BIRLA INSTITUTE OF TECHNOLOGY

MESRA, RANCHI 835215

CA590 – Massive Open Online Course

Project on:

Mobile Price Classification using Machine Learning

Submitted to:

RASHMI RATHI UPADHAYAY

Submitted by:

NAME: AIMAN EQBAL

ROLL NO.: MCA/10038/19

Index

Abstract

Mobile Price Classification categorizes or classifies different mobile into similar price categories by finding relations between features of a mobile phone (E.g.: - RAM, Internal Memory etc).

This idea of this project is based on a mobile website that helps you find your phone based on your budget. However, with this project I am to categorise the mobile phones in sub categories of “Budget”, “Mid-Range”, “Premium” and “Ultra-Premium” price segment.

In this project I have used 3 different models to classify the price and then used the most accurate model for the final prediction.

The data for training and testing has been taken from Kaggle. The model takes a set of 20 parameters then according to those parameters it predicts the in which category should the mobile price fall, 0 being budget and 3 being ultra-premium.

Advantage: This project can help in small business start-ups who struggle to enter the market with a range of new products.

Disadvantage: The prediction model might fail as the new companies are now forcing aggressive prices on mobile phones meaning they cost less but have tons of high-end specifications.

Introduction

Mobile Price Classification can categories mobile phone based on specs from a range of 0 to 3, 0 being the budget category and 3 being the ultra-premium category.

This project is majorly divided into three parts:

- Data Analysis
- Model Building
- Prediction

The dataset for this project is split into 80:20 ratio for training and testing purposes before model building phase and in the 3rd phase the whole test data set is predicted.

Data Analysis

In the Data Analysis, the first phase of the project the input data is taken and various tasks like data pre-processing and feature extractions are performed.

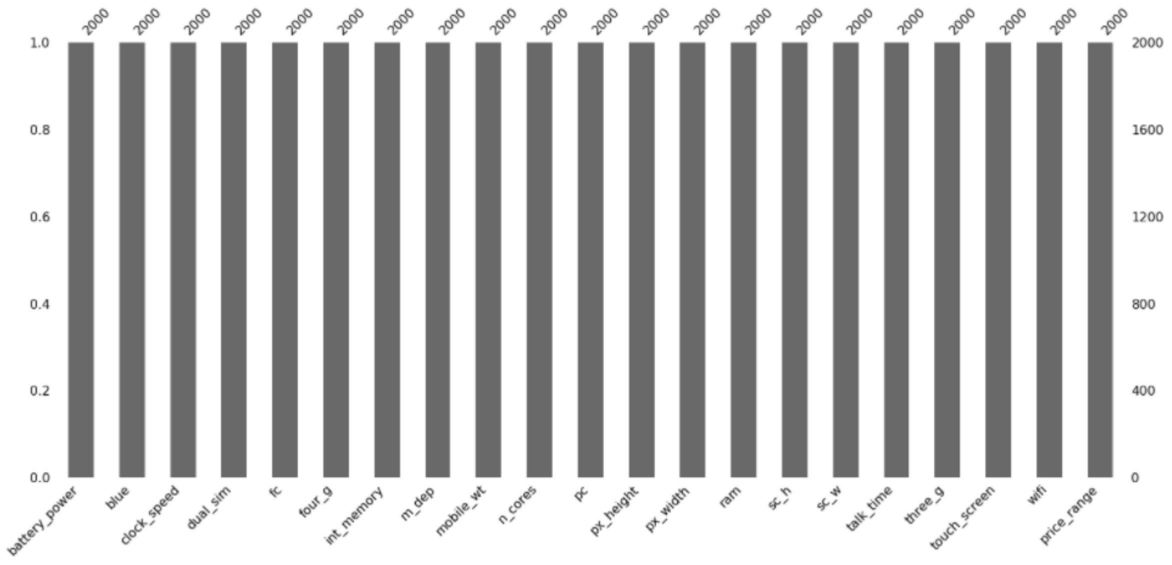
In this phase we create a data dictionary based on features and check for any missing in the dataset.

