Mobile Price Range Prediction Using Pytorch and Plotly

Abstract
Predicting the mobile price range based on specification is a major problem for companies. The purpose of this report is to estimate the price of new phones. In such a competitive environment, we cannot simply assume things. The focus of this report is to analyse relation between different features of the phone like RAM size, clock speed, Mobile weights etc and accordingly predict the mobile price range. The dataset used for this research is taken from various sales data and it can be downloaded from Kaggle (Link shown in Reference Section). Pytorch is used to create a model that can help in predicting the price range of new mobile phones launched by a company. Plotly is used to plot graphs which provides various insights into the data. Initially, a brief introduction of the dataset is provided then some of the functions of Plotly and Pytorch that are used in the analysis are introduced. Finally, some insights into dataset are given using Plotly and the conclusion that we can derive from them.

Mobile price range prediction

Before starting with the analysis, it is important to understand the structure of the dataset. The dataset consists of 21 columns which are described in the below table.

Parameter Name	Description	Parameter Name	Description
id	A unique Id of mobile phone.	battery_power	Total energy a battery can store in one time measured in mAh.
blue	Has bluetooth or not.	clock_speed	Speed at which instruction are executed.
dual_sim	Has dual sim support or not.	fc	Front Camera megapixels.
four_g	Has 4G or not.	int_memory	Internal Memory in Gigabytes.
m_dep	Mobile Depth in cm.	mobile_wt	Weight of mobile phone.
n_cores	Number of cores of processor.	pc	Primary Camera in megapixels.
px_height	Pixel Resolution Height.	px_width	Pixel Resolution Width.
ram	Random Access Memory in Megabytes	sc_h	Screen Width of mobile in cm.

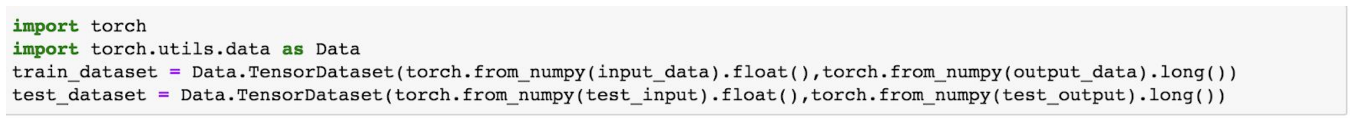
Parameter Name	Description	Parameter Name	Description
sc_w	Screen Width of mobile in cm	talk_time	longest time that a single battery charge.
three_g	Has 3G or not	touch_screen	Has touch screen or not
wifi	Has wifi or not	price_range	Price range of the given mobile.

Pytorch

PyTorch is an open-source machine learning library for Python, based on Torch, used for applications such as natural language processing. PyTorch provides two high level features.

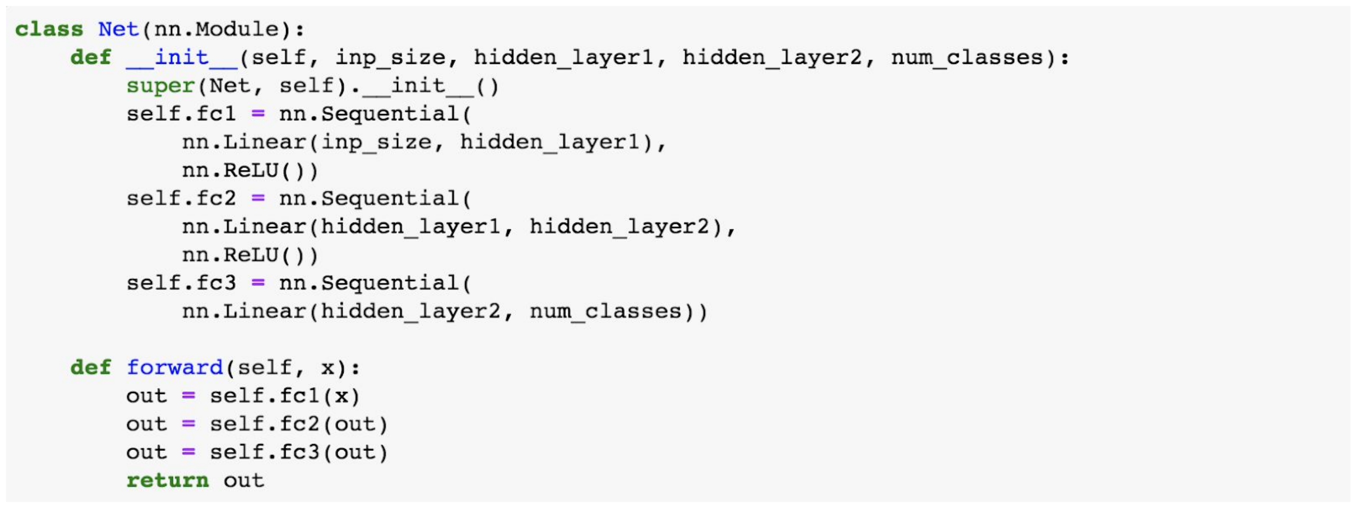
- Tensor computation (like NumPy) with strong GPU acceleration
- Deep Neural Networks built on a tape-based autodiff system

