Machine Learning on Banking Marketing Campaign Strategy

COMP9417 Assignment 2

Topic 0: Self-proposed

z5011499 RUOYU YANG

z5029972 CHANG MA

z5094756 HAONAN LIU

z5004850 TENGYU MA

1.Introduction

1.1 Overview

Machine Learning(ML) plays a key role in contemporary business. Studying consumers’ behavior helps companies to maximize profit with the least manpower in marketing campaign. Client information collected by previous telemarketing campaign is a very useful machine learning training set. ML system analyze these attributes provided in training dataset, and predict the final purchasing decision of customer. Therefore, in telemarketing campaign, ML system will become a recommendation system which provide contacts of clients who are potential clients for telemarketing saler. Through this system, company can minimize marketing manpower and maximize deals.

We proposed a Machine Learning project which predicts the success of tele-market call for selling long-term deposits. The data set was collected from a bank in China, which launched a telemarketing promotion for its long-term deposits in 2011. Eight classical and typical ML algorithms were adopted in building prediction model and tested by a data set collected in 2012. This report will compare all prediction models and analyze bias generated by each model, explain accuracy by using knowledges in COMP9417 course and our study.

1.2 Data set

**Title:** Bank Marketing

**Relevant Information:**

The data is related with direct marketing campaigns of a Chinese bank. The marketing campaigns were based on phone calls records with 45000+ contacts. Often, more than one contact to the same client was required, in order to access if the product (long-term deposit) would be (or not) subscribed.

There are two data set, larger one called bank-full.csv contains 45211 instance of contact records from campaign in 2011 will be used as training data. Data set bank.csv collected in 2012 campaign will be used as verifying data set, which contains 4521 new records. Each data set contains 16 attributes (6 numeric variables, 9 categorical variables) and 1 output attribute (whether the client buy a long-term deposit).

2. Preparation

**Dimension Reduction:** Correlation Analysis

Correlation analysis calculates the correlation coefficient between attributes by measuring linear association between variables. The calculation values are always between 1 and -1.

The absolute value of correlation coefficient indicate the strength of the association between two variables. In order to prevent overfitting, dataset should be cleaned by removing high associated attributes.

In this project, we set threshold of correlation coefficient at [-0.4, 0.4]. Any two or more variables obtain the correlation coefficient larger than 0.4 or less than -0.4 will be treated as high correlation. In practice, if a group of attributes are detected as high related, an analysis of correlation between each attribute and decision y will be measured to decide which attribute in the group should be reserved (Generally, the one with highest relation with decision y will be reserved).

|  |
| --- |
| > cor(data\_num\_1)                   pdays             previous            poutcome  pdays         1.000000000          0.454819635         0.79080600  previous      0.454819635          1.000000000         0.48134118  poutcome      0.790806005          0.481341179         1.00000000  > cor(data\_num\_1$poutcome,data\_num$y) #keep out poutcome var; remove pdays previous  [1] -0.2211275  > cor(data\_num\_1$pdays,data\_num$y)  [1] -0.1036215  > cor(data\_num\_1$previous,data\_num$y)  [1] -0.09323577 |

The correlation computed shows that the attributes: pdays, previous and poutcome are highly related to each other(According to definition of those attribute and general knowledge, they are indeed highly associated). Meanwhile, poutcome is the most highly related to client decision y. Therefore, we remove the attributes: pdays, previous.

3. Method and Analysis:

3.1 Association rule learning (Apriori Algorithm)

Apriori algorithm measures transactional databases over frequent item set mining and association learning. It take a bottom-up method, extended frequent item set levelly by identifying whether each individual frequent item set’s sufficiently appears in database. The frequent item set generated by Apriori could use to identifying association rule in machine learning. This method is generally applied in marketing analysis.

A brief explanation of the target could be: analyze the dataset and determine contact with what kinds of attributes have more possibility to buy or not buy a long-term deposit. Therefore association rule learning shall be applicable.

Data generalization should be processed for this data set. Here is an prediction of result: if we get a result like {age = 58, balance = 1000, duration = 600, pday = 103} =>{y=yes}, this rule is useless in general as there is very low probability that another contact happen to be 58 with exactly 1000 balance and talking 600s with the telemarketer and there were 103 days past since the last time he or she was contacted in previous campaign. Furthermore, without data classifying, it is almost impossible to get a specific rule because some numeric attributes got more than 1000 factors such as “age”, “balance”. So attributes with numeric value were generalized, which described in Attribute.txt file.

