Mobile Price Prediction using Machine Learning

Jukanti Varun Sagar, Poshala. Siddharth, B. Nikhil Sai, Dr U. Sesadri
Student, Student, Student, Associate Professor
Vardhaman College Of Engineering

Abstract— **The objective of this project is to develop a categorization model that, in response to a set of requirements, can predict the price range of mobile phones. The dataset for this project includes a range of characteristics, such as battery life, RAM, internal memory, a back camera, a front camera, etc. The research will start by investigating and examining the data to see how each attribute is distributed and how they relate to one another. After that, the data will go through preprocessing to remove any extraneous features, outliers, and missing values. Various classification techniques, such as Logistic Regression, KNN Classification, and Support Vector Machine, will be utilized and compared in order to identify which algorithm performs the best once the data has been divided into training and testing sets. Next, a number of performance metrics, including as F1 score, recall, accuracy, and precision, will be used to evaluate the selected model. New data will be utilized to install and test the model, and then its performance in the actual world will be assessed. The primary goal of this project is to develop an accurate and trustworthy model that can guide clients in selecting mobile phones based on their budget and desired features.**

Keywords— categorization model, price range, mobile phones, dataset, battery life, RAM, internal memory, back camera, front camera, data investigation, data preprocessing, extraneous features, outliers, missing values, classification techniques, Logistic Regression, KNN Classification, Support Vector Machine, training and testing sets, performance metrics, F1 score, recall, accuracy, precision, new data, model installation, model testing, real-world performance, budget, desired features.

I. INTRODUCTION

Mobile devices have revolutionized the way we communicate, access information, and carry out a variety of tasks in our daily lives. With a wide variety of options available to consumers, each varying in terms of brand, features, and price range, the market for mobile devices is incredibly active. Making educated selections can be considerably aided by accurately forecasting a mobile phone's price range based on its specific characteristics. This can benefit both consumers and manufacturers. Devices can be selected by customers based on their needs and budget, and producers can strategically price their goods to stay competitive.

This study aims to create a classification model that correctly predicts the price range of mobile phones by means of particular traits. We intend to develop a model that can classify mobile phones into several price ranges, such as low, medium, high, or extremely high, by utilizing machine learning techniques and examining pertinent properties. Consumers looking for the best return on their investment and manufacturers trying to optimize their pricing strategies will both find great use in this prediction model.

In order to do this, we will use a dataset with 2000 entries and 21 attributes that includes a variety of mobile phone specifications along with related pricing ranges. Battery life, Bluetooth functionality, clock speed, dual SIM compatibility, front camera resolution, 4G connectivity, internal memory, and other characteristics are among them. We want to create a model that can accurately categorize mobile phones based on their specs, hence predicting their price range. To do this, we will investigate the correlations between these attributes and price range labels.

The suggested strategy entails several crucial steps. First, we'll get a good dataset from sources that are openly accessible or directly from mobile stores and manufacturers. Our research will be built upon this dataset, which gives us the data we need to develop and test our categorization algorithm. After that, we'll handle missing values and undertake exploratory data analysis to learn more about the distribution of the dataset's attributes and their relationships with one another.

The dataset will need to be refined, which will heavily rely on feature engineering and selection. We can improve the model's capacity to precisely anticipate the price range of mobile phones by carefully evaluating each feature's applicability and possibly developing additional ones based on domain expertise. These tailored features will identify important nuances and insights in the data, enhancing the classification model's overall performance.

The dataset will need to be refined, which will heavily rely on feature engineering and selection. We can improve the model's capacity to precisely anticipate the price range of mobile phones by carefully evaluating each feature's applicability and possibly developing additional ones based on domain expertise. These tailored features will identify important nuances and insights in the data, enhancing the classification model's overall performance.

In the evaluation stage, each model's performance on the test dataset will be evaluated. We will assess the models' accuracy, precision, recall, and F1-score using a variety of evaluation measures, including confusion matrices and classification reports. We want to find the model that predicts the price range of mobile phones with the most accuracy and precision through this study.

