A Machine Learning Practice in Car Insurance Risk Rating

Yuanjian Zhou
School of Economics
University of California-Los Angeles
Los Angeles, California 90095
Email: zhouyj9696@g.ucla.edu

Yite Zhu
School of Economics
University of California-Los Angeles
Los Angeles, California 90095
Email: yzhu72@g.ucla.edu

***Abstract*—This report analyzes the influence of different parameters and argument specifications in five classification models in insurance risk rating using a comprehensive dataset describing a set of vehicles. The purpose of these trials is to understand how these parameters and arguments affect the classification results of these models and try to find out the best choice of parameters for each algorithm. We compare these models in terms of accuracy, precision and recall.**

I. Introduction

In the application of machine learning, apart from choosing algorithms, it is also crucial to adjust model parameters, for this will assist us find out the best model amongst all candidates to predict a new testing sample. In this report, we use a comprehensive data set regarding various vehicles to investigate the effects of model parameters.

This vehicle data set consists of insurance risk rating and various characteristics of 205 vehicle products. The insurance risk rating variable serves the interests of the car renting firms since it indicates the degree to which the product is riskier relative to its current insurance rating. For instance, a +3 label means that the vehicle is subject to much more risk compared to what it is currently rated, and should have been rated 3 levels riskier. This is the reason the label is called symboling. In this data set, symboling ranges from -2 to +3, among which -2 indicates the safest and +3 indicates the riskiest. So we use various attributes of a car to train the model to predict the symboling, risks in other words, of a particular car. This is identified as a supervised multi-class classification problem as our label will be several discrete symbol numbers.

A. Challenges. The straight up challenge is that correlation between label and features is not quite apparent. It is rather hard to discuss analytically without any knowledge in mechanics what factors have attributed to the riskiness of these vehicles. Even if we can conclude what variables do contribute to the value or classification of the label, the question still remains which classification model performs the best in terms of prediction accuracy. Also, the data set in discussion provides a discrete label with six classes. This could have been a simple supervised classification problem, if only we have enough data, which is, nevertheless, unattainable in this case, because some classes have too small a sample, as is shown in Fig 1. The data set is rather too small to create six-class classification models with acceptable fit.

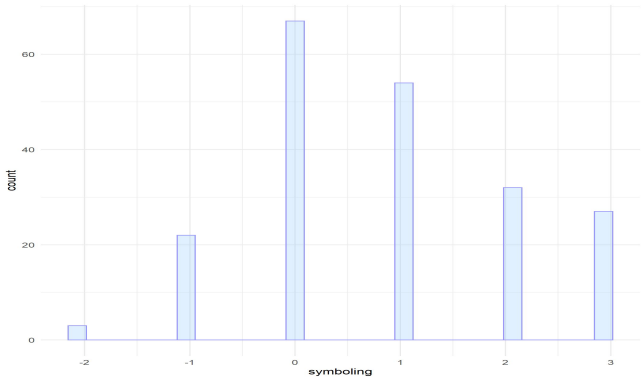

Figure 1. Histogram of Labels

B. Solution. We incorporated, as diversified as possible, multiple features that describe the mechanical, as well as commercial, characteristics of a vehicle, including fuel system, engine systems, brand, product line, power, energy consumption etc. The correlation, thereby, can be ascertained from different characteristics of a product. To exploit the effects of each variable, we utilize different families of machine learning models. To improve the fit of the model, we decide to relabel the labels of our data, specifically, we denote "-2" as -1, "-1" ,"0" and "+1" as 0, "2" and "3" as 1 such that labels represent highly safe(negative), neutral(neutral) and risky(positive) products clearly, then we have a 3-class classification problem.

II. Setting and Data PreProcessing

A. Setting

Considering pros and cons of different programming languages, we used both Python and R for this experiment. We mainly use Python because it offers us a convenient and powerful machine learning development environment on Jupyter-Notebook, with assistance from sklearn library, which is a helpful library for utilizing machine learning algorithms. Besides, for the sake of the convenience of data cleaning and plotting, we use R to do the data wrangling and results visualization.

