Mobile Price Classification

Saeid Cheshmi - 98222027

1. Data Exploration

Mobile price dataset has 2000 rows and 21 columns where each column is a feature of a mobile (e.g RAM, battery power, …). Our goal here is predict price range of a mobile.

The dataset has the following attributes:

- **“id”**: id of each record
- **“battery_power”**: total energy a battery can store in one time measured in mAh
- **“blue”**: has bluetooth or not
- **“clock_speed”**: speed at which microprocessor executes instructions
- **“dual_sim”**: has dual sim support or not
- **“fc”**: front camera mega pixels
- **“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”**: cumber of cores of processor
- **“pc”**: primary camera mega pixels
- **“px_height”**: pixel resolution height
- **“px_width”**: pixel resolution width
- **“ram”**: random access memory in megabytes
- **“sc_h”**: screen height of mobile in cm
- **“sc_w”**: screen width of mobile in cm
- **“talk_time”**: longest time that a single battery charge will last when you are
- **“three_g”**: has 3G or not
- **“touch_screen”**: has touch screen or not
- **“wifi”**: has wifi or not

Also, our target feature (price_range) has 4 different classes (0,1,2,3) but in this exercise we reduce them to 2 classes, ‘low’ and ‘high’, ‘low’ class contains 0 and 1 and, ‘high’ class contains 2 and 3.

2. Preprocessing

Based on previous report, this dataset doesn’t have any null value and for outlier removal unlike previous exercise, here we used IQR method. After computing inter quartile range, we add 1.5x of IQR to third quartile and subtract 1.5x of IQR from first quartile, data which aren’t in this range are considered as outlier.

3. Exploratory Data Analysis

We did this step in last exercise and we didn’t change it. So, we use it in this exercise too.

4. Forward Feature Selection

For implementing forward feature selection at first, we divide our dataset into 3 parts, train set, validation set and test set. At first step, we select a feature that has the highest AUC score on validation set then select another feature and add to first feature which the combination of these two features has highest AUC score between two-feature model. We do so until we select desired number of features. Here we select all feature and based on their scores we pick up best features.

After picking up the features, we will evaluate our model on test set. According to the scores after 4 top features, we got AUC score of 0.999 and after that we didn’t have significant improvement. So, we select these features, ‘ram’, 'battery_power', 'px_width' and 'px_height'.

We got accuracy of 98.5% on test set and in figure 1, you can see recall, precision and f1-score of our model.

	precision	recall	f1-score	support
high	0.99	0.97	0.98	192
low	0.98	1.00	0.99	204
accuracy			0.98	396
macro avg	0.99	0.98	0.98	396
weighted avg	0.99	0.98	0.98	396

Figure 1. Performance of model with selected features

5. Applying PCA

As we have been asked, we set ‘n_components’ parameter of PCA as same as number of selected features in above section. So, we set ‘n_components’ to 4 and transformed our dataset with it then, evaluate our model. By doing so we got accuracy of about 99% on test set and in the figure 2 you can see recall, precision and f1-score of our model.
As you can see, we got almost the same results as previous section, there is only a little improvement.

	precision	recall	f1-score	support
high	0.99	0.99	0.99	192
low	0.99	0.99	0.99	204
accuracy			0.99	396
macro avg	0.99	0.99	0.99	396
weighted avg	0.99	0.99	0.99	396

Figure 2. Performance of model with PCA transformation

6. Feature Engineering

6.1. Binning battery power

We bin battery power feature in 3 ways, first we consider bins with same size and label each bin respectively, ‘low’, ‘mid’ and ‘high’. You can see distribution of them in figure 3.