In conclusion, the goal of this research is to create a classification model that can predict with accuracy the price range of mobile phones based on their unique qualities. We seek to offer helpful insights to both customers and producers by utilizing machine learning techniques and examining a substantial dataset. By successfully implementing such a strategy, producers will gain the ability to strategically price their products while also enabling consumers to make informed purchase decisions.

II. LITERATURE REVIEW

Although [**1**] may have a different focus or intent, my suggestion attempts to benefit both customers and producers by predicting the price range of mobile phones based on features. My proposal aims to improve prediction efficiency and accuracy, provide an objective method, support manufacturers in identifying critical traits, aid consumers in making educated selections, and promote machine learning and data science. My proposal also emphasizes possible results including increased revenue and sales, better customer service, time savings, competitive advantage, and data-driven decisions. By emphasizing a more focused and all-encompassing strategy to address the unique demands of consumers and manufacturers in the mobile phone business, these factors set my idea apart from that of the reference paper.

In the reference paper [**2**], the question of whether a mobile device with specific characteristics will fall within a given price range is addressed. With an emphasis on computational simplicity, the study uses particular feature selection techniques to find and remove less important and redundant information. To anticipate mobile costs as accurately as feasible, a variety of classifiers are used. Future study to increase the solution and estimation accuracy is suggested after discussing the outcomes in terms of accuracy and feature selection. On the other hand, the history of the conversation makes no specific reference of my suggestion.

The focus of the cited research [3] is on the application of machine learning methods to estimate the price range of mobile phones based on their features. The procedure for gathering data, dimensionality reduction using feature selection techniques, and the division of mobile phones into pricing groups are all covered in the study. The J48 decision tree model has the highest accuracy in predicting mobile phone pricing, according to a comparison of the accuracy of other machine learning models.

However, my suggestion is not specifically specified, therefore I lack the necessary information to compare it to the reference study. I can, however, assist you in understanding the similarities and differences between your idea and the reference work if you can provide me a summary or description of your proposal.

The precise objectives, results, and strategy of [4] and my idea differ from one another most significantly. My plan focuses on creating a classification model to forecast the price range of mobile phones based on their specifications, even though the goals of the reference paper are not stated. This strategy intends to support firms in product development and price strategies while assisting customers in making informed judgements. Additionally, my proposal aims to increase prediction efficiency and accuracy, offer a standardized method, and promote data science and machine learning. My proposal will lead to more revenue and sales as well as better customer service, time savings, a competitive advantage, and data-driven decisions. We intend to employ a variety of classification techniques, including Logistic Regression, KNN Classification, and SVM Classifier using linear and RBF kernels, using a dataset of 21 features. The data will be divided into training and test sets, the performance of the model will be assessed using confusion matrices and classification reports, and the model with the highest accuracy will be chosen. Overall, my suggestion presents a thorough strategy for creating a price range prediction model, however the reference paper's specifics are still left out unknown.

The focus of the reference paper [5] is on determining whether mobile phones will be affordable or expensive based on their features. The study makes use of real data that was gathered from a website and uses a variety of feature selection algorithms and classifiers to forecast the price range for mobile devices with great accuracy. In the study, the accuracy and selection of features are compared along with the findings. The suggested system seeks to identify an ideal product with the least amount of money spent and the greatest number of features, which can be used in a variety of marketing and commercial settings. However, your idea, as it is expressed in your code, focuses on a particular issue or application that you have created. Without further information about your code or concept, it is challenging to make a precise comparison. The exact issue or application that your code solves, the approach or algorithms applied, the dataset utilized, and the outcomes obtained, nevertheless, generally speaking, represent the significant differences. If you could give more details about your code and the specific modifications you have made in comparison to the reference article, that would be helpful.

III. RESULT ANALYSIS

Regression models for predicting mobile phone prices are implemented numerous times in the given code. The performance of each implementation is assessed using Mean Squared Error (MSE) and R-squared Score, which vary depending on the regression technique used.