B. Variable Regrouping

Figure 2. Pie Plot of Factor Variables

In our data set, as shown in Fig 2, there are several factor variables with too many levels, and some levels have very few observations. In order to improve the fit, we regroup these factor variables based on the knowledge of mechanics.

C. Normalization

Also, in our data set, continuous variables with drastically different magnitudes, as is shown in Fig 3. For safety, we applied min-max normalization to restrict these valuables to the range of $[0, 1]$, as shown in Fig 4.

After these preprocessing activities, we splitted the data into 4 parts: X_train,X_test,y_train and y_test. And we set the proportion of training and testing data as 66.7% and 33.2%. This prepared us for the implementation of the following models.

III. Model Implementation

A. Logistic regression

1) Parameters and arguments.

- multi_class - Multi-class argument can be either ovr or multinomial. If the argument is set as ovr, then it is the so-called One-vs-Rest approach for multi label classification, which means we fit a binary logistic regression for each label and get several probabilities, before we predict each sample with the label of the highest probability. If the argument is set as multinomial, then we utilize a single multinomial logistic regression to the data set, it is often called “One-vs-One” as compared to OVR.

- solver - Algorithm used for find the optimal value of parameters, here we have “lib-linear”,”lbgs”,”sag” and “saga”.

- penalty- Indicate whether we use regularization or not, and which regularization we use, here we try “none”,”l1” and “l2”.

- C - Inverse of the regularization hyper-parameter lambda

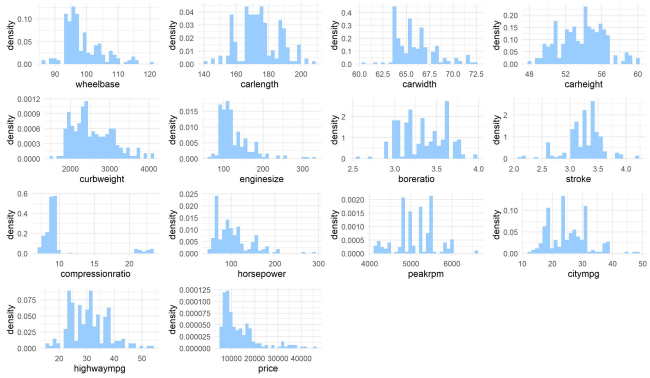

Figure 3. Histogram of Original Continuous Variables

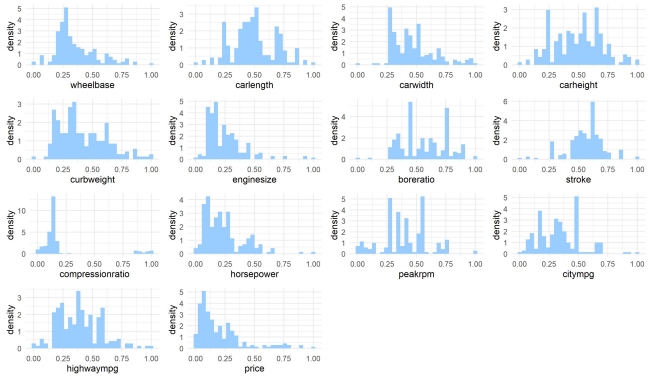

Figure 4. Histogram of Normalized Continuous Variables

2) Test results. Please refer to Fig 5.

3) Discoveries. We discuss the results with three metrics

- Accuracy: Generally, there is no significant difference between “OVR” and “OVO”; and it is

