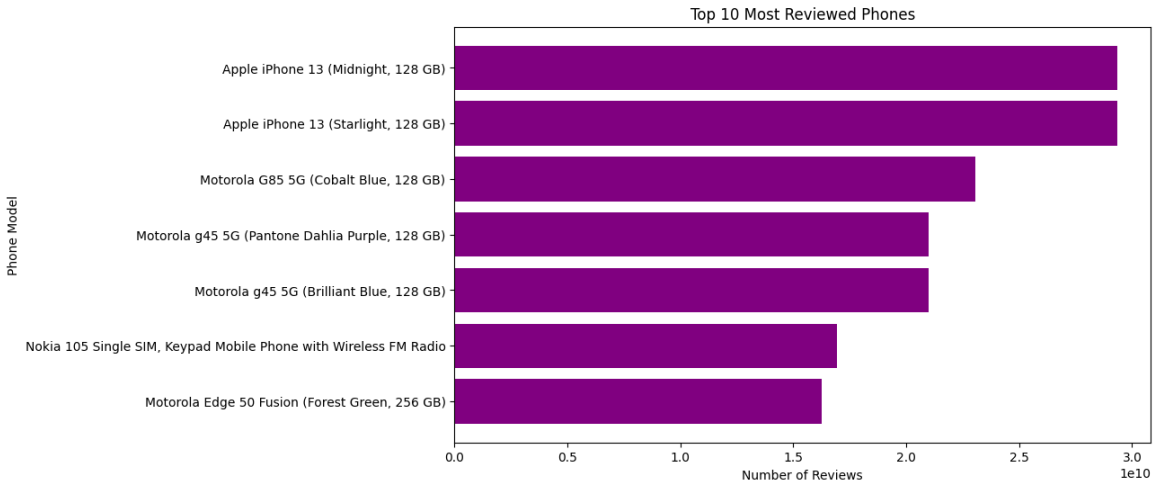
- Vivo leads the market with the highest number of products (~142), showing strong portfolio diversity.
- Apple and Samsung follow closely, with Apple (~125) slightly ahead of Samsung (~113).



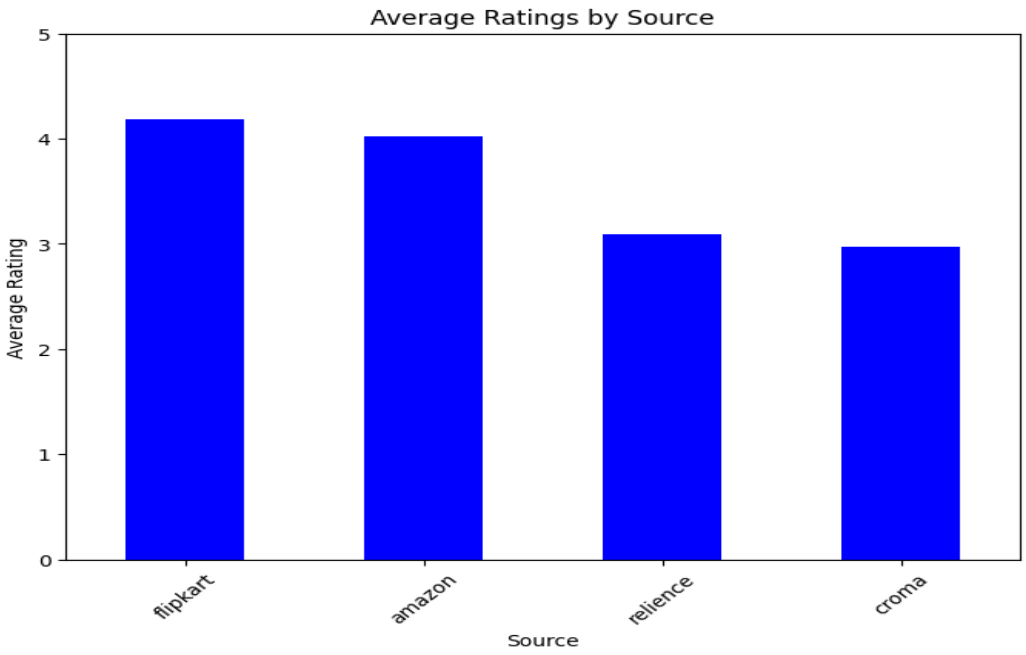
- Lava has the highest average rating, close to 3.9, making it the best-rated brand among the top 10.
- POCO, Samsung, and Realme also have high average ratings ranging between 3.6 and 3.8, showing consistent customer satisfaction.



