Car Evaluation

Yuhan Meng
meng46@wisc.edu

Xueqian Zhang
xzhang2278@wisc.edu

Yuhang Lan
ylan27@wisc.edu

Abstract

As cars become increasingly more common, it could be really helpful if there is a useful algorithm to evaluate different cars. After getting the dataset, including 6 features and the class labels of cars, we transformed our data so that we could employ some algorithms like k-NN to fit the model. To find out which model has the best performance, we tried 7 models and they were k-NN, bagging with k-NN, decision tree,10 fold cross validation with k-NN and decision tree, random forest and feature selection, as well as ordinal regression algorithm. During our process of fitting models, we applied grid search to tune the hyper-parameter for each algorithm . After comparing the performance of models, we could conclude that random forest had the best performance.

key words: K-NN, decision tree, bagging, random forest, cross validation, feature selection, ordinal regression

1. Introduction

In recent years, cars play a really important role in our daily life. With them, it is convenient for us to go outside no matter what the weather is. Also, cars greatly reduce the amount of time people spend on the road. In order to give some recommendations to those people who are not sure which car to buy, we used K-NN, bagging with k-NN, 10-fold cross validation with k-NN, decision tree, random forest, 10-fold cross validation with decision tree and ordinary regression to label cars. In this report,we use Car Evaluation dataset[1] to generate our algorithms.

Since there are so many models,there is no doubt to find the best one among them is a significant and complexed task. For algorithm k-NN, bagging with k-NN, decision tree, bagging with k-NN, we split our dataset into three parts that are training set,validation set and test set. As for the rest algorithm we split our dataset into two parts (training set,test set).We use the training set to fit the model with different hyper-parameters in each kind of models and use validation set to select the best hyper-parameter for each algorithm. To find the hyper parameter in a more accurate way, we also apply cross-validation method.Then we fit the model with the hyper-parameter selected. To find the best model among all the models, we compare and evaluate the performance of these models using the test set. Finally, the best model is the model with the highest accuracy.

Once we obtain the final model,we could give suggestions to the customers about the evaluations of cars that were chosen.For example, a customer who is desired to buy a car can evaluate the condition of it by entering basic information like number of doors and capacity in terms of person to carry. What's more,it is expected not just to provide reference and standards when customers make selections on purchase, but also make the industry standardized and specialized.

[1]http://mlr.cs.umass.edu/ml/machine-learning-databases/car/

2. Related Work

In recent years, researches on machine learning has developed rapidly. It is related to the study of generalization ability, the transfer of supervised learning algorithms to multi-example learning algorithms, the application of machine learning techniques in workflow model setting, and machine learning techniques. Besides, the application of commercial applications in data mining, machine learning-based intrusion detection technology, and artificial intelligence principles in human learning also becomes more popular.

The methods that can be used for classification are: decision tree, genetic algorithm, neural network, naive Bayes, support vector machine, voting-based method, Rocchio classification, KNN classification, maximum entropy, and etc. The KNN algorithm is one of the best classification algorithms under the Vector Space Model (VSM), which was published by Cover and Hart in 1967. The basic idea is: under the VSM representation, the similarity between the sample to be classified and the training sample is calculated separately. k neighbors which are most similar to the sample to be classified are found, and the sample to be classified is determined according to the categories of the k neighbors.The core idea of the k-NN algorithm is that if the majority of the k most nearest neighbors in the feature space have a certain label, the sample also has the same label and characteristics with its neighbors. According to the concept, Nearest neighbor algorithm is really easy to understand and use. However, as a lazy learning method, the KNN algorithm does not establish a classifier until a sample needs to be classified. During the classification, the algorithm will calculate the degree of similarity with the training sample one by one. This is a huge task and the classification efficiency is low. In addition, there are deficiencies such as the influence of sample space density, similarity measure, and category judgment without considering weight.Nearest neighbor algorithm is one of the supervised machine learning.

Nonetheless, there are also some disadvantage in this algorithm. First, k-NN algorithm is particularly susceptible to the curse of dimensionality[3]. What's more, if the class distribution is skewed the outcome will be dominate by one feature and ignore other features. To overcome this problem we use weighted k-NN [2]. In 1988, Valiant took the PAC (Probably Approximately Correct) learning model, and then Valiant and Kearns also developed the concept of strong learning and weak learning, of which the recognition error rate is less than 50%. That is to say , the algorithm with a slightly higher accuracy than the random guess is always called a weak learning algorithm; a learning algorithm with a high recognition accuracy and can be completed in polynomial time is then called a strong learning algorithm. Schapire then proved that the weak learning algorithm and the strong learning algorithm were equivalent. That is, a strong learner can be integrated with multiple weak learners. In this way, during the learning process, you can find multiple weak learners, and then integrate these weak learners, after which use the integrated learner to classify. The integrated learner (that is, the strong classifier),generally has better classification performance than the normal learner. It is obviously much easier to find a single strong learner that is difficult to obtain. Therefore, the Boosting algorithm was greatly concerned in the field of machine learning.

