Smart Mobile Phone Price using Machine Learning Techniques

A Nitish and M Ganga Vasudha and MV Sai Keerthana
Alliance University - Central Campus,
Chandapura-Anekal, Main Road -
Bengaluru, Karnataka - 562106, India.

ABSTRACT

The goal of this research is to create a model that accurately predicts the price of a mobile phone when provided with its specifications. To achieve this, research will leverage predictive analytics, specifically through the use of machine learning. By utilizing machine learning algorithms, the research aims to develop and train a prediction model using input data. The trained model will then be employed to forecast the prices of future mobile phone instances. Supervised machine learning algorithms will be utilized, which rely on data that includes a pre-defined class label, in this case, the mobile phone price. To train the prediction model, the Mobile Price Class dataset from the Kaggle data science community-website: (https://www.kaggle.com/iabhishekofficial/mobile-price-classification).
This dataset categorizes mobile phones into different price ranges. Python was chosen as the programming language due to its extensive ML libraries. Several classification algorithms were implemented to train the model, with the objective of identifying the algorithm that offers the most accurate predictions of mobile phone prices. Performance metrics such as accuracy score and confusion matrix were utilized to evaluate the trained models and determine the most suitable algorithm among those used.

I. INTRODUCTION

The pricing of a product holds immense significance in its marketing, often serving as the ultimate determining factor for consumer purchase decisions. In a constantly evolving and unpredictable market, pricing can make or break the success of a product. Hence, it is crucial for companies to establish an optimal price before launching a product. A valuable tool in this regard is the ability to estimate prices based on product features, enabling informed pricing decisions. This estimation tool proves beneficial not only for companies but also for consumers, who can obtain estimated prices based on their desired product features.

Machine learning algorithms offer a wide range of capabilities that can be tailored to specific data and task objectives. Various tools and programming languages, such as Python, MATLAB, Java, WEKA, Cygwin, and Octave, are available to facilitate machine learning tasks. Commonly used algorithms include Naïve Bayes, K-NN, among others. Feature selection algorithms play a crucial role in training models by identifying and extracting the most relevant parameters. This process optimizes accuracy and reduces computational time.

The choice of the specific method for predicting product prices depends on the available data and the specific requirements of the model training process.

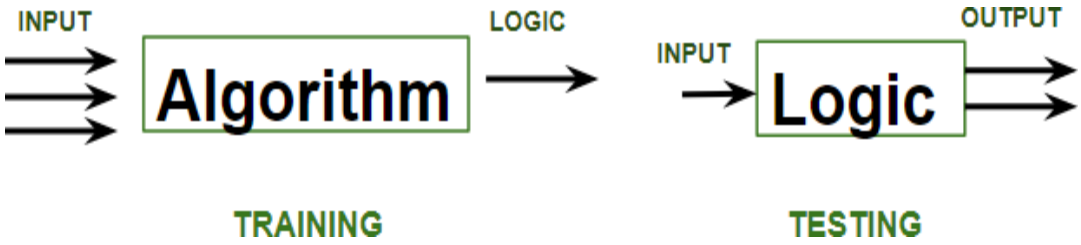
In the present era, cellphones have become an indispensable accessory for individuals. They occupy a prominent position in the continuous evolution and advancement of technology. The market witnesses the rapid introduction of new mobile models with upgraded features, and the daily sales volume reaches thousands of units. In this highly dynamic and competitive market, mobile companies are compelled to establish optimal prices in order to effectively compete with their rivals.

The initial step in determining a suitable price for a mobile phone is to estimate it based on the device's features. The objective of this research is to develop a machine learning (ML) model that accurately predicts the price of a mobile phone by analyzing its specifications. This model can also be beneficial to potential buyers who wish to estimate the price of a mobile phone based on their desired features.

The same approach used to create the prediction model can be applied to develop price estimation models for other products with similar independent variables. Mobile phone prices are influenced by various features, such as the processor, battery capacity, camera quality, display size, and thickness. These features can be utilized to classify phones into different categories, such as entry-level, mid-range, flagship, or premium. To accomplish this, supervised ML algorithms are employed, leveraging a dataset that includes definitive class labels for different price ranges.