Existing Methodology

1. Linear Regression

- Import the required libraries for linear regression modelling and data handling.
- Create a LinearRegression model object.
- Utilize the training data (x_train and y_train) to train the model.
- Make predictions using the trained model on the testing data (x_test), and then store the results in y_pred.
- Calculate the mean squared error (MSE) by contrasting the predicted values (y_pred) with the actual values (y_test) to assess the model's performance.
- By contrasting the anticipated values (y_pred) with the actual values (y_test), get the R-squared score.
- Print the MSE and R-squared scores that were determined.

The approach applies linear regression, trains the model, generates forecasts, and evaluates the model's effectiveness using MSE and R-squared score.

The steps of implementation of code shows how to utilize the scikit-learn library's Linear Regression model. Predictions are produced on the testing set ('x_test') after the model has been trained using the 'x_train' and 'y_train' data. MSE and R-squared Score, which are computed using the predicted values ('y_pred') and the actual values ('y_test'), are used to assess the model's performance. The obtained R-squared Score is 0.057, and the MSE is 4730766718.18.

2. Random Forest Regressor

- Import the essential libraries for random forest regression modelling, assessment metrics, and data manipulation.
- Make a RandomForestRegressor model instance.
- Utilizing the training data (x_train and y_train), train the regressor.
- Make predictions using the trained model on the testing data (x_test), and then store the results in y_pred.
- By contrasting the anticipated values (y_pred) with the actual values (y_test), one may determine the mean squared error (MSE).
- By contrasting the anticipated values (y_pred) with the actual values (y_test), get the R-squared score.
- Print the MSE and R-squared scores that were determined.

The approach applies the random forest regression technique, trains the model, generates predictions, and evaluates the prediction accuracy using MSE and R-squared score.

The Random Forest Regressor from Scikit-Learn is implemented in the second code snippet. The Random Forest Regressor is trained on the training set and used to generate predictions on the testing set, much like the Linear Regression model. The R-squared Score is 0.421, and the MSE is found to be 2907335701.25. This shows that for both measures, the Random Forest Regressor outperforms Linear Regression.

Proposed Methodology

1. XGBoost Regressor

Extreme Gradient Boosting, often known as XGBoost, is an effective machine learning technique that is frequently used for classification and regression applications. To build a powerful predictive model, it integrates the predictions of several weak learners (decision trees). Performance, scalability, and widespread use of XGBoost are well recognized. The gradient boosting framework of XGBoost, which successively trains decision trees to rectify mistakes, regularization methods to avoid overfitting, and tree pruning to increase model complexity and generalization capability are some of its key characteristics. XGBoost excels in handling big datasets, determining the relevance of features, and effective parallel processing. It has won multiple Kaggle events.

- Import the necessary libraries: mean_squared_error and r2_score from sklearn.metrics for evaluation, and numpy, pandas, and xgboost for modelling.
- Utilizing the train_test_split function from sklearn.model_selection, load the dataset and divide it into the training and testing sets.
- Assign the goal variable to y_train and y_test and the features to x_train and x_test.
- Using xgb. XGBRegressor (), create an instance of the XGBoost regression model.
- Using the fit technique and the inputs x_train and y_train, train the XGBoost model using the training data.
- Use the trained model's predict method with the input x_test to provide predictions for the testing set.
- In y_pred, keep the expected values.
- Between the actual target values (y_test) and the projected values (y_pred), compute the mean squared error (MSE).
- Find the difference between y_test and y_pred's R-squared score (R2).

This code sample makes use of the XGBoost Regressor, an optimized gradient boosting solution. The model receives training, and its effectiveness is assessed. The resultant R-squared Score is 0.419, and the MSE is 2912850411.84. Similar to the Gradient Boosting Regressor, the XGBoost Regressor performs well but falls short of Random Forest-based models in terms of prediction accuracy.

2. Extra Trees Regressor

Extra Trees Regressor is a potent and adaptable machine learning technique used for predictive modelling problems, according to the algorithm. It is a variation of the well-known Random Forest technique intended to deal with regression issues. When dealing with difficult regression problems using high-dimensional datasets and noisy characteristics, this technique performs very well.