Data Dictionary with extracted features:

1. **battery_power** = Total energy a battery can store in one time measured in mAh.
2. **blue** = Has bluetooth or not. | 1: Has, 0: doesn't have
3. **clock_speed** = Speed at which microprocessor executes instructions.
4. **dual_sim** = Has dual sim support or not | 1: support, 0: doesn't support
5. **fc** = Front Camera mega pixels.
6. **four_g** = Has 4G or not. | 1: Has, 0: doesn't have
7. **int_memory** = Internal memory in gigabytes.
8. **m_dep** = Mobile Depth in cm.
9. **mobile_wt** = Weight of mobile phone.
10. **n_cores** = Number of cores of 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.
18. **three_g** = Has 3G or not. | 1: Has, 0: Doesn't have
19. **touch_screen** = Has touch screen or not. | 1: Has, 0: Doesn't have

20. **wifi** = Has wifi or not. | 1: Has, 0: Doesn’t have
21. **price_range** = This is the target variable. | 3: Very High Cost, 2: High Cost, 1: Medium Cost, 0: Low Cost

Missing values is graphically plotted as:



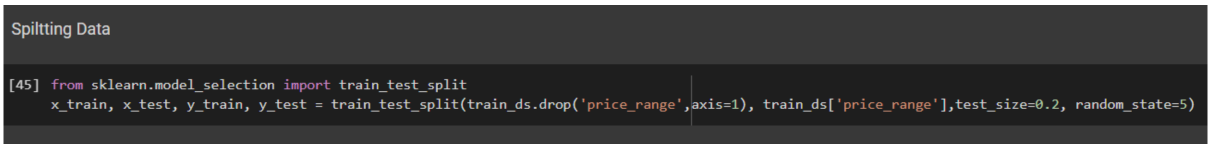
Since, there were no missing values any the data set, we move to the next phase of the project, i.e., Model Building.

Model Building

I have used 3 different algorithmic models in this project.

1. Logistic Regression
2. KNN – K Nearest Neighbor
3. SVM – Support Vector Machines

The training dataset is spilt and is fitted into all the above-mentioned models and then the model with most accuracy is selected to predict the range for the test dataset.


Spiltting Data

```
[45] from sklearn.model_selection import train_test_split
     x_train, x_test, y_train, y_test = train_test_split(train_ds.drop('price_range',axis=1), train_ds['price_range'],test_size=0.2, random_state=5)
```

After each model training a confusion matrix was generated to classify the test data.

1. Logistic Regression

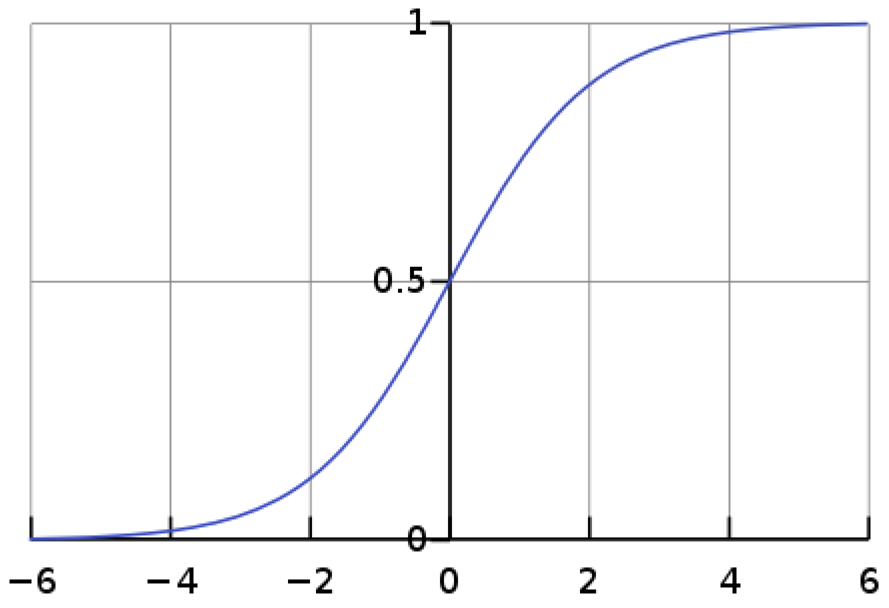
Logistic regression, despite its name, is a linear model for classification rather than regression. Logistic regression is also known in the literature as logit regression, maximum-entropy classification (MaxEnt) or the log-linear classifier. In this model, the probabilities describing the possible outcomes of a single trial are modelled using a logistic function.