To begin with Pytorch it is important to understand Pytorch Tensors. In terms of programming, Tensors can simply be considered multidimensional arrays. Tensors in PyTorch are similar to NumPy arrays, with the addition being that Tensors can also be used on a GPU that supports CUDA. The code to convert numpy array to Tensor is shown in the following snapshot:-



```python
import torch
import torch.utils.data as Data
train_dataset = Data.TensorDataset(torch.from_numpy(input_data).float(),torch.from_numpy(output_data).long())
test_dataset = Data.TensorDataset(torch.from_numpy(test_input).float(),torch.from_numpy(test_output).long())
```

The next step is to define a class which will contain layers of the neural network and the forward function which will help in providing output of the current layer to the next layer. The following snapshot shows three layers that were created for our research:-



```python
class Net(nn.Module):
    def __init__(self, inp_size, hidden_layer1, hidden_layer2, num_classes):
        super(Net, self).__init__()
        self.fc1 = nn.Sequential(
            nn.Linear(inp_size, hidden_layer1),
            nn.ReLU())
        self.fc2 = nn.Sequential(
            nn.Linear(hidden_layer1, hidden_layer2),
            nn.ReLU())
        self.fc3 = nn.Sequential(
            nn.Linear(hidden_layer2, num_classes))

    def forward(self, x):
        out = self.fc1(x)
        out = self.fc2(out)
        out = self.fc3(out)
        return out
```

Each layer has its own input and output size. The activation function that will be applied to each layer is also defined. For our dataset, **RELU** is used in the first two layers and the final layer will provide the predicted price range.

The next step is to define the optimizer and the loss function which is shown in the below snapshot:-

```
# Loss and Optimizer
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(net.parameters(), lr=learning_rate)
```

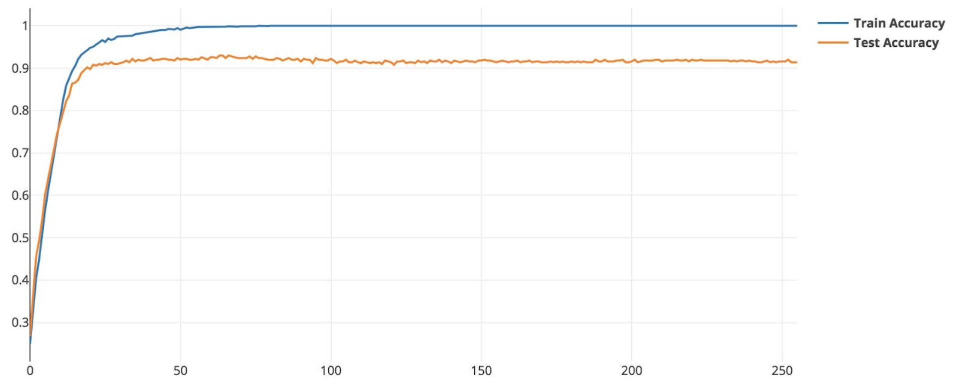
The final step is to train the model where a condition is checked whether GPU is available or not. This is done since GPU is more faster in computing instructions per second. It is shown in the following snapshot:-

```
if use_gpu:
    model_ft, train_acc, test_acc = train_model(net.cuda(), criterion, optimizer, num_epochs)
else:
    model_ft, train_acc, test_acc = train_model(net, criterion, optimizer, num_epochs)
```

