**G. H. RAISONI COLLEGE OF ENGINEERING &**

**MANAGEMENT, WAGHOLI, PUNE**

**(An Autonomous Institute Affiliated to SPPU)**

**Department of Information Technology**

# TAE-2

**Subject:** **OBJECT ORIENTED PROGRAMMING Mini Project Report On**

|  |  |  |
| --- | --- | --- |
| “ | **Mobile Price Detection** | ” |

## Academic Year 2023-24 Semester-IV

**G. H. RAISONI COLLEGE OF ENGINEERING & MANAGEMENT, WAGHOLI, PUNE (An Autonomous Institute Affiliated to SPPU)**

**Department of Information Technology**

**Academic Year 2023-24**

**Semester-IV**

**Guided By:-** Prof. Priya Ujawe

**Submitted by:**

1. SY\_IT\_A48 Harshad Hadule
2. SY\_IT\_A50 Harshal Gawali
3. SY\_IT\_A61Mahesh Jadhav



**CERTIFICATE**

This is to certify that **“**Mobile Price Detection**”** embodies the original work done by

### Mr. Harshad Hadule, Mr. Harshal Gawali, Mr. Mahesh Jadhav. This project submission as a partial fulfilment of the requirement for the Mini Project in subject Machine learning And Algorithm of B.Tech. Degree, IV Semester, of Pune University during the academic year 2023-2024.

**Date: 2-4-2024**

**Place:** Pune

**Project Guide Head of Department**

**(Prof. Priya Ujawe ) (Dr. Poonam Gupta)**

## ACKNOWLEDGEMENT

We would like to express our sincere thanks to Prof. Priya Ujawe under whose valuable guidance and light of knowledge, we could complete this project.

We take this opportunity to thank all the staff members of Department Of Information Technology Engineering for their help whenever required. Finally we express sincere thanks to all those who have helped us directly or indirectly in many ways in completion of this project work and I would like to extend my Deep appreciation to all my group members, without their support and Coordination we would not have been able to complete this Project.

## INDEX

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr.No** |  | **TITLE** | **Page No** |
| 1. | Abstract |  | 6 |
| 2. | Objective |  | 6 |
| 3. | Introduction |  | 6 |
| 4. | Source Code |  | 9 |
| 5. | Output Screens |  | 11 |
| 6. | Reference |  | 15 |

### ABSTRACT

Mobile price detection is a task aimed at predicting the price range of a mobile phone based on various features such as battery power, camera specifications, memory, connectivity options, etc. This abstract presents an overview of the objective, methodology, and potential applications of mobile price detection.

### OBJECTIVE

* Develop a machine learning model to predict the price range of mobile phones accurately, aiding consumers in making informed purchasing decisions by considering various features and their corresponding price implications.
* Create a tool for manufacturers to strategically price their mobile phone models by analyzing market trends and competitors' offerings, thus optimizing profitability and market competitiveness.
* Simplify the mobile phone purchasing process for consumers by providing a reliable and efficient price range estimation tool, reducing the time and effort required for product research and comparison.

**INTRODUCTION**

In this implementation of mobile price detection, we utilize machine learning techniques to predict the price range of mobile phones based on various features. The algorithm employed is the Random Forest Classifier, a powerful ensemble learning method known for its robustness and versatility in handling both classification and regression tasks. By leveraging the capabilities of the Random Forest algorithm, we aim to develop a predictive model that accurately categorizes mobile phones into different price ranges, enabling consumers and industry stakeholders to make informed decisions.

### Parameters Used to Detect the Price of Mobile Phones:

1. **battery\_power**: Total energy a battery can store in one time measured in mAh
2. **blue**: Has Bluetooth or not
3. **clock\_speed**: Speed at which the microprocessor executes instructions
4. **dual\_sim**: Has dual SIM support or not
5. **fc**: Front Camera mega pixels
6. **four\_g**: Has 4G or not
7. **int\_memory**: Internal Memory in Gigabytes
8. **m\_dep**: Mobile Depth in cm
9. **mobile\_wt**: Weight of the mobile phone
10. **n\_cores**: Number of cores of the processor
11. **pc**: Primary Camera mega pixels
12. **px\_height**: Pixel Resolution Height
13. **px\_width**: Pixel Resolution Width
14. **ram**: Random Access Memory in Mega Bytes
15. **sc\_h**: Screen Height of mobile in cm
16. **sc\_w**: Screen Width of mobile in cm
17. **talk\_time**: Longest time that a single battery charge will last when you are talking
18. **three\_g**: Has 3G or not
19. **touch\_screen**: Has touch screen or not
20. **wifi**: Has Wi-Fi or not
21. **price\_range** : This is the target variable with value of 0(low cost), 1(medium cost), 2(high cost) and 3(very high cost) in the introduction part give all parameters are used to detect the price of mobile,algorithm used in this then how many dataset and how many parametes in your datasetbattery\_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 are parameters then inducude varous libraries used in this in seperatly in systematic way.

