Mobile Price Range Prediction

Delivery 3

Siyu Wang

RECAP

- Use *Mobile Price Classification* dataset to build models to predict the price range indicating how high the price is of mobile phone base on mobile specifications
- Random access memory in megabytes and battery power are the two most important feature

Data Preprocess

- Standardize features by removing the mean and scaling to unit variance
- Split Train(80%) and Test data(20%)
- 22 Features and 1600 train data

Data Slice

- Trian

	0	1	2	3	4	5	6	7	8	9	...	12	13	14	15
155	1.032262	-0.990050	0.953358	0.981177	-0.071307	0.957886	0.769162	1.381165	-0.035292	-1.539175	...	1.558662	-0.673342	0.876859	0.971917
1113	0.233279	1.010051	0.708200	0.981177	-0.992890	-1.043966	0.217930	-0.352878	-0.939481	1.521249	...	-1.567991	0.069884	1.351672	1.890337
771	-1.567278	1.010051	-1.253064	-1.019184	2.693441	0.957886	-0.498672	-1.393304	0.134244	-1.101971	...	1.570234	-1.680294	1.351672	-0.176107
109	1.221196	1.010051	-0.027274	-1.019184	-0.301703	-1.043966	0.493546	-1.393304	0.642851	-1.539175	...	1.429060	0.201747	1.589078	1.660732
1924	1.692391	1.010051	0.095305	-1.019184	-0.532099	0.957886	-1.105027	-1.046495	-1.702391	0.209639	...	-1.524019	-0.276832	0.164641	-0.635317

5 rows × 22 columns

- Test

```
155     1
1113    1
771     0
109     3
1924    1
Name: price_range, dtype: int64
```

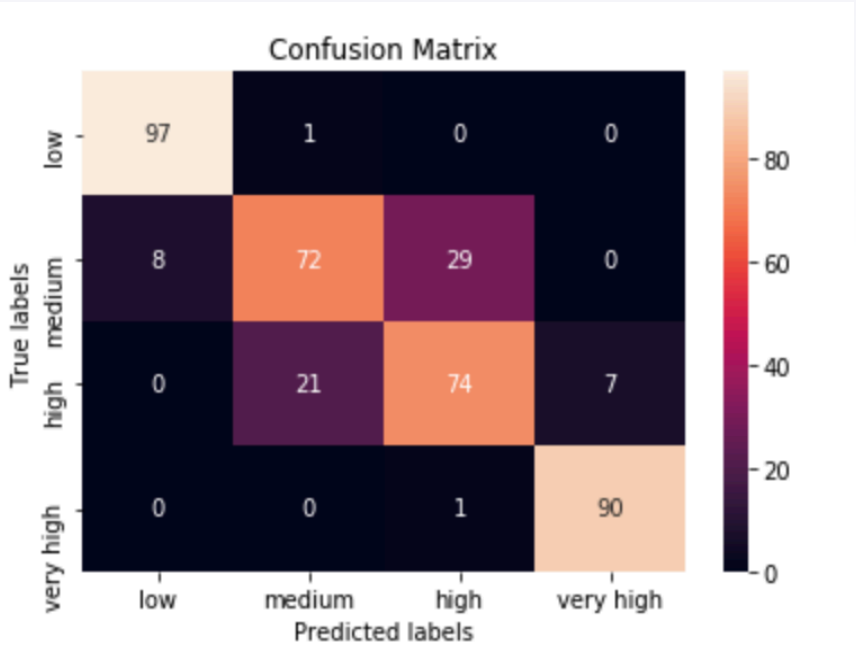
Problem Determination

- **Output labels**: 0: low cost, 1: medium cost, 2: high cost, 3: very high cost
- It is a multiclass classification problem
- Package used: sklearn
- Metric to use: Mean Accuracy
- Models:
  - **Logistic Regression(OvR)**
  - **Neural Network**
  - **Random Forest**