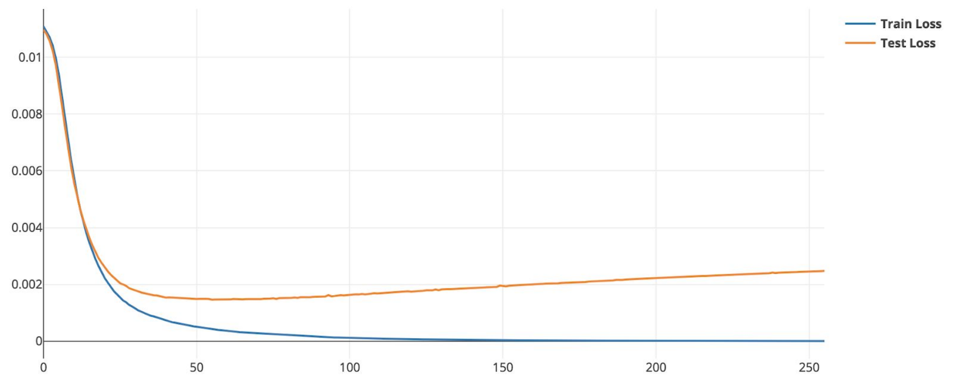
The following snapshot shows the accuracy and loss that was achieved after final epoch using train and test set respectively

```
Epoch 255/255
----------
train Loss: 0.00000848 Accuracy: 1.0000
test Loss: 0.00247639 Accuracy: 0.9140
Best val Acc: 0.930000
```

The graph shows the train and test accuracy that was obtained on different epochs.



The graph shows the train and test loss that was obtained on different epochs.



Visualization using Plotly

For plotly we need to create account on plotly dashboard and obtain the API key for the project as shown in the following snapshot:-

```
plotly.tools.set_credentials_file(username='ali_mirza', api_key='AIl8g6LMPjsv8255sPy5')
plotly.tools.set_config_file(world_readable=True,sharing='public')
```