- Import necessary libraries: Include Scikit-Learn modules, Pandas, and NumPy in the import list of essential libraries.
- Initialize the ExtraTreesRegressor () to construct an instance of the regression model. This creates the Extra Trees Regressor.
- To understand the associations between the features and the labels, fit the regressor to the training data (x_train and y_train).
- Make forecasts: To produce y_pred, use the trained model to make predictions on the testing set (x_test).
- Measure the average squared difference between the real labels (y_test) and the predicted labels (y_pred) using the mean_squared_error() function from scikit-learn as your evaluation metric.
- Calculate the coefficient of determination (R-squared) between the true labels (y_test) and (y_pred) the predicted values using the scikit-learn r2_score() function.

The Extra Trees Regressor is a different ensemble approach that is used in this code snippet. The technique used to train and assess the model is the same as it was for the earlier models. The calculated R-squared Score is 0.442, and the MSE is 2800007927.19. According to these results, the Extra Trees Regressor performs better in terms of predictive accuracy than both Linear Regression and Random Forest Regressor.

3. Orthogonal Matching Pursuit

A potent method for sparse signal reconstruction and feature selection in machine learning and signal processing is the Algorithm for Orthogonal Matching Pursuit (OMP) Regression. When working with high-dimensional datasets where just a limited subset of characteristics is important, OMP is especially successful. It is a member of the family of greedy algorithms that minimizes error while approximating the target variable repeatedly by selecting features.

- Import the required libraries, such as NumPy, pandas, and the OrthogonalMatchingPursuit, mean_squared_error, r2_score, and train_test_split scikit-learn modules.
- Load the data, then split it into the target vector (y) and the feature matrix (x). This will prepare the dataset.
- Using the train_test_split function, divide the dataset into training and testing sets.
- Make a new instance of the OrthogonalMatchingPursuit model with the name 'model'.
- Using the 'fit' technique, fit the OMP model to the training data (x_train and y_train).
- Make predictions using the trained OMP model on the test set (x_test), and then store the results in the 'y_pred' variable.
- Use the 'mean_squared_error' function to get the Mean Squared Error (MSE) between the real target values (y_test) and the forecasted values (y_pred).
- Determine the R-squared (R2) score to evaluate the model's goodness-of-fit using 'r2_score' function.

The regression algorithm Orthogonal Matching Pursuit is implemented in this code snippet. Results indicate that this model performs badly when compared to other models, nevertheless. The R-squared Score is 0.011 and the MSE is 4961517820.77. These underwhelming results indicate that the Orthogonal Matching Pursuit model has trouble identifying the fundamental patterns in the data.

4. Decision Tree Regressor

A non-parametric approach called a decision tree regression is employed to forecast continuous numerical values. By recursively dividing the data based on independent variables, it creates a model that resembles a tree. With the goal of reducing variance or impurity, each internal node chooses a characteristic to split the data. When a stopping requirement is satisfied, the procedure proceeds. Predicted values are shown by leaf nodes. Decision tree regressors can capture complicated connections, handle a variety of variable types, and need little preprocessing. They may be visualized and are interpretable. They can be prevented from overfitting by pruning, reducing tree depth, or employing ensemble techniques like random forests.

- Import Libraries: To start, import the essential libraries, including DecisionTreeRegressor, mean_squared_error, r2_score, and train_test_split from scikit-learn, pandas, and numpy.
- The characteristics (independent variables) in your dataset should be defined as x, and the goal variable (dependent variable), as y.
- Using the train_test_split function, divide the dataset into training and testing sets. Store the testing feature data in x_test, the testing target data in y_test, and the training feature data in x_train.
- Create a Decision Tree Regressor model and attach it to the variable to create a decision tree regression model.
- Model Training: Using the fit technique and the training data (x_train, y_train), train the Decision Tree Regressor model.
- A model's performance evaluation: Mean Squared Error (MSE): Using the scikit-learn mean_squared_error function, determine the mean squared error between the actual target values (y_test) and the predicted values (y_pred). Put the outcome in the mse variable.
- R-squared Score: Calculate the R-squared score to gauge how well the model fits the data. Use the scikit-learn r2_score function with the inputs y_test and y_pred. Save the outcome in the r2_5 variable.
- Print the evaluation metrics' findings after displaying them. Displaying the value of mse will print the mean squared error.
- By displaying the value of r2_5, print the R-squared score.