|  |
| --- |
| > rules <- apriori(myfacdata,parameter = list(minlen=5, supp=0.3, conf=0.95),appearance = list(rhs=c("y=yes", "y=no"),default="lhs"),control = list(verbose=F))  > subset.matrix <- is.subset(rules.sorted, rules.sorted)  > subset.matrix[lower.tri(subset.matrix, diag=T)] <- NA  > redundant <- colSums(subset.matrix, na.rm=T) >= 1  > rules.pruned <- rules.sorted[!redundant]  > inspect(rules.pruned)  lhs                                                                  rhs    support   confidence lift  1 {Fac.age=mid,Fac.duration=long,Fac.pday=new,Fac.pervious=new}     => {y=no} 0.3158302 0.9586438  1.085648  2 {poutcome=unknown,Fac.age=mid,Fac.duration=long,Fac.pervious=new} => {y=no} 0.3158302 0.9586438  1.085648  4 {poutcome=unknown,Fac.age=mid,Fac.duration=long,Fac.pday=new}     => {y=no} 0.3158302 0.9586438  1.085648  6 {default=no,poutcome=unknown,Fac.age=mid,Fac.duration=long}       => {y=no} 0.3087523 0.9577358  1.084620  3 {default=no,Fac.age=mid,Fac.duration=long,Fac.pervious=new}       => {y=no} 0.3086859 0.9577271  1.084610  5 {default=no,Fac.age=mid,Fac.duration=long,Fac.pday=new}           => {y=no} 0.3086859 0.9577271  1.084610 |

When setting support as 0.3, confidence as 0.95, above results were generated. Consider that this data set was about telemarketing sells, the success rate is no higher than 12%(num(instances with decision yes)/num(instances with decision no) ), with a confidence request at about 95%, generating positive rules might be extremely hard or even impossible. Although no positive result was found, the item datasets identified by Apriori could still be meaningful in reducing cost in tele-marketing as negative instances could be detected by this model.

3.2 Naive Bayes

Naive Bayes classifier is a kind of probabilistic classifier based on Bayes’ theorem. It assume that all attributes in dataset are independent with each other. Training dataset with classification {C1, C2,...,Ck}is used as calculation base. When a new instance with Attribute {F1,F2,...,Fk} coming, we calculate the probability P = P(F1F2...Fk | Cj)P(Cj), 0<j<k-1, then find the classification with highest P, which is the classification the new instance shall belong to.

For dataset adopted by this project, some potential problem may lead to a poor accuracy result. The aim of the project is to divide contacts into two groups: potential client(y=yes) and other (y = no). The training set shows that only 1 of 9 above all contacts bring us a long-term deposit (From general knowledge, telemarketing sells normally has a low success rate, which 1 of 9 is already above average). This means in previous formula P = P(F1F2...Fk | Cj)P(Cj), 0<j<k-1, the probability of two classification (P(Cj) in the formula) got quite huge gap. Without classifying rest of attribute, Naive Bayes classifier will identify most of instance in verifying dataset as “y = no”. Since in training data set, most of factors (Fj, 0<j<k) in each attribute got nearly same weight, some attribute have very low proportion for each factor (e.g. “age”, “balance”). To solve this problem, we use same measure as what have done in Apriori algorithm, classifying attributes into different factors. For example, attribute “age” in training set obeys the normal distribution, dividing age into young(less than 30), mid-age(between 30 and 60), old(greater than 60) could increase P(age|y), which would also improve NB classifier’s performance.

|  |
| --- |
| model <- naiveBayes(y ~ ., data =my\_data\_fac)  pred <- predict(model, my\_testdata\_fac)  confusionMatrix(table(pred, my\_testdata\_fac $y))  pred    no   yes  no    3720  327  yes    280  194  Accuracy： 0.8657377 |

Generally, in a commercial application, companies interest more on whether the classifier could find potential client, in another word, the accuracy of positive prediction. We could see in NB model, 62.76% of positive instance were classified as negative, low performance in identifying positive instance, which means NB might be applicable in reducing managing cost in tele-market campaign, but it could not perfectly meet demand of companies.

3.3 Support Vector Machine(SVM)

Support vector machine constructs a [hyperplane](https://en.wikipedia.org/wiki/Hyperplane) or set of hyperplanes in a [high](https://en.wikipedia.org/wiki/High-dimensional_space) or infinite-dimensional space, which can be used for classification. It is an extension of linear classifier.