Currently in the field of machine learning, integrated learning based on homogenous learners is generally used. What we mean by integrated learning is by default a homogeneous integrated learner. Among them, the most widely used in homogeneous integration learning is the CRAT decision tree. The decision tree model is derived from machine learning techniques in the field of artificial intelligence and can be applied to data classification and prediction. As a classical research method, it is constructed by approximating the value of discrete functions and recursively from top to bottom. As early as the 1960s, decision tree methods were produced and widely used, and a variety of algorithms were developed. Among them, Quinlan (1986) issued ID3 algorithm. The algorithm can effectively reduce the depth of the tree. Subsequently, based on the ID3 algorithm, the decision tree algorithm was further extended, including C4.5, CART, CHAID, QUEST, and so on. Among them, the C4.5 algorithm based on Information Theory (Quinlan, 1993) has greatly improved the predictive variable's missing value processing, pruning technology and derivative rules, which makes it less computationally complex, fast processing and available to wider application range (Berretti, Thampi and Srivastava, 2015).

The homogenous basic learner can be obtained in two ways. The first way is that different basic learners can be generated in parallel, and there is no strong dependency. Based on this algorithm, we have the bagging algorithm. Another way is that different basic learners are generated serially, but there are relatively large dependencies between different basic learners. Based on this algorithm, we have another boosting algorithm.

In 1994, LeoBreiman extracted the Bagging (Bootstrap Aggregation) algorithm similar to Boosting [8]. The Bagging algorithm uses repeated sampling to construct the classifier. Bagging is similar to Boosting in that it constructs a weak classifier with weak learning ability, and then integrates the weak classifier to a strong classifier with strong learning ability. The advantage of this algorithm is that it has good classification performance in unstable learning tasks, but the disadvantage is that there is often a loss of precision in more stable learning tasks.

Also in 1995, Tin Kam Ho of Bell Labs took out the Random Decision Tree (RDF) algorithm [10]. Although the traditional decision tree algorithm can achieve fast classification and high classification accuracy, it has the flaw of over-fitting when predicting test samples. The emergence of RDF solves the above shortcomings. In the RDF algorithm, the training samples used by each tree and the feature attributes of the nodes in the tree are randomly selected, and the decision tree is not pruned and completely free to grow.

In 2001, Leo Breiman combined Bagging [8] algorithm and RS [11] (Random SubSpace) algorithm to formally elaborate on Random Forest and proved that the random forest immuned over-fitting characteristic. In the random forest algorithm, for each node on each tree in the forest, it is necessary to randomly extract the attribute as the candidate for the attribute space, and then calculate each possible attribute corresponding to the candidate attribute subspace. The split value is then used to take the attribute of the optimal split value as the split attribute of the node. In addition, the training data of each tree in the random forest is also randomly selected in the original training set.

In 2004, Marko Robnik-Sikonja [5] split the decision tree nodes in random forests by combining information gain and other metrics, replacing the splitting mode of decision tree node singularity selection in the original random forest algorithm. Besides, a weighted voting method was used to predict the sample class label.

In 2009, random forest algorithm appeared in medical

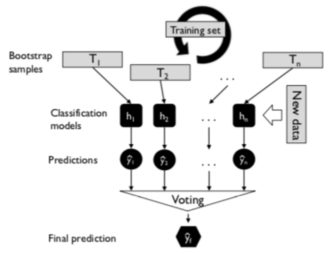


Figure 1. Bagging concept

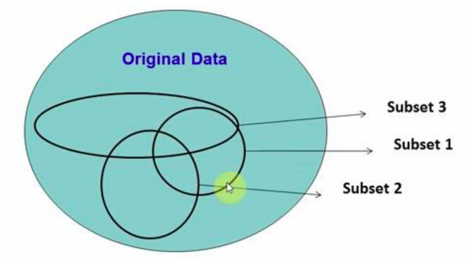


Figure 2. Random Forest

imaging diagnosis [4]. In addition, in 2011, Bjoern H. Menze and other scholars presented the Oblique Random Forest [7] algorithm, which can achieve large-scale data compression and dimensionality reduction, thereby reducing the amount of calculation.