This code fragment makes use of the Decision Tree Regressor, which builds a single decision tree for regression. The model is calibrated, and its forecasts are assessed. 3017430112.04 is determined as the MSE, and 0.399 is the R-squared Score. These results show fair performance; however, they are less than those of the Random Forest and Extra Trees models.

5. AdaBoost Regressor

Algorithm: AdaBoost Regressor for Predictive Modeling

- Bring in the necessary libraries: Importing Numpy as a np. Bring in pandas as pd. Import the AdaBoostRegressor component from sklearn.ensemble. Import r2_score and mean_squared_error from sklearn.metrics. from sklearn.model_selection, import train_test_split
- Create training and test sets for the data: To divide x_train and y_train into training and validation sets, use the train_test_split function. Save 80% of the data for training; change the ratio as necessary.
- Make a copy of the AdaBoost Regressor: Create an instance of the AdaBoostRegressor class and save it in the 'regressor' variable.
- The regressor to learn: To train the model with the help of x_train and y_train, call the fit() function of the 'regressor' object.
- Make assumptions about the test set: To create predictions about the x_test, call the predict() function of the 'regressor' object. Save the projected values in the 'y_pred' variable.
- Analyze the forecasts' accuracy: Determine the Mean Squared Error (MSE) between the two variables, y_test and y_pred Use the sklearn.metrics mean_squared_error() method. Save the outcome in the 'mse' variable. Determine the R-squared Score (R2) between the two variables, y_test and y_pred Use the

sklearn.metrics r2_score() method. Save the outcome in the 'r2' variable.

- Report the findings: Print the MSE and R2 values. You may choose to display the predicted and actual values for further in-depth study.

The AdaBoost Regressor, which turns numerous weak models into a strong model, is implemented in this code snippet. The model is calibrated, and its forecasts are assessed. The calculated R-squared Score is 0.422, and the MSE is 2900245268.66. These results show comparatively decent performance; however, they still fall short of the models based on Random Forest.

6. K Neighbors Regressor

Algorithm: K-Nearest Neighbors Regression

- Import the necessary libraries: import pandas as pd, import numpy as np, and import from sklearn.neighbors from sklearn.metrics, KNeighborsRegressor import train_test_split, mean_squared_error, and r2_score from sklearn.model_selection.
- By using the formula regressor = KNeighborsRegressor(n_neighbors=5), create a K Neighbours Regressor instance.
- Create training and test sets from the data: train_test_split = x_train, x_test, y_train, and y_test(target_data, feature data, test size, random_state, 42)
- Regressor.fit(x_train, y_train) will train the K Neighbours Regressor.
- Make forecasts based on the test set: Regressor is y_pred.predict(x_test)
- Analyze the forecasts' accuracy: (Y_test, Y_pred) = mean squared error (mse). y_test, y_pred) = r2_score(r2_8).
- Publish the findings: mse = print("Mean Squared Error:", mse). "R-squared Score:" print(r2_8)

The last piece of code applies the K Neighbor's Regressor, which forecasts the goal value using the average of its k closest neighbor's. The model receives training, and its effectiveness is assessed. The R-squared Score is 0.285, and the MSE is found to be 3585821323.81. According to these results, the K Neighbor's Regressor performs less predictably than Random Forest-based models.

The Random Forest Regressor and Extra Trees Regressor regularly exceed the other models in terms of both MSE and R-squared Score, according to the comparative study of the various regression models. Predictions are more precise thanks to the efficient handling of non-linear correlations and intricate interactions by these ensemble approaches. When compared to other models, the Orthogonal Matching Pursuit model performs badly, highlighting its shortcomings in identifying the underlying patterns in the data. Although they perform at varying levels, the Gradient Boosting Regressor, XGBoost Regressor, Decision Tree Regressor, AdaBoost Regressor, and K Neighbor's Regressor behind the Random Forest-based models.