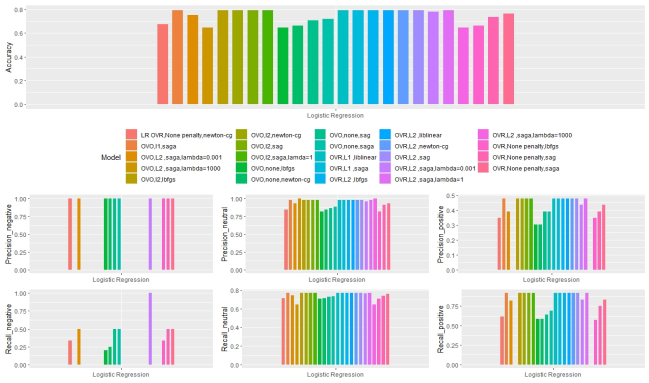

Figure 5. Metrics Results of Logistic Regression Family

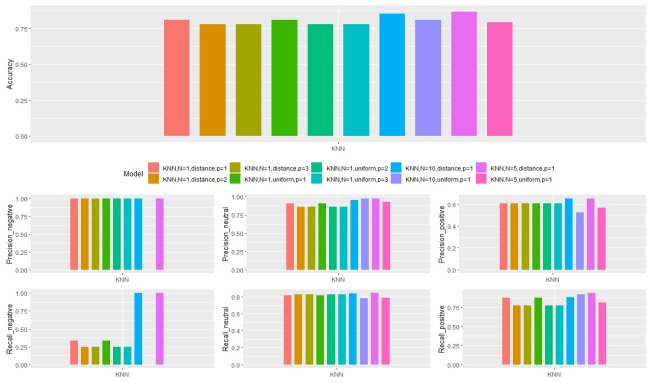

Figure 6. Metrics Results of K Nearest Neighbors Family

the same case for solvers. Besides, regularization significantly improves the accuracy, and a moderate regularization parameter is better.

- Precision and Recall: Due to the scarcity of negative symboling samples, these two metrics for some values are absent. The phenomenon will repeat itself in virtually all the models presented above. But generally, these two metrics share the similar pattern, for neutral symboling, both recall and precision are almost high for each model. For positive symboling, apparently models without regularization work worse than others.

B. K nearest neighbors

1) Parameters and arguments.

- n_neighbors - Number of neighbors used to vote for a new point.
- weights - The weights of neighbors used to vote for a new point, "uniform" indicates the same weight while "distance" indicates that closer neighbours have higher weights.
- p - specification of distance metrics, if p = 1, use Manhattan distance, if p = 2, use Euclidean distance, if p>=2, use Minkowski distance.

2) Test results. Please refer to Fig 6.

3) Discoveries.

- Accuracy: Basically, the accuracy is over 0.75 for each model, but we can find a small difference between each other. First of all, ceteris paribus, Manhattan distance works better than the other two for this data set. Situations for other parameters are a little bit more complicated. When we have a neighbour, two ways of weights work similarly while "distance" works better when we have multiple neighbors. It is reasonable since choosing weight when we have only one neighbour is meaningless. Last but not least, 10 neighbours work better with "distance" weight and 5 neighbours work better with "uniform" weight.
- Precision and Recall: There is no huge difference between precision and recall for neutral and positive symboling, while for negative symboling, recall is extremely low in all models except two.

C. Multi-layer perceptron classifier

1) Parameters and arguments.

- hidden_layer_size: A tuple indicating number of perceptrons in each hidden layer, the length of the tuple is the number of hidden layers.
- activation_function: Activation function used in the neural network. Here we use sigmoid, ReLU and tanh.
- solver- Algorithm used for optimization. Here we use adam,lbfgs and sgd.
- learning_rate: The hyper-parameter used in gradient descent variants, which indicates the update speed, when using "constant", we utilizes an invariant default learning rate 0.001, when using "adaptive", the learning rate will be adapted automatically in each iteration. For convenience, we will use "adaptive".

2) Test results. Please refer to Fig 7.

3) Discoveries.

- Accuracy: Among all the models, sigmoid activation function with lbfgs solver works the best, while sigmoid with sgd solver works the best. Ceteris paribus, with ReLU activation function, increasing hidden layer sizes improves the accuracy, while it seems to have no effect with tanh. Generally, there is no significant difference between three activation functions. When it comes to solver, lbfgs fits sigmoid, sgd fits ReLU and lbfgs fits tanh.