Ordinal regression is one kind of model that has totally different mechanism. To apply ordinal regression model, the response should be ordered structured. And can be fitted by generalized linear models (GLM) with coefficients and intercepts [6]. In the domain of machine learning, models of ordinal regression have been widely proposed. For example, the PRank, an algorithm that found multiple parallel hyperplanes separating the various ranks; with output being a weight vector and a sorted vector of K-1 threshold, as in the ordered logit/probit models.[1]

3. Proposed Method

3.1. k-NN

K-NN is an algorithm for supervised learning and it is also a lazy learning algorithm because it only stores the labeled training data during training process.In order to predict new data,the k-NN algorithm first find k nearest neighbors around the new data point and then get the label of new data based on the majority voting of these neighbors.Instead of predicting the new data based on all training data,k-NN algorithm make predictions locally.It is also important to notice that this algorithm is easy to be affected by dimensionality because the more dimensions we add, the harder to find the neighbors which have the similar features with the query data.

3.2. Bagging with k-NN

Bagging with k-NN algorithm fits models on different bootstrap samples according to the same k-NN algorithm. Using bagging we can improve the accuracy of models which have a tendency to overfit. The concept of bagging is shown in Figure 1 .

3.3. Decision tree

There are many kinds of decision tree algorithms for us to choose, such as ID3, C4.5, CART and etc. CART can handle both continuous and discrete features. Besides, binary splits can generate better trees than C4.5. Although the trees will be much taller and harder to interpret, we can still choose CART algorithm.What's more, using cross validation method to tune the max depth hyper-parameter for the tree or using the feature selection approach may improve the algorithm.

As for the criterion for the tree growing, in the DecisionTreeClassifier of scikit learn library, both ”gini” and ”entropy” algorithm can be used. The former represents the Gini coefficient and the latter represents the information gain. Generally speaking, the default Gini coefficient ”gini” is always used, that is, the CART algorithm.

3.4. Random forest

The random forest is made up of several CART (Classification And Regression Tree). For each tree, the training set of it is back-sampled from the total training set, which means that some samples in the total training set may appear in the training set of a tree multiple times, or may never appeared in the training set of a tree. For each individual tree, the sets that fit the model will be the subset of the training dataset,as shown in Figure 2.

Like bagging with K-NN, we combine BaggingClassifier and DecisionTreeClassifier in order to get the random forest.

3.5. 10-fold cross validation

In the article, 10-fold cross validation is applied , both with decision tree and k-NN algorithms. In reality, the data is always limited, in order to reuse the data, scholars proposed k-fold cross-validation. 10-fold cross validation could get an average accuracy on 10 trials, which is more stable and efficient. Besides, the article also employs the stratified k-fold cross validation to ensure that the distribution in the training dataset and testing dataset are the same.

3.6. Feature selection

Large feature space may introduce many problems. The curse of dimensionality, the time complexity as well as the cost consuming are all difficulties that scholars may encounter. Feature selection is somewhat a good approach to solve this issue. In this article, we choose PCA (Principle Component Analysis) to determine the number m of features to select. Next, use Column-Drop variant.
For each feature column i:
1. temporarily remove column
2. fit model to reduced dataset
3. compute validation set performance and compare to before
Finally, we build the random forest classification task using m informative features.

3.7. Ordinal regression

In the scope of machine learning, classification with ordered classes is a crucial topic. Ordinal classification is more like a method combining classification and regression, where the response is discrete and has specific orders. [12]

Ordinal regression is a class of regression that predicts an ordered variable, i.e. a variable whose value exists on an arbitrary scale where only the relative ordering between different values is significant [2]. So when the dataset contains responses that have a clear order, we think it would be better to utilize this attribute and then the performance of our model should have a significant improvement.

Ordinal regression can be performed by generalized linear model(GLM). Assuming we have predictors as X, and the linear combination of all the predictors would be βX. The model can be formulaed as

$$f(p_i) = (1 + exp(-\beta X))^{-1}$$

. By estimating the coefficient $\hat{\beta}$, we could fit the whole model.In software R we could use function polr() in package MASS to complete this task. The command name comes from proportional odds logistic regression, highlighting the proportional odds assumption in our model.

One thing that we should pay attention to is that one of the assumptions underlying ordinal logistic (and ordinal probit) regression is that the relationship between each pair of outcome groups is the same. Which means that, ordinal logistic regression assumes the coefficients that describe the relationship between the lowest versus all higher categories of the response variable are the same as those that describe the relationship between the next lowest category and all higher categories, etc. This is called the proportional odds assumption or the parallel regression assumption. So by using ordinal regression model, we have to check this assumption.