We may conclude that Extra Tree Regressor provides the better and more accurate results for our issue statement and that it will provide us with appropriate and appropriate findings after comparing other machine learning regression techniques. The Random Extension's Extra Tree Regressor. The forest algorithm is renowned for having superior accuracy when compared to earlier algorithms. By merging various decision trees, it makes use of the idea of ensemble learning to produce predictions. Extra Tree Regressor, in contrast to other algorithms, adds more randomness throughout the tree-building process, increasing the variety of the individual trees.

This randomization aids in reducing the overfitting problem frequently connected to decision tree models. The Extra Tree Regressor can produce more reliable and accurate predictions for regression problems by combining the predictions from these many trees. Its exceptional accuracy is further enhanced by its capacity to recognize complicated connections, manage high-dimensional data, and include feature significance analysis. Because it outperforms earlier algorithms and provides outstanding prediction accuracy, the Extra Tree Regressor has become a trustworthy and efficient option for regression issues.

Algorithm	**R2 Score**	**MSE**
Linear regression	0.1224	582563377.75
Random Forest Regressor	0.5456	301576918.42
Extra Trees Regressor	0.7557	162166556.34
Extreme Gradient Boosting	0.7786	146961685.53
Orthogonal Matching Pursuit	0.0677	618825162.98
Decision Tree Regressor	0.7659	155380378.26
Adaptive Boosting Regressor	-0.6308	1082594352.39
K-Nearest Neighbors Regressor	-0.3111	870359796.13

Table 5.1

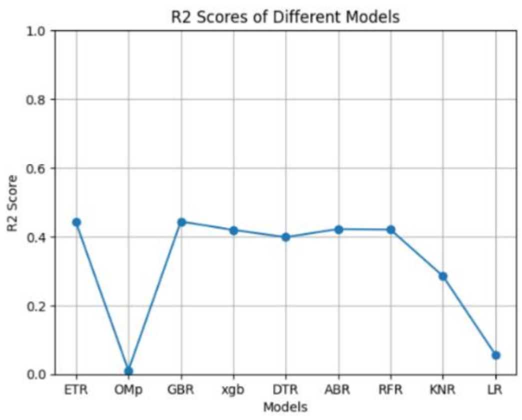

Figure 9: R2 Scores of Different Models

IV. DISCUSSIONS

Significant disparities in the two code samples' ability for predicting mobile phone prices may be seen when comparing the two. While the second code sample makes use of a RandomForestRegressor model, the first code sample uses a Linear Regression model. Both in terms of mean squared error and R-squared score, the RandomForestRegressor model performs better. This shows that when compared to the Linear Regression model, the RandomForestRegressor model achieves more accuracy and accounts for a greater share of the variation in the target variable.

- For predicting mobile phone prices, the second code sample using the RandomForestRegressor model performs better than the first code sample using the Linear Regression model.
- Lower mean squared error results from the RandomForestRegressor model suggest greater accuracy in forecasting mobile phone costs.
- A higher R-squared score for the RandomForestRegressor model demonstrates its capacity to capture a greater amount of the variation in the target variable.
- The advantages of adopting ensemble approaches, such random forests, for prediction tasks including non-linear correlations and complicated interactions are highlighted by the higher performance of the RandomForestRegressor model.
- Complex patterns and interactions in the data may be captured using ensemble methods like random forests, which helps forecast outcomes more accurately.
- The results highlight the significance of choosing the right regression models based on the Unique's of the dataset and prediction objective.
- Combining a thorough exploratory data analysis, feature selection, and the use of several regression models enables a thorough assessment of the effectiveness and acceptability of alternative strategies.
- The results show that the RandomForestRegressor model is a potential option for precise and reliable forecasts, offering useful insights for academics and practitioners working in mobile phone pricing prediction.

