Report of COMP9417 ML & DM

The Analysis of Bank User Financial Status based on Machine Learning

Topic 0: Self-Raised Topic
COMP9417, Assignment 2, 2017s1

Group Members:

Xue Sun z5051698
Ying Chen z5007330
Jiawei He z5086661
Bingqian Shan z5082436

The Analysis of Bank User Financial Status based on Machine Learning

Xue Sun, Jiawei He, Ying Chen, Bingqian Shan

May 31, 2017

Contents

1 Introduction

Machine Learning and Data Mining Techniques are widely used in financial industries in terms of both classification and regression. Since economic crisis starting from 2008, financial institutions, especially banks, are focusing much more on the financial situation of their customers to prevent in advance the debt crisis and delinquency which could possibly lead to a second round of earthquake in the business industry. In this circumstances, banks always collect several information sources from the behaviour of their clients, which is proved to be an important data source used in Machine Learning. Such data could be used as a training data set to predict the financial status of future unknown clients. In this particular case, classification is utilised to label clients which provide the banks with information, such that if this specific client is going to have default payment in the next month. This would enable banks to achieve risk management and capital control in the increasingly competitive, undetermined monetary market.

2 Problem Description

This project is designed to predict the financial status that **whether a customer will have default payment or not in the next month**, which is an important attribute of the evaluation of client financial status in the bank system. The data set was collected from a prominent bank in Taiwan in the form of payment details with the total size of 25,000 customer records from April to October 2005, within which 5529, i.e. around 22.12% of customers have default payment. In the classification problem, a binary class label is used as table 1 to transfer this practical problem into a model of classification problem.

Semantic Meaning	default payment next month	no default payment next month
Label Value	1	0

Table 1: Class Label

Based on the data set described above, 8 classification methods are introduced and implemented by machine learning and prediction tests are conducted on a separate test data set in order to evaluate the performance of each classification method which predicts if a customer will have default payment next month. According to the prediction result which will be compared against the real ground truth in the test dataset, analysis of the reason of such performance will be presented by comparing the prediction performance based on the accuracy metrics (correct proportion, f1 score, etc.) of all classifiers as a practical application of theories learned from the lectures.
In the conclusion part, the best classification method with the optimised parameters will be nominated as a result of this project which is believed to be a best model that can be used by the bank for further predictions.

3 Domain Knowledge of Data Set

There are in total 23 attributes in the date set with all meanings presented below. All these attributes together consist the entire domain knowledge on the data set.

- X1: The amount of given credit including the customer credit and his/her family credit. (Measured in New Taiwan Dollar)
- X2: Gender (1=male and 2=female)
- X3: Education(1=graduate school,2 = university, 3=high school, 4=others)
- X4: Marital Status (1=married,2 = single, 3=others)
- X5: Age (Measured in Year)
- X6: Past Payment Record in September, 2005 (Measured in year)

  – Measured as: -1 = pay duly;1=payment delay for 1 month;2=payment delay for 2 monts;3=payment delay for 3 month; ... ;9=payment delay for 9 month; and the number is increased in this way possible toward infinity.

- X7: Past Payment Record in August, 2005 (Measured in year)
  – Measured as X6 but in August, 2005
- X8: Past Payment Record in July, 2005 (Measured in year)
  – Measured as X6 but in July, 2005
- X9: Past Payment Record in June, 2005 (Measured in year)
  – Measured as X6 but in June, 2005
- X10: Past Payment Record in May, 2005 (Measured in year)
  – Measured as X6 in May, 2005
- X11: Past Payment Record in April, 2005 (Measured in year)
  – Measured as X6 in April, 2005
- X12: The amount of Bill Statement in September, 2005 (Measured in New Taiwan Dollar)
- X13: The amount of Bill Statement in August, 2005 (Measured in New Taiwan Dollar)
- X14: The amount of Bill Statement in July, 2005 (Measured in New Taiwan Dollar)
- X15: The amount of Bill Statement in June, 2005 (Measured in New Taiwan Dollar)
- X16: The amount of Bill Statement in May, 2005 (Measured in New Taiwan Dollar)
- X17:The amount of Bill Statement in April, 2005 (Measured in New Taiwan Dollar)
- X18: The amount of Previous Payment in September, 2005 (Measured in New Taiwan Dollar)
- X19: The amount of Previous Payment in August, 2005 (Measured in New Taiwan Dollar)
- X20: The amount of Previous Payment in July, 2005 (Measured in New Taiwan Dollar)
- X21: The amount of Previous Payment in June, 2005 (Measured in New Taiwan Dollar)
- X22: The amount of Previous Payment in May, 2005 (Measured in New Taiwan Dollar)
- X23:The amount of Previous Payment in April, 2005 (Measured in New Taiwan Dollar)
- **Default Payment Next Month**
  – **This is the label of each instance:**
    ∗ **1 - The customer will have default payment next month.**
    ∗ **0 - The customer will have no default payment next month.**

The dataset described above was collected from the first hand records from the banking system. In order to test the performance of all classifiers implemented by various machine learning methods, cross-validation will be used by splitting the whole dataset in to training set and test set, such that the size for training is 80% of the total dataset while that for testing accounts for 20% of the entire data source.

4 Machine Learning Methods

A wide range of classification methods are used in this project. A brief introduction of each machine learning method is given for reference and information.

4.1 Naive Bayes

Naive Bayes is a probabilistic model based on Bayes theorem

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

When it comes to classification in the formula above presents the class label while B is the data instances. According to this, the revised Naive Bayes classifier equation is

$$P(C_i|x_1, x_2, ..., x_n) = \frac{P(x_1, x_2, ..., x_n|C_i)P(C_i)}{P(x_1, x_2, ..., x_n)}$$

Since the probability of the data instances is irrelevant to the classes predicted by the classifier so the formula can be converted to

$$P(C_i|x_1, x_2, ..., x_n) \propto P(x_1, x_2, ..., x_n|C_i)P(C_i)$$

The most significant characteristic of Naive Bayes is the assumption of independence between attributes, so the actual formula used in Naive Bayes is

$$\widehat{y} = \underset{i \in \{1,2,3,...,n\}}{\arg\max} = P(x_1|C_i)P(x_2|C_i), ..., P(x_n|C_i)|P(C_i)$$

Notice that n is the total number of classes and is equal to 2 in this binary classification case.

