Semester Project

Chris Lin, Kwong Yuet Michael Fadillah Wong, Xinze Fan

Abstract

For this project, our group utilized varies Data Mining algorithms such as K-th Nearest Neighbors classification, Naive Bayes Algorithm and Neural Network. The goal for this project is to predict whether or not the driver is going to have an file an auto insurance claim for the next year based on his/hers driving habit data this year. We faced multiple challenges while approaching this problem. We are facing a relatively large data set with more than 890000 entries as testing data set, therefore, we tried make use of IU's supercomputer Carbonate and Karst to do a faster computation of data as well as making use of multiple online libraries available.

Keywords

Data Mining — Machine Learning — Supervised Learning – Naive Bayes – KNN

[1] *Computer Science, School of Informatics , Computing and Engineering, Indiana University, Bloomington, IN, USA*

Contents

1. Problem and Data Description

The problem we are facing in this project is to predict whether or not that a driver will initiate an auto insurance claim in the next year based on the driver's past driving experience. The data provided to us including a set of training data and a set of testing data. There are total of 58 attributes for each entry and similar groupings are tagged as such in the variable names (e.g., ind, reg, car, calc). Furthermore, there are also postfix bin to helps indicate binary features and cat to indicate categorical features. In addition to that, missing values are indicated by value of -1. Target columns show whether or not a claim was filed by the driver in the training set.

2. Data Preprocessing & Exploratory Data Analysis

2.1 Handling Missing Values

Since there are missing value in both training set and testing set. All of the missing values are replaced by the value of -1, therefore, we were trying to decide whether or not to process the missing values at all. After the discussion, we decided that we will run all the algorithms both with and without processing missing values to decide which method is better to use. Our first attempt to process missing values was to replace all of the missing values with the mean of the corresponding column in the data.

As we learned more about the data itself, we decided to handle missing values in a similar but different approach. This time, we classified data between binary and non binary. For binary data, we replaced the missing values with the most common element excluding -1. On the other hand, we replaced the missing values with the mean of the attributes. We chose that approach because it did not make sense to have decimal as a binary data value.

Because of limited time, we do not have time to have run denosing autoencoder in order to improve data representation.For denosing autoencoder, it works like a neural network, but the input and the output length are the same. With this approach, we can have training set as our corrupted input, and try to reproduce the input feature. Therefore, it could be a optimal way to handle missing value.

2.2 Cross Validation for model evaluation

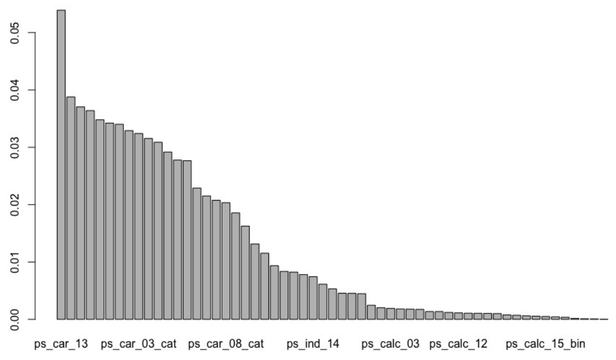
For further data analysis, We divided the training set into 5 folds using K-fold cross validation, then we ran through all these training sets over the test set in order to examine a better result. We used our K-fold implementation on homework 3 to perform cross validation. In our implementation, the function has two parameters, data and k, and the function is able to split the data into k part, then randomly choose one part as the testing set, and combine the remaining parts as a training set, after that the function returns a list of training set and testing set. We chose k equals five for our model. The goal for this algorithm was to improve the accuracy when we are predicting the label using different machine learning algorithm. Basically, we trained our model using five training sets that we produced using cross validation, and predicted the label based on the model results. Since we have five models$'$ result, so we were be able to compute the mean of the result, and used it for submission.

2.3 Correlation Analysis

We also analyzed the correlation between all 56 variables with the target variable to see which variable has the most impact on the target classification. We found out that variables $ps-car-13, ps-car-12, ps-ind-17-bin, ps-car-$

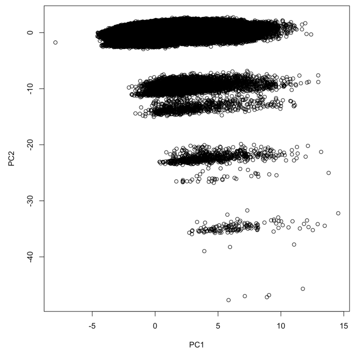
$07 - cat, ps - reg - 02$ has the highest correlation to the target variable.

