Smart Mobile Phone Price Prediction Using Machine Learning

Team Name: Infinity

Team Members

S. VISHNU VARDHAN REDDY

D. VENKATA SHIVA

B. ANKITHA

COLLEGE MENTOR: MRS D. KALPANA

ABSTRACT

This paper proposes a predictive system for estimating mobile phone prices using machine learning algorithms. By analyzing historical data, market trends, and phone specifications, the system generates accurate price predictions. Users can input phone specifications, and the system provides an estimated price range. The objective is to aid consumers, retailers, and manufacturers in making informed decisions in the fast-paced mobile phone market by predicting prices in real-time. The project proposes a predictive system for estimating mobile phone prices. The system utilizes machine learning algorithms to analyze historical data, market trends, and phone specifications. Users can input phone specifications, and the system generates an estimated price range. The objective is to assist consumers, retailers, and manufacturers in making informed decisions. Real-time price predictions help navigate the dynamic and competitive mobile phone market. The system aims to improve decision-making by providing accurate and up-to-date price forecasts. By leveraging historical data and market trends, the system generates reliable price ranges.

1. INTRODUCTION

Smartphone price prediction using machine learning involves building a model to estimate the price of a smartphone based on various features and characteristics. This helps both consumers and sellers in understanding the factors that influence smartphone prices. The process begins by collecting a dataset containing information such as specifications, brand, release year, storage capacity, camera quality, battery life, and screen size, along with corresponding prices.

The dataset is then preprocessed by handling missing values, encoding categorical variables, and normalizing numerical features. It is split into a training set and a test set. Various regression algorithms like linear regression, decision trees, random forests, gradient boosting, and neural networks are evaluated for their performance using metrics like mean squared error or mean absolute error.

The best-performing algorithm is selected and trained using the training set, adjusting its parameters and hyperparameters to optimize performance. The model is then evaluated on the test set to assess accuracy and generalization capabilities. If the model performs well, it is deployed to make predictions on new, unseen data. By inputting the specifications of a smartphone, the model estimates its price based on learned patterns from the training data. The accuracy of the predictions depends on the dataset quality, chosen algorithm, and relevant features. Regular updates and refinement are crucial to improve the model's accuracy and relevance in predicting smartphone prices.

2. PREVIOUS WORK

The present study contributes to the field of mobile price prediction by introducing a machine learning-based approach to determine the price class of mobile phones. Through the collection of a comprehensive dataset from www.GSMArena.com, the authors employed various feature selection algorithms to identify the most influential factors for accurate price prediction. Subsequently, a range of classifiers, including decision trees,

random forests, and support vector machines, were employed to predict the price class of mobile phones.

The results demonstrated that the random forest classifier, utilizing 10 carefully selected features, achieved the highest accuracy of 84%. This suggests that the proposed model can effectively predict the price class of mobile phones with a notable level of accuracy.

The findings further revealed that specific features such as the processor, screen size, and amount of RAM exhibited a significant influence on price prediction. These features directly impact the overall performance of mobile phones, consequently affecting their market value.

The implications of this research are multifaceted. Firstly, it can assist consumers in making informed decisions when selecting mobile phones that align with their budget and requirements. Secondly, businesses within the mobile industry can leverage the developed model to set appropriate prices for their products. Lastly, researchers interested in exploring the factors influencing mobile phone prices can utilize this work as a foundation for their investigations.