4.2 Decision Tree

The decision tree combines various rules in a hierarchical way with each node splitting instances into several subsets based on the best division criteria. All instances are divided in subsets using divide-and-conquer way until certain convergence threshold is met or pruning technique is applied. In this project, the tree depth is set manually by several experiments under cross-validation to find the best tree depth in order to avoid overfitting.

Normally, in order to choose the best attribute to split, two methods are used to target the best feature to select at each node, i.e. Information Gain and Gini.

4.2.1 Information Gain

Using entropy, the attribute with the largest splitting information gain is chosen.

$$Information\,Gain(S, A) = E(A) - E(S, A)$$

$$E(S, A) = \sum_{i=1}^{n} \frac{|S_i|}{S} E(S_i, A)$$

The information gain for each attribute splitting is calculated as above where $E(A)$ stands for the entropy of all instances at node A and $E(S_i, A)$ is the entropy of one of n subsets generated by splitting on node A. The attribute with the highest information gain is chosen as the 'best' attribute. Notice that n is the total number of classes.

4.2.2 Gini Index

Similar to information gain, Gini index uses probability squares to choose the best attribute. Gini index is calculated as below.

$$Gini(T) = 1 - \sum_{i=1}^{n} p_i^2$$

$$Gini_{split}(T) = \sum_{i=1}^{n} \frac{N_i}{N} Gini(T_i)$$

In the formula above, $Gini(T)$ is the Gini index at node T where p_i is the probability of the instances with the label i in node T. $Gini(T_i)$ is the Gini index of one of the subset after splitting and $Gini_{split}(T)$ is weighted by all nodes that are generated by splitting the node T to extend the tree. The attribute that gives the smallest $Gini_{split}(T)$ is chosen because it is expected that the impurity after splitting is as small as possible.

In this project, due to the type of tree (CART) that is chosen to be the tree to deal with numerical values, information gain, instead of gini, is used to split nodes.

4.3 Bagging on Random Forest

The Random forest essentially is an ensemble learning method on trees with baggin with replacement as a way to improve the performance of a single tree.The bootstrap trees with randomly chosen features are generated to predict on their own random sampling set respectively. Since this project only concerns the classification result, the resulting output of the random forest will be viewed as the majority voting of all bootstrap trees in the forest. This could be regarded as a ensemble extension of a single tree into a forest, which will generate better result for unstable models, i.e. trees.

4.4 Boosting on Decision Tree

Boosting is the second ensemble learning method used in this project. The most prominent difference between bagging and boosting on decision trees is:

- Bagging uses random sampling to reduce the size of the data set while Boosting operates on the whole data set.
- Boosting changes the proportion of misclassified and correctly classified instances based on the error rate in each iteration.
- The weight of each ensemble member is updated in each iteration and the final classification result is a weighted average of all outputs of ensemble members while Bagging simply takes the majority voting result.

4.5 KNN Classifier

KNN(K Nearest Neighbour) is a non-parametric classifier which predict an instance based on its K nearest neighbours. The advantage of KNN is that such algorithm does not require training which makes the model construction faster than its counterparts using the following selection method.

$$LabelClass = The\,Majority\,Class\,of\,\mathbf{K}\,NearestNeighbours$$

This guarantees the efficiency of model construction but requires much more computation in prediction because in the worst case all training data need to be considered, which exponentially increases the computation. In this project, the number of neighbours referred to will be optimized by Cross Validation of 10 folds.

4.6 SVM

Support Vector Machine is an augmented linear classification with boundaries consisting of support vectors. SVM is created to cope with classifications in one space within which all data points can not be linearly separated by mapping the original dimensions into higher dimensions with the hope that they can be linearly separated by a hyper plane in higher dimensional space. Assume the original data space is p-dimensional. The data instance is

$$\overrightarrow{v} = \langle v_1, v_2, v_3, ..., v_{p-1}, v_p \rangle$$

By using kernel tricks, the original data based on selected support vectors is mapped into q-dimensional space where $q \geq p$:

$$\overrightarrow{s} = Kernel(\overrightarrow{v}) == \langle s_1, s_2, s_3, ..., s_{q-1}, s_q \rangle$$

Linear functions are applied on the space of q dimensions using the function as below:

$$\widehat{y} = \theta_0 + \theta_1 s_1 + \theta_2 s_2 + ... + \theta_{q-1} s_{q-1} + \theta_q s_q$$

In this higher dimension, the data set which was not linearly separable can be separated using the hyperplane and the cost function of SVM can guarantee that the margin between the gap of two classes is as large as possible.

4.7 Logistic Regression

Logistic Regression is the classification method based on Linear Regression, with the sigmoid function to convert the output between the range of 0 to 1. The model function of Logistic Regression is

$$\widehat{y} = \frac{1}{1 + e^{-h_\theta(x)}}$$

$$h_\theta(x) = \theta_0 + \theta_1 s_1 + \theta_2 s_2 + ... + \theta_{q-1} s_{q-1} + \theta_q s_q)$$

The output is the probability of the label being 1, therefore Logistic Regression predicts 1 if $\widehat{y} \geq 0.5$ and predicts 0 otherwise. In order to learn the best parameters of $\{\theta_0, \theta_1, \theta_2, ...\theta_{q-1}, \theta_q\}$, the gradient descent on the cost function with regularization

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)} log(h_\theta(x^{(i)})) + (1 - y^{(i)}) log(1 - h_\theta(x^{(i)}))] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

will guarantee that the MSE of the classification will be minimised with restriction on overfitting.

4.8 Neural Network

In this project, multi-layer neural network with sigmoid function is used to achieve the non-linear classification with hidden nodes and output nodes in the network. The weight of each neural unit is updated using backpropagation to ensure MSE is minimised. In this neural network, batch mode is implemented by gradient descent

$$\omega = \omega - \eta \nabla E_D[\omega]$$

In the backpropagation process, the gradient descent Error function is defined as

$$E_D[\omega] = \frac{1}{2}\sum_{d \in D}(t_d - o_d)$$

where if the weights are of hidden nodes, t_d is the target output of such hidden node and o_d is the actual output of such hidden node while if the weights are of the output node, t_d is the target output of such ouput node and o_d is the actual output of such output node.

4.9 Summary

All brief descriptions for all machine learning methods are presented above which explains how each method works. However, in specific implementation scenarios, detailed parameters and classification results will be provided and elaborated in the chapter 6 and the chapter 7.