The bar chart show the correlation of price range as compared to other features

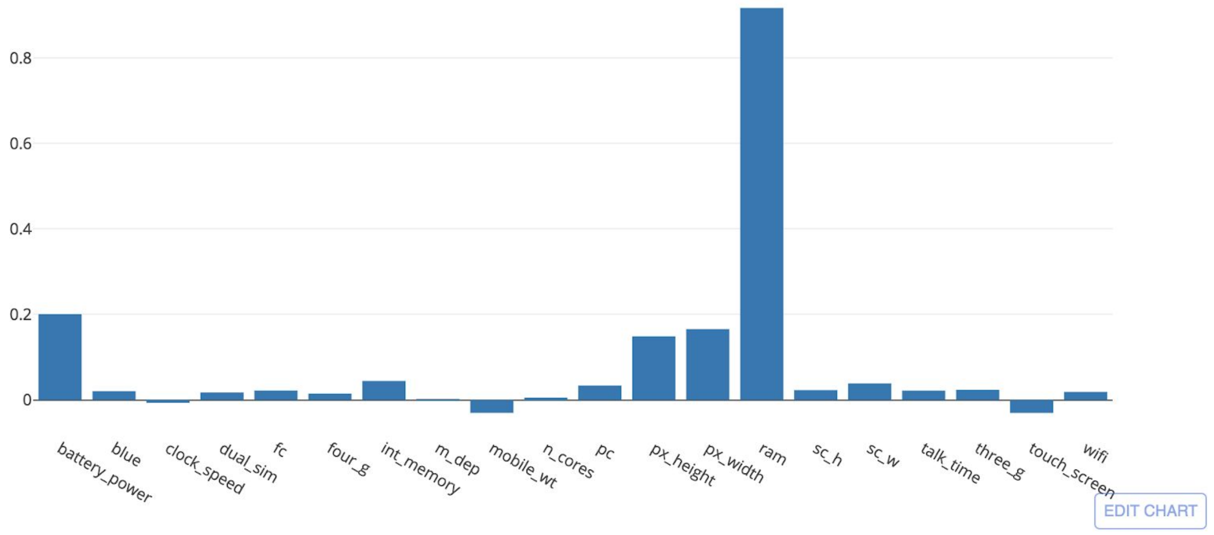


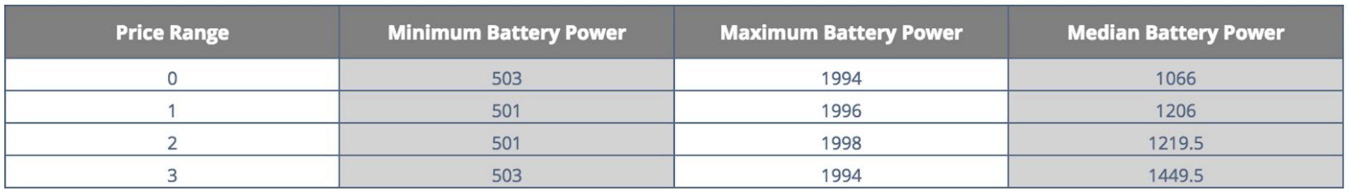
Fig 1: Bar Chart showing correlation of price range as compared to other features

Analysis

- From the graph we can observe that with the increase in battery power, height, width, RAM the price range also increases. All this attributed shows **positive correlation** with price range
- It is also evident that if there is no touch screen then the price range reduces as the graph shows **negative correlation**.
- Similarly with increase in mobile weight the price range reduces as the graph shows **negative correlation**.

Now we will verify all the above correlated attributes in terms of price range

The table shows minimum, maximum and median battery power for various price range



Price Range	Minimum Battery Power	Maximum Battery Power	Median Battery Power
0	503	1994	1066
1	501	1996	1206
2	501	1998	1219.5
3	503	1994	1449.5

Fig 2: Table showing Minimum,Maximum and Median Battery Power

The table shows minimum, maximum and median values of RAM for various price range

Price Range	Minimum RAM	Maximum RAM	Median RAM
0	256	1974	719.5
1	387	2811	1686.5
2	1185	3916	2577
3	2259	3998	3509.5

Fig 3: Table showing Minimum,Maximum and Median RAM

The table shows minimum, maximum and median values of Pixel Height for various price range.

Price Range	Minimum Pixel Height	Maximum Pixel Height	Median Pixel Height
0	1	1878	465.5
1	0	1914	606
2	10	1960	538.5
3	0	1949	674

Fig 4: Table showing Minimum,Maximum and Median Pixel Height

The table shows minimum, maximum and median values of Pixel Width for various price range.

Price Range	Minimum Pixel Width	Maximum Pixel Widyh	Median Pixel Width
0	500	1989	1132.5
1	500	1998	1223
2	508	1997	1221.5
3	501	1995	1415.5

Fig 5: Table showing Minimum,Maximum and Median Pixel Width

From median values as shown shown in figure 2, figure 3, figure 4, figure 5 it is clear that there is a positive correlation of price range with battery power, RAM, pixel width and pixel height.

The following graph shows the count of phones available with Categorical features like bluetooth availability, dual sim, four_g, three_g, touch_screen and wifi.

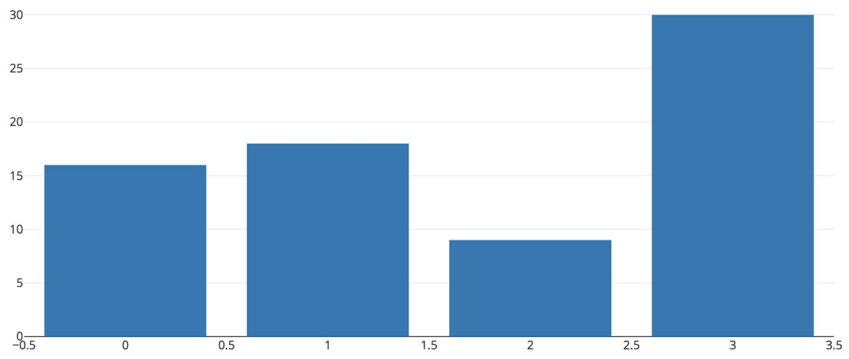
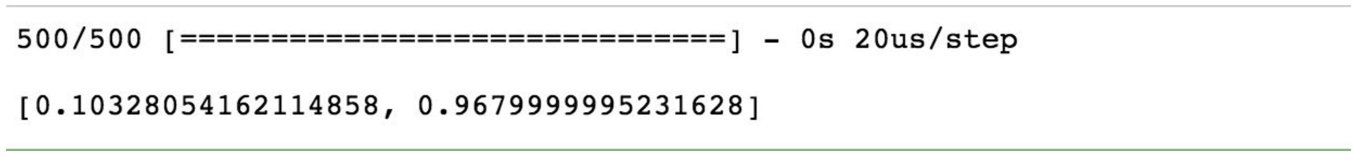


