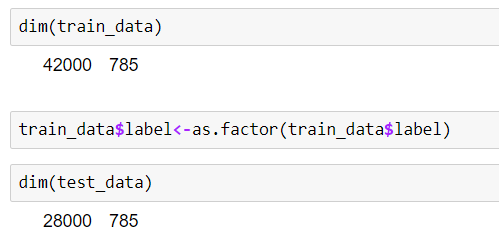
1. **Introduction**

The data set comes from the Kaggle Digit Recognizer competition. The goal is to recognize digits 0 to 9 in handwriting images. We are going to use the sampled data to construct prediction models using naïve Bayes and decision tree algorithms. Tune their parameters to get the best model (measured by cross validation) and compare which algorithms provide better model for this task.

Loading the required packages in R:

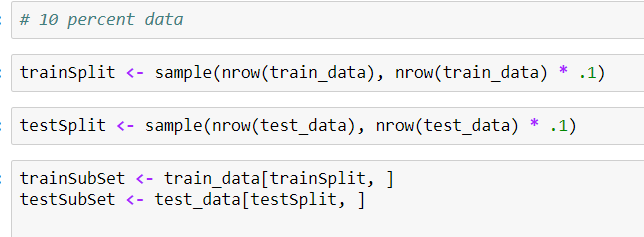


Summary:



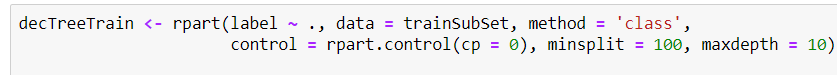
Splitting:

Dataset is huge we split data into 10% and use it for our classifiers.



1. **Decision Tree**

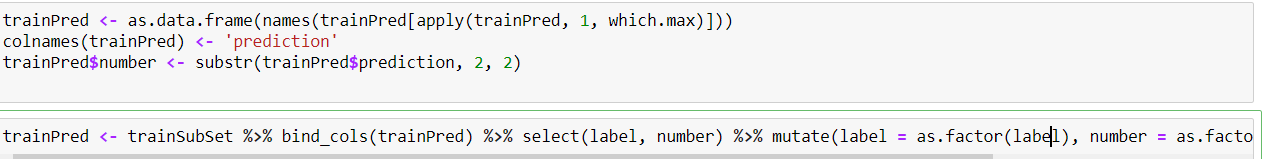
**Model:**



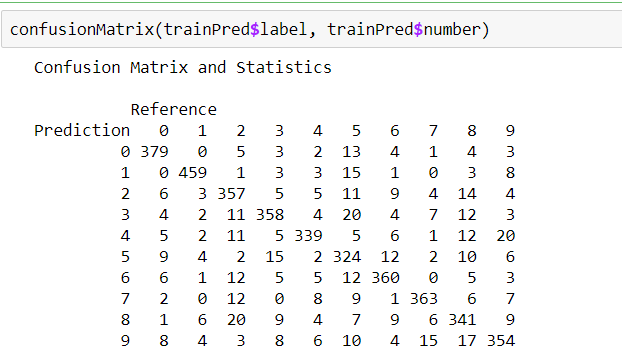
**Testing accuracy of our model on Training data**