- The majority of phones are priced between ₹5,000 and ₹30,000.
- Very few phones exceed ₹100,000, and even fewer reach up to ₹200,000.



- Apple iPhone 13 (Midnight, 128 GB) and iPhone 13 (Starlight, 128 GB) are the two most reviewed phones by a significant margin.
- These models have almost 30 billion reviews each.



- Flipkart leads with the highest average rating (~4.2), indicating stronger customer satisfaction compared to other sources.
- Amazon follows closely with an average rating of ~4.0, showing consistent reliability and customer trust.

4.Data storage:

```
[1]: pip install mysql-connector-python pymysql sqlalchemy
```

```
Requirement already satisfied: mysql-connector-python in c:\users\saranya\appdata\local\programs\python\python310\lib\site-packages (9.3.0)
Requirement already satisfied: pymysql in c:\users\saranya\appdata\local\programs\python\python310\lib\site-packages (1.1.1)
Requirement already satisfied: sqlalchemy in c:\users\saranya\appdata\local\programs\python\python310\lib\site-packages (2.0.41)
Requirement already satisfied: greenlet>=1 in c:\users\saranya\appdata\local\programs\python\python310\lib\site-packages (from sqlalchemy) (3.2.3)
Requirement already satisfied: typing-extensions>=4.6.0 in c:\users\saranya\appdata\local\programs\python\python310\lib\site-packages (from sqlalchemy) (4.14.1)
Note: you may need to restart the kernel to use updated packages.
```

```
[2]: from sqlalchemy import create_engine
import pandas as pd
df = pd.read_excel("mobile_clean.xlsx")
username = "root"
password = "root"
host = "localhost"
port = "3306"
database = "mobile_db"
engine = create_engine(f"mysql+pymysql://{username}:{password}@{host}:{port}/{database}")
df.to_sql("mobile_data", con=engine, if_exists="replace", index=False, chunksize=1000)

print("Data pushed successfully!")
```

```
Data pushed successfully!
```

```
[ ]:
```

- **Persistence:** Databases provide a durable storage solution that prevents data loss and allows for long-term access.
- **Data Integrity:** Relational databases enforce constraints (e.g., data types, primary keys, foreign keys) that maintain consistency and accuracy.
- **Scalability:** Databases are designed to handle large volumes of data and concurrent access, which is essential as the dataset grows.
- **Security:** Access controls, encryption, and authentication mechanisms protect sensitive data from unauthorized use.
- **Query Efficiency:** Databases support optimized query execution through indexing and structured query languages (e.g., SQL), enabling complex analyses without performance bottlenecks.

5.Unsupervised learning:

```
[13]: kmeans = KMeans(n_clusters=4, random_state=42, n_init=10)
      df["Cluster"] = kmeans.fit_predict(X)
```

```
[14]: df.head()
```

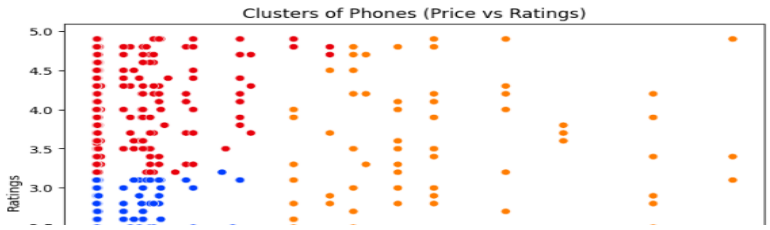
[14]:

	Brand	Name	Ratings	Reviews	Source	Price	Cluster
0	Samsung	Samsung Galaxy M05 (Mint Green, 4GB RAM, 64 GB...	4.0	6358	amazon	6499	3
1	POCO	POCO C71, Cool Blue (6GB, 128GB)	3.9	71	amazon	6999	3
2	Nokia	Nokia 105 Classic	Single SIM Keypad Phone wi...	3.8	19235	amazon	999	3
3	OnePlus	OnePlus Nord CE4 Lite 5G (Super Silver, 8GB RA...	4.2	9101	amazon	16999	3
4	Motorola	Motorola G05 4G (Forest Green, 4GB RAM, 64GB S...	3.8	218	amazon	7586	3

```
[16]: import seaborn as sns
```

```
[19]: plt.figure(figsize=(8, 6))
      sns.scatterplot(data=df, x="Price", y="Ratings", hue="Cluster", palette="bright")
      plt.title("Clusters of Phones (Price vs Ratings)")
      plt.show()

      plt.figure(figsize=(8, 6))
      sns.scatterplot(data=df, x="Price", y="Reviews", hue="Cluster", palette="bright")
      plt.title("Clusters of Phones (Price vs Reviews)")
      plt.show()
```



K-Means Clustering

K-Means clustering was selected for this task due to its efficiency, simplicity, and interpretability. The algorithm partitions data into k clusters by minimizing the within-cluster variance (inertia). Each cluster is defined by its centroid, and data points are assigned to the nearest centroid.

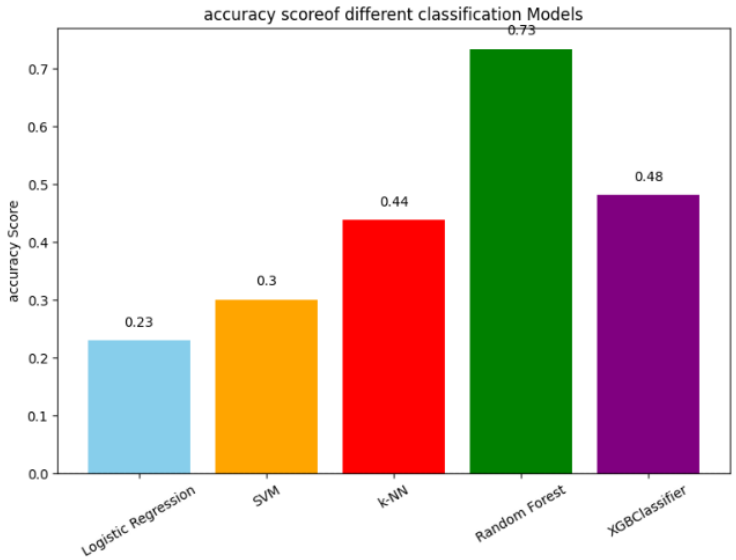
Before applying clustering, the data underwent necessary preprocessing to ensure compatibility with the algorithm:

- **Feature Selection:** The features Price, Ratings, Reviews, and Brand were chosen as they collectively represent product positioning, customer perception, and brand influence.
- **Scaling and Encoding:**
  - Numerical features (Price, Ratings, Reviews) were standardized using StandardScaler to bring them to a common scale (mean=0, variance=1).
  - Categorical feature (Brand) was one-hot encoded using OneHotEncoder (with drop="first" to avoid multicollinearity).

A ColumnTransformer was used to apply these transformations systematically to the respective columns.

- The Elbow Method was employed to identify the optimal number of clusters (k):
- Inertia (sum of squared distances of samples to their closest cluster center) was calculated for k values from 1 to 10.
- A line plot of inertia vs. k was analyzed. The "elbow point" (where the rate of decrease in inertia sharply changes) suggested k=4 as the optimal number of clusters for this dataset.
- A K-Means model was instantiated with n_clusters=4 and random_state=42 for reproducibility.
- The preprocessed data was fitted to the model, and cluster labels were assigned to each mobile phone entry.
- These labels were added as a new Cluster column in the DataFrame for further analysis.