- Precision and Recall: Precision for negative symboling is extremely high for each model and recall is relatively low except two three-hidden-layer

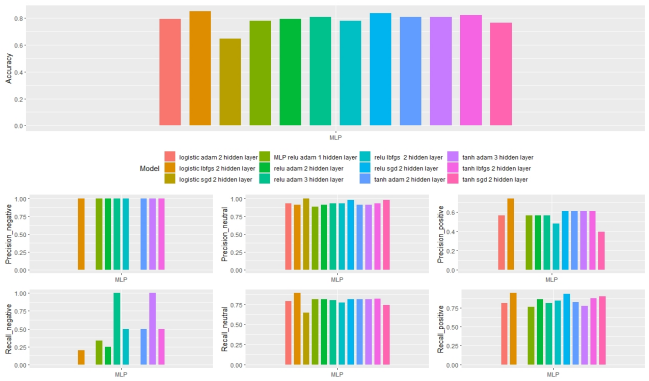

Figure 7. Metrics Results of Multi-Layer Perceptron Classifier Family

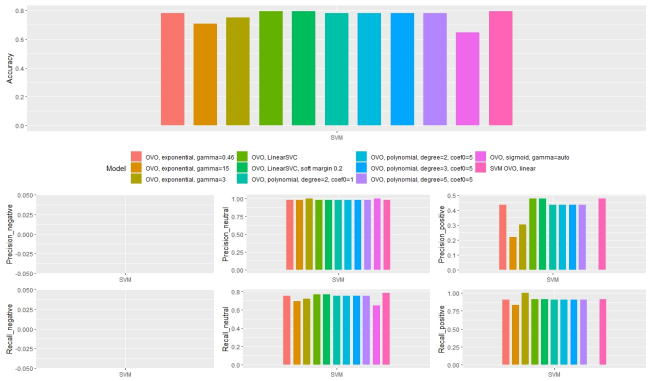

Figure 8. Metrics Results of Supporting Vector Machine Family

models. For neutral symboling, both precision and recall are high for each model except a low recall for 2 hidden layers model with sigmoid activation and sgd solver. For positive symboling, precision is relatively high for 2 hidden layers model with sigmoid activation and sgd solver, and relatively low for 2 hidden layers model with tanh activation and sgd solver, and the highest precision model also has the highest recall.

D. Support Vector Machine

1) Parameters and arguments.

- In applying the SVM models, we study only OVO models.
- kernel_functions: This is the function to map the feature space into different spaces, such that boundaries can be more flexible to split the feature space. The choices include linear, polynomial, rbf exponential, and sigmoid. The following parameters are used in forming these families of kernels.
- degree and coef0: Describes the parameter d and r in polynomial kernel functions $K(x, x') = (\gamma x^T x' + r)^d$.
- gamma: Refers not only to the γ in polynomial kernel, but also γ in rbf exponential kernel $K(x, x') = exp(-\gamma||x - x'||^2)$. In sigmoid, the parameters stands for γ in $K(x, x') = tanh(\gamma x^T x' + r)$.
- C: Describes whether to initial soft margins in linear kernels. A smaller C means a more relaxed condition and more robustness against complex data.

2) Test results. Please refer to Fig 8. This family of models suffers the most from the scarcity of data with positive symboling. Due to the complexity of the data, svm algorithms are bound to experience difficulties in determining a highly flexible boundary to divide the feature spaces. This problem has led to more missing values in evaluation metrics compared to those for other algorithms, given each of them are facing the same skewed label distribution.

3) Discoveries.

- Accuracy: The result is surprising in that, given the ill-behaving precision and recall due to the data complexity, the linear kernel seems to be the most accurate in terms of overall accuracy. The performance is not elevated much using soft margins. The models who follow are rbf exponential kernels. A gamma of 0.46 is performing the best. Polynomial kernels are giving average performance, where adjustment of parameters does not mean much. Sigmoid kernel is performing the worst.
- Precision and Recall: Taking into account that the neutral label is the most populated in the data set, this result full of missing values can make more sense if we look at just the neutral precision and recall. And it turns out that sometimes, sigmoid kernel and rbf exponential kernel with $\gamma = 3$ outperforms linear kernels. This is one justification for using these two kernels instead of simple linear kernels.
- The results suggest that linear kernels, rbf exponential kernels with $\gamma = 3$, and sigmoid kernels can be possible choices given a relatively complete data set, but linear kernels seem to outperform the other two when labels are skewly distributed.

