ISEN-613 Engineering Data Analysis

Course Project

Car Insurance Classification Problem

Team Member:
1. Saurabh Vaichal (527004772)
2. Gaurav Kamath (727006808)
3. Krutarth Mehta (127003604)
4. Madan Taldevkar (523007674)
5. Naman Soni (627007858)

EXECUTIVE SUMMARY

The primary challenge in the insurance industry is a prediction of claims and ultimately deciding the insurance premium to be charged from consumers based on the likelihood whether a customer will make a claim. This differential charging of insurance premium can be attributed to multiple parameters. The deciding parameters are like vehicle type, make, safety features available, previous claims, vehicle age, etc. Each parameter would contribute in determining the risk that insuring a particular customer would accompany. For one, a trend in crashes of cars with same make & year of production leading to insurance claims should alarm the insurance providing firms. The more accurately the company identifies claim instances based on customer profile and customer's car features, the better the company profit figures would look.

We have a dataset that lists 100,000 policies from years 2005 and 2006. In the present work, we have used this data to build a predictive model that has been tested on the 50,000 observations from the calendar year 2007 to check the performance of the model built. The prediction model reported here was built after application of multiple statistical prediction methods and the best method resulting in highest reliability from the company's point of view has been selected.

Various challenges that this data brings along are, 1) Dataset with only a few claim instances, 2) Variable names not known, 3) Missing values from data and 4) High number of variables. Many methods focus primarily on the accuracy of correctly identifying both type of responses, in this case, whether a claim was made or not. Base assumption in this approach is that, misclassification of both type of responses would cost equally. However in our case, failure in predicting claim would cost company hefty claim amounts. On the flip side, incorrectly tagging customers as high risk(prone to make insurance claims) would marginally increase the premium amount, which would still be acceptable.

In the present work, we have used multiple methods for claim prediction. These methods range from primitive methods to more advanced methods that learn from the errors that it makes. We have also considered an additional parameter apart from accuracy that would help the insurance company to correctly predict the claim instances.

Based on the best model, we were able to achieve prediction accuracy of 59% and a recall of 45%. Recall is the measure of accurately predicting that the customer will claim insurance. We have included methods in the report to further improve the recall. All team members have equally contributed in this project.

1. DATA INTERPRETATION AND PREPROCESSING

The data set contains 100,000 policies from calendar years of 2005-2006. There are 33 input variables having both numeric as well as categorical values. We have 720 claims in total. To treat it as a classification problem **`Claim_Amount`** was replaced by a categorical response **`C_Claim`**. Collinearity between variables was checked using scatterplot as shown below.

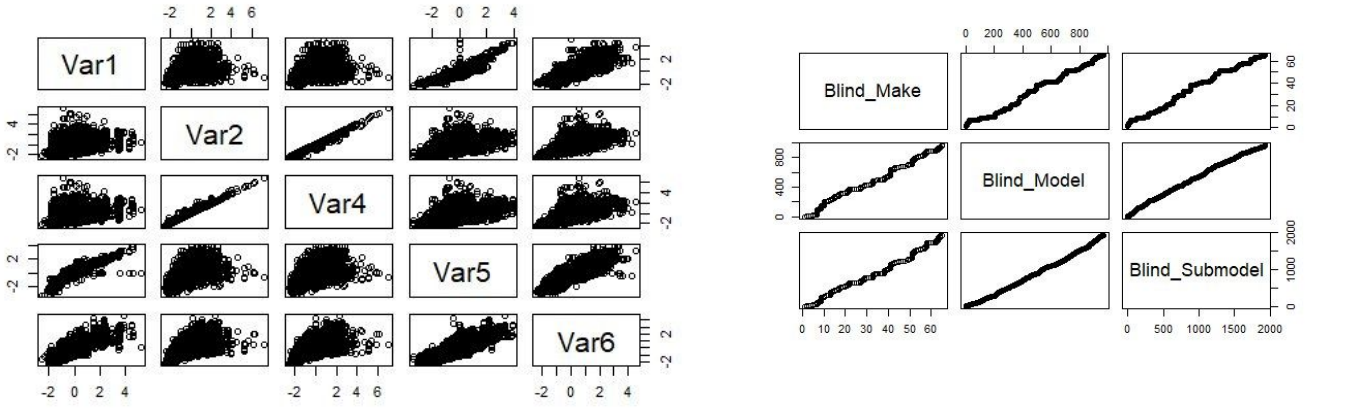

FIGURE 1.1. *Left:* **`Var1-Var5`**, **`Var2-Var4`** *and* **`Var5-Var6`** *are pairs with high collinearity and hence* ***`Var4`*** *and* ***`Var5`*** *can be omitted from the model. Right:* **`Blind_Model`** *and* **`Blind_Submodel`** *are also omitted from model.*

`Cat1-Cat12` have several missing values. To tackle this, two approaches were used, data omission and data imputation. Data omission by Listwise deletion(deletion of entire row when at least one missing value was found) resulted in 74% data loss and was not chosen (**`Cat2, Cat4, Cat5, Cat7`**). Dropping variables with 40% of data missing resulted in 99% of observations being retained This was called **`Data1`**. For data imputation, MICE was implemented as replacing missing values with Mode would result in biased data. This imputed data was called **`Data2`**.