Correlation:



Therefore, throughout our computation, we tried to emphasized the importance of those variables over other variables by giving it more weights. Although, It does not have much impact when it comes to neural network computation because the weights are updated after each iteration, but it does have influence on the result of KNN and Naive Bayes Algorithm as they use the initial input weights and never updates them. Furthermore, we tried to do a feature selection and only use those variables as the input to train our model to save the computation time. However, the result shows that we got way lower score when feature selection comes into place, because the information loss. It clearly shows that even one feature does not have much impact on the target variable, it can still has influence to the outcome.

Plot of PC1 and PC2:



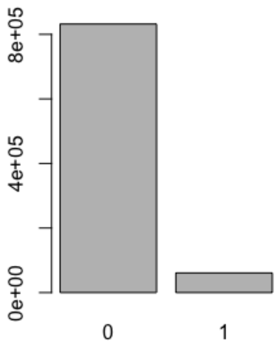
3. Algorithm and Methodology

3.1 Naive Bayes Algorithm

Naive Bayes algorithm is an algorithm of constructing classifiers.Using this algorithm, we were able to calculate the probability of each attribute resulting in different class label, then, we sorted the importance of each attribute probability in order to build the decision tree. After that, we predicted the test set label by using the decision tree produced by Naive Bayes.

After we performed Naive Bayes algorithm, we had the result of how each attribute is related to the class label. Thus, we marked down the attributes that were more likely to affect the class label, and the attributes that were unlikely to affect the class label for further analysis. Then, we predicted the test sets class labels based on the result. For the predicted result, There were 831941 zeros, and 60875 ones. After getting the predicted result, we extracted each record′s probability of resulting in the final class label of one, and saved it.

Label plot:

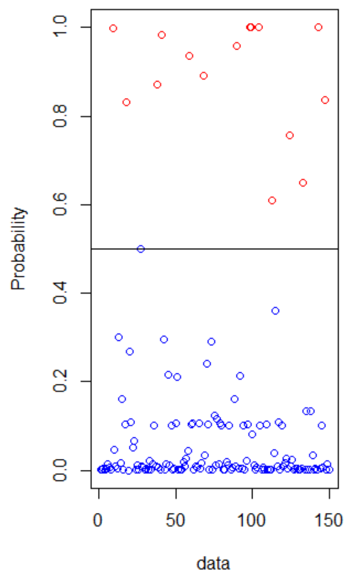


3.2 K-th Nearest Neighbors

K-th Nearest Neighbors classification algorithm classifies the class label of each test data entry by calculating the distance between the test data against all training data set. After that, we get the k-th closest training data and decide the class label of our test data based on the majority of class label of k-th training data selected.

We utilized fastknn algorithm that we found from github (https://github.com/davpinto/fastknn), because the computing time is extremely long due to the size of the data set. After performing the algorithm, we did not get a result that we wanted. The reason behind that might be because we did not format our data correctly in order to make the fastknn method works. Thus, we decided to take advantage of Karst IU server to perform KNN algorithm from R's default package. We did not get as good of a result as Naive Bayes and Neural Network and we think the main reason behind that is there are way more data entries with class label 0 than 1 in our dataset.

KNN result plot:

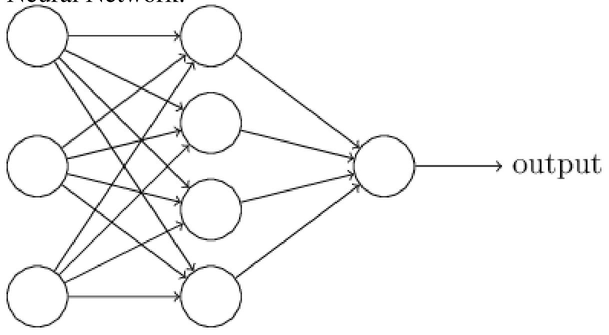


3.3 Neural Network

Neural network is a supervised machine learning algorithm. In other words, if we are given an ideal output and its input, neural network is able to find how the input is related to the ideal output and computes the weight between input and output. With that being said, neural network is usually splits into two layers: input layer, and output layer, and there are weights associating between the layers. However, this kind of neural network is only able to solve linear tasks. In order to solve non-linear tasks, there is a neural network with hidden layer between input layer and output layer. In this algorithm, it is trying to compute the weights between each layer in order to reach the ideal output. Throughout the learning phase, we back propagate the neural network, and use the delta rule to update the weights based on the error between the estimate output and ideal output.

Finally, we are able to obtain the weights after error is minimized. One of the most important parameters is the number of hidden unit, as the number of hidden unit increases, the starting MSE increases, converging speed increases, and the run time also increase because of computing a large data matrix. Therefore, there are tradeoffs when selecting number of hidden units, and we chose five hundred hidden units in our approach.