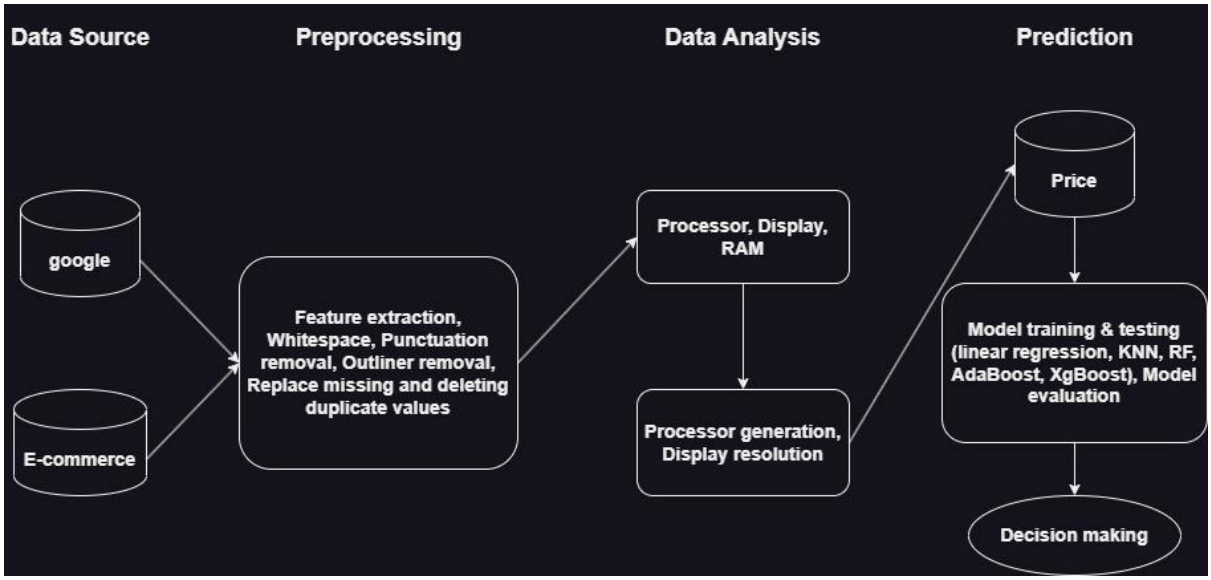
In summary, the current study demonstrates the efficiency of employing machine learning techniques for mobile price prediction. By identifying crucial features and employing advanced classifiers, the model achieves an impressive accuracy rate of 84%. The significance of factors such as the processor, screen size, and RAM on price prediction underscores their relevance in the mobile market. The findings offer practical benefits to consumers, businesses, and researchers alike.

This project aims to classify mobile phone prices into different categories such as very economical, economical, expensive, and very expensive. The provided price range is as follows:

Range (EUR)	Class
<150	Very economical
151-300	Economical
301-450	Expensive
>450	Very expensive

Table: classification criteria

3. Architecture



4. METHODOLGY

4.1 DATA COLLECTION

Ten features of mobiles are collected from www.GSMArena.com.

Category (whether the given mobile is made by apple, Samsung, Lenovo, lava, intex, infinix, oppo, vivo, realme, redmi, poco, moto, iqoo, asus, nokia, oneplus.)

Size of Display (inches, x-resolution, y-resolution), **RAM size** (GB), **ROM size** (GB), **CAMERA** (MP), **PROCESSOR** (mediatek helio, mediatek dimensity, snapdragon, etc..), **OS** (android, IOS).

In the previous study, the price attribute was classified into classes based on price ranges. However, in our approach, we consider the original prices instead of classifying them. To evaluate the data, we divided it into a training set and a test set, with 930 instances used for training and 150 instances used for testing. In total, we worked with 1082 instances.

4.2 DATA PREPROCESSING

Initially, our dataset consisted of columns 4abelled as brand name, RAM, storage, camera, processor, display, battery, flash charge, operating system (OS), and price.

4.2.1 Processing RAM

Initially, the RAM column contained values in string format such as ‘4GB’, ‘6GB’, etc. We converted these string values into integer values by removing the ‘GB’ substring.

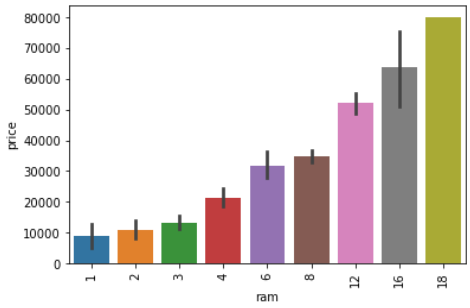


Fig: relation b/w RAM and price

4.2.2 *Processing ROM*

Initially, the ROM column contained string values such as '64GB, 128GB, 256GB,' and so on. We converted these string values into integer values by removing the 'GB' portion. Additionally, for ROM values specified as '1TB,' we converted it to 1024GB.