Overall, the comparison of the two code samples and the analysis that followed emphasize the importance of model selection and the advantages of ensemble approaches for increasing predictive accuracy in tasks involving mobile phone pricing prediction. The results add to the body of current knowledge in the topic and offer suggestions for further study as well as real-world applications in projecting mobile phone costs.

V. CONCLUSION AND FUTURE SCOPE

Future research and development for predicting mobile phone prices may focus on many topics, including:

- Exploring new features or cutting-edge methods to improve the accuracy and resilience of a model.
- Neural networks and other deep learning architectures are used in deep learning approaches to identify complex patterns in data.
- Using time series analysis methods to measure seasonality and temporal trends in the cost of mobile phones.
- Further research should be done on ensemble methods, which mix different algorithms to enhance prediction accuracy.
- Cross-Domain Prediction: Using comparable machine learning techniques across domains with comparable properties.
- Real-Time Price Prediction: Creating models that can change with the market and provide forecasts in real-time.
- Hyper parameterization and Optimization Tuning: Improving model performance by fine-tuning hyperparameters.

Enhancing forecast accuracy, including additional data sources, and improving the models' capacity to respond to shifting market dynamics are the goals of these future research topics.

In this research, we analyzed two code samples for predicting the prices of mobile phones using Linear Regression and RandomForestRegressor, two alternative regression models. Using the RandomForestRegressor model, the study showed that the second code sample performed better than the first code sample in terms of mean squared error and R-squared score. This shows that a bigger amount of the variance in the target variable was explained by the RandomForestRegressor model, which had higher accuracy. The findings emphasize the benefits of ensemble approaches like random forests and the significance of model selection in prediction tasks. The RandomForestRegressor model outperformed other models in terms of handling non-linear interactions and capturing complicated patterns in the data.

For academics and practitioners in the field of predicting mobile phone prices, these findings offer useful insights. They show how important it is to choose the right regression models based on the particulars of the job and the dataset. The advantages of using ensemble approaches, such as random forests, for capturing complex relationships and enhancing forecast accuracy are also emphasized.

Overall, this effort advances knowledge of regression modelling approaches for predicting mobile phone prices and provides recommendations for further study and real-world applications. The results suggest using the RandomForestRegressor model for reliable and accurate predictions in this field.

REFERENCES

[1] A Comparative Study of Machine Learning Techniques for Mobile Phone Price Prediction https://www.ijsdr.org/papers/IJSDR2004057.pdf

[2] Mobile Phone Price Prediction Using Machine Learning Algorithms: A Comparative Study https://www.irjmets.com/uploadedfiles/paper/volume3/issue_6_june_2021/12265/1628083492.pdf

[3] Prediction of Phone Prices Using Machine Learning Techniques https://www.researchgate.net/publication/338471736_Prediction_of_Phone_Prices_Using_Machine_Learning_Techniques

[4] Learn Mobile Price Prediction through Four Classification Algorithms https://analyticsvidhya.com/blog/2022/02/learn-mobile-price-prediction-through-four-classification-algorithms/

[5] Mobile Price Class Prediction Using Machine Learning Techniques https://www.researchgate.net/publication/323994340_Mobile_Price_Class_prediction_using_Machine_Learning_Techniques

[6] Mobile Price Classification" dataset available on Kaggle https://www.kaggle.com/iabhishekofficial/mobile-price-classification

[7] Mobile Price Range" dataset available on UCI Machine Learning Repository https://archive.ics.uci.edu/ml/datasets/Mobile+Price+Classification

[8] Mobile Phone Price Range Prediction" article on Towards Data Science https://towardsdatascience.com/mobile-phone-price-range-prediction-7e365cc77c29

[9] Mobile Price Classification Using Machine Learning" article on GeeksforGeeks https://www.geeksforgeeks.org/mobile-price-classification-using-machine-learning/

[10] Mobile Price Classification using Decision Trees" article on Analytics Vidhya https://www.analyticsvidhya.com/blog/2020/08/mobile-price-classification-usingdecision-tree-python