II. RESEARCH METHODOLOGY

The research was performed using the Python kernel in Google Colab, specifically focusing on the workflow diagram for supervised machine learning tasks is as follows:



(Ref: https://www.superannotate.com/blog/supervised-learning-and-other-machine-learning-tasks)

The dataset is split into two sections: the training set and the test set. The purpose of the training set is to train the model, while the test set is used to assess its performance. Through analysing the features of mobile phones, the computer strives to comprehend the underlying logic behind their pricing. This understanding is then applied to make accurate predictions for future instances.

III. UNDERSTANDING THE DATASET

The Mobile Price Class dataset that used in our project, obtained from the Kaggle data science community website (https://www.kaggle.com/iabhishekofficial/mobile-price-classification), was utilized to train the prediction model. This dataset categorizes mobile phones into different price ranges.

The dataset consists of a total of 21 attributes, which include 20 features and a class label representing the price range. The features encompass various aspects such as battery capacity, RAM, weight, camera pixels, and more. The class label indicates the price range and is categorized into four ordinal values: 0, 1, 2, and 3. These values correspond to increasing degrees of price, where higher values indicate higher price ranges. Specifically, these values can be interpreted as economical, mid-range, flagship, and premium categories.

Despite price being traditionally treated as a numeric problem, in this case, the machine learning task falls under classification rather than regression. This is because the class label consists of discrete values rather than continuous ones. The advantage of classification algorithms, such as Naïve Bayes and Decision Tree, is their compatibility with discrete data, which makes them suitable for this problem.

The dataset contains 2000 records in total.

```
import seaborn as sns
import matplotlib.pylab as plt
import numpy as np
import pandas as pd
%matplotlib inline

df=pd.read_csv('/kaggle/input/mobile-price-classification1/Mobile-data-train.csv')
df.head(5)
```

[1] Python

```
/opt/conda/lib/python3.10/site-packages/scipy/__init__.py:146: UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required for this version of SciPy (detected version 1.23.5
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	...	px_height	px_width	ram	sc_h	sc_w	talk_time	three_g	touch_screen	wifi	pri
0	842	0	2.2	0	1	0	7	0.6	188	2	...	20	756	2549	9	7	19	0	0	1	
1	1021	1	0.5	1	0	1	53	0.7	136	3	...	905	1988	2631	17	3	7	1	1	0	
2	563	1	0.5	1	2	1	41	0.9	145	5	...	1263	1716	2603	11	2	9	1	1	0	
3	615	1	2.5	0	0	0	10	0.8	131	6	...	1216	1786	2769	16	8	11	1	0	0	
4	1821	1	1.2	0	13	1	44	0.6	141	2	...	1208	1212	1411	8	2	15	1	1	0	

5 rows × 21 columns

```
df.info()
```

Python

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   battery_power  2000 non-null   int64
 1   blue           2000 non-null   int64
 2   clock_speed    2000 non-null   float64
 3   dual_sim       2000 non-null   int64
 4   fc             2000 non-null   int64
 5   four_g         2000 non-null   int64
 6   int_memory     2000 non-null   int64
 7   m_dep          2000 non-null   float64
 8   mobile_wt      2000 non-null   int64
 9   n_cores        2000 non-null   int64
 10  pc             2000 non-null   int64
 11  px_height      2000 non-null   int64
 12  px_width       2000 non-null   int64
 13  ram            2000 non-null   int64
 14  sc_h           2000 non-null   int64
 15  sc_w           2000 non-null   int64
 16  talk_time      2000 non-null   int64
 17  three_g        2000 non-null   int64
 18  touch_screen   2000 non-null   int64
 19  wifi           2000 non-null   int64
 20  price_range    2000 non-null   int64
dtypes: float64(2), int64(19)
memory usage: 328.2 KB
```

The df.info() function is commonly used in pandas, a popular data manipulation and analysis library in Python. When applied to a DataFrame object named df, it provides a summary of the dataset's information.