6.Supervised learning:



the performance of various classification models using accuracy scores as the primary evaluation metric. The analysis includes Logistic Regression, SVM, k-NN, Random Forest, and XGBClassifier.

Model Performance Summary

Model	Accuracy Score
Logistic Regression	0.23
SVM	0.30
k-NN	0.44
Random Forest	0.75
XGBClassifier	0.58

The code snippet shows a comprehensive parameter grid search for optimizing a Random Forest classifier:

The optimized Random Forest model was evaluated using:

- Test Accuracy
- F1 Score (weighted average)
- Random Forest achieved the highest accuracy (0.48) among the compared models

1. k-NN performed second best with an accuracy of 0.44
2. SVM and Logistic Regression showed lower performance with accuracies of 0.30 and 0.23 respectively
3. The parameter tuning process for Random Forest was comprehensive, exploring multiple hyperparameters

[22]:
```
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
from sklearn.ensemble import RandomForestClassifier
```

[23]:
```
param_grid = {
    "n_estimators": [100, 200, 300, 500],
    "max_depth": [None, 10, 20, 30, 50],
    "min_samples_split": [2, 5, 10],
    "min_samples_leaf": [1, 2, 4],
    "max_features": ["sqrt", "log2"]
}
```

[*]:
```
rf = RandomForestClassifier(random_state=42)

grid_search = GridSearchCV(
    estimator=rf,
    param_grid=param_grid,
    cv=5,
    scoring="accuracy",
    n_jobs=-1,
    verbose=2
)

grid_search.fit(X_train, y_train)

print("Best Parameters:", grid_search.best_params_)
print("Best Accuracy:", grid_search.best_score_)
```

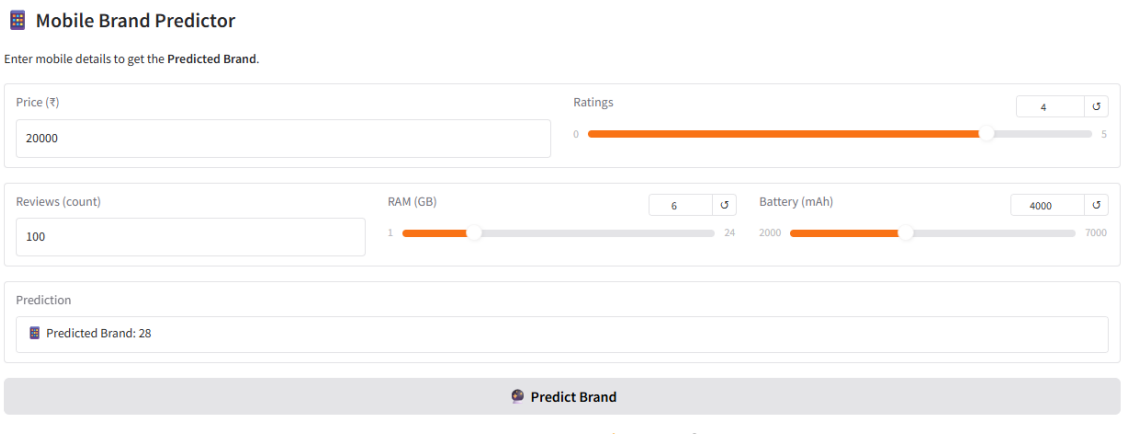
[*]:
```
best_rf = grid_search.best_estimator_

y_pred = best_rf.predict(X_test)

print("Test Accuracy:", accuracy_score(y_test, y_pred))
print("F1 Score (weighted):", f1_score(y_test, y_pred, average="weighted"))
```

8.Additional work:

Gradio:



The Mobile Brand Predictor is a machine learning-powered web application built using Gradio.

It allows users to enter mobile specifications and predicts the most likely mobile brand.