The logistic function is a sigmoid function, which takes any real input, and **outputs a value between zero and one**.

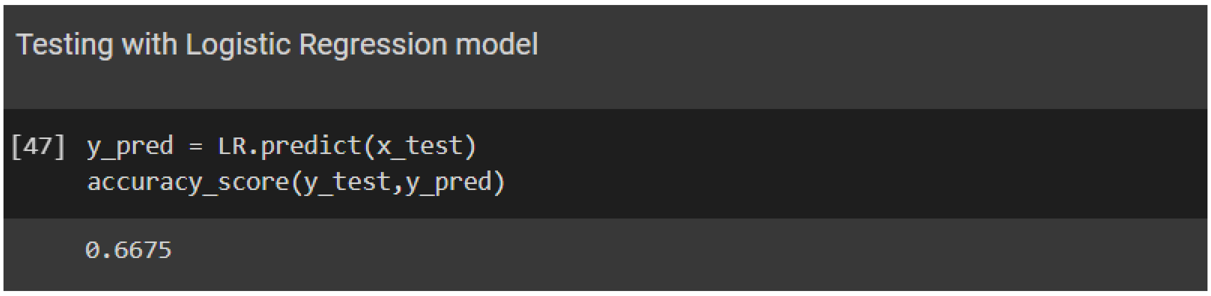
The *standard* logistic function $p : \mathbb{R} \to (0,1)$ is defined as follows:

$$\sigma(t) = \frac{e^t}{e^t + 1} = \frac{1}{1 + e^{-t}}$$

A graph of the logistic function on the *t*-interval is shown below:



The standard logistic function σ(*t*); σ(*t*) ϵ (0,1) for all *t*



Testing with Logistic Regression model

```
[47] y_pred = LR.predict(x_test)
     accuracy_score(y_test,y_pred)

     0.6675
```

However, after applying this logistic regression to our training data and after testing, we get an accuracy 66%.

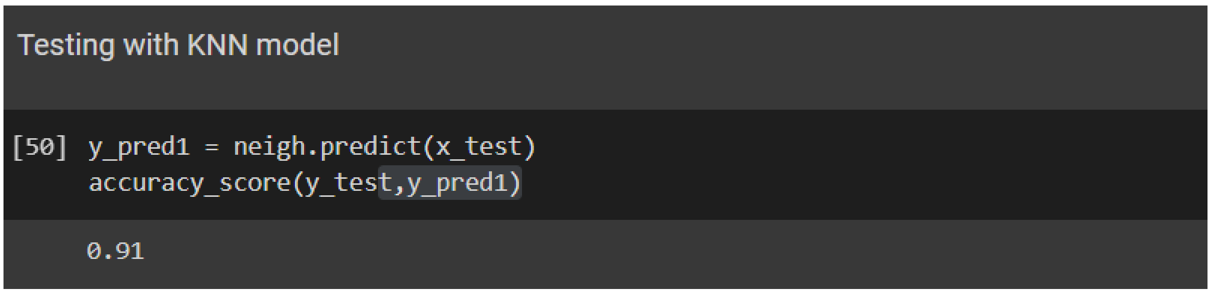
2. K Nearest Neighbors

The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other. KNN can be used for classification as well as regression. For this model we are using classification.