Logistic Regression

- Logistic Regression is mainly used for binary classification
- Uses the one-vs-rest (OvR) scheme for multiclass classification
- **LogisticRegression(multi_class = "ovr", C = 10 ,solver = "liblinear", max_iter = 200)**

Result



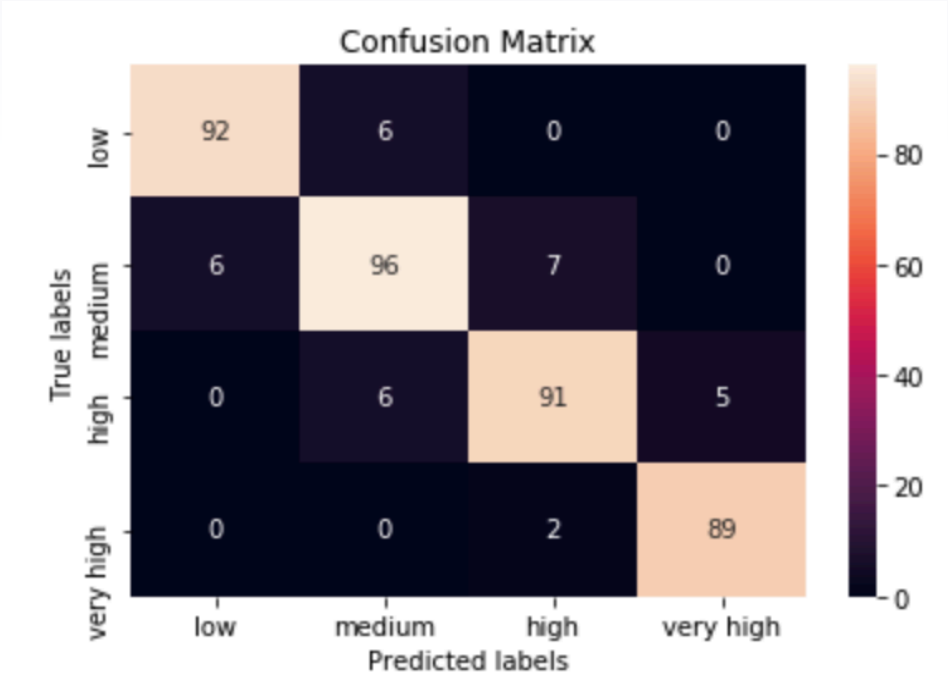
- Mean Accuracy for the Train Data: 86.75%
- Mean Accuracy for the Test Data: 83.25%