5 Data Preprocessing and Data Reuction

5.1 Data Correlation Analysis

Due to high dimensionality curse, uncorrelated features should be eliminated from the data set in order to reduce the noise and bias. All features described in chapter 3 was tested on correlation coefficient with the class label- **Default Payment Next Month**-with together with an definitely unrelated extra and added feature ID which works as a counter of dataset instance from 1 to 25,000 (the total size of the training dataset). The result is displayed in table 2.

Feature	Correlation Coefficient	Absolute Correlation Coefficient	Selection
Class Label	1.000000	1.000000	Selected
X6	0.324794	0.324794	-
X7	0.263551	0.263551	Selected
X8	0.235253	0.235253	Selected
X9	0.216614	0.216614	Selected
X10	0.204149	0.204149	Selected
X11	0.186866	0.186866	Selected
X1	-0.153520	0.153520	Selected
X18	-0.072929	0.072929	Selected
X19	-0.058579	0.058579	Selected
X21	-0.056827	0.056827	Selected
X20	-0.056250	0.056250	Selected
X22	-0.055124	0.055124	Selected
X23	-0.053183	0.053183	Selected
X2	-0.039961	0.039961	Selected
X3	0.028006	0.028006	Selected
X4	-0.024339	0.024339	Selected
X12	-0.019644	0.019644	Selected
X13	-0.014193	0.014193	Selected
X14	-0.014076	0.014076	Selected
ID	-0.013952	0.013952	NA
X5	0.013890	0.013890	Not Selected due to Low Correlation
X15	-0.010156	0.010156	Not Selected due to Low Correlation
X16	-0.006760	0.006760	Not Selected due to Low Correlation
X17	-0.005372	0.005372	Not Selected due to Low Correlation

Table 2: Correlation Coefficients

From table 2, It can be seen that attributes including age(X5), the amount Bill Statement from April to July (X15-X17) have extreme low correlation, even lower than the counter ID. Since data reduction based on correlation coefficients is quite primitive and straightforward, therefore, such dimension reduction is used on the classifiers such as **Decision Tree, Logistic Regress, Naive Bayes, Random Forest, KNN**. The data used by Neural Network, SVM and Boosting will be treated using a more complicated data reduction methods, i.e. PCA to further reduce dimensions to fewer numbers since these methods requires more computation. As a result, the new vectors have 19 features instead.

Notice that after data reduction, all features are also normalised using max-min normalisation function

$$v_{new} = \frac{v_{old} - v_{max}}{v_{max} - v_{min}}$$

in order to formalise the influence of each feature.

5.2 Principle Component Analysis

Principle Component Analysis is a more efficient way to reduce the data dimensions so that the computation intensity is greatly reduced while the main characteristics and distribution of the data set is maintained. The main steps of PCA is described below.

- Calculate the covariance matrix of the feature vectors $\overrightarrow{v}$ as Matrix Σ which is a n by n matrix where n is the original number of dimensions of the data.
- Assume the new dimension is $k \leq n$, find n Eigenvectors of the covariance matrix and choose the first k Eigenvectors which have the top k highest eigenvalues to form a n by k matrix Σ'.
- The new vector can be calculated as
$$\overrightarrow{v'} = \Sigma'^T \overrightarrow{v}$$
$\overrightarrow{v'}$ is a k-dimension vector as a result.

In this project, the original data set has 23 dimensions and the covariance matrix calculated can be found in the document of 'PCA figure.docx'. Based on the covariance matrix, the sorted eigenvalues are given in table 3 .

λ_1	λ_2	λ_3	λ_4
0.610437013	**0.295353813**	3.05241925e-02	1.69285926e-02
λ_5	λ_6	λ_7	λ_8
9.42041926e-03	9.04174545e-03	7.54445790e-03	6.38481379e-03
λ_9	λ_{10}	λ_{11}	λ_{12}
5.83708771e-03	2.96671412e-03	2.38506002e-03	1.69774353e-03
λ_{13}	λ_{14}	λ_{15}	λ_{16}
1.47834530e-03	1.83162337e-09	9.37745809e-11	1.94256625e-11
λ_{17}	λ_{18}	λ_{19}	λ_{20}
1.24761563e-11	1.17818918e-11	7.83531981e-12	5.77630757e-12
λ_{21}	λ_{22}	λ_{23}	
5.37275222e-12	4.64414890e-12	3.85781359e-12	

Table 3: Eigen Values

It is obvious that the first two eigenvalues dominate all others so that the dimension of the new space can be reduced to two, which is believed to be a efficient and drastic reduction on the redundancy of data. The classifiers which require high computation will use this reduced data set as a way to improve the efficiency of the algorithms, including **Neural Network, SVM and Boosting on Decision Trees**.

In summary, correlation coefficient reduction reduces the number of dimensions to 19 while PCA converts the vector space to two. And the match-ups between classifiers and the dimension-reduced data set is used are summarised in table 4.

Classifier	Data Dimension Reduction Method
Decision Tree	Cerrelation Coefficient
Logistic Regression	Cerrelation Coefficient
Random Forest	Cerrelation Coefficient
KNN	Cerrelation Coefficient
SVM	PCA
Neural Network	PCA
Boosting	PCA

Table 4: Classifiers and their input data preprocessing methods

6 Classification Results and Analysis

6.1 Naive Bayes

The Naive Bayes Model used in this project is Multinomial Naive Bayes which takes the number of occurrence into consideration. However, as Multinomial Naive Bayes odes not work well on negative data and also each attribute should have a similar range such that bias data distribution is avoided. In order to normalise features, max-min normalisation is achieved to convert the value of features between [0,1] by the formula below.

$$v_{new} = \frac{v_{old} - v_{max}}{v_{max} - v_{min}}$$

Also, 10 fold cross validation is used to test the accuracy of Naive Bayes classifier prediction. The classification result is shown in table 5

CV Round	Accuracy on Training Set	Accuracy on Test Set	f1 score Micro
1	0.778541666667	0.779833333333	0.779833333333
2	0.780166666667	0.773333333333	0.773333333333
3	0.7785	0.78	0.78
4	0.777791666667	0.782833333333	0.782833333333
5	0.779416666667	0.776333333333	0.776333333333
6	0.779875	0.7745	0.7745
7	0.775625	0.7915	0.7915
8	0.778083333333	0.781666666667	0.781666666667
9	0.778458333333	0.780166666667	0.780166666667
10	0.778166666667	0.781333333333	0.781333333333
Average Accuracy on Test Set	**0.78015**		

Table 5: Naive Bayes Classification Result

The Average Accuracy on Test Set -namely 0.78015- is the result of prediction on test data and it can be noticed that the accuracy on test set is always equal to the f1 micro score. Judging from the results of the classification, the performance of Naive Bayes is not satisfactory to meet the prediction accuracy demand as 78% of correct prediction may cause many clients to be misclassified. Several reasons originated from the natural essence of Naive Bayes implementation causes this phenomenon.

