Mobile Price Classification Final Report

Onurcan İşler *150120825*
Erkam Karaca *150118021*
Berk Kırtay *150118043*

Abstract

With technology becoming a part of our lives, we witness an incredible increase in phone usage. People of all ages and geographies now use smartphones. Even a shoe cleaner in Mumbai can be seen using a smartphone, or a young child who has just learned to read and write. Moreover, many giant technology companies are launching various phones every month that offer many new features. But even this cannot meet the demand. In an environment where there is such a demand for smartphones, it can be difficult to determine which phone is how good. In this study, we use the dozens of features of the phones to classify the phones and make a kind of catalog for the people who want to buy a smart phone.

Index Terms

Mobile Prices; Machine Learning; Classification; Support Vector Machines; Logistic Regression; K-neighbors; Random Forest

I. ACCOMPLISHMENTS

A. Work Sharing among Team Members

This study was carried out by undergraduate computer engeineering students, Erkam Karaca, Onurcan İşler and Berk Kırtay.

- Although this work was shared among the members, Berk was mainly responsible for the implementation of the necessary classification algorithms for the classification of phones.
- Visualizing and analyzing the data was mostly Onurcan's task. He mainly sketched the plots and heat graphs for the presentation.
- Erkam was the person who found the problem on Kaggle and the author of proposal document. He also checked our commits on GitHub, and reported if there is any issue. He also helped Berk to write the classification algorithms to be used.

B. Tasks Accomplished

In fact, the team successfully solved the problem with the help of various classification algorithms and extensively visualized the whole data. Later in this report, we share some of our codes, charts, and algorithms. To summarize,

- Project subject and the problem has been determined.
- Project Proposal has been written.
- An extensive research for possible classification algorithms was carried out, and formulations of the algorithms reviewed.
- Project Proposal has been written.
- A few data visualization methods are applied. Sketched many charts and tables to better understand the data.
- Written a Midterm report to briefly describe the problem and possible solutions.
- The problem is completely solved. Used various classification techniques including K-neighbors, Random Forests, SVMs etc.
- Experimental results are visualized. These visualizations then explained in detail with their reasons.
- Written Final Report.
- Prepared slides for the presentation day. Here, the data visualizations and tables are extensively used.

II. PROBLEM STATEMENT

Given features of phones, classify the phones as ”cheap”, ”standard”, ”expensive” or ”very expensive”.

Here, total of 21 features are provided for the phones. To better learn about the metadata for these features we write the following two lines.

```
df = pd.read_csv('train.csv')
df.info()
```

```
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
```

Then, using these feature values, price category of the phones are determined.

- 0 means the phone is ”cheap”,
- 1 means the phone is ”standard”,
- 2 means the phone is ”expensive”,
- 3 means the phone is ”very expensive”.

In fact, the provided test data does not contain the price category for the phones. Only the train data contains the correct classifications for the phones. In this case, some portion of the train data is used for training and the remain portion is used for testing. Experimental results and algorithms are shared later in this report.

III. EXPLORATORY DATA ANALYSIS

Let’s start by better understanding about these parameters. We first use pandas.DataFrame.describe() function. In this case, count, mean, std, min and max values of these parameters are listed. See,

```
                count        mean          std    min     max
battery_power  2000.0  1238.51850   439.418206  501.0  1998.0
blue           2000.0     0.49500     0.500100    0.0     1.0
clock_speed    2000.0     1.52225     0.816004    0.5     3.0
dual_sim       2000.0     0.50950     0.500035    0.0     1.0
fc             2000.0     4.30950     4.341444    0.0    19.0
four_g         2000.0     0.52150     0.499662    0.0     1.0
int_memory     2000.0    32.04650    18.145715    2.0    64.0
m_dep          2000.0     0.50175     0.288416    0.1     1.0
mobile_wt      2000.0   140.24900    35.399655   80.0   200.0
n_cores        2000.0     4.52050     2.287837    1.0     8.0
pc             2000.0     9.91650     6.064315    0.0    20.0
px_height      2000.0   645.10800   443.780811    0.0  1960.0
px_width       2000.0  1251.51550   432.199447  500.0  1998.0
ram            2000.0  2124.21300  1084.732044  256.0  3998.0
sc_h           2000.0    12.30650     4.213245    5.0    19.0
sc_w           2000.0     5.76700     4.356398    0.0    18.0
talk_time      2000.0    11.01100     5.463955    2.0    20.0
three_g        2000.0     0.76150     0.426273    0.0    1.0
touch_screen   2000.0     0.50300     0.500116    0.0    1.0
wifi           2000.0     0.50700     0.500076    0.0    1.0
price_range    2000.0     1.50000     1.118314    0.0    3.0
```