Response is highly skewed, with only 0.7% claims and needed resampling to generate balanced train data. Several sampling methods were tried including, under-sampling, SMOTE, K-modes clustering. K-modes clustering proved to be best method for data preprocessing where following algorithm was used.

	Positives	**Negatives**	**Ratio**
Original Data	720	99280	0.73%
Training Data	500	500	50.00%
Test Data	217	29603	0.73%

TABLE 1.1. *Skewness of data compared for original data, undersampled training data and test data*

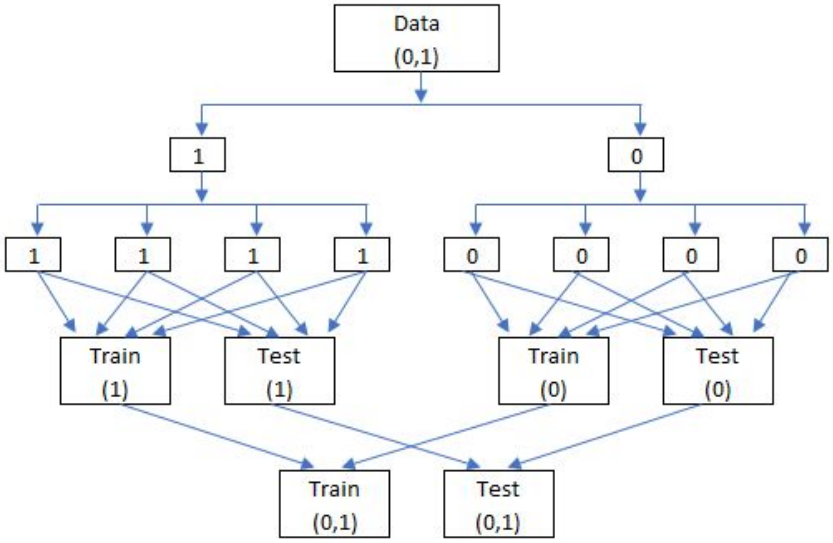

FIGURE 1.2. *Algorithm used to split data into train and test using clustering. This will ensure proper representation of 0s and 1s in train data.*

Although, for all the models mentioned in this report, Data1(without K-modes clustering) was considered as it produced superior results.

This problem being insurance claim (dataset is unbalanced), predicting true positives is of utmost importance. We want the customers who are likely to claim insurance to be identified correctly. Focusing only on accuracy would not serve the purpose of classification as we might correctly identify non-claim customers but if we do not identify the claiming customers, the accuracy will still be very high. Hence, it accuracy is not the parameter to be looked at while assessing the quality of a model. To address this important issue, sensitivity (also called as recall) is considered in the present report as prime parameter for model comparison along with AUC.

$$\text{Recall} = \frac{TP}{TP + FN}$$

1.1 LOGISTIC REGRESSION

We have a classification problem at hand. The response variable assumes two values, 0 and 1. Linear regression would result in a output ranging from $-\infty$ to $+\infty$ resulting in reduced interpretability. Hence, a logistic function as described below, is used by logistic regression resulting in an output ranging from 0 to 1[1].

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}}.$$

The logistic regression predicts the probability P(Y=1) and a probability threshold determines the classification of observations to either 0 or 1. Logistic model was applied to train data considering all the variables. Predictions were done for both train and test data. The results obtained test data are listed in FIGURE 1.3