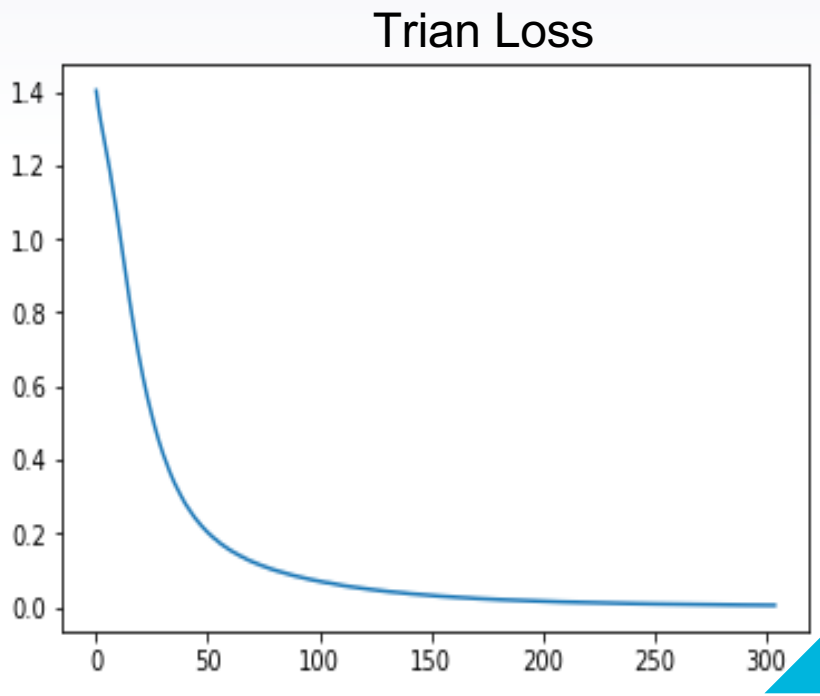
Neural Network

- Activation function: Relu
- Optimization Algorithm: Adam
- **MLPClassifier(hidden_layer_sizes=(20,40),max_iter=400,activation ='relu',solver='adam',learning_rate_init = 0.001, n_iter_no_change = 20)**

Result

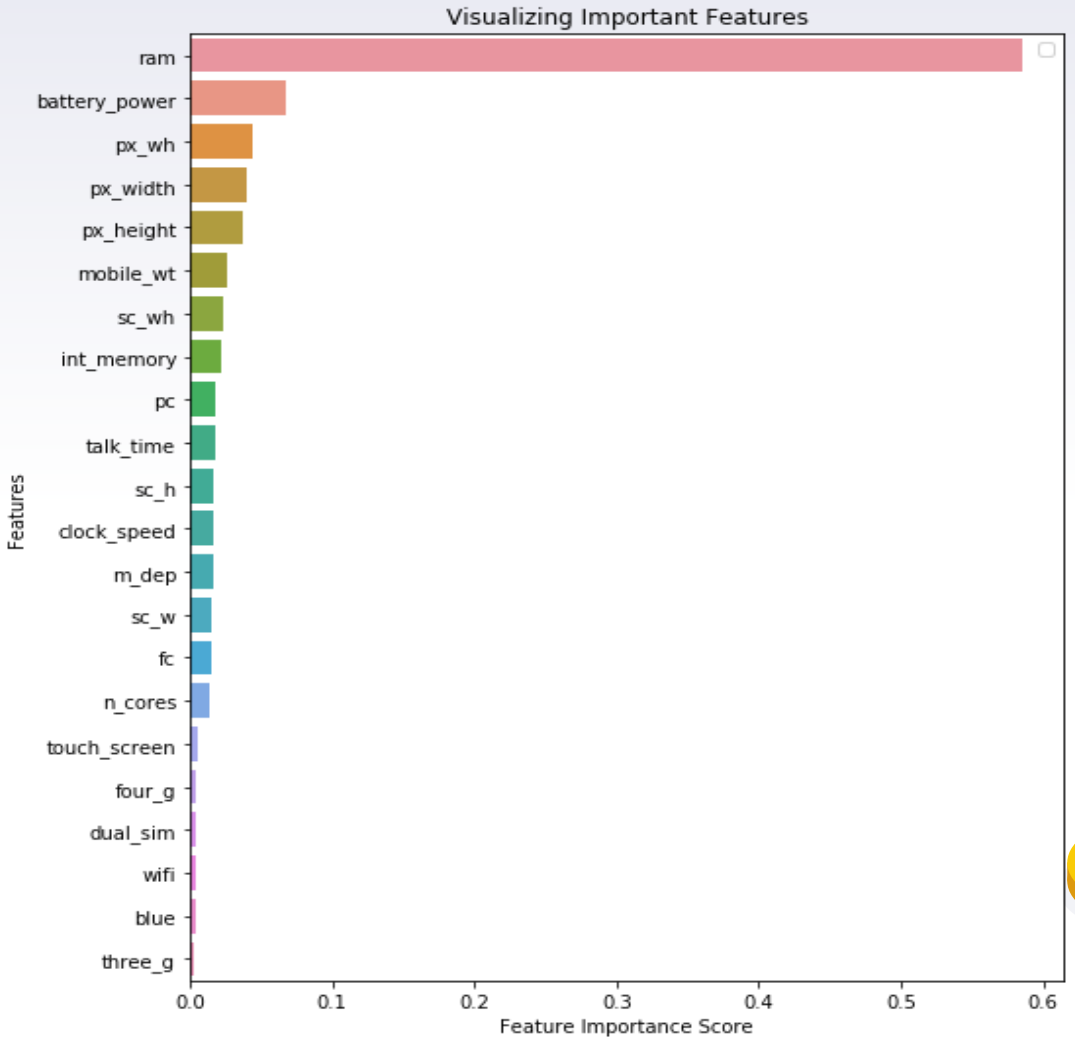
- Mean Accuracy for the Train Data: 100%
- Mean Accuracy for the Test Data: 92%



