Car Insurance Claim Forecasting Using Fusion of Machine Learning Techniques

Wenting Li
Department of Computer Science
Wayne State University
Detroit, USA
gg3181@wayne.edu

***Abstract*—In this project, machine learning techniques are explored to improve accuracy in classification prediction. Combined classification techniques are used to build a model that predicts the probability that a driver will initiate an auto insurance claim. Among these techniques, XGBoost outperforms all the other algorithms and gives best forecasting accuracy.**

***Keywords*—car insurance claim; forecasting; gradient boosting machine learning; XGBoost; grid search**

I. Background Introduction

For car insurance companies, it is important to make accurate predictions and evaluate driver's performance. Otherwise, bad predictions will result in raising the cost of insurance of good drivers and reducing the price for good drivers.

In recently years, a lot of curiosity has been generated in machine learning field. Numerous methods have been tried an applied to analyze and forecast the market. The baseline methods usually used in classification include: Logistic Regression (LR), Naïve Bayes, K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Decision Trees, Neural networks. Naïve Bayes is a conditional probability model. A problem instance to be classified is represented by a vector representing features [1]. The instance probabilities are assigned for each of K possible outcomes or classes. LR measures the relationship between categorical dependent variable with one or more independent variables by estimating probabilities using cumulative logistic function [2]. SVM performs classification by constructing an N-dimensional hyper plane to separate data into two categories optimally [3]. KNN algorithm is a non-parametric method used for classification and regression [4]. Decision Tree is a decision support tool. It uses a tree-like model to make decisions and possible consequences, including resource costs, chance event outcomes, and utility [5]. Neural Networks are computing systems inspired by biological neural networks [6]. It learns to do tasks by considering examples.

From past experiences, researchers have found that certain models perform better and have better accuracy in prediction compared to other machine learning models. "Extreme Gradient Boosting", known as XGBoost is one among such models which has obtained rave from machine learning practitioners. XGBoost model is based on Gradient boosting, which believes that single trees are so weak that they cannot give accurate prediction. Therefore, ensembles of decision trees are used. Trees are added in a way that the current error is optimized. Now XGBoost has attracted a lot of attention and is widely used for supervised learning problem.

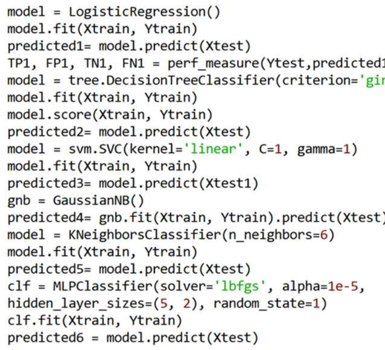
II. Methods

A. Preprocessing

For the baseline test, data were split into 75% of training data and 25% of testing data. There were missing values labeled with "-1" in the data set. After more careful observation, most of the missing values came from column z and column ab. It's also true that most values from these two columns were missing. Thus, these two columns were deleted. There are also missing values in other features. For these missing ones, mean values of the features have been used to replace them. Another fact I noticed is that the data set is a mixed one, consisting of both continuous variables and binary variables. Binary attributes were left untouched, while continuous attributes were rescaled into the range between 0 and 1.

B. Baselines in Python

All the Baselines in this project were performed on Python platform. The baselines used in this project include: Logistic Regression (LR), Naïve Bayes, K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Decision Trees, Neural networks. The preprocessed data have been trained with arguments in Fig 1.



```
model = LogisticRegression()
model.fit(Xtrain, Ytrain)
predicted1= model.predict(Xtest)
TP1, FP1, TN1, FN1 = perf_measure(Ytest,predicted1)
model = tree.DecisionTreeClassifier(criterion='gini')
model.fit(Xtrain, Ytrain)
model.score(Xtrain, Ytrain)
predicted2= model.predict(Xtest)
model = svm.SVC(kernel='linear', C=1, gamma=1)
model.fit(Xtrain, Ytrain)
predicted3= model.predict(Xtest1)
gnb = GaussianNB()
predicted4= gnb.fit(Xtrain, Ytrain).predict(Xtest)
model = KNeighborsClassifier(n_neighbors=6)
model.fit(Xtrain, Ytrain)
predicted5= model.predict(Xtest)
clf = MLPClassifier(solver='lbfgs', alpha=1e-5,
hidden_layer_sizes=(5, 2), random_state=1)
clf.fit(Xtrain, Ytrain)
predicted6 = model.predict(Xtest)
```

Fig. 1. Baseline methods training defination

C. XGBoost

XGBoost model is achieved by implementing xgboost package in python. XGBoost hyperparameters are tuned by running randomized grid search. Parameters like 'min_child_weight', 'gmma', 'subsample', 'max_depth' were searched thoroughly via different combinations. The total number of combinations for the parameter sets is a product of options for each parameter. The total number of each data-fitting run is the product of number of combinations and cross-validation folds. Firstly, a coarse grid was used to locate the approximate range the parameters. Then, a fine grid was used to search for the optimal parameter combination more thoroughly.