Then, to predict the relationship between the variables, we sketch the correlation matrix. Correlation matrix simply tells you how pairs of variables are linearly related. It is useful to summarize a large data set and to identify the patterns in the given data. To plot the correlation matrix we make use of seaborn.heatmap() function. See,

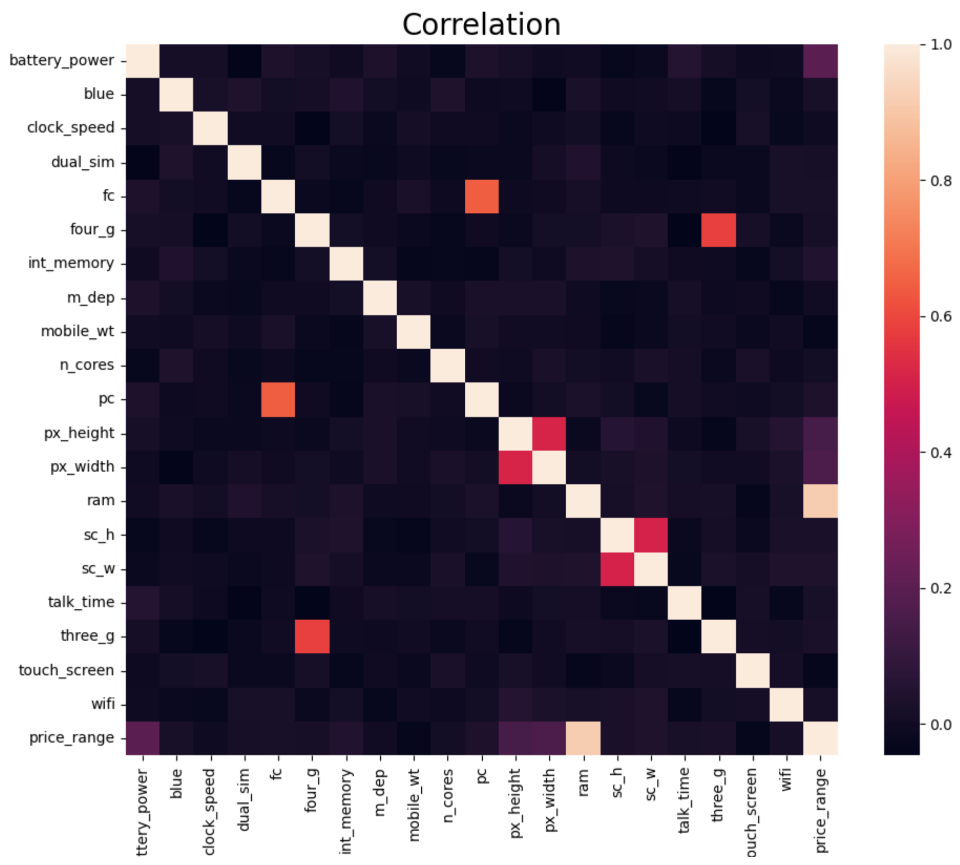
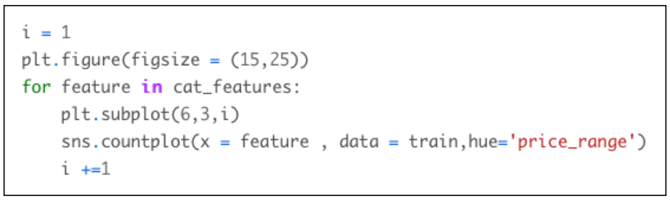


Fig. 1: Correlation matrix to identify the patterns in the given data.

Then, we determine how many "cheap", "standard", "expensive", "very expensive" phones are there with each discrete value of a parameter. We run the code below,



```
i = 1
plt.figure(figsize = (15,25))
for feature in cat_features:
    plt.subplot(6,3,i)
    sns.countplot(x = feature , data = train,hue='price_range')
    i +=1
```

Fig. 2: Python code to obtain the bar charts on the next page.