[2]https://en.wikipedia.org/wiki/Ordinal_regression

buying	Low-0	Med-1	High-2	Vhigh-3
maint	Low-0	Med-1	High-2	Vhigh-3
doors	2-2	3-3	4-4	5more-5
persons	2-2	4-4	More-6	
Lug Boot	Small-0	Med-1	Big-2	
safety	Low-0	Med-1	High-2	
Class Labels	Unacc-0	Acc-1	Good-2	Vgood-3

Table 1. Transformation of data

4. Experiments

4.1. Dataset

Our data come from UCL machine learning repository called Car Evaluation Data Set.[3] There are 1728 observations 6 variables4 class labels in this dataset. The features we use to classify the cars are the buying price, price of the maintenance, number of doors, capacity in terms of person to carry, the size of the luggage boot and estimated safety of the car".The abbreviate of these variables and the level of them are shown below:

-buying: vhigh,high.med,low
-maint:vhigh,high,med,low
-doors:2,3,4,5more
-persons:2,4,more
-lug-boot:small,med,big
-safety:low,med,high
Besides, the labels of the cars are unacceptable, acceptable, good and very good.

4.2. k-NN

We have known that k-NN is easy to implement and also works well in practice, thus we first use the k-NN algorithm to fit the model. Since our data are not numerical, we transformed the data as shown in the Table 4.2.

After transforming the data, we can fit the model on the training set which is mentioned above. In order to find the hyper-parameter in this algorithm, we fit the model for each k from 1 to 16 and get the accuracy of the validation set for each k. The results are shown in Table 4.2

From Table 4.2,we can identify that when k=7, our model has the highest accuracy on the validation test. The Figure 3 shows the accuracy intuitively.

After selecting the hyper-parameter k=7 for the k-NN algorithm, we can fit the model reasonably. Also, we can get the accuracy of the k-NN algorithm on the test set, which is 91.33%.

[3]http://mlr.cs.umass.edu/ml/machine-learning-databases/car/

K-value	1	2	3	4
accuracy	88.4892	89.2086	92.8058	90.6475
K-value	5	6	7	8
accuracy	92.0863	89.9281	94.2446	93.5252
K-value	9	10	11	12
accuracy	90.6475	89.2086	90.6475	89.2086
K-value	13	14	15	16
accuracy	89.2086	90.6475	93.5252	89.9281

Table 2. Accuracy with different hyper-parameter k

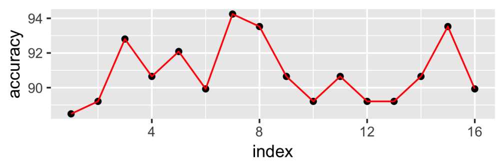


Figure 3. Accuracy with different hyper-parameter k

K-value	1	2	3	4
accuracy	94.2446	87.7690	93.5252	89.2086
K-value	5	6	7	8
accuracy	94.2446	90.6475	93.5252	92.0863
K-value	9	10	11	12
accuracy	92.8058	92.0863	92.0863	91.3669
K-value	13	14	15	16
accuracy	92.0863	92.0863	91.3669	91.3669

Table 3. Bagging Accuracy with different hyper-parameter k

4.3. Bagging with k-NN

To improve the accuracy of unstable models, which are more likely to overfit, we use bagging method. Bagging with k-NN uses the k-NN algorithm to fit models on bootstrap samples of the training data and get the predictions based on a concept which is similar to majority voting. We use 200 bootstrap samples for each k value (k=1,2,...16) and the evaluations of validation set for each algorithm are in Table 4.3

From the Table 4.3 we can find when k=1 and k=5 the accuracy is the highest which is 94.2446%. However, when k=1 it's more likely to overfit and have a bad performance because if the query point is close to a noise point, k-NN will predict the label of the noise point in all cases. Hence, to limit the influence of noise points we choose k=5 as our hyper-parameter. With k=5, the accuracy of k-NN algorithm on test set is 93.06%. It shows that the ensemble method has higher accuracy than the single method.

4.4. 10-fold cross validation with k-NN

Also we use 10 fold stratified cross validation to find the hyper-parameter of k-NN. Using k-fold cross validation method let each sample in our dataset has the opportunity to be tested. Because our data isn't that large, 10-fold stratified cross validation is applied. In each round we split the dataset into 10 parts. Each time one part of them is used for validation and the rest parts will be the training subset. After fitting the model for each k, the hyper-parameter for k-NN is 5 and the accuracy for test set is 93.06% which is the same as the result of bagging with k-NN algorithm.

max-depth	1	2	3	4
accuracy	69.7842	80.5755	82.0144	85.6115
max-depth	5	6	7	8
accuracy	88.4892	92.8058	92.0863	96.4029
max-depth	9	10	11	12
accuracy	95.6835	97.8417	98.5612	98.5612