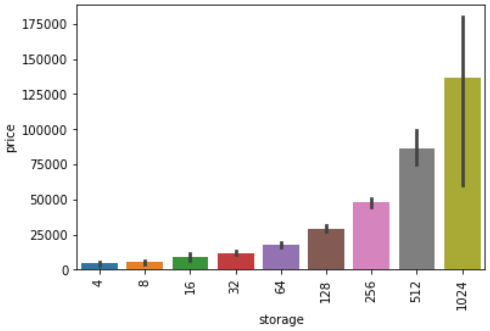


Fig: relation b/w storage and price

4.2.3 *Processing Camera*

Initially, the camera column contained values such as '48mp', '108mp', '50mp', and so on, which were in string format. We converted these string values into integer values by removing the 'MP' substring.

4.2.4 *Processing Operating System*

During the processing of the OS column, we encountered a problem regarding categorizing the operating systems (Android and iOS). To overcome this, we first transformed the values into lowercase and then applied a user-defined function to return the corresponding values as either Android or iOS.

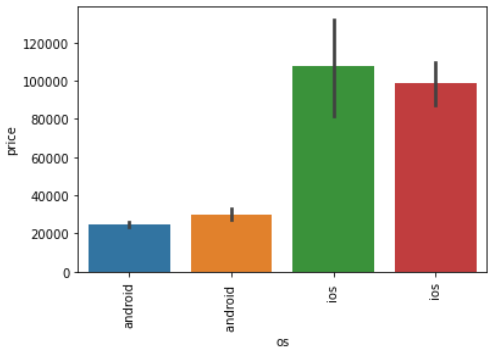


Fig: Before resolving

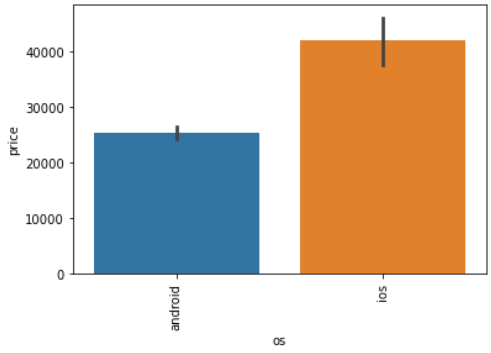


Fig: After resolving

4.2.5 *Extracting PPI value from display*

To calculate the PPI (pixels per inch) value, we require the X resolution, Y resolution, and diagonal size in inches. In our dataset, the display column includes the diagonal size along with the type of display and the mobile screen resolution. To extract the screen resolution and diagonal size, we need to use a regular expression. By implementing this regular expression, we can separate the screen resolutions and diagonal size into distinct columns. The formula to calculate the PPI is provided below:

$$pixel\ density = \frac{\sqrt{width^2 + lenght^2}}{screen\ size\ (diagonal)}$$

```
df['price'].corr(df['ppi'])
0.46704343805155546
```

Fig: Correlation between price and PPI

4.2.6 *Extracting Processor*

The dataset contains various types of processors, including different variants of Snapdragon, MediaTek Helio processors, MediaTek Dimensity processors, iPhone processors, and other processors. In total, there are 219 types of processors in the dataset. To simplify the analysis, we categorized these 219 types into five categories: MediaTek Helio, MediaTek Dimensity, Apple, Snapdragon, and Other.

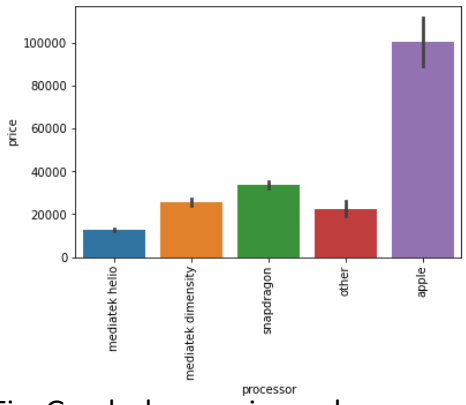


Fig: Graph shows price and processor

5. Algorithms & its Metrics

5.1 *Linear Regression*

We utilized a column transformer with OneHotEncoder to handle categorical features like name, processor, and OS. To address multicollinearity, we dropped the first column. Using a pipeline with column transformation and linear regression, we obtained an R2 score of 0.80164 and an MAE of 0.255.