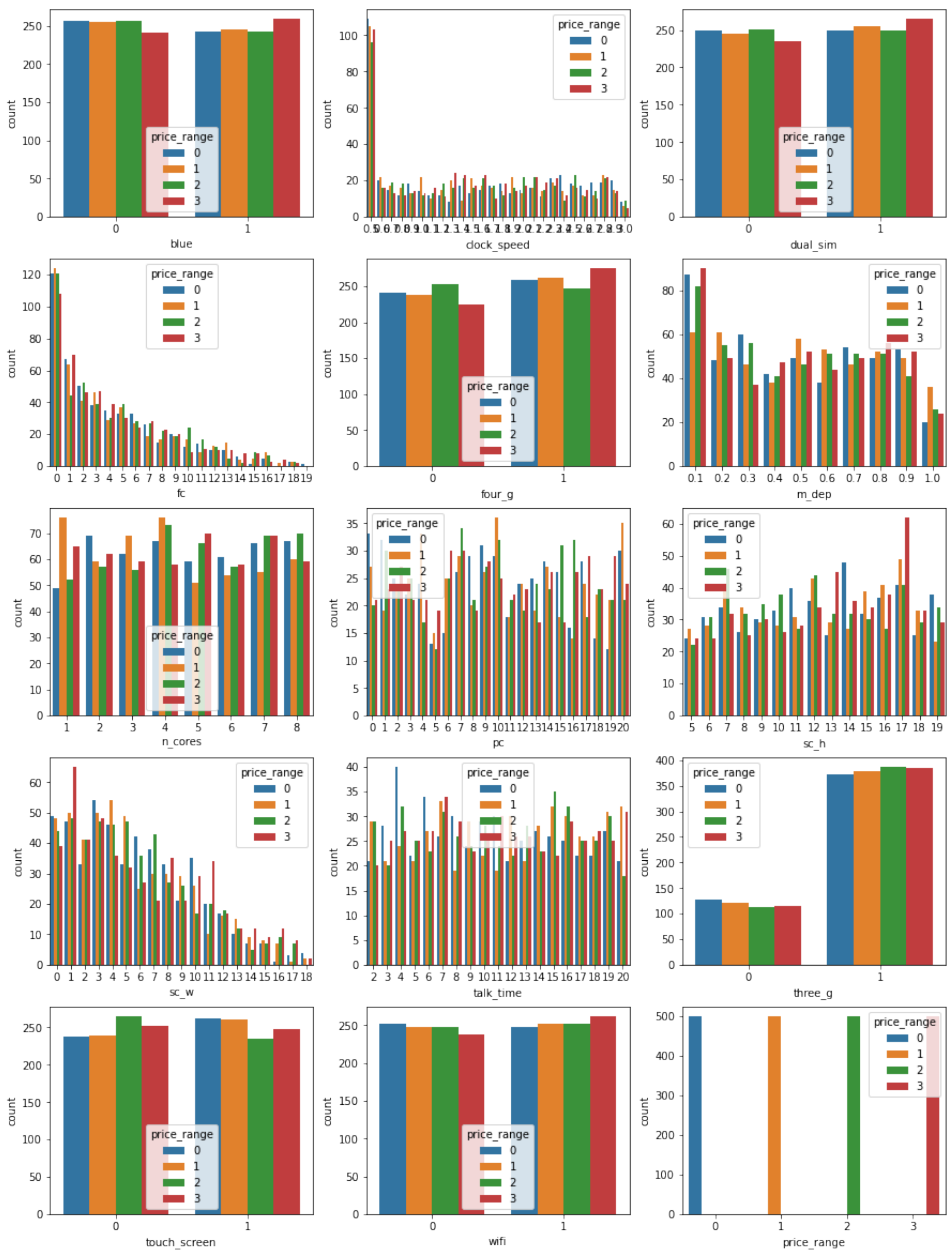


Fig. 3: Bar charts to determine the count of phones for each feature value.

For example, in above figure we can observe that around 400 ”very expensive” phones, painted by red, have 3G feature and around 120 ”very expensive” phones does not have 3G feature. We see around 250 ”cheap” phones, painted by blue, does not have WiFi feature.