In k-NN classification, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor.

Algorithm:

i. Load the data
ii. Initialize K to your chosen number of neighbors
iii. For each example in the data
   - Calculate the distance between the query example and the current example from the data.
   - Add the distance and the index of the example to an ordered collection
iv. Sort the ordered collection of distances and indices from smallest to largest (in ascending order) by the distances
v. Pick the first K entries from the sorted collection
vi. Get the labels of the selected K entries
vii. If classification, return the mode of the K labels


Testing with KNN model

```
[50] y_pred1 = neigh.predict(x_test)
     accuracy_score(y_test,y_pred1)

     0.91
```

After applying this KNN classification to our training data and after testing, we get an accuracy of 91% which is much higher than the previous model.

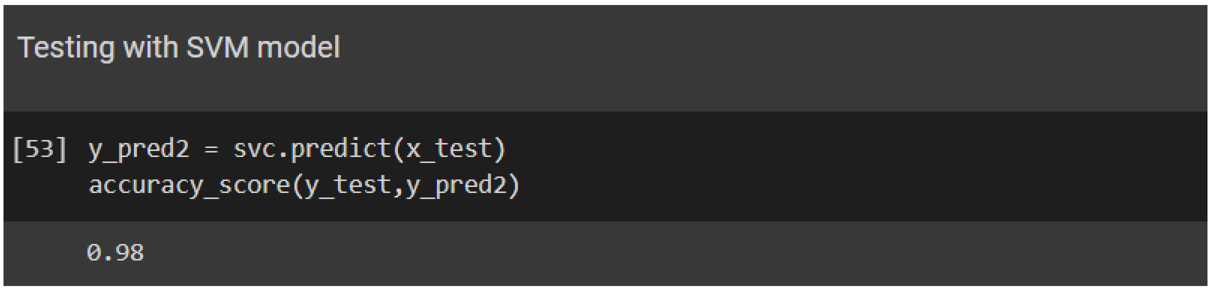
3. Support Vector Machines

Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outliers detection.

The advantages of support vector machines are:

i. Effective in high dimensional spaces.
ii. Still effective in cases where number of dimensions is greater than the number of samples.
iii. Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
iv. Versatile: different Kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.

The kernel used in this model was linear kernel.


Testing with SVM model

```
[53] y_pred2 = svc.predict(x_test)
     accuracy_score(y_test,y_pred2)

     0.98
```