Fig 6: Bar Chart showing Number of phones with categorical Features

It can be observed from the bar graph as the number of categorical feature increases the price range also increases.

Keras Vs Pytorch

The same dataset was used to build the model using keras by keeping the same parameters and better accuracy was obtained as shown in the following snapshot:-



```
500/500 [==============================] - 0s 20us/step

[0.10328054162114858, 0.9679999995231628]
```

The graph shows the training set accuracy of Keras and Pytorch at different epoch

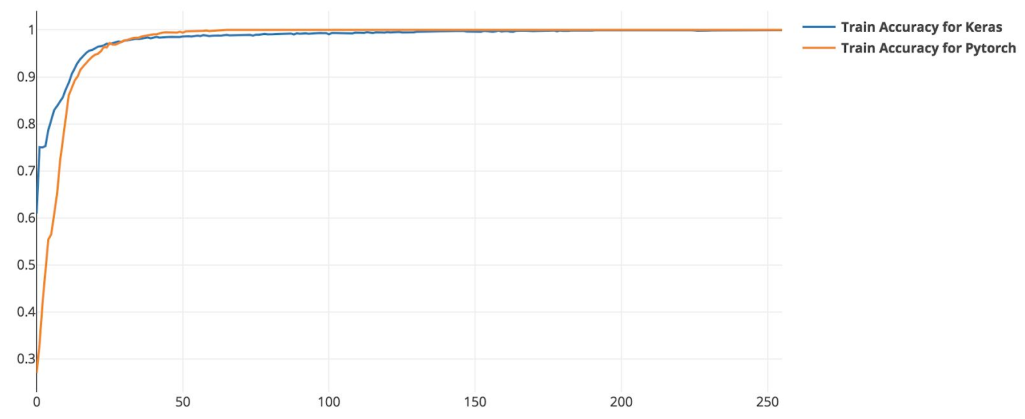


Fig 7: Line Chart showing training set accuracy of Keras and Pytorch at different epoch

The graph shows the test set accuracy of Keras and Pytorch at different epoch

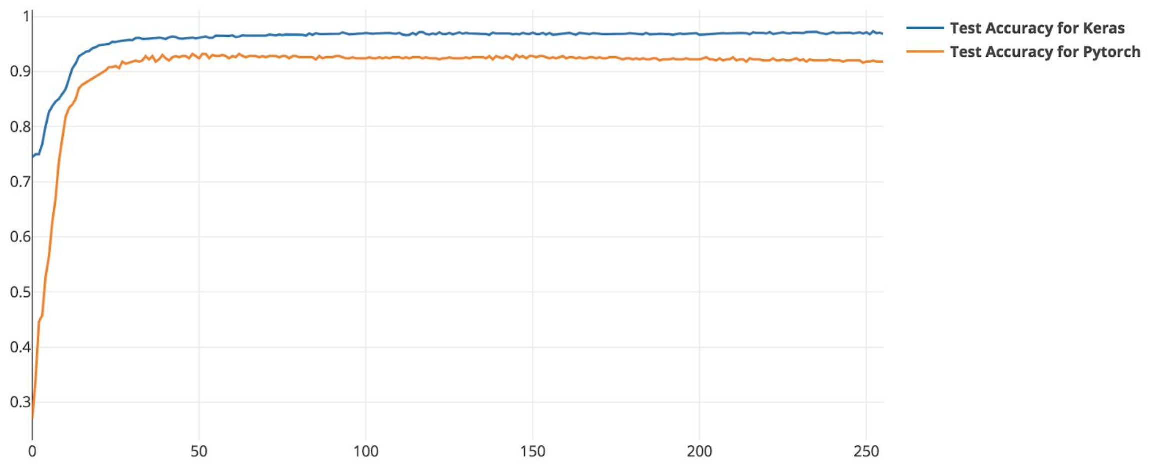


Fig 8: Line Chart showing test set accuracy of Keras and Pytorch at different epoch