Now, let's try to determine how much battery power and RAM size of a phone can effect price category of a phone. To do that we need to plot a chart where only battery power and RAM sizes are shown. Phones will be represented by colored points in the coordinates. At this point, we make use of seaborn.FacetGrid() and seaborn.map() functions. See,

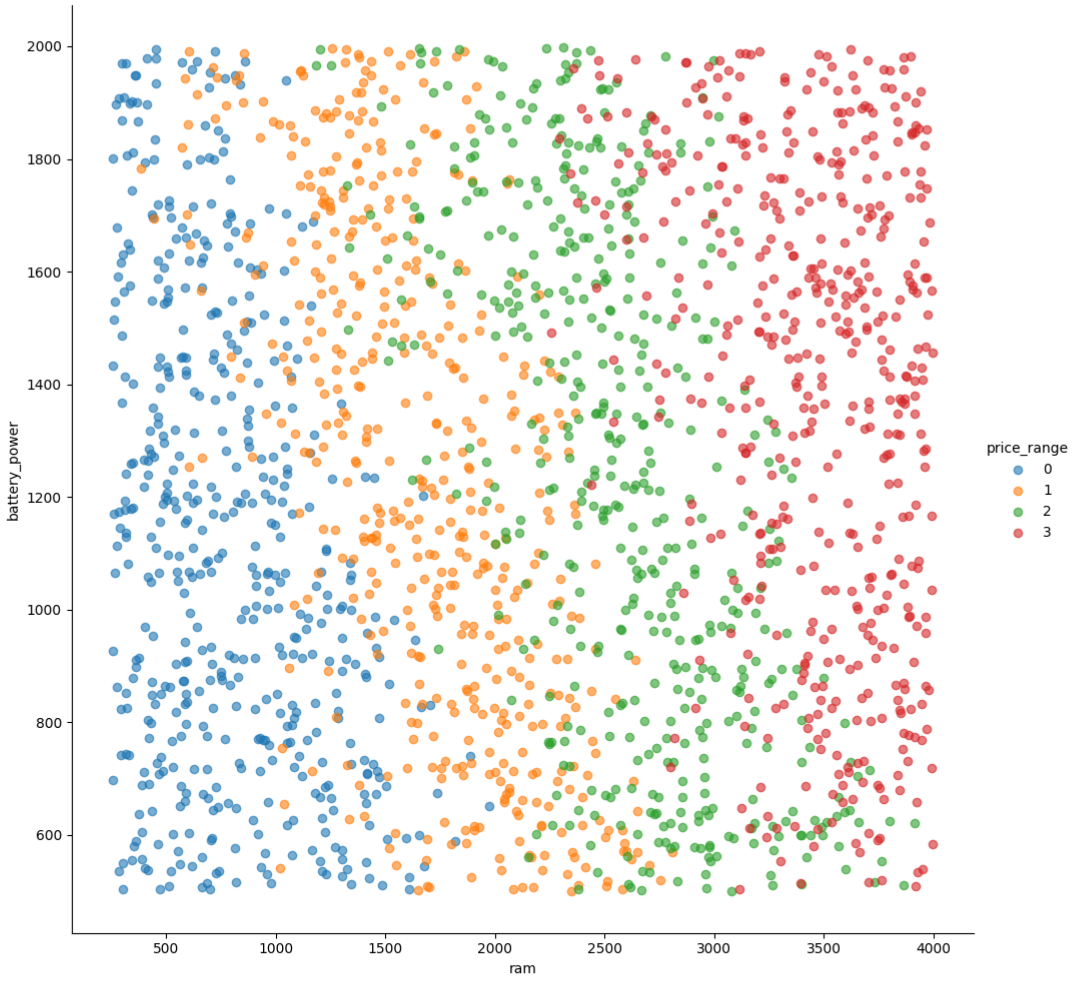


Fig. 4: Phones with varying RAM sizes (GB) and Battery Powers (mAh).

By looking at the graph, we observe that as the RAM size and Battery Power increases the phones are falling into expensive categories. The phones with 3 to 4 GB of RAM and 1500 to 2000 mAh battery power falls in to ”very expensive” category.

IV. SOLUTION METHODS

In this section we compare the classification algorithms with each other. At the end, we would have accuracty rate for each classification algorithm on the test data. We then look at these accuracy rates and decide on the performace of the algorithms. On the otherhand, there is another metric that is commonly used to measure performance of classification algorithms: Confusion matrices.

Confusion matrices is an NxN matrix used for evaluating the performance of a classification model, where N is the number of target classes. In our case, N equals to 4. In this matrix, it can be easly seen whether the model is confusing two classes. For example, a confusion matrix with main diagonal cells being are extremely large means, classification is done very well.