### Dataset Information:

* **Dataset Name**: data\_mobile\_price\_range.csv
* **Number of Datasets**: 1
* **Number of Rows**: 2001
* **Number of Parameters in the Dataset**: 21 (including the target variable)
* **Target Variable**: price\_range (with values 0 for low cost, 1 for medium cost, 2 for high cost, and 3 for very high cost)

### Libraries Used:

* **pandas**: Data manipulation and analysis library
* **scikit-learn (sklearn)**:
  + **train\_test\_split**: Splits the dataset into training and testing sets
  + **RandomForestClassifier**: Implements the Random Forest Classifier algorithm
  + **classification\_report**: Generates a text report showing the main classification metrics
  + **accuracy\_score**: Computes the accuracy classification score
  + **confusion\_matrix**: Computes the confusion matrix to evaluate the accuracy of a classification
* **matplotlib.pyplot**: Plotting library for creating static, animated, and interactive visualizations
* **seaborn**: Data visualization library based on matplotlib for statistical graphics.

### Algorithm Used:

* **Random Forest Classifier**: Ensemble learning method that builds multiple decision trees during training and merges them together to get a more accurate and stable prediction.

In this project, the Random Forest algorithm works as follows:

1. **Training Phase**:
   * Random Forest builds multiple decision trees during the training phase. The number of trees is determined by the **n\_estimators** parameter, which is set to 100 in this project.
   * Each decision tree is trained on a random subset of the training data (selected with replacement), as well as a random subset of features. This process is known as bagging (Bootstrap Aggregating).
   * At each node of the decision tree, a random subset of features is considered for splitting, rather than considering all features. This randomness helps to decorrelate the trees and make the model more robust.
   * The trees are grown until some stopping criteria are met, such as reaching a maximum depth or having nodes with minimum samples.
2. **Prediction Phase**:
   * During the prediction phase, each decision tree in the forest independently predicts the target variable (price range) for unseen instances (test data).
   * For classification tasks like this one, where the target variable has discrete values (price range 0, 1, 2, or 3), the mode (most common) prediction among all the trees is taken as the final prediction. This means the predicted price range with the highest frequency across all trees is selected.
   * For regression tasks, the average prediction of all the trees is considered.
3. **Ensemble Learning**:
   * By combining the predictions of multiple decision trees, Random Forest reduces overfitting and improves generalization performance.
   * Since each tree is trained on a different subset of data and features, the ensemble of trees captures different aspects of the relationship between features and the target variable.
   * This diversity among trees helps Random Forest to make more accurate predictions and be less sensitive to noise in the data.
4. **Hyperparameter Tuning**:
   * Random Forest has several hyperparameters that can be tuned to optimize its performance, such as the number of trees (**n\_estimators**), the maximum depth of trees (**max\_depth**), the minimum number of samples required to split a node (**min\_samples\_split**), and others.
   * Tuning these hyperparameters is essential to prevent overfitting and achieve the best possible performance on unseen data.

Overall, Random Forest is an effective and popular algorithm for both classification and regression tasks, known for its robustness, scalability, and ability to handle high-dimensional datasets. In this project, it is employed to predict the price range of mobile phones based on various features, leveraging the power of ensemble learning and decision trees.

### SOURCE CODE

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification\_report, accuracy\_score, confusion\_matrix

import matplotlib.pyplot as plt

import seaborn as sns

# Load the dataset

data = pd.read\_csv('data\_mobile\_price\_range.csv')

# Print the first few rows of the dataset

print("Full Dataset:")

print(data.head())

# Split the data into features (X) and target (y)

X = data.drop('price\_range', axis=1)

y = data['price\_range']

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Print the training and testing datasets

print("\nTraining Set Features:")

print(X\_train.head())

print("\nTraining Set Labels:")

print(y\_train.head())

print("\nTesting Set Features:")

print(X\_test.head())

print("\nTesting Set Labels:")

print(y\_test.head())

