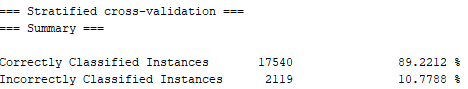
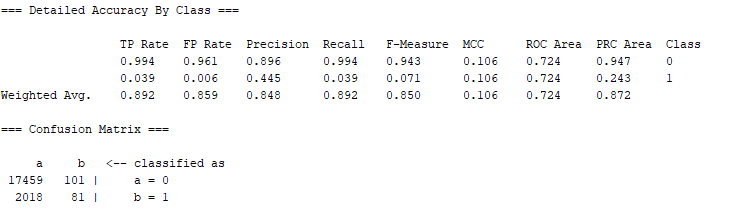
**Slightly Helpful Link** (<https://www.kaggle.com/nareshbhat/fraud-detection-feature-selection-over-sampling>)

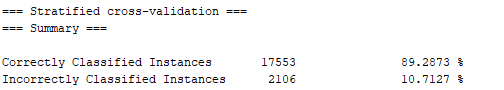
**Logistic Regression – Weka (functions.Logistic)**

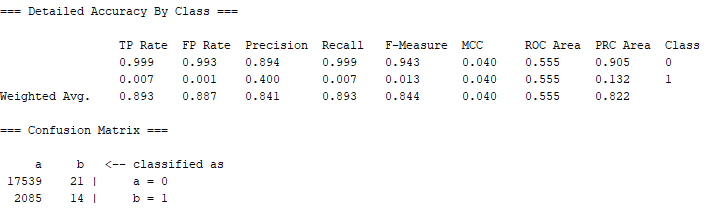
****

****

**Decision tree - Weka (trees. J48)**

We decided to use a decision tree, since decision trees work best with nominal data which the majority of our dataset consists of.





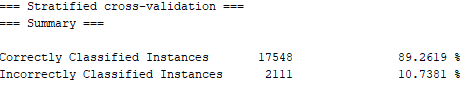
It is worth noting that the accuracy of this dataset is rather high with a nearly 100% True Positive Rate for evaluating non-fraud transactions. Furthermore, the accuracy for evaluating precision, recall and F-Measure for Class 0 is also very high. However, the accuracy for Class 1 for fraudulent transactions is much lower and this in part can be attributed to the low sample size.

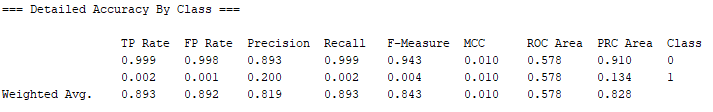
**KNN – Weka (lazy.IBK)**

KNN stores all available instances and then classifies any new instances based on a similarity measure. In the nearest-neighbor classification method, each new instance is compared with existing ones by using a distance metric and the closest existing instance is used to assign the class to the new one.

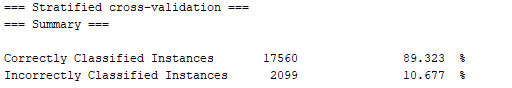
In fraud detection (high risk), KNN allows us to classify an incoming transaction by calculating a nearest point to the new incoming transaction. Then if the nearest neighbor is fraudulent, we classify the transaction as fraudulent. Since we are working with a large dataset, we decided to use a large K value to help reduce the effect of a noisy dataset.C

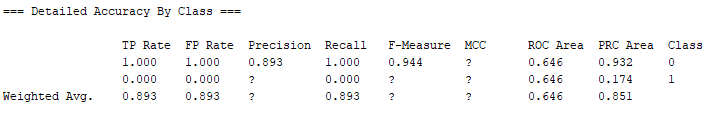
**K=10**





**K=100**

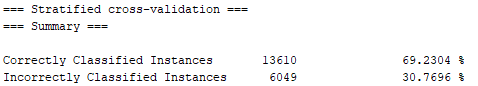


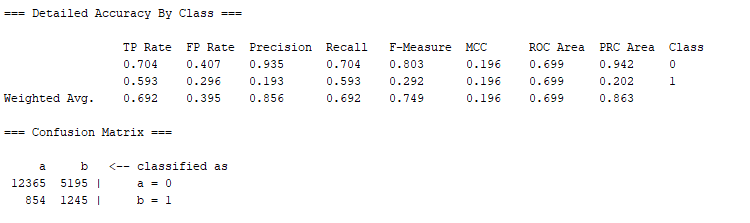
****

Testing for when K=10 seems to be most accurate, for tests that go beyond 10 such as when K= 20, 100 and 1000 the results start to make much less sense. Almost all measures besides for ROC Area share similarities with the metrics in the decision tree results.

**Naïve Bayes – Weka (bayes.NaiveBayes)**

Naïve Bayes is a supervised machine learning method that uses a training dataset with known target classes to make predictions of future instances. This algorithm works best with a small training dataset and can let us solve binary classification problems.