Table 4. (Gini)Accuracy with different max depth h

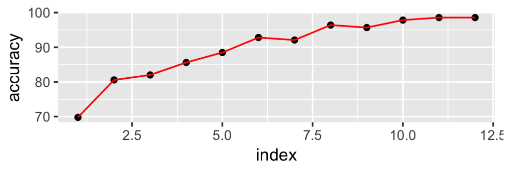


Figure 4. "Gini" validation accuracy

max-depth	1	2	3	4
accuracy	69.7842	80.5755	82.0144	84.8921
max-depth	5	6	7	8
accuracy	86.3309	92.8058	92.0863	96.4029
max-depth	9	10	11	12
accuracy	95.6835	97.8417	97.8417	97.8417

Table 5. (Entropy)Accuracy with different max depth h

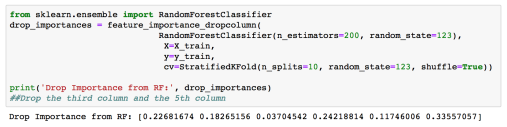
4.5. Decision tree

For an individual decision tree, in order to choose a max depth hyper-parameter so that the validation set's accuracy as well as the test accuracy will be large. We fit the model for each max depth h from 1 to 12 and get the accuracy of the validation set for each h. First, we select the "gini" tree growing algorithm. The results are shown in Table 4.5

We graph the prediction accuracies of the validation set via R studio shown in Figure 4.

From both the graph and the table, we can identify when the tree's max depth is 11, it reaches the maximum accuracy. The validation accuracy under this circumstance is 98.56%, and the test accuracy will be 97.11%.

Similarly,we can also apply the method with entropy criterion and gain another table as is shown in Table 4.5 This time, the optimum max depth is 10. And the validation accuracy is 97.84%. Fitting this model to the test set, we get the test accuracy 96.82%. Both of these results are lower

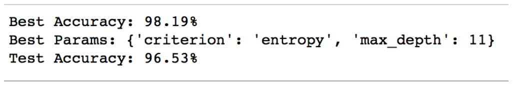


```
from sklearn.ensemble import RandomForestClassifier
drop_importances = feature_importance_dropcolumn(
                            RandomForestClassifier(n_estimators=200, random_state=123),
                             X=X_train,
                             y=y_train,
                             cv=StratifiedKFold(n_splits=10, random_state=123, shuffle=True))

print('Drop Importance from RF:', drop_importances)
##Drop the third column and the 5th column
```

```
Drop Importance from RF: [0.22681674 0.18265156 0.03704542 0.24218814 0.11746006 0.33557057]
```

Figure 5. Random Forest codes



```
Best Accuracy: 98.19%
Best Params: {'criterion': 'entropy', 'max_depth': 11}
Test Accuracy: 96.53%
```

Figure 6. 10-fold cross validation grid search

than the 'gini' criterion decision tree model.

4.6. Random Forest

Random forest model is an ensemble method and can be regarded as bagging with decision trees. We fit 200 bootstrap samples for 200 individual tree classifiers. Here, the hyper-parameters for the classifiers are 11 (max depth) and 'entropy' (criterion). The reason that we opt to these hyper-parameters is based on the output of the grid stratified 10-fold cross validation, which will be shown later in our article.

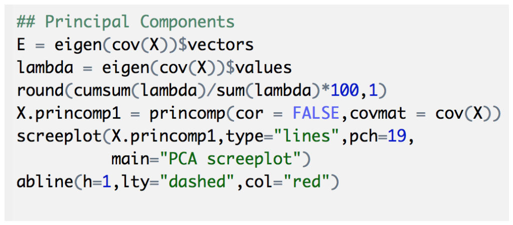
After confirming the individual classifier and fitting the model, we may get the predictions via majority voting of these 200 trees. The validation accuracy and test accuracy are 98.56% and 97.40%, which are higher than the single decision tree model. A few lines of codes are shown in Figure 5

4.7. 10-fold cross validation with decision tree

In order to choose the best hyper-parameters for the random forest model, we use the cross validation method. From the grid search, we identify the best hyper-parameters are 'entropy' criterion and 11 max depth. The accuracies are pretty high for these parameters. The output result is shown in Figure 6