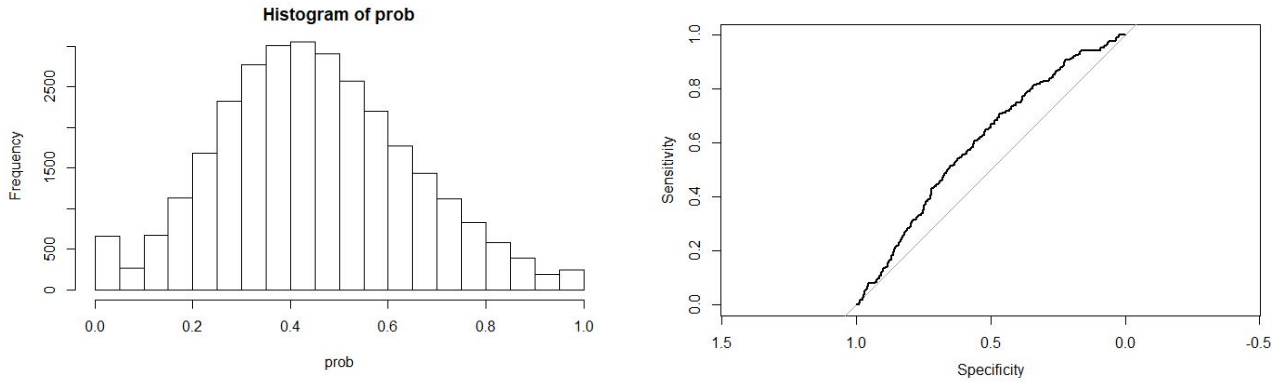

FIGURE 1.3. *Left: histogram of predicted probabilities for test data Right: ROC curve for test data with Area under the curve(AUC) = 0.6035.*

CONFUSION MATRIX		**Actual Claim**	
		0	**1**
Predicted Claim	**0**	18382	99
	1	11221	117

TABLE 1.2. *Confusion matrix comparing Logistic regression predictions with true claim status for train data set. Recall is 54% and Accuracy is 62.04%*

[1] G. James et al., An Introduction to Statistical Learning: with Applications in R, Springer Texts in Statistics 103, DOI 10.1007/978-1-4614-7138-7 4,

1.2 SUPPORT VECTOR MACHINE

SVM is a data classification method that operates by separating data using hyperplanes. Using Kernel function helps to accommodate non linear boundary between the classes. As it is impossible to visualize the data, the SVM was modelled using three types of kernel- linear, polynomial (degree=3) and radial. The best output was obtained when radial kernel was used. We have to decide the value of γ and Cost (C) for the Kernel. A large γ results in high bias and low variance. A large C results in low bias and high variance. To identify the ideal γ and C, a built-in function tune() in the e1071 library is used. Because the tune function minimises training error and does not maximise recall, we will use the results from tune() as reference and not as the best values for γ and C. The radial kernel function is defined as-

$$K(x_i, x_{i'}) = \exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2).$$

However, for our best SVM model, which involved striking a balance between accuracy and recall of the model as tested on the test data, the values of γ and C are taken as 0.01 and 100 respectively. The Results obtained from applying SVM model are as follows-

CONFUSION MATRIX		**Actual Claim**	
		0	**1**
Predicted Claim	**0**	17263	53
	1	12340	163

TABLE 1.3. *Confusion matrix comparing SVM predictions with true claim status for train data set. Recall is 75.46% and Accuracy is 58.44%*

1.3 ADAPTIVE BOOSTING

The boosting methods grow trees sequentially using the information from previous tree. AdaBoost is adaptive boosting in which subsequent weak learners are tweaked in favor of those instances misclassified by previous classifiers[2]. AdaBoost is sensitive to noisy data and outliers. It is less susceptible to the overfitting problem than other learning algorithms. The parameter: coeflearn = "Freund" (**α=1/2ln((1-err)/err)**) was used while applying adaptive boosting. The best results were obtained for when ran the algorithm with 100 iterations to get the results tabulated in the following confusion matrix.
Again, test data and train data were obtained as mentioned in the code for Data1.
The results obtained from application of Adaptive Boosting are listed below,

CONFUSION MATRIX		**Actual Claim**	
		0	**1**
Predicted Claim	**0**	17496	33
	1	12107	183

TABLE 1.4. *Confusion matrix comparing Adaptive boosting predictions with true claim status for train data set Recall is 84.72% and Accuracy is 59.28%*

[2] Freund, Y. and Schapire, R.E. (1996):"Experiments with a new boosting algorithm". In Proceedings of the Thirteenth International Conference on Machine Learning, pp. 148–156, Morgan Kaufmann.

2.0 COMPARISON & METHOD SELECTION

The results obtained by above-mentioned three methods have been encapsulated in the table below,

Sr. No	Method Utilised	Recall	Accuracy
1	Logistic regression	54.17%	62.01%
2	SVM	75.46%	58.44%
3	Adaptive boosting	84.72%	59.28%

TABLE 1.5. *With recall and accuracy, it can be concluded that adaptive boosting gives best results.*

The data is highly skewed with only 720 claim instances. This might result in mis-classification of minor class by traditional methods. The adaptive boosting assigns weights to observations, calculates associated error and reassigns the weights to ultimately reduce error in subsequent iterations.