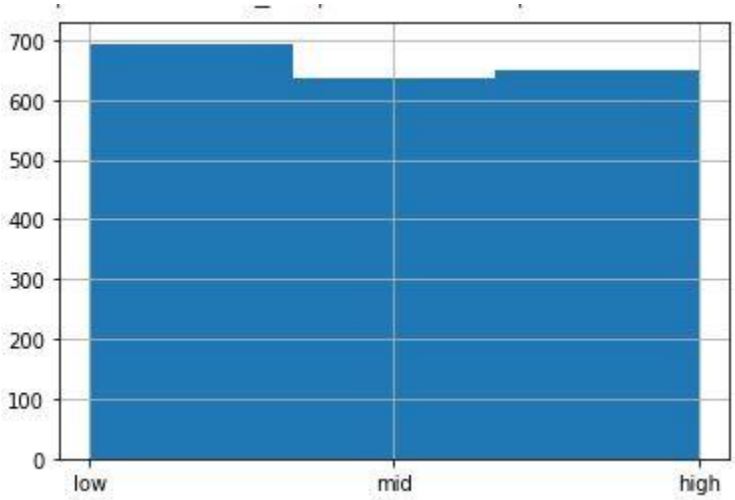

Figure 3. binned battery power (1)

We also bin this feature in 2 another ways and we consider different bin sizes. We consider 0-666 as ‘low’, 666-1333 as ‘mid’ and 1333-2000 as ‘high’. You can see distribution of this binning in figure 4.

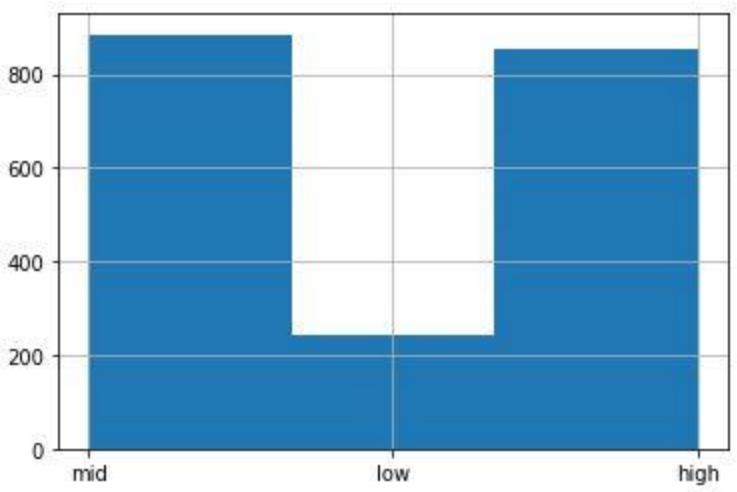

Figure 4. binned battery power (2)

At last, we bin the feature in this way, we consider 0-1000 as ‘low’, 1000-1600 as ‘mid’ and 1600-2000 as ‘high’ bin. You can see distribution of this binning in figure 5.

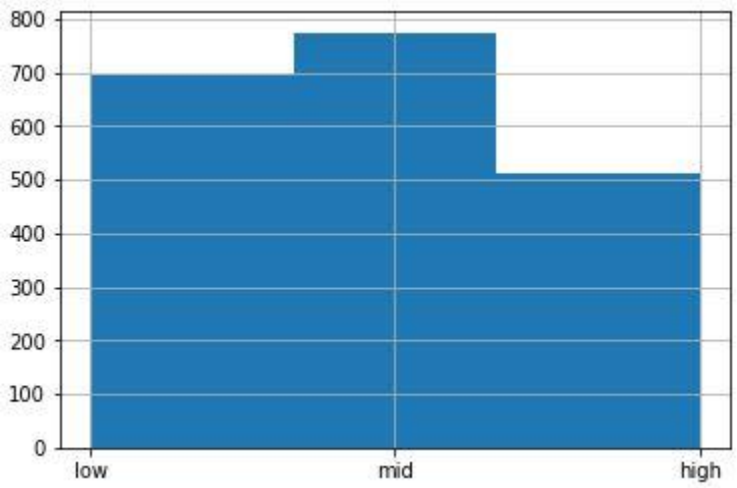

Figure 5. binned battery power (3)

6.2. One-Hot Encoding

Many machine learning algorithm can not work with categorical features directly. They accept only numerical features. This means that categorical features must be converted to numerical. There are some conversion methods like one-hot encoding, ordinal encoding etc. In one-hot encoding, we convert each categorical value into a new categorical column and assign a binary value of 1 or 0 to those columns. One hot encoding is useful for data that has no relationship to each other.