5.2 Ridge Regression

Using Ridge regression with an alpha value of 10, we obtained an R2 score of 0.79102 and an MAE of 0.261. This approach was similar to linear regression but included regularization to mitigate potential overfitting or multicollinearity.

5.3 Lasso Regression

Following a similar approach to Ridge regression, we performed Lasso regression by reducing the alpha value. The resulting R2 score was 0.7861, and the MAE was 0.258. Lasso regression adds a sparsity constraint, encouraging feature selection and reducing the impact of irrelevant variables.

5.4 KNN

We utilized K-Nearest Neighbors (KNN) with a parameter of 3 for the number of nearest neighbors. We experimented with different neighbor values but obtained the best results with an R2 score of 0.76662 and an MAE of 0.249.

5.5 Decision Tree

We applied a decision tree with a maximum depth parameter set to 8. This yielded an R2 score of 0.78375 and an MAE of 0.243. Further improvements can be achieved by exploring and tuning other parameters of the decision tree algorithm.

5.6 SVM

We used SVM with C parameter controlling error tolerance and epsilon set to 0.1 for regression. The R2 score of 0.77383 indicates 77.4% target variance explanation, while the MAE of 0.237 represents an average deviation of predictions from actual values.

5.7 Random Forest

We implemented a Random Forest with the following parameters: n_estimators=100, random_state=3, max_samples=0.5, max_features=0.75, and max_depth=15. These parameter choices were made to achieve a balance between model complexity, generalization ability, and reducing overfitting. The resulting R2 score of 0.87938 and MAE of 0.188 indicate strong performance of the Random Forest model.

5.8 Ada Boost

We implemented AdaBoost with the parameters n_estimators=15 and a learning rate of 1.0. These choices were made to balance the number of weak learners and the impact of each learner on the final model. The resulting R2 score of 0.76267 and MAE of 0.289 indicate slightly lower performance compared to other models. Adjusting the parameters may further improve the results.

5.9 Gradient Boost

We implemented Gradient Boosting with the parameter n_estimators=500. This choice was made to increase the number of boosting iterations, allowing the model to learn complex patterns and improve performance. The resulting top R2 score of 0.91701 and MAE of 0.154 indicate the effectiveness of Gradient Boosting in capturing the underlying relationships in the data.

5.10 XGBoost

We implemented XGBoost with the parameters n_estimators=45, max_depth=5, and learning_rate=0.5. These choices were made to balance the trade-off between model complexity, learning rate, and the depth of individual trees. The resulting second highest R2 score of 0.89255 and MAE of 0.169 demonstrate the effectiveness of XGBoost in capturing complex relationships and achieving high predictive accuracy.

Algorithm	R2 score	MAE
Linear regression	0.8016419099942476	0.2554426549665309
Ridge regression	0.7910269180313019	0.26142295984031805
KNN	0.7666298790374444	0.24928156484974887
Decision tree	0.7837599739802491	0.24366138374033125
Random forest	0.8793828226848077	0.18854058757630346
Ada Boost	0.7626750267354508	0.2893196630582138
Gradient Boost	0.9170113123677978	0.15475473704363685
XGBoost	0.892550579085909	0.1696576861941368
Lasso regression	0.7986187594355987	0.2583128739952594
SVM	0.7738349914841682	0.237784729456288

We achieved a high score in Gradient Boosting compared to other algorithms. Additionally, Random Forest also performed well in terms of the score.

6. Conclusion

This work can be concluded by highlighting the good results obtained using various algorithms. The combination of PPI values, RAM, ROM, processor, and OS has resulted in maximum accuracy while selecting a minimal yet most appropriate set of features. It is worth noting that adding additional instances can further improve the score. One of the main advantages of our project is the consideration of actual prices instead of pricing classes such as expensive and economical. In conclusion, our project achieved a high score in Gradient Boosting and Random Forest, solidifying its success.