Before we start running the classification algorithms, let's discuss a very important issue. Most of the classification methods are in fact binary classifiers. However, in our problem, we may have 4 different outputs 0,1,2,3 i.e., ”cheap”, ”standard”, ”expensive”, ”very expensive”. So, will it be a problem?

No, it won't. In fact, most of the Python libraries also handle multiple outcomes. We do not see, but on the background, the classification algorithms are run for each possible outcome. That way, the entries falling each category is determined. More precisely, One-vs-Rest or One-vs-All method is used for binary classification algorithms for multi-class classification. It involves splitting the multi-class data set into multiple binary classification problems. These are mainly used for Perceptrons, Logistic Regressions and Support Vector Machines.

A. Logistic Regression

By considering independent variables, a probabilistic measure is used to classify the entries. We run the algorithm as follows,

```
lr = LogisticRegression(random_state = 42, max_iter=100000)
lr.fit(X_train, y_train)
y_pred_lr = lr.predict(X_test)
accuracy = metrics.accuracy_score(y_test, y_pred_lr)
```

The resulting confusion matrix for Logistic Regression are shared below.

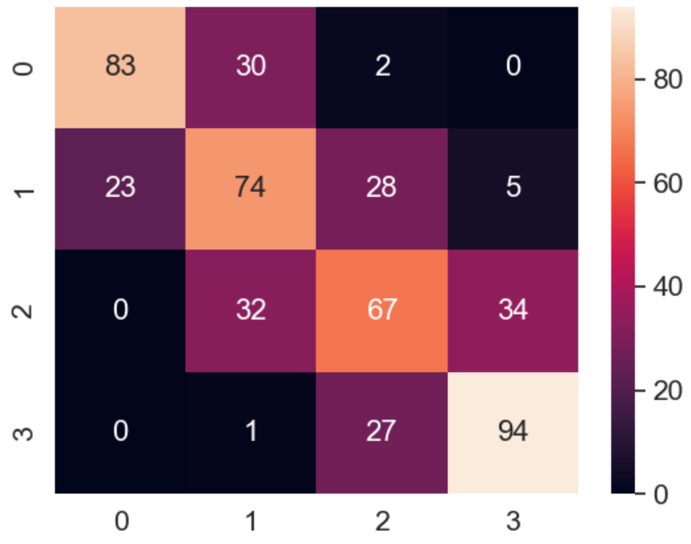


Fig. 5: Confusion matrix for Logistic Regression.

B. SVM

The main idea is to find the line or the hyperplane where the margin is maximum. The whole idea is intiuiative and performs very well. We run the algorithm as follows,

```
svc = SVC()
svc.fit(X_train, y_train)
y_pred_svc = svc.predict(X_test)
accuracy_svc = metrics.accuracy_score(y_test, y_pred_svc)
```

The resulting confusion matrix for SVM are shared below.

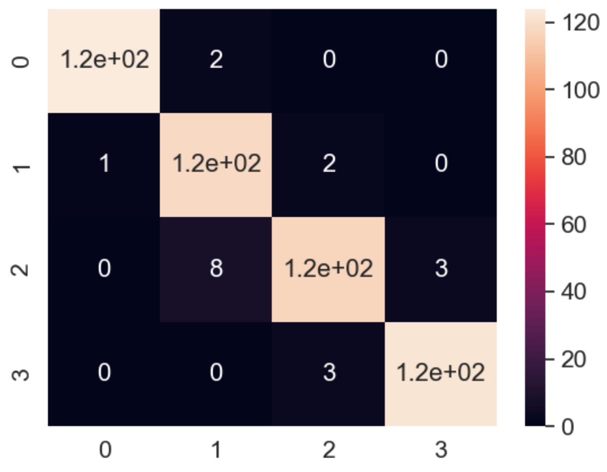


Fig. 6: Confusion matrix for SVM.

C. Decision Tree

Here, a flowchart like structure is generated and branches out with potential outcomes of each decision. Each entry tested the tree and classified. We run the algorithm as follows,

```
first_tree = DecisionTreeClassifier()
first_tree.fit(X_train, y_train)
y_pred=first_tree.predict(X_test)
accuracy = metrics.accuracy_score(y_test,y_pred)
```

The resulting accuracy rate and confusion matrix for Decision Tree are shared below.