E. Decision Tree

1) Parameters and arguments.

- Criterion: The criteria upon which features are selected to split each node when extending the decision tree. The applied choices are gini index Cart algorithms that calculate the impurity of a node, and entropy information gain criterion that detects whether the feature gives new information at a node.
- max_depth: This parameter determines the, literally, the maximum depth a tree can go. The bigger the depth, the more complex the model.
- max_features: The parameter would decide the number of features to include in the model. Like depth, this can be an indication of complexity.

2) Test results. Please refer to Fig 9.The rough result would suggest that tweaking all three parameters above would have an influence on all metrics. Larger depth does not seem to improve the prediction accuracy when talking about gini index Cart algorithms. 15 layers is generally a good choice already.

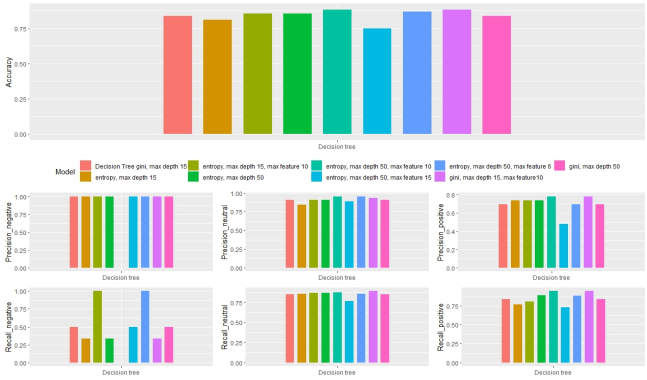

Figure 9. Metrics Results of Decision Tree Family

3) Discoveries.

- Accuracy: Looking at the big picture, entropy algorithms work better at risk rating prediction, compared to gini index Cart algorithms. The tree with 50 layers and 10 maximum features using entropy algorithm has the best accuracy. Increasing the depth improves the accuracy for entropy algorithms. On the contrary, limiting the number of features improves, up to a point, the performance of models, ceteris paribus.
- Precision and Recall: Like what happens with Accuracy, the three parameters are affecting precision and recall in a similar manner, high depth and entropy information gain is better for prediction. This similarity gives us a similar gap in metrics between different settings. But the upper bound for feature number is displaying a higher importance, with 10 as a limit significantly better than 6 or none.
- Overall, the tree with 50 layers and 10 maximum features using entropy algorithm is the top choice. The depth and number of features do not necessarily benefit the power of the model. But depth do help entropy algorithms.

Conclusion

As far as this car insurance rating supervised classification problem is concerned, the study has reached the conclusion that the following models performs the best for predictions:

- Logistic regression: ”OVR” approach with L2 norm penalty and penalty parameter 2, using solver ”saga”.
- KNN: 5 neighbors with ”distance” weight and Manhattan distance.
- MLP: Generally, 2 hidden layers with logistic activation function, using ”adam” optimization.
- SVM: For generality, linear kernels or rbf exponential kernels with $\gamma = 3$.
- Decision Tree: 50 layers and 10 maximum features using entropy algorithm.

This is by-no-means subject to all data or scenarios. In the best cases, the models above can achieve a test accuracy of around 80%, which is subject to more improvement. One basic measure would be to uncover more training samples or more features.

Acknowledgments

The authors would like to thank all deliveries and supermarkets for providing the food essential for the survival of the two authors during this special time. The completion of the article would not have been possible without their whole-hearted, though uneconomic, support.

References

[1] Dua, D. and Graff, C. (2019). *UCI Machine Learning Repository* [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[2] Pedregosa *et al.*(2011): *Scikit-learn: Machine Learning in Python*, JMLR 12, pp. 2825-2830.

[3] Qiu. X(2019) : *Neural Network and Deep Learning*. GitHub repository: https://github.com/nndl/nndl.github.io