In this dataset we have only one categorical feature, ‘battery_power’, and we apply one-hot encoding on it.

6.3. Feature Transformations

We use feature transformation to change the distribution of features. Some machine learning models, like linear regression and logistic regression assume that data follow the normal distribution but in real world it is unlikely to. By applying these transformations, we change the distribution of features to normal or almost normal and this causes improvement of model performance. Also, some optimization algorithms converge faster if features have same range.

Here we use log transformation and Quantile Transformer for some features.

Log transformer: It is used to convert a skewed distribution to a normal distribution/less-skewed distribution.

Quantile Transformer: It converts he variable distribution to a normal distribution and scales it. It is also robust to outliers.

According to figure 6, we can apply log transformation on ‘fc’ feature because it has a skewed distribution. Also, 'clock_speed', ‘px_height’ and ‘sc_w’ aren’t normally distributed, so we can apply quantile transformation on them.

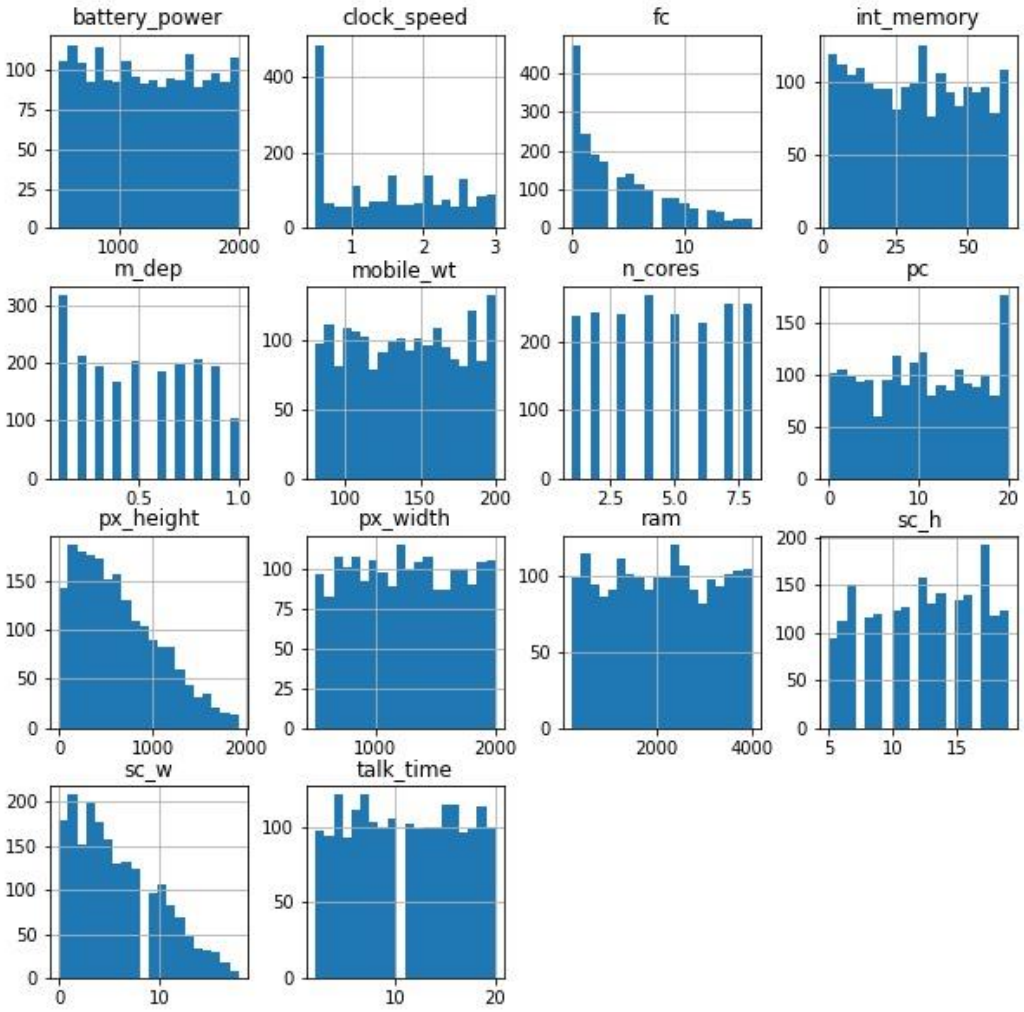

Figure 6. Distributions of numerical features

After applying the above transformation, we got the following distributions.

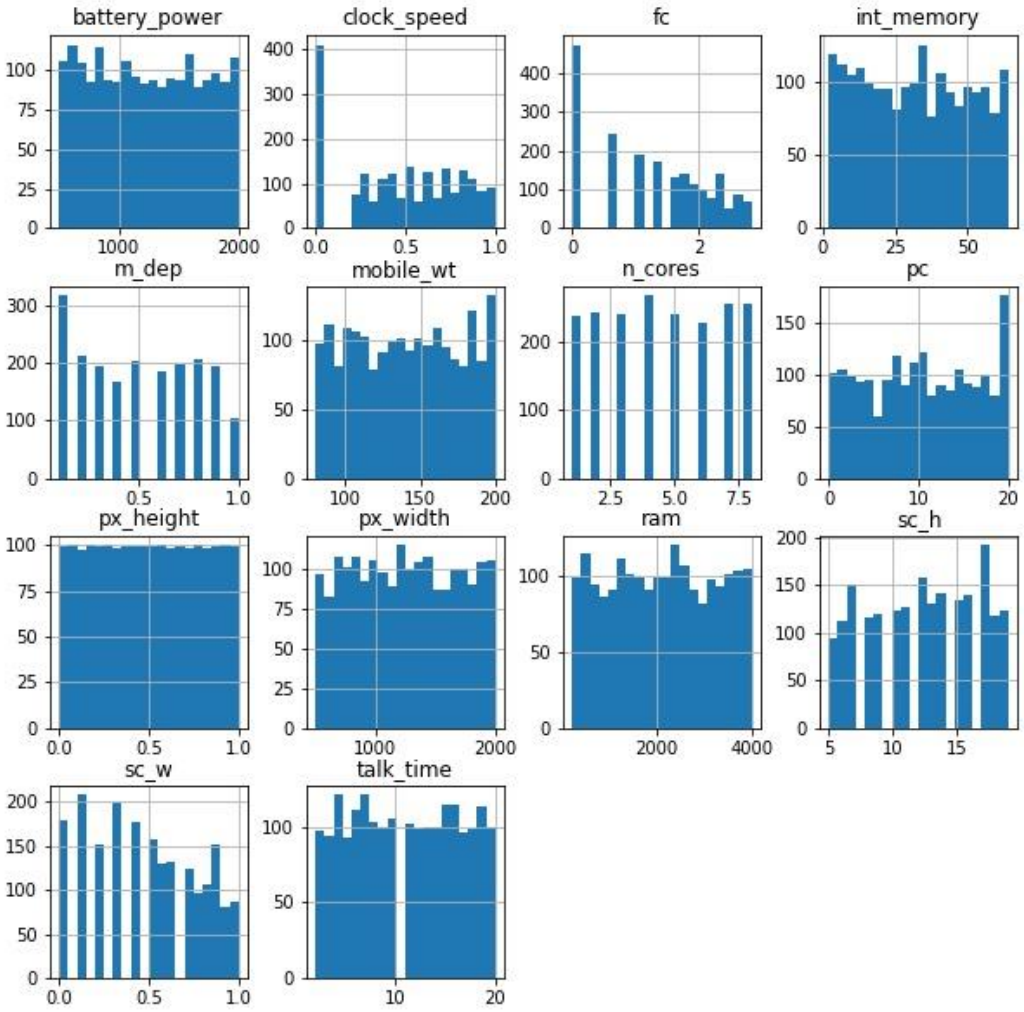

Figure 7. Distributions of numerical features after transformation

Now you can see that distributions of our selected features are more normal than before.