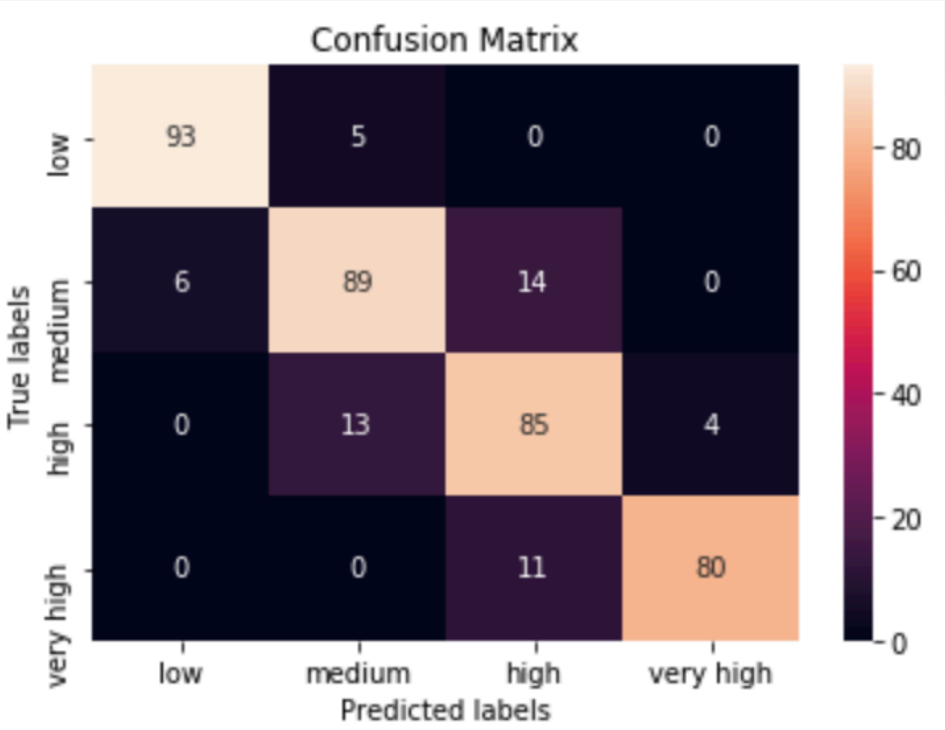


Random Forest

- Able to print out Feature Importance
- RandomForestClassifier(n_estimators= 300,max_depth=10, random_state=6, min_samples_split= 4, min_samples_leaf =5)


Visualizing Important Features
ram
battery_power
px_wh
px_width
px_height
mobile_wt
sc_wh
int_memory
pc
talk_time
sc_h
clock_speed
m_dep
sc_w
fc
n_cores
touch_screen
four_g
dual_sim
wifi
blue
three_g
Features
0.0
0.1
0.2
0.3
0.4
0.5
0.6
Feature Importance Score

Result



- Mean Accuracy for the Train Data: 98.75%
- Mean Accuracy for the Test Data: 86.75%

THANKS!

Any questions?

You can find me at:

- siyuw1@umbc.edu
- https://github.com/ciciwang1/DATA606

Credits

- Presentation template by Slide Carnival
- Dataset by Abhishek Sharma