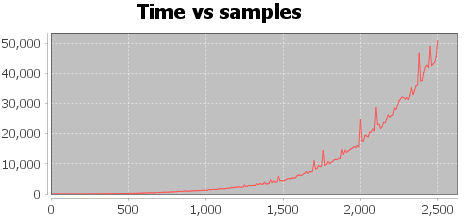


Naive Bayes resulted in the most insignificant and inaccurate results of all the tests conducted. For instance, the CCI drops to 69.23% compared to the 89% found in most other tests.

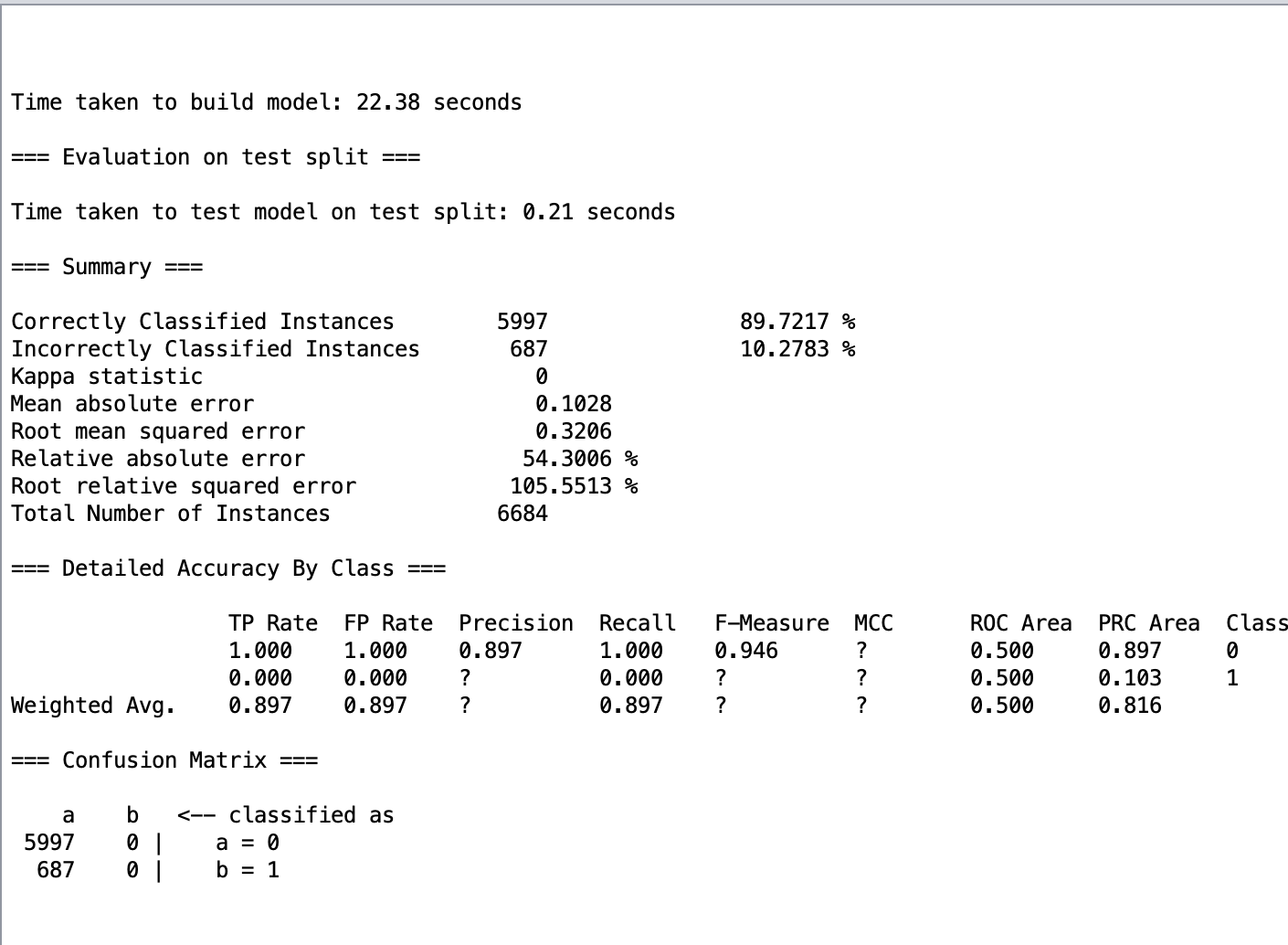
**Support Vector Machines (SVM) – Weka (functions.SMO)**

Support Vector Machine is a technique that is especially suitable for binary classification techniques. In our case, we have legitimate (low-risk) and fraudulent (high-risk) classes. SVM methods require large training dataset sizes to achieve maximum prediction accuracy.

Training time complexity of non-linear SVMs is between O(n2) and O(n3). Weka SMO seems to be closer to the latter. For instance, the execution time in ms for a random set with 205 features with Weka SMO looks like this:



Given the nature of the graph being hard to predict, but it could take anywhere from 1 day (n2) and 124 days (n3). Hence, we chose an alternative algorithm using linear kernel implementation: LIBLINEAR.



The measure of accuracy used here is a percentage split of 66%. We can see that the number of correctly classified instances are almost 89% showing that it is on par with the other classifying algorithms.