Neural Network:



3.4 Comparison of performance

The classifiers we used to compare are KNN algorithm and Naive Bayes algorithm. In the KNN algorithm, we use K = 5 ,euclidean distance and prob = TRUE to return the probability. The running time is about 15-17 hours and successfully return the probability for each objects. Since we pick K = 5 in KNN, the probability we get is shorter and more concise.

Since the Naive Bayes algorithm returned the probability by calculate the conditional probability of each objects, the Naive Bayes cost less running time than KNN did. The running time for Naive Bayes is approximately 20 minutes. Since every variable in the data was calculated in the conditional probability, the probabilities returned by the Naive Bayes are more specific than the KNN did. By comparing the KNN and Naive Bayes, we find that in this case, the Naive Bayes is more suitable since the higher accuracy and less time cost.
For the neural network, this algorithm cost us lots of time since it is a complex model. It runs better than the other two algorithm because it requires more decision on the hidden layer, unit, learning rate and etc. This algorithm gives us a more specific result since it has an unique weights for each objects which are trying to make test data get closer to train data. Especially in such a large dataset, although the Naive Bayes runs faster than the Neural Network, neural network produce a result with the better accuracy since it does better on handling the correlation or dependence between the input variable. Considered the upper results, we think that the Neural Network works best in this case, then the Naive Bayes and the worst one is KNN algorithm.

4. Experiments and Results

Throughout the whole project, we tried multiple strategies to try to improve the performance of our model. We took advantage of cross validation, feature selection, as well as different implementation of the same algorithm. Below are the results for all of our experiment, left score represent the score when running the results against 30% of the testing data, right score represent the score when running the results against 70% of the testing data:

Kaggle Score for Naive Bayes before cross validation:

0.23509 0.22767

Kaggle Score for Naive Bayes after cross validation:

0.23660 0.22916

Kaggle Score for KNN from R without feature selection:

0.21022 0.20542

Kaggle Score for KNN from R with feature selection:

0.10706 0.10521

Kaggle Score for fastknn before cross validation:

0.03210 0.03468

Kaggle Score for fastknn after cross validation:

0.03951 0.04458

Kaggle Score for Neural Network:

0.27046 0.26622

5. Summary and Conclusions

5.1 Challenges

Throughout this project, we faced lots of difficultlies and challenges. One of the biggest challenge is the size of the dataset. The training data has the size of approximately 590,000 entries and testing data has the size close to 890,000 entries. Therefore, our own PCs do not have enough computation power to perform the algorithm on those data set especially when it comes to KNN and Neural Network. Therefore, we utilized IU's supercomputer Karst and Carbonate in order to complete the computation.

In addition, handling missing data was also a challenge because we do not know if we choose the correct method until the computation is over and how we handle missing data can impact our result dramatically. Furthermore, it was also difficult to pick whether or not to perform Neural Network with all the attributes or only pick the attributes that have higher contribution to the target result.

5.2 Conclusion and Summary

After analyzing the dataset and understanding the purpose using correlation and PCA , we handled the missing values, performed k-fold validation, utilized three different classification methods, analyzed the results between the different methods. We finally indicated that the best method for this project is classification using Neural Network. Although the computation process took a while, Neural Network works best based on our accuracy result. In addition to that, with handling the missing value and k-fold, the accuracy got higher each time. In this project, we learned how to utilize the classification algorithm that we learned from class into real life tasks. We also learned that each change to the data will impact the results of the outcome. It is really important to be familiar with different method, because when it comes to solving real-life tasks, it is crucial to test the results using different algorithm since every model has its own advantages and disadvantages on different kinds of datasets.

Acknowledgments

5 functions to do Principal Components Analysis in R. (2012, June 17) Retrieved from http://www.gastonsanchez.com/visually-enforced/how-to/2012/06/17/PCA-in-R/

Davpinto, Fastknn Repository. (2017,Sep 17) Retrieved from https://github.com/davpinto/fastknn

SupriyaDubey, PCA on Seguro′s data. (2017, Jan) Retrieved from https://www.kaggle.com/priyasd/pca-on-seguro-s-data/comments

Michael Jahrer, 1st place with representation learning (2017, Dec) Retrieved from https://www.kaggle.com/c/porto-seguro-safe-driver-prediction/discussion/44629

Alvira Swalin, How to Handle Missing Data(2018, Jan 30) Retrieved from https://towardsdatascience.com/how-to-handle-missing-data-8646b18db0d4

Perceptive Analytics, Understanding Naive Bayes Classifier Using R (2018, Jan 12) Retrieved from https://towardsdatascience.com/how-to-handle-missing-data-8646b18db0d4

Michy Alice, Fitting a neural network in R; neuralnet package (2015, Sep 23) Retrieved from https://www.r-bloggers.com/fitting-a-neural-network-in-r-neuralnet-package/