# Predictions on the training set

y\_train\_pred = rf\_classifier.predict(X\_train)

# Predictions on the test set

y\_test\_pred = rf\_classifier.predict(X\_test)

# Plot regression graph for training set

plt.figure(figsize=(8, 6))

sns.regplot(x=y\_train, y=y\_train\_pred, scatter\_kws={'alpha':0.5})

plt.xlabel('Actual Price Range')

plt.ylabel('Predicted Price Range')

plt.title('Regression Plot - Training Set')

plt.grid(True)

plt.show()

# Plot regression graph for test set

plt.figure(figsize=(8, 6))

sns.regplot(x=y\_test, y=y\_test\_pred, scatter\_kws={'alpha':0.5})

plt.xlabel('Actual Price Range')

plt.ylabel('Predicted Price Range')

plt.title('Regression Plot - Test Set')

plt.grid(True)

plt.show()

# Initialize the Random Forest Classifier

rf\_classifier = RandomForestClassifier(n\_estimators=100, random\_state=42)

# Train the model

rf\_classifier.fit(X\_train, y\_train)

# Predictions on the test set

y\_pred = rf\_classifier.predict(X\_test)

# Evaluate the model

accuracy = accuracy\_score(y\_test, y\_pred)

print('\nAccuracy:', accuracy)

# Print classification report

print(classification\_report(y\_test, y\_pred))

# Compute confusion matrix

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

print('Confusion Matrix:')

print(conf\_matrix)

# Function to get user input

def get\_user\_input():

    features = {}

    for column in X.columns:

        value = input(f"Enter {column}: ")

        features[column] = [float(value)]  # assuming all values are numeric

    return pd.DataFrame(features)

# Get user input

user\_input = get\_user\_input()

# Make prediction using the trained model

predicted\_price\_range = rf\_classifier.predict(user\_input)

print("\nPredicted Price Range:", predicted\_price\_range)

**OUTPUT**

Full Dataset:

battery\_power blue clock\_speed dual\_sim fc four\_g int\_memory m\_dep \

0 842 0 2.2 0 1 0 7 0.6

1 1021 1 0.5 1 0 1 53 0.7

2 563 1 0.5 1 2 1 41 0.9

3 615 1 2.5 0 0 0 10 0.8

4 1821 1 1.2 0 13 1 44 0.6

mobile\_wt n\_cores ... px\_height px\_width ram sc\_h sc\_w talk\_time \

0 188 2 ... 20 756 2549 9 7 19

1 136 3 ... 905 1988 2631 17 3 7

2 145 5 ... 1263 1716 2603 11 2 9

3 131 6 ... 1216 1786 2769 16 8 11

4 141 2 ... 1208 1212 1411 8 2 15

three\_g touch\_screen wifi price\_range

0 0 0 1 1

1 1 1 0 2

2 1 1 0 2

3 1 0 0 2

4 1 1 0 1

[5 rows x 21 columns]

Training Set Features:

battery\_power blue clock\_speed dual\_sim fc four\_g int\_memory \

968 1923 0 0.5 1 7 0 46

240 633 1 2.2 0 0 1 49

819 1236 0 0.9 1 2 1 57

692 781 0 1.1 0 2 0 38

420 1456 1 0.5 1 7 0 7

m\_dep mobile\_wt n\_cores pc px\_height px\_width ram sc\_h sc\_w \

968 0.5 191 1 10 767 1759 1489 10 9

240 0.1 139 8 1 529 1009 3560 11 1

819 0.1 188 1 14 517 809 1406 14 12

692 0.4 198 5 7 304 1674 3508 13 8

420 0.4 105 5 12 823 1104 1587 6 5

talk\_time three\_g touch\_screen wifi

968 3 1 1 1

240 16 1 1 1

819 20 1 0 1

692 5 0 0 1

420 20 1 0 1

Training Set Labels:

968 1

240 2

819 0

692 3

420 1

Name: price\_range, dtype: int64

Testing Set Features:

battery\_power blue clock\_speed dual\_sim fc four\_g int\_memory \

1860 1646 0 2.5 0 3 1 25

353 1182 0 0.5 0 7 1 8

1333 1972 0 2.9 0 9 0 14

905 989 1 2.0 0 4 0 17

1289 615 1 0.5 1 7 0 58

m\_dep mobile\_wt n\_cores pc px\_height px\_width ram sc\_h sc\_w \

1860 0.6 200 2 5 211 1608 686 8 6

353 0.5 138 8 16 275 986 2563 19 17

1333 0.4 196 7 18 293 952 1316 8 1

905 0.2 166 3 19 256 1394 3892 18 7

1289 0.5 130 5 8 1021 1958 1906 14 5

talk\_time three\_g touch\_screen wifi

1860 11 1 1 0

353 19 1 0 0

1333 8 1 1 0

905 19 1 1 0

1289 5 1 0 0

Testing Set Labels:

1860 0

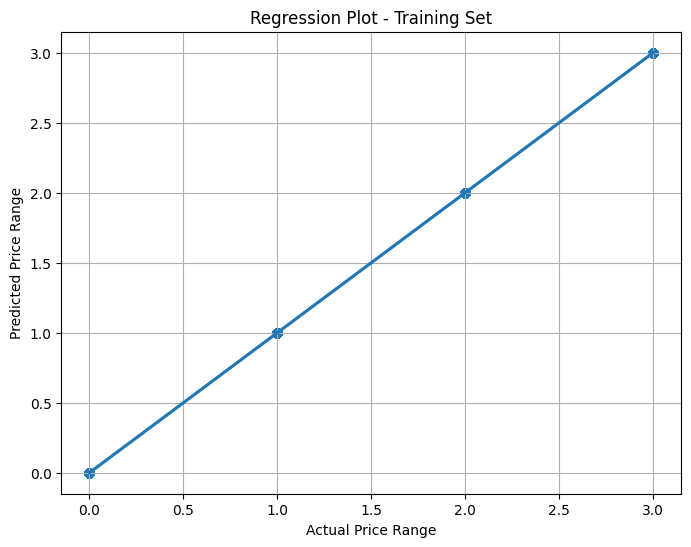
353 2

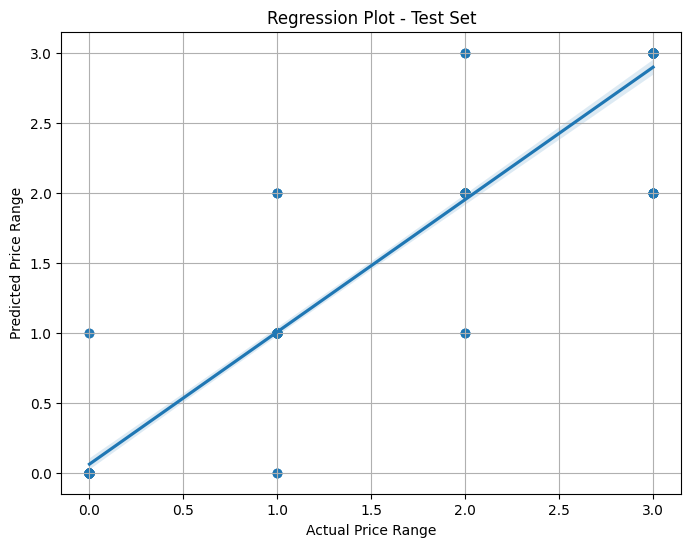
1333 1

905 3

1289 1

Name: price\_range, dtype: int64





Accuracy: 0.8925

precision recall f1-score support

0 0.95 0.96 0.96 105

1 0.89 0.87 0.88 91

2 0.78 0.87 0.82 92

3 0.94 0.87 0.90 112

accuracy 0.89 400

macro avg 0.89 0.89 0.89 400

weighted avg 0.90 0.89 0.89 400

Confusion Matrix:

[[101 4 0 0]

[ 5 79 7 0]

[ 0 6 80 6]

[ 0 0 15 97]]

Enter battery\_power: 1024

Enter blue: 1

Enter clock\_speed: 2.2

Enter dual\_sim: 1

Enter fc: 1

Enter four\_g: 1

Enter int\_memory: 23

Enter m\_dep: 0.6

Enter mobile\_wt: 199

Enter n\_cores: 5

Enter pc: 6

Enter px\_height: 906

Enter px\_width: 1988

Enter ram: 2789

Enter sc\_h: 29

Enter sc\_w: 43

Enter talk\_time: 36

Enter three\_g: 1

Enter touch\_screen: 1

Enter wifi: 1

Predicted Price Range: [2]

**Reference:**Mostly used the concept of Machine learning and Teacher teaching .The data set is used for training and testing from the Kaggle.

**GitHub Link:** <https://github.com/Harshad73735/Mobile-Price-Detection>