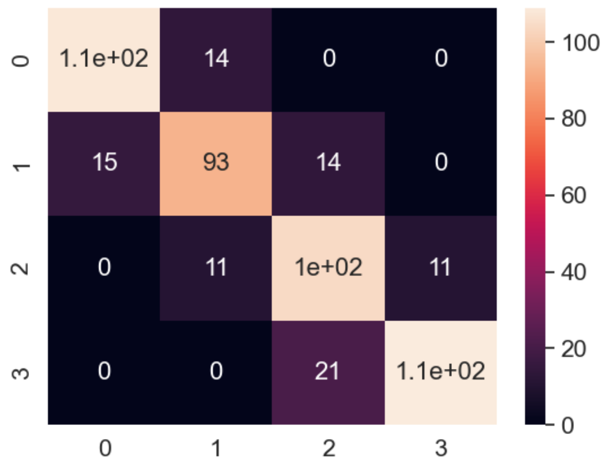


Fig. 7: Confusion matrix for Decision Tree.

D. Random Forest

The random forest is a classification algorithm consisting of many decision trees. Trees are built with randomness and prediction is done by committee of trees instead of an individual tree. We run the algorithm as follows,

```
rc = RandomForestClassifier(random_state=42)
rc.fit(X_train,y_train)
y_pred_rc = rc.predict(X_test)
accuracy = metrics.accuracy_score(y_test, y_pred_rc)
```

The resulting confusion matrix for Random Forest are shared below.

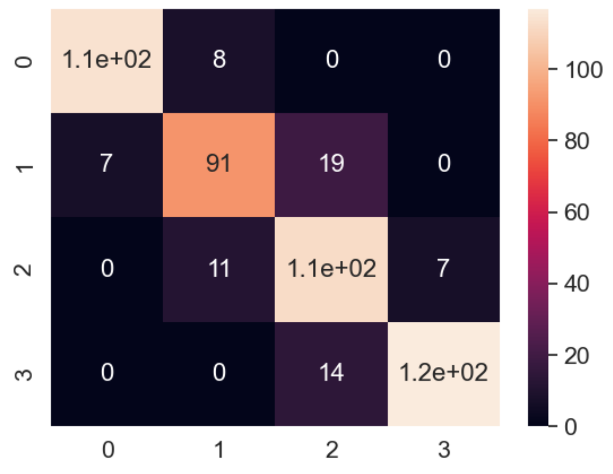


Fig. 8: Confusion matrix for Random Forest.

E. K-neighbors

K-nearest neighbors (KNN) tries to predict the correct class for the test data by calculating the distance between the test data and all the training points. Then select the K number of points which is closet to the test data. We run algorithm as follows,

```
model = KNeighborsClassifier()
model.fit(X_train, y_train)
predicted= model.predict(X_test)
accuracy = metrics.accuracy_score(y_test,predicted)
```

The resulting confusion matrix for K-neighbors are shared below.

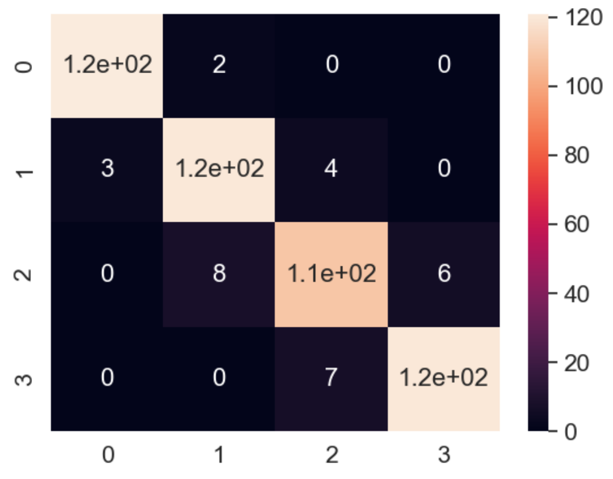


Fig. 9: Confusion matrix for K-neighbors.

V. EXPRIMENTAL RESULTS AND ANALYSIS

So far, we have applied various algorithms and observed the corresponding accuracy rates. Below, we share accuracy rates for algorithms used.