Preprocessing for SVM model construction request converting all categorical attributes in dataset into numeric type. Because it is impossible to predict the influence of each coefficient in hyperplanes function, we randomly gave an integer to each attributes in each categorical variable.

|  |
| --- |
| >model <- svm(y ~ ., data = mydata)  > pred <- predict(model, mytestdata[,-15])  > table(pred, mytestdata$y)  pred    no  yes  no   3925  388  yes    75  133  Accuracy: 0.89758903 |

Result shows that SVM did not provide a good performance in prediction of clients’ decision. Prediction for positive instance is even worse than KNN algorithm. Considering the following assumption, in attribute “job” we have engineer and medical doctor, we simply use 1 to represent medical doctor and 2 for engineer. If function have a positive correlation with “job”, but in fact is medical doctor is more likely to buy a long-term deposits, but it likely give opposite prediction because of our assignment. As it is impossible to predict influence of each attribute (coefficient in hyperplanes function), SVM might be not suitable for telemarketing prediction.

3.4 K-nearest-neighbours Algorithm(KNN)

KNN is a non-parametric method used for classification. When a new instance coming into the system, k closest training example in the feature space will be treated as input. All these instance will comprehensively identifying the prediction of new sample by majority voting.

KNN algorithm is a kind of effective classifier with good robustness against noise. Another reason for choosing KNN to analyze this data set is that KNN could handle attributes with actually value (some ML algorithm require classified data, which means that many attributes in our data need to be classified in these algorithm ), as it KNN got judgement criteria depending on specific instance. The data set adopted in this project contains some attribute with accurate numerical value (such as “age”, “balance” and so on). KNN could handle them perfectly without any kind of transform.

|  |
| --- |
| > cl<-mydata$y  > mydata.knn<-knn(mydata, mytestdata, cl, k = 9, prob=TRUE, use.all=TRUE)  > confusionMatrix(table(mydata.knn, mytestdata$y))  Confusion Matrix and Statistics  mydata.knn   no  yes        no  3899  334        yes  101  187  Accuracy : 0.9038 |

Result shows that the KNN algorithm got accuracy of 90.64%. Although its accuracy is acceptable in this ML project, there is still 9.36% incorrect rate on testing data, and the positive prediction accuracy is only 35.89%, which make KNN a not applicable algorithm for this dataset. A main problem could be curse of the dimensionality of our data set. Some of the attribute has weak relationship with the classification of a instance, these attribute will bestially affect the result of KNN. Another cause might comes from the “majority voting” part. According to the judgement strategy of KNN based on distance. a certain number of neighbors were detected in feature space in to K will consider as same weight in voting, but these neighbors may have different in actual distance.  Among those neighbors, if some neighbors with further distance have larger quantity, the result will be miscalculated.

3.5 Decision Tree

Decision tree is a kind of classic data mining strategy which is widely used in ML project. It could generate a tree-like model of decisions which could be used as a classifier. Decision tree is applicable with telemarketing sells campaign as it could handle variety of attributes(no matter numeric or categorical). Furthermore, decision tree could handle different kinds of data without any transform.

Our dataset is a record set of previous telemarketing campaign which contains 16 attributes in different type (numeric and factor). This sets is exactly the data what Decision Tree is good at handling. The request of preprocessing is converting some attribute with too many factors (e.g. age, balance).

In this project, the newest version c5.0 tree was adopted. c5.0 tree is an extension of c4.5 tree. It has similar strategy with traditional decision tree model, several improvement upgrades were introduced in c5.0 version, such as memory optimization, and rule post-pruning. These functions could improve calculation performance as well as avoid overfitting. All improvements mentioned above could be beneficial for effectively and accurately generating decision tree classifying model as well.

|  |
| --- |
| TreeModel <- C5.0(y ~ ., data = mydata)  pred<-predict(TreeModel,mytestdata[,-15])  table(pred, mytestdata[,15])  pred    no  yes  no   3895  196  yes   105  325  Accuracy: 0.93342181 |

It is clear that accuracy of total classification and accuracy of positive prediction (62.38%) had a significant improvement. This is a quite successful experiment, which indicate decision tree has good capability in telemarketing sells prediction. The following part will describe attempts in improving accuracy based on decision tree.

3.6 Bagging (random forest)

In previous decision tree model, an ideal prediction model was generated, and good capability of decision tree for telemarketing sells was proved. In order to make further improvement, we should focus on improving decision tree algorithm rather than trying new algorithms. So bagging strategy came to our sight.

Random forest is an application of bagging algorithm. It randomly choose n instances (n=400 in our experiment) from training set (and then put them back) to build 100 decision tree models. During each model construction, we randomly remove one attribute (This is quite similar with reduced-error pruning, but it is target-less, just randomly chose one attribute each time). In this way we could generate a set of classifiers. When new data coming into system, each classifier will give independent prediction, after majority voting, an composite decision will comes out.

|  |
| --- |
| > rf <- randomForest(y ~ ., data=bank\_full, ntree=100, proximity=TRUE)  > bank\_test<-read.csv("bank.csv",header=TRUE,sep=";")  > confusionMatrix(table(predict(rf,bank\_test), bank\_test$y))  Confusion Matrix and Statistics       no    yes  no   4000    2  yes     0  519  Accuracy : 0.9996 |

As there is uncertainty in the model, we did ten tests and verifying with small dataset after each test. Bagging model shows nearly perfect result with 99.96% over all accuracy in one of our ten tests, furthermore the accuracy for predicting positive instance is 99.62%, which is very impressive.