- Naive Bayes assumes the conditional independence between attributes however there are a lot of attributes that are correlated with each other, for example, the attributes X6-X11 are past payment records and X12-X23 are the amount of Bill Statement from April-September, 2005. These attributes are clearly not independent with each other becuase it is obvious that the payment and statement amount for one person are likely to have similar amount during the period of six months. Therefore the assumption of independence between attributes would not stand.
- The probabilities of $P(C_{default} = 1)$ and $P(C_{non-default} = 0)$ have large difference such that $P(C_{default} = 1) = 22.12\%$ and $P(C_{non-default} = 0) = 77.88\%$. The factor that reduces the accuracy is the fact the $P(C_{default} = 1) \ll P(C_{default} = 0)$. Since the prior probability of each class is an important factor of the output of Naive Bayes, the output of predicting 'not default 'will be more likely to be much larger than the output of predicting 'default '. This causes the biased prediction of Naive Bayes.

Based on these two arguments, Naive Bayes is not suitable as expected under such data distribution and application scenario.

6.2 Decision Tree

Decision Tree is rule-based classifier which handles the categorical and numerical values efficiently. There are two splitting criteria, i.e. Gini index and information gain that are widely used in the industry of data mining. There are several versions of tree with different methods of pruning and optimization methods to improve the accuracy and avoid overfitting, such as ID3, C4.5, C5.0 and CART. The tree used in this project is Classification and Regression Tree (CART) because such tree treats numerical values better with information gain based on entropy. Also, several experiments with different tree depths were conducted to find the best tree depth for such data set in order to avoid overfitting.

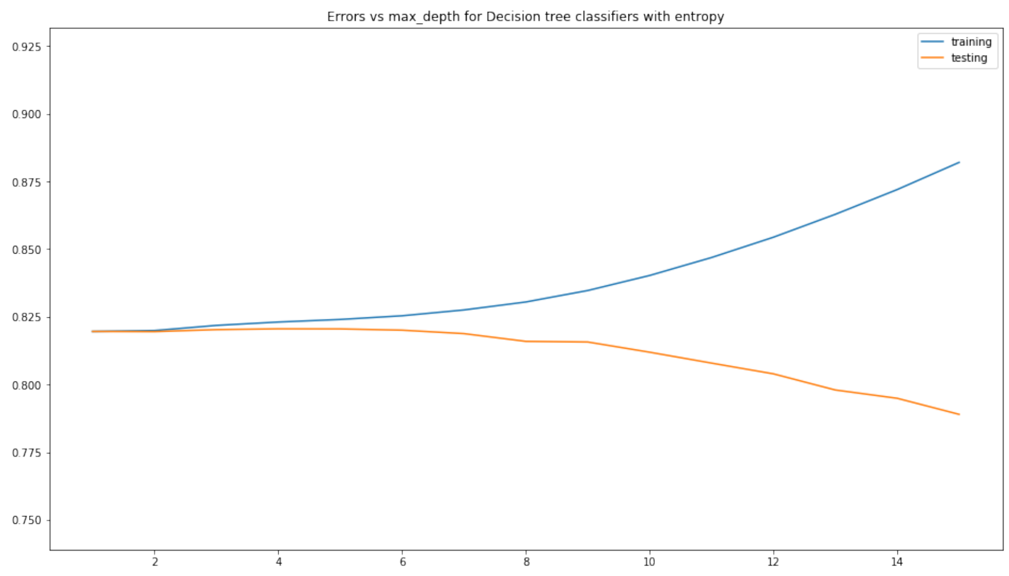

Figure 1: Decision Tree Error with Depth on Training and Test Set

The result of depth experiments are shown in figure 1 from which we can see that when the depth of the tree is lower than five, there is no big difference between the accuracy of training and test data set, i.e. around 0.82. When CART tree is higher than 5, it is obvious that the error on training data set is reducing dramatically while that on test data set is degenerating greatly which is a eminent sign of overfitting. Therefore, the maximum depth of the tree will be limited to five.

CV Round	Accuracy on Training Set	Accuracy on Test Set	F1 micro score
1	0.826208333333	0.814833333333	0.814833333333
2	0.824416666667	0.822	0.822
3	0.823125	0.8225	0.8225
4	0.821708333333	0.8265	0.8265
5	0.825583333333	0.8185	0.8185
6	0.824958333333	0.819166666667	0.819166666667
7	0.825166666667	0.816333333333	0.816333333333
8	0.827416666667	0.808	0.808
9	0.826458333333	0.8115	0.8115
10	0.825666666667	0.8165	0.8165
Average Accuracy on Test Set	**0.817583333333**		

Table 6: DecisionTree Classfier 10 times Cross validation Result

When all parameters are properly set in the Decision Tree classifier, it is run by cross validation with 10 folds and the result is retrieved in the table 6. The resulting accuracy on training and testing set has no significant difference between one and the other, indicating the fact that no overfitting occurs. The average testing accuracy is 0.82 which is better than Naive Bayes. This is because Decision Tree does not sever the dependency between attributes and cope well with the data set with a mixture of numerical and categorical features like the data set used in the project.

6.3 Bagging on Random Forest

Since decision tree has a fairly acceptable accuracy on the bank user prediction of test data, it is worth trying if several trees are combined together in an ensemble. According to the mechanism of bagging which creates the subsets of data by random sampling. In the bootstrap, 80% of data are randomly selected to generate subsets while 80% of features are randomly selected to generate sub-features in order to create models for each tree in the forest. The experiments used to find the best number of trees gives the following results in figure 2.