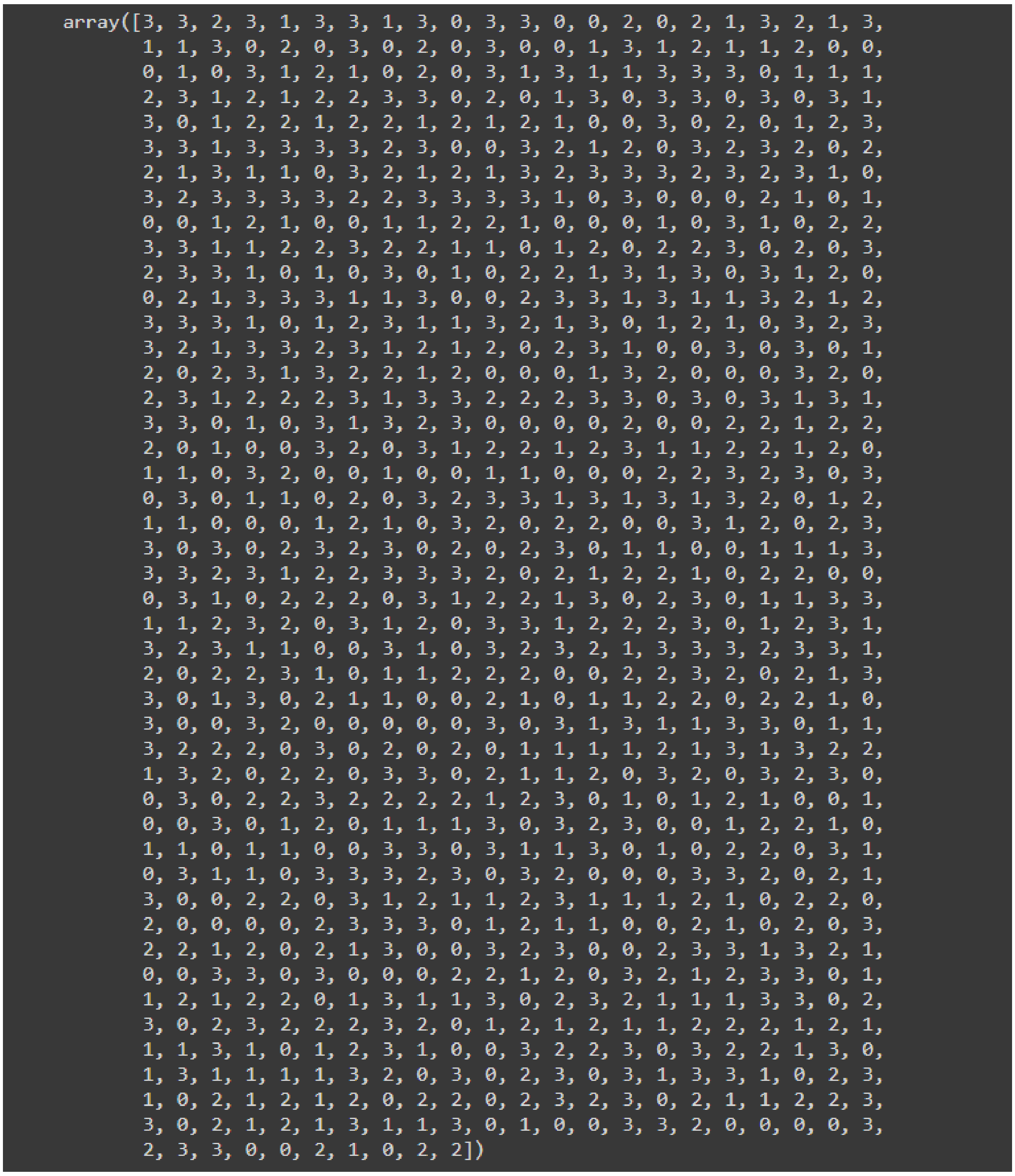
After applying this SVC classification to our training data and after testing, we get an accuracy of 98% which the best accuracy overall.

Prediction

Finally, after comparing all the model's accuracy, SVM's accuracy turns out to be the best.

So now the whole test dataset is taken and the classification is done using the SVM model.