4.8. Feature selection

Consequently, base on the results above, we tend to choose the random forest model. Next, we try to apply feature selection to the random forest model. There is always the question of how many features to retain. There is no definitive answer to this question. Things to consider include the amount of total sample variance explained, the relative sizes of the eigenvalues (the variances of the sample components), and the subject-matter interpretations of the components. A useful visual aid to determining an appropriate number of principal components is a scree plot. With the eigenvalues ordered from largest to smallest, a scree plot is a plot of λ versus ithe magnitude of an eigenvalue versus its number. To determine the appropriate number of com-



```
## Principal Components
E = eigen(cov(X))$vectors
lambda = eigen(cov(X))$values
round(cumsum(lambda)/sum(lambda)*100,1)
X.princomp1 = princomp(cor = FALSE,covmat = cov(X))
screeplot(X.princomp1,type="lines",pch=19,
          main="PCA screeplot")
abline(h=1,lty="dashed",col="red")
```

Figure 7. PCA R codes

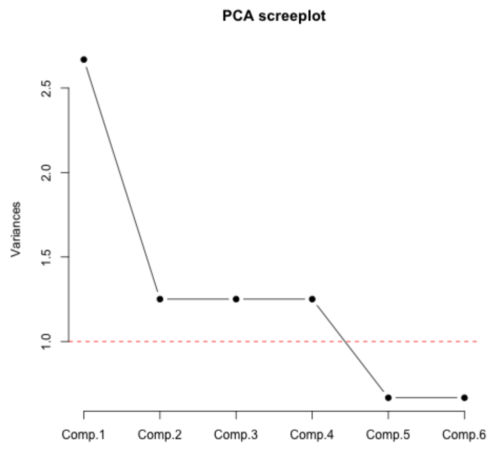
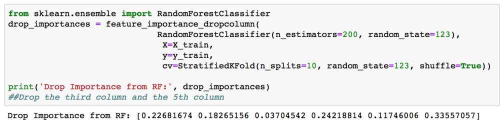


Figure 8. PCA screeplot



```
from sklearn.ensemble import RandomForestClassifier
drop_importances = feature_importance_dropcolumn(
                            RandomForestClassifier(n_estimators=200, random_state=123),
                             X=X_train,
                             y=y_train,
                             cv=StratifiedKFold(n_splits=10, random_state=123, shuffle=True))

print('Drop Importance from RF:', drop_importances)
##Drop the third column and the 5th column
```

```
Drop Importance from RF: [0.22681674 0.18265156 0.03704542 0.24218814 0.11746006 0.33557057]
```

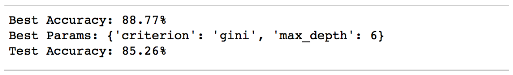
Figure 9. column drop variant method

ponents, we look for an elbow (bend) in the scree plot. Another rule of thumb is to choose those λs greater than 1.[9] The R codes and the screeplot are shown in Figure 8

From the graph, we identify 4 features satisfy this condition($lambda > 1$). Hence,we use column drop variant approach and select 4 features all together. A few lines of codes are shown in Figure 9 .

The output provides the result that the best approach is to drop the third and the fifth columns' features. After dropping these relatively insignificant features, we apply the 10-fold cross validation grid search to tune the parameter. We identify the best hype-parameters changed to 6 (max depth) and "gini"(criterion). Nonetheless, this time if we fit the validation set and test set, both the accuracies will decrease dramatically. The output results are shown in Figure 10

This time we can draw the decision tree and interpret the tree easier because the max depth is much smaller than the



```
Best Accuracy: 88.77%
Best Params: {'criterion': 'gini', 'max_depth': 6}
Test Accuracy: 85.26%
```

Figure 10. grid search with 4 features

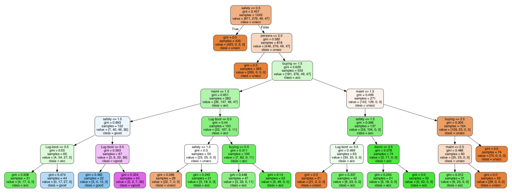


Figure 11. 'Gini' Decision Tree

random forest model. Figure 11

Due to the fact that the accuracy is somewhat much lower than random forest model with all features, and the difference between the test set accuracy and validation set accuracy isn't that large in the previous model, which indicates there isn't obvious overfitting problem, we still choose not to reduce the feature dimensions in order to maintain the accuracy of the prediction.

4.9. Ordinal regression

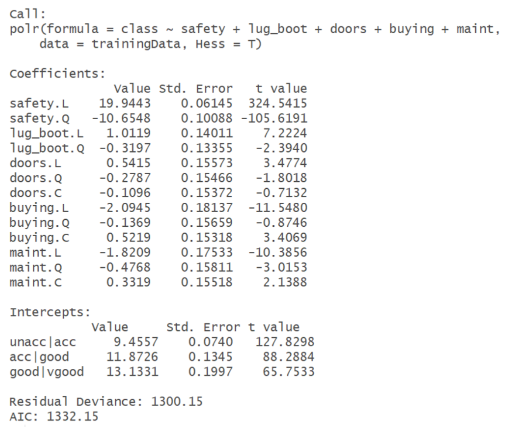
To make full use of the structure of our dataset, we fit the ordinal regression model. As we show earlier, the responses of our dataset is "unacc","acc","good","vgood", which obviously have a clear ordered structure.