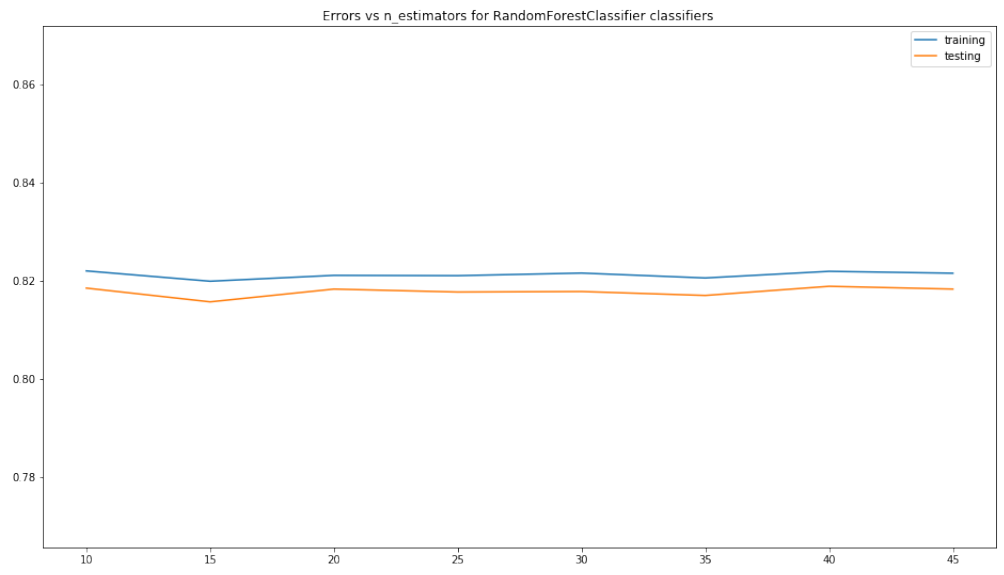

Figure 2: Random Forest Error with Number of Trees in the Ensemble

The figure 2 proves the fact the in this data mode, the number of trees from 10 to 45 does not have to much effect on the accuracy on both training error and testing error. Therefore, the number of trees in the forest is set to 10 in order to make classification faster.

The next parameter in the random forest is the maximum depth of trees in the forest. Figure 3 indicates how the accuracy will be affected along with the maximum depth of trees. Similar to single tree model, when the maximum depth of trees is less than 5, the accuracy of testing and training is pretty similar to each other while the testing error will increase and the training error decreases drastically when the depth of trees rises. Thus, the maximum depth of trees is set to 5 indeed.

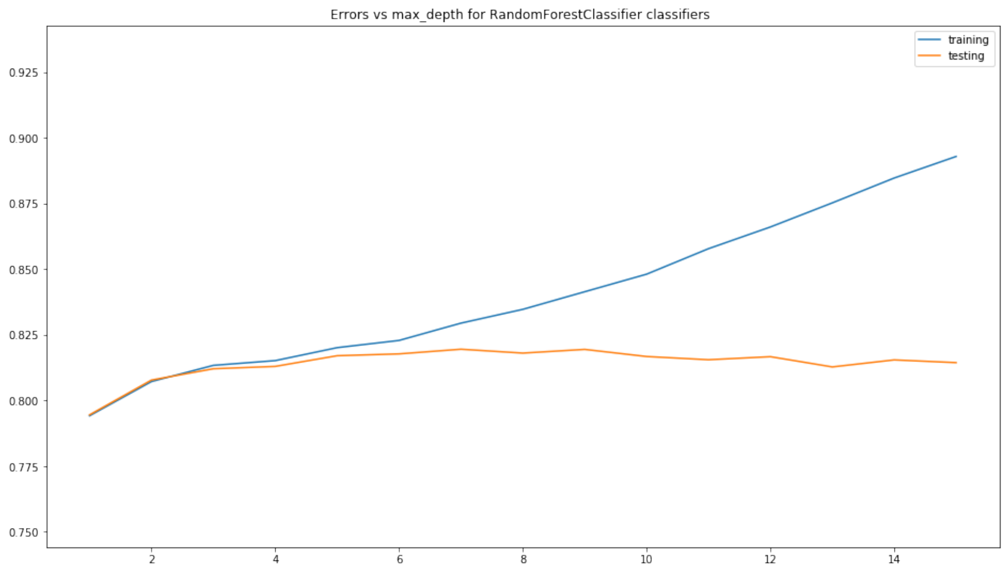

Figure 3: Random Forest Error with Tree Depth

CV Round	Accuracy on Training Set	Accuracy on Test Set	F1 micro score
1	0.831625	0.820666666667	0.820666666667
2	0.8355	0.819166666667	0.819166666667
3	0.835458333333	0.823666666667	0.823666666667
4	0.837416666667	0.811	0.811
5	0.836	0.817166666667	0.817166666667
6	0.834208333333	0.8215	0.8215
7	0.835625	0.815333333333	0.815333333333
8	0.833458333333	0.816833333333	0.816833333333
9	0.83825	0.816	0.816
10	0.838166666667	0.814666666667	0.814666666667
Average Accuracy on Test Set	**0.8176**		

Table 7: RandomForestClassifier 10-estimators 5-depth 10-times Cross validation Result

Since there are 10 trees in the forest and the maximum depth is set to 5, the ensemble model is tested by training and testing, resulting in the accuracy results in table 7. The accuracy of random forest is 0.82 which has acceptable results in the banking industry. Notice that the ensemble actually does not increase the accuracy as much as is expected. This showcases the terminology that in large dataset, sometimes, the simpler the model is, it is likely to have more satisfactory results, however. There is another reason why Bagging can maintain the performance, that is, the classification distribution is biased with the proportion of non-default users being more than default-users and sampling on instances and features randomly can reduce such influence of bias distribution from original data.

6.4 Boosting on Decision Trees

Boosting mechanism includes more than one classifiers in the ensemble, and each classifiers is informative and focus essentially the error made by previous classifiers. Each classifier is trained on the same data set with the proportions of each class being modified according to the performance, error rate, of precedent classifiers. Since tree has impressive on the prediction in the previous classification, boosting is used to promote the performance of groups of trees. The result is shown in figure 4 with the variation of error to the number of trees in the ensemble.