The predicted result came out as:



```
array([3, 3, 2, 3, 1, 3, 3, 1, 3, 0, 3, 3, 0, 0, 2, 0, 2, 1, 3, 2, 1, 3,
       1, 1, 3, 0, 2, 0, 3, 0, 2, 0, 3, 0, 0, 1, 3, 1, 2, 1, 1, 2, 0, 0,
       0, 1, 0, 3, 1, 2, 1, 0, 2, 0, 3, 1, 3, 1, 1, 3, 3, 3, 0, 1, 1, 1,
       2, 3, 1, 2, 1, 2, 2, 3, 3, 0, 2, 0, 1, 3, 0, 3, 3, 0, 3, 0, 3, 1,
       3, 0, 1, 2, 2, 1, 2, 2, 1, 2, 1, 2, 1, 0, 0, 3, 0, 2, 0, 1, 2, 3,
       3, 3, 1, 3, 3, 3, 3, 2, 3, 0, 0, 3, 2, 1, 2, 0, 3, 2, 3, 2, 0, 2,
       2, 1, 3, 1, 1, 0, 3, 2, 1, 2, 1, 3, 2, 3, 3, 3, 2, 3, 2, 3, 1, 0,
       3, 2, 3, 3, 3, 3, 2, 2, 3, 3, 3, 3, 1, 0, 3, 0, 0, 0, 2, 1, 0, 1,
       0, 0, 1, 2, 1, 0, 0, 1, 1, 2, 2, 1, 0, 0, 0, 1, 0, 3, 1, 0, 2, 2,
       3, 3, 1, 1, 2, 2, 3, 2, 2, 1, 1, 0, 1, 2, 0, 2, 2, 3, 0, 2, 0, 3,
       2, 3, 3, 1, 0, 1, 0, 3, 0, 1, 0, 2, 2, 1, 3, 1, 3, 0, 3, 1, 2, 0,
       0, 2, 1, 3, 3, 3, 1, 1, 3, 0, 0, 2, 3, 3, 1, 3, 1, 1, 3, 2, 1, 2,
       3, 3, 3, 1, 0, 1, 2, 3, 1, 1, 3, 2, 1, 3, 0, 1, 2, 1, 0, 3, 2, 3,
       3, 2, 1, 3, 3, 2, 3, 1, 2, 1, 2, 0, 2, 3, 1, 0, 0, 3, 0, 3, 0, 1,
       2, 0, 2, 3, 1, 3, 2, 2, 1, 2, 0, 0, 0, 1, 3, 2, 0, 0, 0, 3, 2, 0,
       2, 3, 1, 2, 2, 2, 3, 1, 3, 3, 2, 2, 2, 3, 3, 0, 3, 0, 3, 1, 3, 1,
       3, 3, 0, 1, 0, 3, 1, 3, 2, 3, 0, 0, 0, 0, 2, 0, 0, 2, 2, 1, 2, 2,
       2, 0, 1, 0, 0, 3, 2, 0, 3, 1, 2, 2, 1, 2, 3, 1, 1, 2, 2, 1, 2, 0,
       1, 1, 0, 3, 2, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 2, 2, 3, 2, 3, 0, 3,
       0, 3, 0, 1, 1, 0, 2, 0, 3, 2, 3, 3, 1, 3, 1, 3, 1, 3, 2, 0, 1, 2,
       1, 1, 0, 0, 0, 1, 2, 1, 0, 3, 2, 0, 2, 2, 0, 0, 3, 1, 2, 0, 2, 3,
       3, 0, 3, 0, 2, 3, 2, 3, 0, 2, 0, 2, 3, 0, 1, 1, 0, 0, 1, 1, 1, 3,
       3, 3, 2, 3, 1, 2, 2, 3, 3, 3, 2, 0, 2, 1, 2, 2, 1, 0, 2, 2, 0, 0,
       0, 3, 1, 0, 2, 2, 2, 0, 3, 1, 2, 2, 1, 3, 0, 2, 3, 0, 1, 1, 3, 3,
       1, 1, 2, 3, 2, 0, 3, 1, 2, 0, 3, 3, 1, 2, 2, 2, 3, 0, 1, 2, 3, 1,
       3, 2, 3, 1, 1, 0, 0, 3, 1, 0, 3, 2, 3, 2, 1, 3, 3, 3, 2, 3, 3, 1,
       2, 0, 2, 2, 3, 1, 0, 1, 1, 2, 2, 2, 0, 0, 2, 2, 3, 2, 0, 2, 1, 3,
       3, 0, 1, 3, 0, 2, 1, 1, 0, 0, 2, 1, 0, 1, 1, 2, 2, 0, 2, 2, 1, 0,
       3, 0, 0, 3, 2, 0, 0, 0, 0, 0, 3, 0, 3, 1, 3, 1, 1, 3, 3, 0, 1, 1,
       3, 2, 2, 2, 0, 3, 0, 2, 0, 2, 0, 1, 1, 1, 1, 2, 1, 3, 1, 3, 2, 2,
       1, 3, 2, 0, 2, 2, 0, 3, 3, 0, 2, 1, 1, 2, 0, 3, 2, 0, 3, 2, 3, 0,
       0, 3, 0, 2, 2, 3, 2, 2, 2, 2, 1, 2, 3, 0, 1, 0, 1, 2, 1, 0, 0, 1,
       0, 0, 3, 0, 1, 2, 0, 1, 1, 1, 3, 0, 3, 2, 3, 0, 0, 1, 2, 2, 1, 0,
       1, 1, 0, 1, 1, 0, 0, 3, 3, 0, 3, 1, 1, 3, 0, 1, 0, 2, 2, 0, 3, 1,
       0, 3, 1, 1, 0, 3, 3, 3, 2, 3, 0, 3, 2, 0, 0, 0, 3, 3, 2, 0, 2, 1,
       3, 0, 0, 2, 2, 0, 3, 1, 2, 1, 1, 2, 3, 1, 1, 1, 2, 1, 0, 2, 2, 0,
       2, 0, 0, 0, 0, 2, 3, 3, 3, 0, 1, 2, 1, 1, 0, 0, 2, 1, 0, 2, 0, 3,
       2, 2, 1, 2, 0, 2, 1, 3, 0, 0, 3, 2, 3, 0, 0, 2, 3, 3, 1, 3, 2, 1,
       0, 0, 3, 3, 0, 3, 0, 0, 0, 2, 2, 1, 2, 0, 3, 2, 1, 2, 3, 3, 0, 1,
       1, 2, 1, 2, 2, 0, 1, 3, 1, 1, 3, 0, 2, 3, 2, 1, 1, 1, 3, 3, 0, 2,
       3, 0, 2, 3, 2, 2, 2, 3, 2, 0, 1, 2, 1, 2, 1, 1, 2, 2, 2, 1, 2, 1,
       1, 1, 3, 1, 0, 1, 2, 3, 1, 0, 0, 3, 2, 2, 3, 0, 3, 2, 2, 1, 3, 0,
       1, 3, 1, 1, 1, 1, 3, 2, 0, 3, 0, 2, 3, 0, 3, 1, 3, 3, 1, 0, 2, 3,
       1, 0, 2, 1, 2, 1, 2, 0, 2, 2, 0, 2, 3, 2, 3, 0, 2, 1, 1, 2, 2, 3,
       3, 0, 2, 1, 2, 1, 3, 1, 1, 3, 0, 1, 0, 0, 3, 3, 2, 0, 0, 0, 0, 3,
       2, 3, 3, 0, 0, 2, 1, 0, 2, 2])
```