The graph shows the training set loss of Keras and Pytorch at different epoch

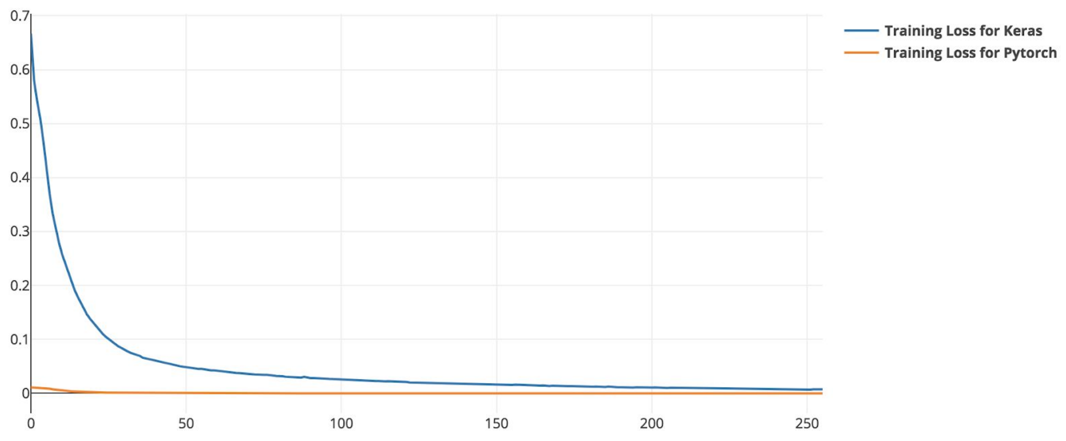


Fig 9: Line Chart showing Train set Loss of Keras and Pytorch at different epoch

The graph shows the test set loss of Keras and Pytorch at different epoch

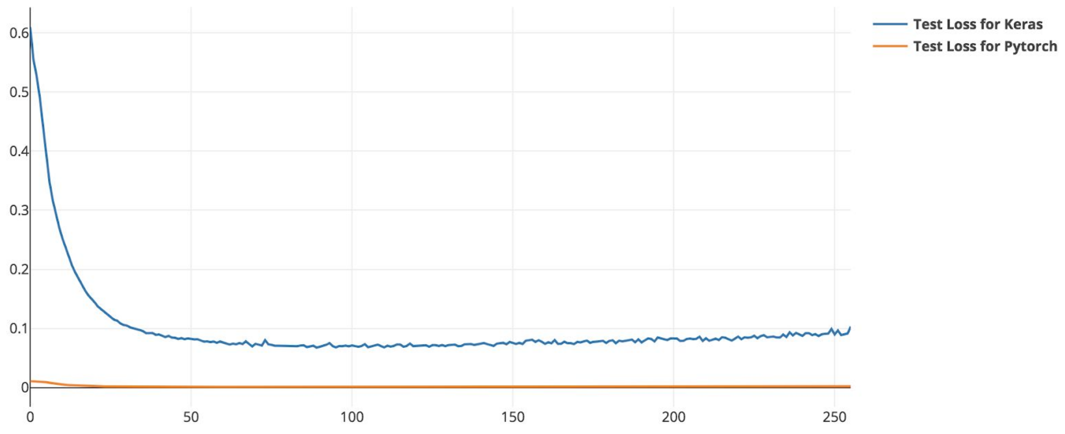


Fig 10: Line Chart showing Test set Loss of Keras and Pytorch at different epoch

<u>Analysis</u>

- From the graphs shown fig.9 and fig.10 we can conclude that for our dataset convergence is better in case of Pytorch as compared to Keras. This holds true for both train and test
- From the graphs shown fig.7 and fig.8 we can conclude that for our dataset performance of keras is better than pytorch in terms of Accuracy.

<u>References</u>

https://plot.ly/python/
https://pytorch.org/tutorials/
https://www.kaggle.com/iabhishekofficial/mobile-price-classification