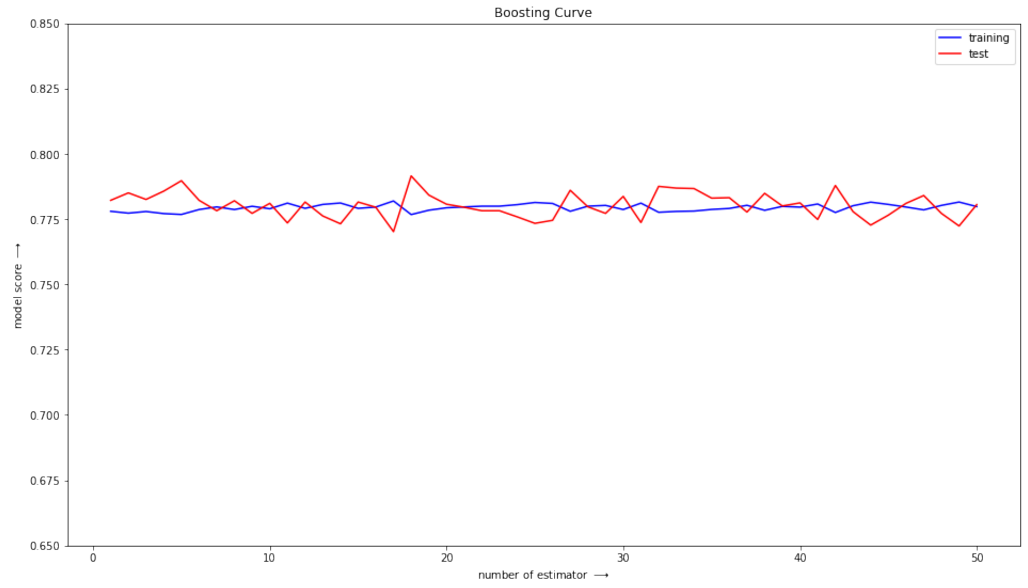

Figure 4: Boosting Error with Number of Trees in the Ensemble

From 4, it can be calculated that the average error rate is **0.77995** with the number of estimators ranging from 1 to 50. The error of test data varies drastically beyond and below the training error as the number of estimators is increasing. This means the number of trees combined by boosting does not cooperate well with each other. This is because all classifiers in the ensemble are trees so that the following classifier can not cope much better under the same error that was made by the previous classifier with the same model. Generally, boosting performs better than other ensemble methods because it can integrate various kinds of classifiers. In this occasion, because it is just a comparison of performance under different implementation of trees, boosting, on the contrary, does not perform as good as expected. It is believe that combining more different kinds of classifiers would increase the accuracy much further.

6.5 KNN

K-Nearest Neighbour classification does not use training to learn, it will just create a hypothesis space using all data instances. When a new instance arrives, its classification is subject to the majority or the weighted combination of the nearest k neighbours.

KNN will be robust against noise while the number of neighbours are increasing so does the computation demand. KNN can behave well on data set with numerical value and categorical values all together. In order to find the best number of nearest neighbours, the classifier is tested by different numbers of neighbours from 1 neighbour to 16 neighbours. The result is attached in figure 5.

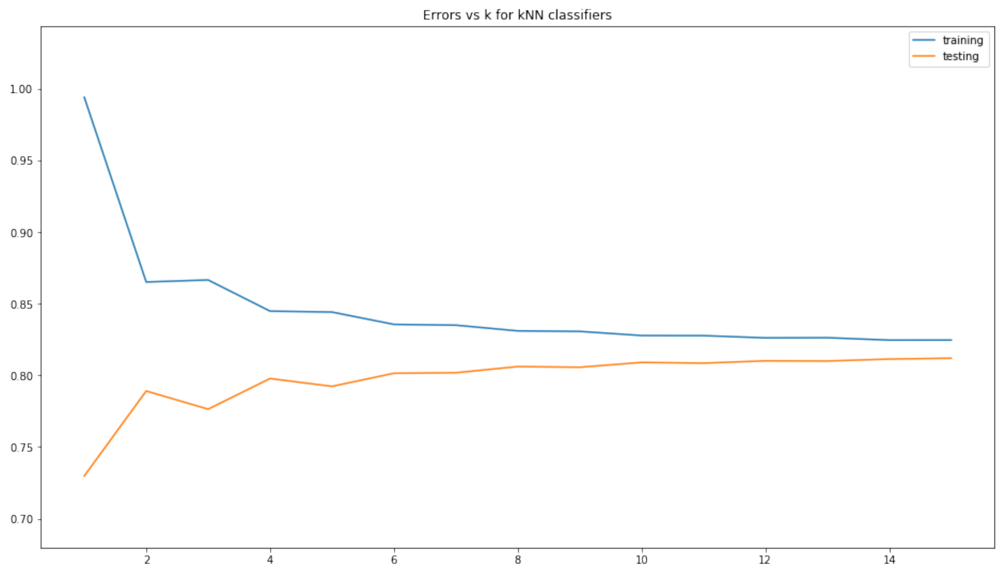

Figure 5: KNN Error with the Number of Neighbours

From the result in 5, the number of nearest numbers will raise the accuracy greatly when such number increases from 1 16. However, when the total number of neighbours is larger than 8, the accuracy will not increase much further and the accuracy remains relatively stabel. This make it meaningless to further increase the number of neighbours beyond 8. So k is set 8 as a result. The training error and the test error is display in table 8.

CV Round	Accuracy on Training Set	Accuracy on Test Set	F1 micro score
1	0.830916666667	0.810666666667	0.810666666667
2	0.833208333333	0.803833333333	0.803833333333
3	0.830166666667	0.811333333333	0.811333333333
4	0.829416666667	0.812	0.812
5	0.83075	0.798333333333	0.798333333333
6	0.831125	0.8035	0.8035
7	0.8305	0.804333333333	0.804333333333
8	0.830625	0.806166666667	0.806166666667
9	0.831875	0.798833333333	0.798833333333
10	0.83	0.810666666667	0.810666666667
Average Accuracy on Test Set	**0.805966666667**		

Table 8: KNeighborsClassifier 10 times Cross validation Result

Given table 8, KNN with 8 neighbours gives the prediction accuracy on average of 0.8 which is better than Naive Bayes. It is worth mentioning that the main reason why KNN does not have essential improvements compared to Tree and Random Forest is that since there are about 20 dimensions. KNN can easier get stuck in to the 'the curse of dimension'which means two similar instances can be far apart by any unrelated features in high dimension.

Now consider an other dimension reduction process, i.e. PCA. Since PCA will only remain two most significant features in this project, when running KNN on a data set with only two features, the classification would be too arbitrary so that the accuracy will not improve too much. The experimental results of KNN with PCA was also conducted and the average accuracy was 0.77 which is not as good as the data set presented in this report, therefore the detailed analysis of KNN with PCA will not be presented but the relative codes will be attached for reference. In conclusion, KNN does not behave well in both scenarios so KNN is not one of the best candidate of good classifier.