6.4. Creating new feature

We create new feature ‘area’ by multiplying the ‘sc_w’ and ‘sc_h’ of a mobile. It might improve model performance. We evaluate effect of this feature creation in next section.

7. SVM

We train an SVM model for each above feature engineering steps and will compare the results. We use cross validation for evaluation of our models. In the following table you can see the results.

	One-hot encoding	Transformation	New feature	All steps
Accuracy	92.9%	96.5%	98.2%	92.1%

8. Bootstrapping vs cross validation

Bootstrapping is a resampling method used to estimate statistics on a population by sampling a dataset with replacement. It can used to estimate performance of a machine learning model. On the other hand, cross validation is another resampling method which used to estimate machine learning model performance, too. In cross validation, the dataset is splitted into k folds then, k-1 folds are used for training and 1 fold is used for evaluation. Generally, cross validation is mostly used for estimation of model performance while bootstrapping is used for data ensembling for some models like random forest.

9. 5 x 2 cross validation

In 5x2 cross validation, we do cross validation 5 times and each time k equals to 2. We can use the average of these results to approximate the model error and their variances. These data also can be used for statistical tests and comparing the performance of different models.

10. Elbow method

Error of predicted values and accuracy of model depend on model complexity and degree of model, the lower complexity, the higher error and bias, this is the result of underfitting. On the other hand, by increasing the complexity of model, variance increases and this results in overfitting. In real world dataset due to their noises, by increasing the model complexity, bias decreases and variance increases. In these situations, we want to find a model with low bias and variance which depend on dataset size and amount of noise it has. If our dataset isn't very noisy or a little noisy, we can realize that both of bias and variance decreases with increasing the degree of model. So, we can't use elbow method everywhere, instead of it we can plot the bias and variance and select degree of model.

11. Extra

11.1. Backward Selection

We implement backward feature selection then select some features and train a model based on them. As you can see, like forward selection method, the same features are selected. So, we got the same results as before.

	precision	recall	f1-score	support
high	0.99	0.97	0.98	192
low	0.98	1.00	0.99	204
accuracy			0.98	396
macro avg	0.99	0.98	0.98	396
weighted avg	0.99	0.98	0.98	396

Figure 8. Performance of model with selected features

11.2. Statistical Significance Tests

Machine learning models usually evaluated by resampling methods like k-fold cross validation which mean score is calculated and models are compared. But, we don't know difference between these scores is real or because of statistical issues.

Statistical significance tests are designed to address this issue and quantifying the likelihood of the samples of scores with assumption that they were drawn from the same distribution.

Some of the methods that we can use for these purposes are McNemar's test or 5x2 cross validation which described earlier.

11.3. Matthews Correlation Coefficient

MCC is a metric for evaluation of binary or multi-class classifier. MCC treats the predicted class and true class as two variables and computes the correlation between them. The higher correlation, the better prediction. MCC is always between -1 and 1, 0 means the classifier is no better than flipping a coin.

$$\mathrm{MCC} = \frac{\mathrm{TP} \times \mathrm{TN} - \mathrm{FP} \times \mathrm{FN}}{\sqrt{(\mathrm{TP}+\mathrm{FP})(\mathrm{TP}+\mathrm{FN})(\mathrm{TN}+\mathrm{FP})(\mathrm{TN}+\mathrm{FN})}}$$

12. References

- https://towardsdatascience.com/the-best-classification-metric-youve-never-heard-of-the-matthews-correlation-coefficient-3bf50a2f3e9a
- https://www.analyticsvidhya.com/blog/2021/05/feature-transformations-in-data-science-a-detailed-walkthrough/#h2_3
- https://www.analyticsvidhya.com/blog/2020/07/types-of-feature-transformation-and-scaling/#h2_9
- https://machinelearningmastery.com/statistical-significance-tests-for-comparing-machine-learning-algorithms/