Conclusion:
Observing from the Table 1.5, the recall is increasing from 54% to 84% with a marginal reduction ~3% in accuracy. As mentioned earlier, recall is considered prime factor for prediction.This concludes that the adaptive boosting should be testing the new data.

3.0 EVALUATING TEST POINTS ON BEST MODEL

As mentioned in section 2.0, the results obtained by Adaptive Boosting are the best and same would be used to predict results for test data. Some of categorical variables have less categories as compared to train data, which would give error while prediction. To tackle this issue, dummy variables that were present in original model were added to test data as separate columns with all 0 values. By applying the model to test dataset from year 2007 (50,000 observations), following predictions are obtained.

CONFUSION MATRIX		**Actual Claim**	
		0	**1**
Predicted Claim	**0**	29352	202
	1	20281	165

TABLE 1.6. *Confusion matrix comparing Adaptive boosting predictions with true claim status for test data set Recall is 44.96% and Accuracy is 59.03%*

45% of claim instances were correctly predicted by our model. On the other hand 40% false positives would not matter much, apart from premium of those misclassified customers would marginally increase. This decrease in accuracy and recall could be possibly dealt by increasing the variability captured by the training data which is explained as below.

4.0 STEPS FOR IMPROVING THE BEST MODEL

The model finalised from train data still has scope for improvement. Following parameters are tuned so as to have better prediction.

1. `mfinal = 200`, an integer, the number of iterations for which boosting is run or the number of trees to use. Defaults was mfinal=100 iterations.
2. `boos = TRUE` (by default), a bootstrap sample of the training set is drawn using the weights for each observation on that iteration. If FALSE, every observation is used with its weights.
3. `minsplit=0,` the minimum number of observations that must exist in a node in order for a split to be attempted.
4. `maxdepth = 8`, Set the maximum depth of any node of the final tree, with the root node counted as depth 0.
5. `xval = 5,` number of cross-validations,

To improve it further, multiple samples were randomly selected from train data using different seed values. Using adaboost would be fit on each of the sample creating several submodel. Final prediction can be made with following possible approaches.

1. we can record the class predicted by each of the submodel, and take a majority vote to assign most commonly occurring class as our prediction
2. The second approach is to classify based on the average of probabilities predicted by each submodel.

The model was run with 12 different seed values. Accuracy and recall were plotted for predictions of test data by 12 different submodels.

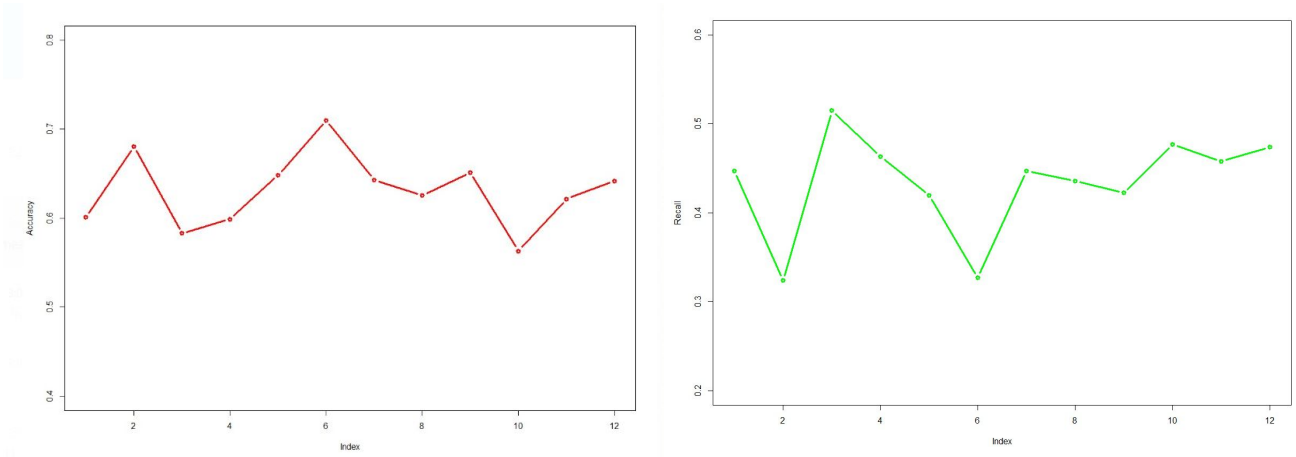

FIGURE 1.4. *Left: Accuracy plotted against different submodels, Right: Recall plotted against different submodels. Final mode is expected to have Accuracy ~ 65% and Recall ~ 50% which is a considerable improvement*