6.6 Logistic Regression

Logistic regression is the implementation of linear regression as classification using sigmoid function to re-scale the value of output between 0 and 1. The best advantage of logistic regression is that this classifier can deal with high dimensional feature space fast and efficiently using gradient ascent to maximise the likelihood of the probability of classes. The classification result of Logistic Regression is presented in table 9.

CV Round	Accuracy on Training Set	Accuracy on Test Set	F1 micro score
1	0.81275	0.8	0.8
2	0.808166666667	0.818166666667	0.818166666667
3	0.810583333333	0.806166666667	0.806166666667
4	0.808791666667	0.8125	0.8125
5	0.809625	0.809166666667	0.809166666667
6	0.810416666667	0.810333333333	0.810333333333
7	0.811875	0.806833333333	0.806833333333
8	0.811708333333	0.803166666667	0.803166666667
9	0.808083333333	0.819166666667	0.819166666667
10	0.809833333333	0.810833333333	0.810833333333
Average Accuracy on Test Set	**0.809633333333**		

Table 9: LogisticRegression 10 times Cross validation Result

The average accuracy on test set is 0.81 which is quite similar to KNN but still performs better than Naive Bayes. This is mainly because Logistic Regression also takes the conditional independence into consideration. This makes the learning process longer and harder where each weight is updated using gradient ascent, moving to the optimal position gradually but Naive Bayes has nothing to do with it. It can be shown that the test accuracy is at the same level with training data accuracy which means the model is not overfitting to the training data.

6.7 SVM

SVM transfer the original data into other data space for the hope that the data will be linearly separable in new feature space, which is a promotion of linear classifier. Since Logistic Regression has good performance on prediction, in SVM, instead of using very complicated kernel, Linear Kernel is the right target model to do SVM Machine Learning to predict. SVM in this project is tested with different penalty parameter C in the cost function of linear kernel, which is shown in figure . The average accuracy of SVM is **0.78** which is worse than logistic regression.

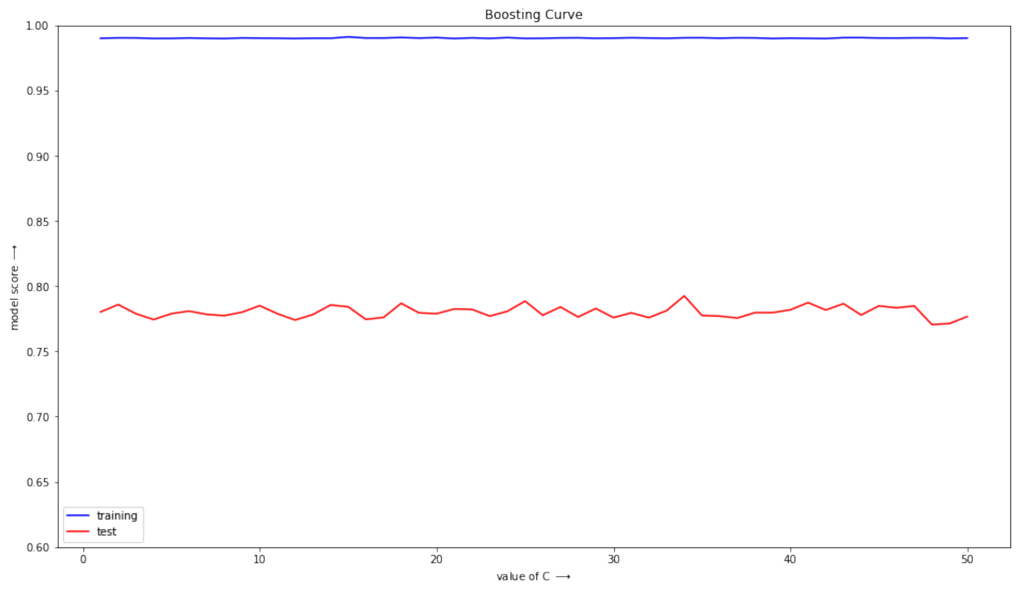

Figure 6: SVM Error with Regularization Penalty

From figure 6, the reason why SVM performs worse than logistic regression is overfitting which is obvious and essential even though the overfitting penalty parameter ranges from 0 to 50. The accuracy of training data is always close to zero while such accuracy on test set is only less than 0.8. This means if SVM is used, we should choose more complicated kernel instead of simply using linear kernel, This is due to the fact that the kernel suits training data much better than test data, resulting in overfitting.

6.8 Neural Network

Neural network consists of several neurons formed in multiple layers. In order to update the weights in hidden layers and output layers, back-propagation is used to find the optimal weights using gradient descent. Notice that the model runs on PCA data of two dimensions which makes neural network faster. The number of layers is related to the accuracy of training and testing. The number of layers from 1 to 50 is used to test which kind of neural networks works better.

Figure 7 displays the variance of accuracy with the number of layers in the neural network with back propagation gradient decent. It is obvious that the accuracy of ANN fluctuates drastically when the number of layers is fewer than 25, with the worst case of 0.57 of 14 layer. On the contrary, when the number of layers is larger than 25, the accuracy, both in training set and testing set, stables at 0.77. Therefore, in the bank default payment scenario, the number of layers in neural network should be large than 25.

Moreover, the average accuracy of prediction of neural network of 1 to 50 layers is calculated and the result of the average mean is **0.76393**. This accuracy is even lower than Naive Bayes because Neural Network using gradient descent can get stuck in the local optimal so that when stuck into the local optimal, it is impossible for ANN to get rid of the local optimal by increasing the error function temporarily to find the global optimal once and for all.