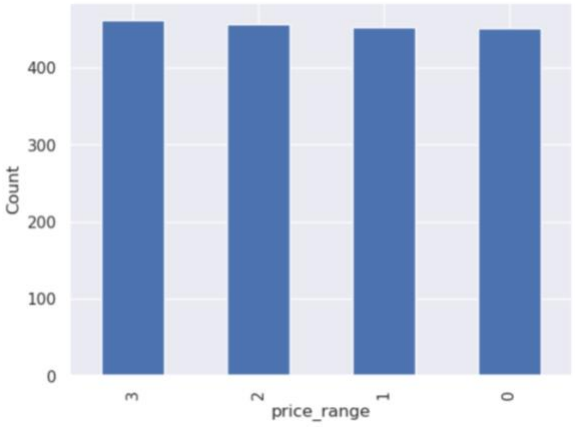
```
tdf= df[df['sc_w'] != 0]
tdf.shape
```

[3]

```
(1820, 21)
```

```
sns.set()
price_plot=tdf['price_range'].value_counts().plot(kind='bar')
plt.xlabel('price_range')
plt.ylabel('Count')
plt.show()
```

[4]



The code tdf = df[df['sc_w'] != 0] filters the DataFrame df to select rows where the 'sc_w' column is not equal to 0. The resulting DataFrame tdf contains the filtered data. tdf.shape retrieves the dimensions (number of rows and columns) of the DataFrame tdf.

IV. TRAINING THE PREDICTION MODEL

The first step in creating a model is to extract the required features for training from the dataset and assigning the parameter that is to be the class label.

```
X=tdf.drop(['price_range'], axis=1)
y=tdf['price_range']
X.isna().any()
```

In this code snippet, the first 20 attributes are being extracted to serve as the training parameters and the final attribute (price_range) is used as the class label.

```
[ ] from sklearn.model_selection import train_test_split
    X_train,X_test,y_train,y_test = train_test_split(X,y,test_size = 0.2,random_state = 0)
```

Prediction Model

1.Random Forest

```
#building random forest model
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier(bootstrap= True,
                           max_depth= 7,
                           max_features= 15,
                           min_samples_leaf= 3,
                           min_samples_split= 10,
                           n_estimators= 200,
                           random_state=7)
```

2.Naive Bayes

```
# using Guassian Naive Bayes model
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(X_train, y_train)
y_pred_gnb=gnb.predict(X_valid)
```

3.KNN

```
#using KNN classifier
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=3,leaf_size=25)
knn.fit(X_train, y_train)
y_pred_knn=knn.predict(X_valid)
print('KNN Classifier Accuracy Score: ',accuracy_score(y_valid,y_pred_knn))
cm_rfc=my_confusion_matrix(y_valid, y_pred_knn, 'KNN Confusion Matrix')
```

4.SVM

```
#using SVM classifier
from sklearn import svm
svm_clf = svm.SVC(decision_function_shape='ovo')
svm_clf.fit(X_train, y_train)
y_pred_svm=svm_clf.predict(X_valid)
print('SVM Classifier Accuracy Score: ',accuracy_score(y_valid,y_pred_svm))
cm_rfc=my_confusion_matrix(y_valid, y_pred_svm, 'SVM Confusion Matrix')
```

V. RESULTS AND DISCUSSION

A random forest system is composed of multiple decision trees working together. Each decision tree has its own structure, including a root node, leaf nodes representing choices, and intermediate nodes reflecting the decision-making process. The final output of the random forest system is determined through a majority vote, where the choice selected by the majority of decision trees becomes the ultimate output. Now, let's proceed with the application of the random forest approach.

```
Random Forest Classifier Accuracy Score:  0.9093406593406593
              precision    recall  f1-score   support

           0       0.98      0.97      0.97        95
           1       0.90      0.92      0.91        92
           2       0.82      0.86      0.84        86
           3       0.93      0.88      0.90        91

    accuracy                           0.91       364
   macro avg       0.91      0.91      0.91       364
weighted avg       0.91      0.91      0.91       364
```