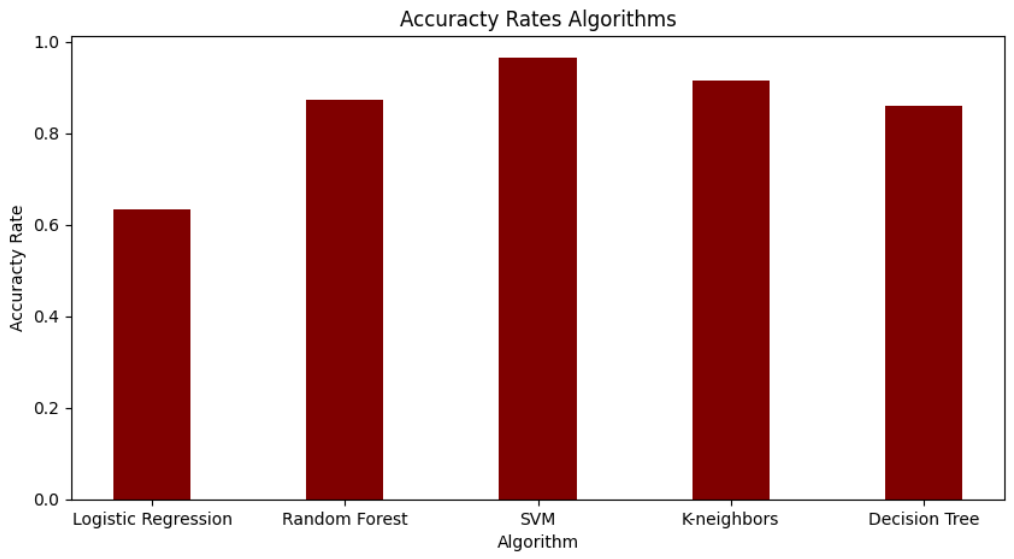


Fig. 10: Accuracy rates for all classification algorithms used.

We see SVM has the highest accuracy rate and performed slightly better than the other algorithms. In fact, it is not a coincide. SVM works relatively well when there is a clear margin of separation between classes. The train data contains very well separated data entries so SVM peformed well on these. Another thing is that SVMs really good at seperating high dimensionality data. In our problem, we gave 21 parameters even though the entire data contains only 2000 entries.

Logistic Regression has relatively small accuracy rate compared to others. The reason is that there is a defined maximum number of iterations for the model. By default, it is set to 1000. During the run of the program we observed error:

```
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
Increase the number of iterations (max_iter) or scale the data
as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver
options:
    https://scikit-learn.org/stable/modules/linear_model.
    html#logistic-regression
  n_iter_i = _check_optimize_result(
```

So, Linear Regression was not able to converge in 1000 iterations. We set iteration count to 1 million but there was extremely a small change in the accuracy. The accuracy was now 0.752. Increasing iterations to millions would only make slightly changes to the accuracy and still the algorithm was not able to converge. We guess that if enough iteration count is passed accuracy rate would probably reach around 0.79 which is again way below than the other algorithms.

VI. APPENDIX

All the tables and plots listed in this report are created by simple plotting codes in Python. Matplotlib and Seaborn offers many visualization tools in Python. We make great use of these libraries when creating the tables and plots.

We also would like to express our most profound appreciation to Abhishek Sharm who is the author of the problem and have written problem description in detail.

REFERENCES

[1] A. Sharma, ”Mobile Price Classification, Kaggle,” Kaggle, 2018. [Online]. Available: https://www.kaggle.com/datasets/iabhishekofficial/mobile-price-classification. [Accessed 2 November 2022].

[2] 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), 2017.

[3] C. Mair et al., ”An investigation of machine learning based prediction systems”, J. Syst.Softw., 2000.

[4] N. Hu, ”Classification of Mobile Phone Price Dataset Using Machine Learning Algorithms,” 2022 3rd International Conference on Pattern Recognition and Machine Learning (PRML), 2022, pp. 438-443.

[5] M. Çetın and Y. Koç, ”Mobile Phone Price Class Prediction Using Different Classification Algorithms with Feature Selection and Parameter Optimization,” 2021 5th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT), 2021, pp. 483-487.

[6] Harshitha, B., Maria Rufina, P. & Shilpa, B.L. Comparison of Different Classification Algorithms for Prediction of Heart Disease by Machine Learning Techniques. SN COMPUT. SCI. 4, 128 (2023).

[7] Y. S. Abu-Mostafa, M. Magdon-Ismail, and H.-T. Lin, Learning from data : a short course. Pasadena, Ca: Aml Book, 2012.