3.7 Boosting

Boosting is another applicable algorithm usually used in ML. Compared with Bagging, Boosting algorithm builds multiple classification model by training on same dataset different times, during each training, an error rate based weight will be given to each model. Boosting algorithm classify instance by combining all decisions made by sub-classifier according to their weight.

As there is uncertainty in building decision trees, training round was set as 100 in order to get a higher prediction accuracy (when trials set to 10, the result given by boosting was even worse than decision tree based on c5.0). Following result shows that Boosting could also generate high accuracy prediction. Generally speaking, boosting have a better classifying performances than bagging because sub-classifier generated by boosting is more diverse than bagging. If we could have enough resource (hardware and time ) to run a huge number of training rounds.

|  |
| --- |
| > TreeModel <- C5.0(y~., data = mydata, trials = 100) ##number of boosting iteration  > pred<-predict(TreeModel,mytestdata)  > confusionMatrix(table(pred, mytestdata$y))  Confusion Matrix and Statistics  pred    no  yes  no   4000   36  yes     0  485  Accuracy : 0.992 |

3.8 Neural networks

Basically, neural networks is an algorithm for auto\_built multi-layer non-linear classification model.  The weight for each case keeps change in the process of model evolvution. As cutting-edged classification algorithm, we decided to try this method for prediction.

However, in this bank data, we have 9 factor variables and 6 numeric variables. Coercing transform raw data into all-numeric dataset, and assigning numeric number to raw data proper would lost and misrepresent information provided by dataset. Therefore, the accuracy of prediction through this algorithm is relatively low.

|  |
| --- |
| > nn <- nnet(y ~ ., data = mydata, size = 2, rang = 0.1,decay = 5e-4, maxit = 200)  >  confusionMatrix(table(mytestdata$y, predict(nn, mytestdata, type = "class")))  Confusion Matrix and Statistics        no  yes  no  3875  125  yes  329  192  Accuracy : 0.8996 |

4 Results

|  |  |  |  |
| --- | --- | --- | --- |
| No | Algorithm | Accuracy | Positive prediction accuracy |
| 1 | Aprior | N/A | N/A |
| 2 | Naive Bayes | 0.8657 | 0.3724 |
| 3 | Support Vector Machine(SVM) | 0.8976 | 0.2553 |
| 4 | K-nearest-neighbours Algorithm(KNN) | 0.9038 | 0.3589 |
| 5 | Decision Tree | 0.9334 | 0.6238 |
| 6 | Bagging (random forest) | 0.9996 | 0.9962 |
| 7 | Boosting | 0.992 | 0.9309 |
| 8 | Neural networks | 0.8996 | 0.6057 |

5 Discussion and Conclusions

Through above implementation and analysis, we can see that Bagging achieved best performance in this dataset, and gave us highest accuracy (0.9996) and positive prediction accuracy (0.9962). It perfectly meets the demand of save marketing manpower and prevention of lossing potential clients. Hence, marketing team of this bank can use this bagging (random forest) model as a prediction model in the future marketing activities.

We shouldn’t be surprised that bagging which bootstrap aggregated by decision tree algorithm achieved much more higher accuracy than many other complicated and advanced algorithm. Unlike decision tree algorithm, support vector machine and knn can be used in numeric data, Navie Bayes can only be used in categorical data. Transforming data from categorical to numeric would probably misrepresent data by assigning inappropriate value to attributes of categorical variables. Similiarly, in the process of transforming numeric data to categorical data,  inappropriate split will also lost the information contained in the transform variables which could be crucial to build prediction model.

6 References

All the packages we used in the assignment are:

* Klaus Schliep. "Package ‘kknn’". 2016.
* David Myer “Package ‘e1071’ ”.2015
* Max Kuhn “Package ‘C50’”. 2015
* Micheal Hasher “Package ‘arule’”. 2016
* Alfaro Estban “Package adabag’”. 2015
* Brian Ripley “Package ‘nnet’”. 2016
* Andy Liaw “Package ‘randomForest’”.2015
* Brian Ripley “Package ‘class’”. 2015
* Mitchell, T. "Machine Learning". McGraw Hill, 1997.
* Peter Flach. “Machine Learning”. Cambridge, 2012