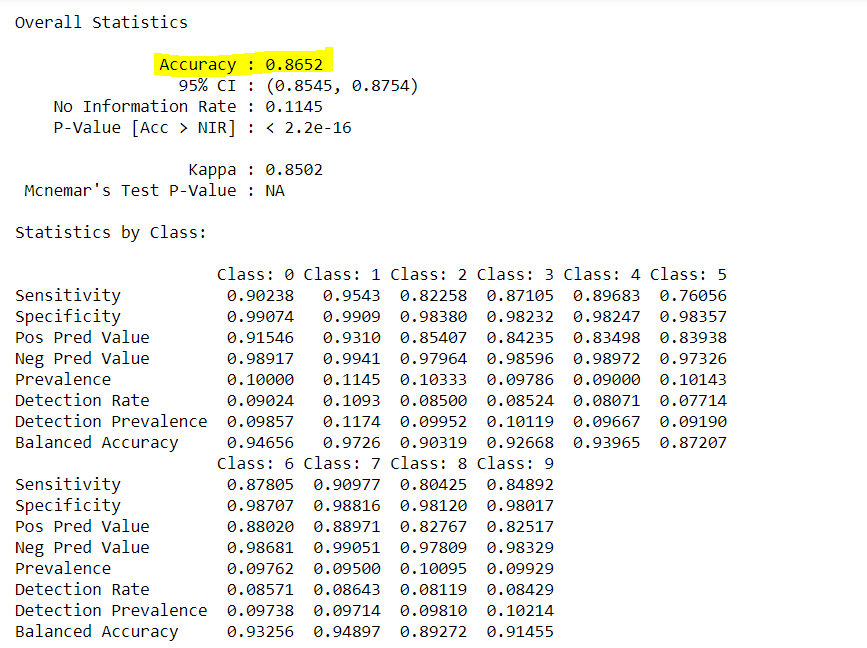
**Maximum likelihood Number**



**Confusion Matrix**

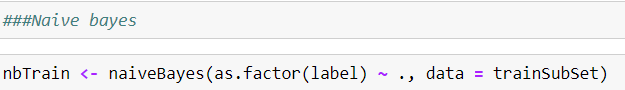


**Output**

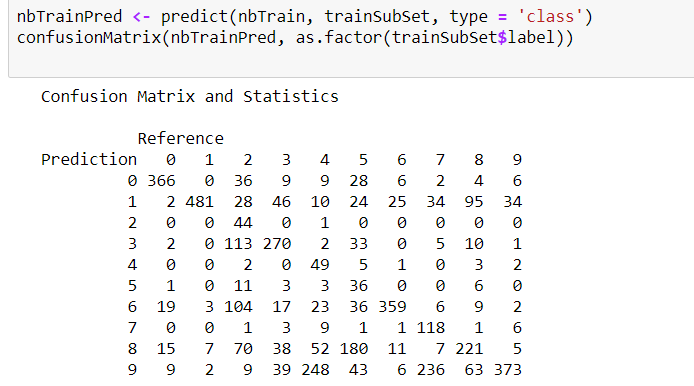


1. **Naïve Bayes**

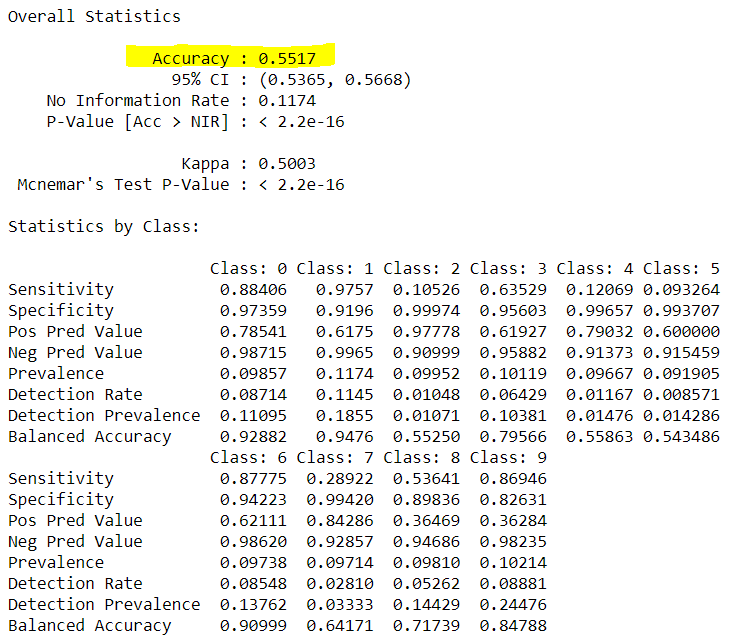
**Model**



**Confusion Matrix**



**Output**



1. **Algorithm Performance Comparison**

According to our Algorithmic models, Decision tree performs better than Naïve Bayes for this set of data. Decision Tree has a predicting accuracyof **86.52%** while Naïve Bayes has a predicting ability of **55.17%** which is quite low. Naïve Bayes had problem determining 9 which can be fixed if the dataset was large. However, computations were performed on random sample of 10% data, but if we choose anther random sample or large sample data results could have been better.

Decision Trees are very flexible, easy to understand, and easy to debug. They will work with classification problems and regression problems. Naive Bayes requires you build a classification by hand. There's no way to just toss a bunch of tabular data at it and have it pick the best features it will use to classify.

As such there’s no better classifier, it depends upon problem to problem.

Naive Bayes:

1. Work well with small dataset compared to DT which need more data

2. Lesser overfitting

3. Smaller in size and faster in processing

Decision Tree:

1. Decision Trees are very flexible, easy to understand, and easy to debug

2. No preprocessing or transformation of features required

3. Prone to overfitting but you can user pruning or Random forests to avoid that.