Gaussian Naive Bayes is a variant of the Naive Bayes algorithm designed to handle continuous data following a Gaussian normal distribution. The Naive Bayes theorem serves as the fundamental principle for a family of supervised machine learning classification algorithms, collectively known as Naive Bayes. This approach offers a simple yet powerful categorization technique, particularly useful when dealing with high-dimensional input data. The Naive Bayes Classifier can effectively address complex classification problems as well.

Gaussian NB Classifier Accuracy Score: 0.8461538461538461

	precision	recall	f1-score	support
0	0.93	0.92	0.92	95
1	0.79	0.73	0.76	92
2	0.74	0.80	0.77	86
3	0.92	0.93	0.93	91
accuracy			0.85	364
macro avg	0.84	0.85	0.84	364
weighted avg	0.85	0.85	0.85	364

The K Nearest Neighbor (KNN) method is a versatile supervised learning technique employed for both classification and regression tasks. It enables the prediction of the class or continuous value of a new data point by analyzing the K nearest neighbors, as the name suggests. KNN can also be utilized to handle missing values and perform dataset resampling, making it a flexible approach in various data analysis scenarios.

KNN Classifier Accuracy Score: 0.9340659340659341

	precision	recall	f1-score	support
0	0.99	0.98	0.98	95
1	0.93	0.97	0.95	92
2	0.87	0.88	0.88	86
3	0.94	0.90	0.92	91
accuracy			0.93	364
macro avg	0.93	0.93	0.93	364
weighted avg	0.93	0.93	0.93	364

The purpose of the Support Vector Machine (SVM) algorithm is to identify the optimal line or decision boundary in order to classify data points in an n-dimensional space accurately. By finding the optimal hyperplane, SVM ensures that future data points can be easily assigned to their appropriate categories. A hyperplane serves as the ideal decision boundary in SVM, enabling effective separation and classification of data points.

SVM Classifier Accuracy Score: 0.9587912087912088

	precision	recall	f1-score	support
0	0.98	0.98	0.98	95
1	0.93	0.97	0.95	92
2	0.94	0.93	0.94	86
3	0.99	0.96	0.97	91
accuracy			0.96	364
macro avg	0.96	0.96	0.96	364
weighted avg	0.96	0.96	0.96	364

VI. CONCLUSION

The SVM model demonstrated the highest accuracy of 96% in accurately predicting mobile price classes. To further improve the model's performance, data pre-processing techniques like normalization and standardization can be employed. Additionally, the application of feature selection and extraction algorithms can help eliminate unsuitable and redundant features, leading to more refined results. This research methodology can be extended to predict prices for various other products such as cars, bikes, and houses, utilizing historical data that incorporates relevant features such as cost and specifications. Such an approach would provide valuable insights for organizations and consumers, enabling them to make well-informed pricing decisions.

VII. REFERENCES

1. D. Banerjee and S. Dutta, "Predicting the housing price direction using machine learning techniques", 2017 IEEE International Conference on Power Control Signals and Instrumentation Engineering (ICPCSI), pp. 2998-3000, 2017.
2. Kanwal Noor and Sadaqat Jan, “Vehicle Price Prediction System using Machine Learning Techniques”, International Journal of Computer Applications (0975 – 8887) Volume 167 – No.9, June 2017.
3. R. Gareta, L.M. Romeo and A. Gil, "Forecasting of electricity prices with neural networks", Energy Conversion and Management, vol. 47, pp. 1770-1778, 2006.
4. https://www.kaggle.com/iabhishekofficial/mobile-price- classification
5. https://www.gsmarena.com/
6. Sameerchand Pudaruth . “Predicting the Price of Used Cars using Machine Learning Techniques”, International Journal of Information & Computation Technology. ISSN 0974-2239 Volume 4, Number 7 (2014), pp. 753-764.