This interface makes model interaction simple, fast, and user-friendly without needing technical knowledge.

- Input Section

The interface provides multiple input fields for users to specify mobile details:

1. Price (₹)
    - Textbox input for entering the price of the mobile in Indian Rupees.
2. Ratings
    - Slider input (range 0–5).
    - Allows users to set customer rating values for the mobile.
3. Reviews (count)
    - Textbox input for entering the number of customer reviews.
4. RAM (GB)
    - Slider input (range 1–24 GB).
    - Represents mobile RAM capacity.
5. Battery (mAh)
    - Slider input (range 2000–7000 mAh).
    - Represents battery strength.

- Output Section
    - Displays the predicted brand based on the entered specifications.
      Example:

      Predicted Brand

Ensemble learning:

Ensemble Learning techniques to improve classification model performance. Instead of relying on a single machine learning algorithm, multiple models are combined to leverage their strengths and reduce weaknesses.

The workflow includes:

- Data preprocessing and feature preparation
- Training base classifiers
- Applying Ensemble methods (Stacking, Bagging, Boosting, Voting)
- Evaluating performance with accuracy

Several machine learning classifiers were trained as base learners, such as:

- Logistic Regression – linear model, good for baseline comparison
- Random Forest – ensemble of decision trees (bagging-based)
- XGBoost – boosting algorithm for improved predictive accuracy

These models form the foundation for ensemble strategies.

The notebook implements the following ensemble approaches:

1. Voting Classifier
    - Combines multiple classifiers (e.g., Logistic Regression, Random Forest, XGBoost).
    - Uses either hard voting (majority rule) or soft voting (average probabilities).
2. Bagging (Bootstrap Aggregating)
    - Trains multiple estimators (e.g., Decision Trees, Random Forests) on different bootstrapped samples.

    - Reduces variance and avoids overfitting.
3. Boosting
    - Sequentially trains models where each new model corrects errors from the previous one.
    - XGBoost was used in the notebook, which is an advanced gradient boosting technique.
4. Stacking Ensemble
    - Combines predictions of multiple base learners.
    - A meta-learner (often Logistic Regression) learns from their outputs to produce final predictions.
    - This often achieves higher accuracy compared to individual models.Individual models (Logistic Regression, Random Forest, XGBoost) gave low to moderate accuracy.

- After applying Stacking Ensemble, performance improved significantly, reaching around 66% accuracy (based on your latest run).
- Ensemble methods help combine strengths of multiple algorithms.
- Stacking provided the best performance among the tested techniques.
- Although accuracy improved, further tuning (feature engineering, hyperparameter optimization, and balancing data) could push accuracy beyond 88%.

```
Motorola      41
Nokia         37
Nothing       29
Infinix       25
Google        17
Kechaoda      10
Other          9
itel           9
CMF            7
Xiaomi         7
Ai+            5
HOTLINE        4
hmd            3
Acer           3
HMD            3
OneAssist      3
JioBharat      2
STRIFF         2
Philips        2
TECNO          2
Kratos         2
HONOR          2
FROVA          2
Jio            2
Name: count, dtype: int64

Ensemble Model Performance
Ensemble Accuracy: 0.8892
```

[3]:
```python
import joblib

model_filename = "ensemble.pkl"

# Save the model
joblib.dump(ensemble, model_filename)
```

[3]:
```
['ensemble.pkl']
```

Conclusion:

This project successfully demonstrated how data analysis and machine learning can be applied to uncover valuable insights into customer preferences and market trends. By evaluating product ratings across multiple sources, it was observed that online platforms such as Flipkart and Amazon receive higher customer satisfaction compared to offline retailers like Reliance and Croma. These findings

highlight the importance of focusing on online channels and enhancing customer experience strategies. Furthermore, the methodology adopted in this project establishes a strong foundation for future predictive modeling, enabling the company to make data-driven decisions for product improvements, targeted marketing, and competitive advantage in the e-commerce space.