Data Description

There are two input files:

- Input.csv
    - It contains almost 2000 entries of data.
    - This dataset has 20 feature columns
    - This dataset is split into a ratio of 80:20
    - The 80% of this data set is dedicated to model training.
    - The rest 20% of the data is used to testing the accuracy of the trained model.
- Test.csv
    - This dataset contains 1000 values.
    - The feature of this dataset is similar to the input data set, but it has only "price_range" feature column.
    - The best accurate model, here SVM, is used to predict the "price_range" feature for this dataset.

Tools Used

The following tools are used in creation of this project:

Programming language used:

1. Python: This project if fully made using python programming lanuage
2. Python libraries used

    • Pandas: It is used to read from the data values from the dataset or input files

    • Numpy: It provides the basic data structure for the entire project. It is used to store and pre-process data.

    • Scikit learn: It provides the necessary tools needed to construct the models used in this project. Along with the construction tools it also provides tools for training and testing the data.

    • Matplotlib: It is used for drawing the bar graph that shows if there are any missing values in the dataset.

Platform used for coding: Google Colab

References

NPTL - Machine Learning, ML by Prof. Carl Gustaf Jansson KTH, The Royal Institute of Technology

Udemy - https://www.udemy.com/share/102vAMCUEaeV5aRn4=/

Dataset - https://www.kaggle.com/iabhishekofficial/mobile-price-classification

Scikit Learn

- Logistic Regression – https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
- SVM – https://scikit-learn.org/stable/modules/svm.html
- KNN – https://scikit-learn.org/stable/modules/neighbors.html#nearest-neighbors-classification