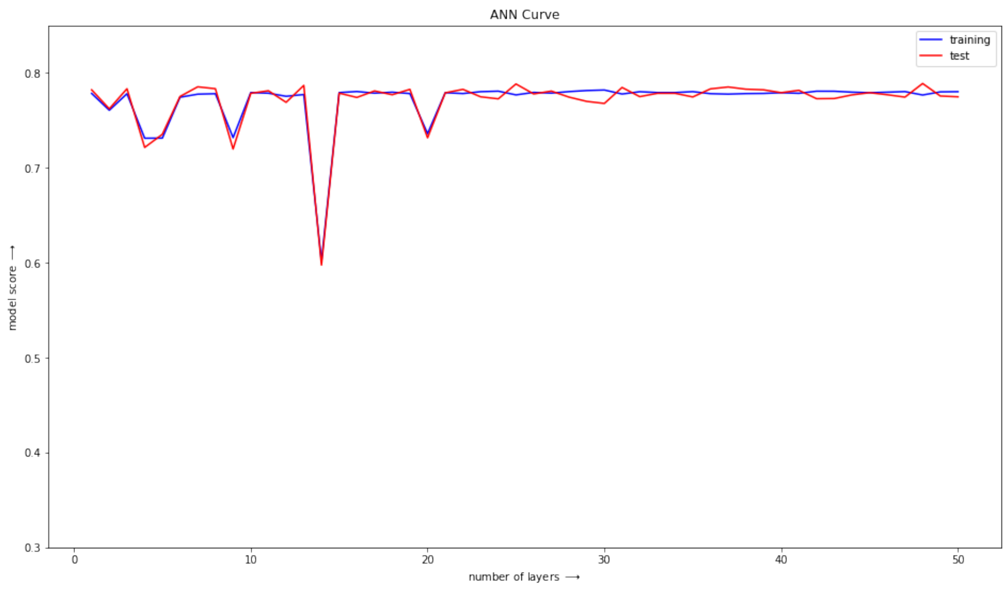

Figure 7: Neural Network Error with Network Layers

7 Conclusions

All results are displayed in table 10.

Machine Learning Algorithm	Average Accuracy
Naive Bayes	0.78
Decision Tree	***0.82***
Bagging on Random Forest	***0.82***
Boosting on Decision Tree	0.78
KNN	0.81
Logistic Regression	0.81
SVM	0.78
Neural Network	0.76

Table 10: Summary of Performance

Based on the training, testing, analysis of 8 different classifiers, it is shown that Decision Tree and Bagging on Random Forest give the best performance on test data with an accuracy of 82% which can be regarded as a valuable reference for bank mangers to predict if a customer will have default payment in the next month with the confidence of over 80%.

When it comes to specific algorithms, it can be observed that non-linear classifiers , for example Naive Bayes, SVM and Neural Network, do not have good performance on the test data while linear classifiers such as Logistic Regression has better accuracy compared to its non-linear counterparts. On the other hand, non-parametric method, e.g. KNN, can compete with linear classifiers because KNN with 8 neighbours possibly forms similar decision boundaries with Logistic Regression, making the performance pretty similar. Most importantly, Decision Trees with depth of 5 and entropy based splitting presents the best performance. This is because Decision Trees are rule based classifier which makes decision in a way much similar to what real human are doing by setting rules and making predictions based on the results of each rule.

To make things complete, Bagging and Boosting are introduced as two different methods of promoting the score of a single Decision Tree. However, these two methods have discrepancy in their performance of prediction on test set. Bagging obviously performs much better than Boosting because the training data set is skewed, with the number of non-default instances more than its non-default companions in the dataset. *The random sampling in the bagging ensemble can reduce such biased distribution so that* *the accuracy will not be affected.* On the contrary, since all models in boosting are trees, it is hard for the following same model to solve the previous problem left by precedent trees.

To make things clear, the data set is much suitable for rule based classification and Decision Trees and Random Forests are the two classifiers that give the best performance.

8 Code Attached

There are several code files in python source code files or jupyter notebooks to support the argument in this report. There names and brief introduction are referred to here. This part is also written in README.txt. All relevant materials are stored in the folder called files submitted together with this report.

In the folder of Classifier:

- KNN_Coeff.ipynb: The implementation and test for KNN on data dimension reduction by correlation coefficient.
- KNN_PCA.ipynb: The implementation and test for KNN on data dimension reduction by correlation coefficient.
- Decision tree classifier.ipynb: The implementation and test for Decision Tree.
- LogisticRegression.ipynb: The implementation and test for Logistic Regression.
- MultinomialNB.ipynb: The implementation and test for Naive Bayes
- RandomForestClassifier.ipynb: The implementation and test for Random Forest.
- SVM.ipynb: The implementation and test for SVM.
- Boosting.ipynb: The implementation and test for Boosting Ensemble.
- ANN.ipynb: The implementation and test for Neural Network.

In the folder of Data Pre-processing:

- Data preprocessing correlation.ipynb: the data pre-processing code for correlation coefficient.
- PCA_Transfer.py: the codes used for PCA data pre-processing.
- PCA figure.docx : PCA eigenvalues and PCA coefficient matrix.

In the folder of Data Source:

Notice that due to the data quota limit in the submission system, these three data files are stored on-line instead in Github. In the link below.

`https://github.com/railsgem/ML-DM-Bank-Analysis-Data-Source`

Just open the file Data Source, all three files below can be found.

- default of credit card clients.xls: the original data set in xls form.
- default of credit card clients.csv: the data set in csv form for the convenience of input.
- PCA_Transform.csv: the PCA transferred data in csv form.

Please also notice that all files in Data Source have to be put in the ***SAME Directory, i.e SAME FILE*** with code files and pre-processing files in order to run the code without error.

If there is anything wrong with compiling or running the codes, please contact any member of this group via UNSW e-mail. Thank you.

9 Acknowledgement

- Some materials derived for this report is from the Machine Learning tutorial of Andrew Ng 'Machine learning coursera', https://www.youtube.com/playlist?list=PLZ9qNFMHZ-A4rycgrgOYma6zxF4BZGGPW
- Some implementation of machine learning methods, e.g. Decision Trees, Naive Bayes, Logistic Regression, etc. are based on sklearn machine learning packages in Python, http://scikit-learn.org/stable/

References

[1] Yeh, I-Cheng, and Che-hui Lien. ”The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients.” Expert Systems with Applications 36.2 (2009): 2473-2480.

[2] Pedregosa, Fabian, et al. ”Scikit-learn: Machine learning in Python.” Journal of Machine Learning Research 12.Oct (2011): 2825-2830.

[3] Dietterich, Thomas G. ”An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization.” Machine learning 40.2 (2000): 139-157.

[4] McCallum, Andrew, and Kamal Nigam. ”A comparison of event models for naive bayes text classification.” AAAI-98 workshop on learning for text categorization. Vol. 752. 1998.

[5] Witten, Ian H., et al. Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann, 2016.