III. RESULTS AND DISCUSSION

A. Baseline Methods Results

Baseline methods were applied on the preprocessed training data. For each method, specific classification model is obtained by fitting the training data. This model was then applied to the test data. Predictions and true target values were compared to calculate precision, recall, F-value, etc. Among the six baseline techniques, Decision Tree, KNN, and Naïve Bayes performed better than the others. Their F-measures were 0.046, 0.012, and 0.084. Different ways of data preprocessing, data selection, and data transformation were used. However, the prediction accuracy didn't have noticeable improvement.

B. XGBoost Results

In XGBoost algorithm, grid search was carried out to optimize hyperparameters. At first a coarse grid was used to roughly locate the range of the parameters. From all the parameter combinations, the optimal 5 parameter sets were selected. Then a fine grid was used to search parameters around the optimal parameter set selected from the coarse grid. The criterion of parameter selection is Area Under the Curve(AUC). Normalized Gini Coefficient was also calculated for each prediction result using certain parameter set. One reason of using Normalized Gini Coefficient (Gini) is that it is the main evaluation adopted in Kaggle Competition.

For the results, it was found that XGBoost outperformed all the baseline single methods and has given best forecasting prediction. Table 1 is a sample summary of one small grid search trial.

TABLE I. SUMMARY OF ONE GRID SEARCH TRIAL

AUC	Gini	P1	P2	P3	P4	P5	rank
0.6393	0.2785	0.8	1	5	5	1	4
0.6402	0.2804	0.6	1	5	5	1	1
0.6391	0.2781	0.8	5	5	5	1	5
0.6399	0.2799	0.6	5	5	5	1	3
0.6401	0.2802	0.6	1	5	2	1	2

Here P1 represents for parameter "colsample_bytree", P2 represents for "gamma", P3 represents for "max depth", P4 represents "min_child_weight", and P5 represents for "subsample". From this table, we can see that in this trial, the optimal test score is obtained from a combination of P1 = 0.6, P2 = 1, P3 = 5, P4 = 5, P5 = 1. The optimal AUC score is 0.6402, while the optima Normalized Gini Coefficient is 0.2804.

C. Discussion

Apparently, the prediction accuracy has been dramatically improved by using XGBoost Algorithm. The test prediction results were submitted to Kaggle and the final score is 0.278. Due to time limit, the grid search was not thorough enough and only a 3-fold cross-validation was performed. It's reasonable to expect a further improvement by adjusting learning rate, using higher number of iterations, carrying out more in-depth grid search, etc.

IV. CONCLUSION

In this project, XGBoost outperformed all the individual baseline classification techniques and has given best forecasting accuracy. Gradient boosting has proven to be an effective prediction algorithm for both classification and regression tasks. This project has just proved this one more time. XGBoost has the strengths of dealing with nominal values, numeric values, and missing values. It has immensely high predictive power and computes much faster than existing gradient booster techniques. In this project, we provide an efficient way of tuning parameters by combining XGBoost with grid search method. We can expect that in the future, XGBoost will play a more and more significant role in machine learning field.

REFERENCES

[1] Murphy, Kevin P, "Naïve Bayes Classifiers," University of British Columbia

[2] Kurt, Imran, MevlutTure, and A. TurhanKurum, "Comparing performances of logistic regression, classification and regression tree, and neural networks for predicting coronary."

[3] Alfaro, Rene, et al. "Forests for the New Millennium-Making Forest Work for People and Nature," Selected Books 1 (2005).

[4] Alman, N.S. "An introduction to kernel and nearest-neighbor nonparametric regression," The American Statistician. 46. 3: pp. 175-185, 1992

[5] Utgoff, P.E. "Increment induction of decision trees", Machine learning, 4(2), 161-186, 1989.

[6] Ng, Andrew; Dean, Jeff, "Building High-level Features using Large Scale Unsupervised Learning", 2012

[7] T. Chen and T. He, XGBoost: extreme gradient boosting. R package version, 2015

[8] Gumus, Mesut and Kiran M.S. "Crude Oil Price Forecasting Using XGBoost," IEEE, 2017.

[9] Gurnani, Mohit, "Forecasting of Sales by using fusion of Machine Learning Techniques", ICMAI, 2017

[10] BiswaRanjanSmal, "Performance Analysis of Supervised Machine Learning Techniques for Sentiment Analysis", ICSSS, 2017