# **KNN & Naive Bayes part**

# zID: z5235682

Name: Genyuan Liang

# **Nearest Neighbour classification**

1. Nearest Neighbour classification can be used as a machine learning methods. For each unknown data points, calculating the distance of k number of the most adjacent data points and then calculating the average in order to classify.

Unlike most machine learning methods, the KNN algorithm does not have a clear model fitting step to the training data. We only need to achieve predictions through this method. For distance based KNN, due to its sensitive to scale of the features, higher dimensions will lead to the observations in each partition rapidly approach zero, and the distance from one observation point to another tends to be far away. Therefore, in this model, we need to select the features which are more highly correlated with the target to reduce the dimensions.

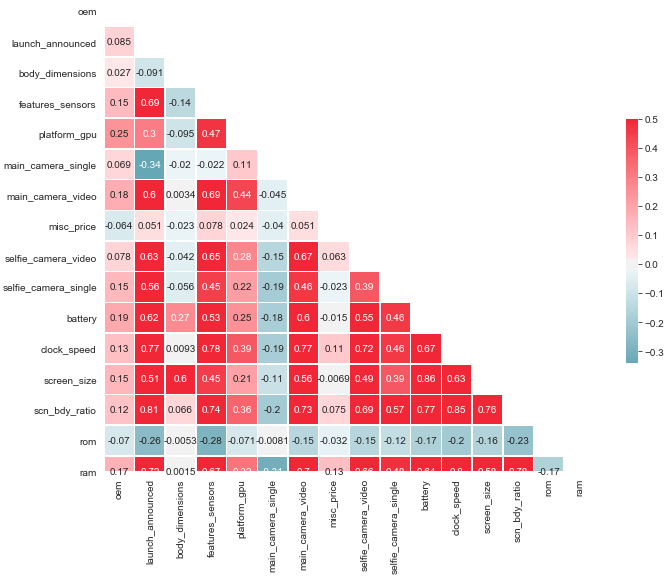


Figure 1: Correlation heatmap of filtered dataset

According to the correlation heatmap, the correlation between price and other attributes is quiet low. The top 5 features of correlation are "ram", "clock\_speed", "features\_sensors", "scn\_bdy\_ratio" and "oem".

First we look at all the features. In this model we use GridSearchCV from scikit-learn to exhaustive search over specified parameter values for KNeighborsClassifier class. When we set the algorithm in the parameters to "auto", "n\_neighbors" to 9, and "weight" to "distance", we find the best accuracy which is 0.662379421221865. which means using 9 Nearest Neighbour classification and weighting points by the inverse of their distance can reach the highest accuracy.

Then we look at the top 5 features. Using the same approach above, the best result using top 5 features ('ram', 'clock\_speed', 'features\_sensors', 'scn\_bdy\_ratio', 'oem') is : 0.7041800643086816, and the algorithm in the parameters to "brute", "n\_neighbors" to 10, and "weight" to "distance".

# **Naive Bayes**

In machine learning, the naive bayes classifier assumes that the features are independent of each other, and calculates the probability according to the Bayes formula to determine the label. This assumption is often not true in practical applications. In this model, according to the correlation heatmap of training dataset above, it can be seen that there may actually be a large relationship between attributes, and the number of attributes is relatively large, so the performance of Bayesian classification is not very good, and the actual accuracy obtained will not be very high.

There are several naive bayes classification algorithm classes in sklearn, such as BernoulliNB, MultinomialNB , GaussianNB and ComplementNB. We use these four models to train the data. For our dataset, the accuracy of the each kind of naive bayes classification are showed in table.

Table 1: Accuracy of naive bayes models

|  |  |
| --- | --- |
| **model** | **accuracy** |
| BernoulliNB | 0.4983922829581994 |
| MultinomialNB | 0.42765273311897106 |
| GaussianNB | 0.5305466237942122 |
| ComplementNB | 0.5852090032154341 |

In this dataset, it shows that ComplementNB model has the highest accuracy, while the accuracy of GaussianNB is the second highest, but they are all below 60%. Gaussian Naive Bayes which prior is Gaussian distribution can perform better if the distribution of sample features are mostly continuous values. Multinomial Naive Bayes which prior is a polynomial distribution is more appropriate to use when the distribution of sample features is mostly multivariate discrete values. Bernoulli Naive Bayes which prior is Bernoulli distribution is suitable for sample features that are binary discrete values or very sparse multivariate discrete values. Complement Naive Bayes is designed to correct the serious assumption of the standard polynomial naive Bayes classifier. It is particularly suited to unbalanced data sets.

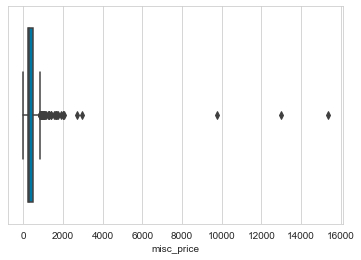


Figure 2: Distribution of the unlabeled prices

Our data sets is unbalanced, so it is suitable to use ComplementNB. Before we label the target, we can see that distribution of the unlabeled target, the data is is severely skewed to the right. There are a few of data have the extremely high price (Figure 2), while most of the data are under $2000

Table 2: The number of each label in test data

|  |  |  |
| --- | --- | --- |
| Label | Price range | Test number |
| 0 | lower than $300 | 147 |
| 1 | lower than $700 & higher than $300 | 135 |
| 2 | higher than $700 | 29 |

Then we divide the misc price into three labels, and use train\_test\_split to split dataset randomly(In this model, we set test\_size = 0.3, random\_state=0). As it shown as below, the amount of label 2 in the test data is much lower than the other labels.