First of all, we coded all of our predictors and variables into ordered factors for practical reasons. Then by experience we divided 70% of our dataset into training set and 30% into testing set. After that, we fitted an ordinal model, the coefficients and intercepts are as below in the Figure 13. From this model we conducted predictions on the testing set we preserved earlier. And drew a confusion matrix based on the true classes and predicted classes. The results of the confusion matrix is as below in Table 6.

In the R output in Figure 13 we can see that

- Call: the type of model we ran.
- Coefficients: regression output coefficient table including the value of each coefficient, standard errors, and t value.
- Intercepts: where the latent variable is cut to make the three groups that we observe in our data. However, in our model, since the response variable is discrete, this quantity is not really meaningful.
- Residual Deviance and AIC.

Typically, we calculate the accuracy by summing up the diagonal numbers of the confusion matrix and dividing by



```
Call:
polr(formula = class ~ safety + lug_boot + doors + buying + maint,
    data = trainingData, Hess = T)

Coefficients:
             Value Std. Error   t value
safety.L   19.9443    0.06145  324.5415
safety.Q  -10.6548    0.10088 -105.6191
lug_boot.L  1.0119    0.14011    7.2224
lug_boot.Q -0.3197    0.13355   -2.3940
doors.L     0.5415    0.15573    3.4774
doors.Q    -0.2787    0.15466   -1.8018
doors.C    -0.1096    0.15372   -0.7132
buying.L   -2.0945    0.18137  -11.5480
buying.Q   -0.1369    0.15659   -0.8746
buying.C    0.5219    0.15318    3.4069
maint.L    -1.8209    0.17533  -10.3856
maint.Q    -0.4768    0.15811   -3.0153
maint.C     0.3319    0.15518    2.1388

Intercepts:
            Value      Std. Error t value
unacc|acc     9.4557     0.0740   127.8298
acc|good     11.8726     0.1345    88.2884
good|vgood   13.1331     0.1997    65.7533

Residual Deviance: 1300.15
AIC: 1332.15
```

Figure 12. The fitted the coefficients and intercepts of the ordinal regression model.

	unacc	acc	good	vgood
unacc	305	45	0	4
acc	60	60	0	0
unacc	0	17	0	4
acc 2	0	18	0	10

Table 6. This is an example of a table.

the total number of testing set. In this case, the "accuracy" calculated in this way would be 72.25%. However, in this particular case, since the response is ordered, it would be a waste if we do not use the information that a specific variable is closer to some variables while further from other variables. But how can we utilize this information is a remaining problem we need to figure out. Here we tried a traditional method, as calculating the MSE, i.e. mean square errors. The MSE in this particular circumstance is 0.4432.

Further more, we used 10 fold cross validation to find a more stable estimation of all these coefficients. The mean MSE of all these validations is 0.4144.

Maybe you have noticed that, the predicted result of ordinal regression is kind of surprising and counter-intuitive. Since we utilized the structure of the dataset, we would have expected a much higher accuracy. However, the accuracy is lower in reality. Now we have to check if the assumption of our model is satisfied. We have known that for each predictor variable, distance between the symbols for each set of categories of the dependent variables should be the same. And we can diagnose this assumption by a graphical method for assessing the parallel slopes assumption. The values displayed in this graph are essentially (linear) predictions from a logit model, which are used to model the probability that y is greater than or equal to a given value (for each level of y), using one predictor (x) variable at a time. The graph is plotted below as Figure 14.

According to this plot, we can conclude that the distance

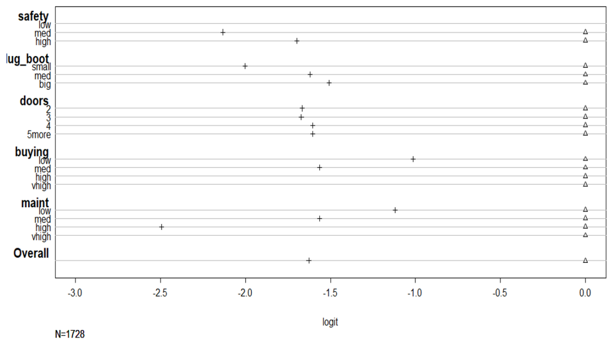


Figure 13. Model diagnostic to assess the parallel slopes assumption

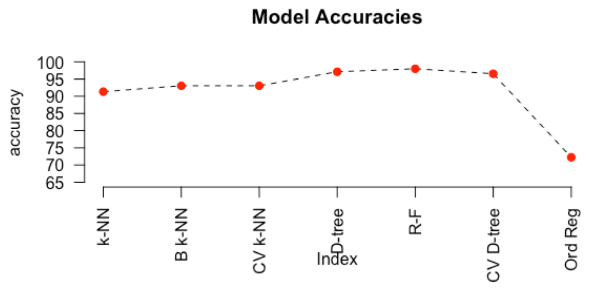


Figure 14. Prediction accuracies of difference models

between the sets of coefficients of doors is similar. In contrast, the distances between the estimates for ”maint”, ”buying” and ”safety” are very different, suggesting that the proportional odds assumption may not hold. And that might be the reason why our model has such a low accuracy: the ordinal regression might not be suitable for our data.

4.10. Software

For most parst, we used Python to do the calculation and modeling, including K-NN, decision tree, bagging, random forest, cross validation and feature selection(except PCA). Python has very complete functions for them. And for the ordinal regression part, since we don’t have a specific library to do that kind of work, we use R instead.

5. Results and Discussion

Results: Based on all the output of our models, it’s obvious that the performance of the random forest model is the best. It has both the highest validation accuracy and test accuracy. Besides, the difference of these two accuracies isn’t dramatic, which indicates no serious overfitting problem. The prediction accuracies of each model we fit before are graphed as Figure 14:

Discussion: What we should do next is to apply this model to the new real-world data. For instance, by collecting the feature values of different types of cars, like the buying price,the number of the doors, the size of the luggage boot,etc, we might then predict the evaluation labels, either ’acceptable’ or ’unacceptable’, ’good’ or ’very good’. Then, we might give customers some reference for the feasibility and preference for buying those types of cars. Moreover, we might also do some extension to this model, if possible, add more features to the dataset and fit a new model to provide even better assistance to the customer for buying cars.

6. Conclusions

According to all the discussions above, we have several conclusions. First of all, random forest, as a method to process classification, has the best performance. If we plan to do some classification in the future, random forest should be the first method to consider. Secondly, to choose a perfect car, there are plenty of factors to take into account, and the most important things to think about include safety index, maintenance fees, price and so on. Thirdly, always remember to check the assumption before fitting a model, before it is too late and you have run all the procedures. Sometimes, choosing a good method doesn’t mean that everything is done. A good method can perform badly on your specific dataset.

7. Acknowledgements

First of all, thanks to Marko Bohanec and Blaz Zupan, two generous gentlemen who provided us the amazingly complete and useful dataset. And much gratitude to Professor Sebastian Raschka, who is always full of energy and answers tons of our stupid questions. And he always encourages us to pursue more of our work. Last but not least, thanks to my amazing teammates. Finally our work pays with all these gorgeous models.

8. Contributions

Although the dataset we choose isn’t that large, and the model we choose are quite basic, we gain great prediction accuracy and we are really appreciated for the final result. During the process of working, each member does their task earnestly. Yuhan Meng finished data preprocessing, KNN model as well as bagging with KNN ensemble method. Xueqian Zhang constructed decision tree, random forest model and tried to apply feature selection approach for the model. Yuhang Lan applied ordinal regression model and did some model diagnostic. All of us tuned the hyper-parameter through 10-fold cross validation grid search. Our professor also provided us great assistance and solved many problems during the process of our project. These all together leads to our final project completion.

References

[1] S. Y. Crammer, Koby. Pranking with ranking. *NIPS*, 2001.

[2] D. M. D. Coomans. Alternative k-nearest neighbour rules in supervised pattern recognition. *Analytica Chimica Acta*, 1982.

[3] D. L. D. et al. High-dimensional data analysis :the curses and blessing of dimensionality. *AMS math challenges lecture 1.2000*, 2000.

[4] N. A.]Lempitsky V. Verhoek M. Random forest classification for automatic delineation of myocardium in real-time 3d echocardiography. 2009.

[5] R.-I. M. Improving random forests. 2014.

[6] P. McCullagh. Regression models for ordinal data. *Journal of the Royal Statistical Society. Series B (Methodological), vol. 42, no. 2*, 1980.

[7] S. D.]Menze B.H., Kelm B.M. On oblique random forests. 2011.

[8] B. L. B. predictors. Machine learning. 1996.

[9] D. W. Richard Johnson. Applied multivariate statistical analysis. 2014.

[10] H. T.K. Random decision forests. 1995.

[11] H. T.K. The random subspace method for constructing decision forests. 1998.

[12] R. D. Winship, Christopher; Mare. Regression models with ordinal variables. *American Sociological Review. 49*, 1984.