Automobile Prediction

Toby Chappell

Abstract

The goal of this project is to accurately predict the insurance risk rating in addition to the make of a car. To help predict the symbol of a car, it is categorized into two categories which allows for greater accuracy during the prediction process. Prior to running different algorithms, the data is cleaned up and preprocessed using PCA to retrieve the three most relevant attributes. Each attribute is then processed using three different machine learning algorithms including: linear regression, logistic regression, k-nearest neighbors, and decision trees. In addition, each model is compared using PCA and domain-expert selection as inputs.

Introduction

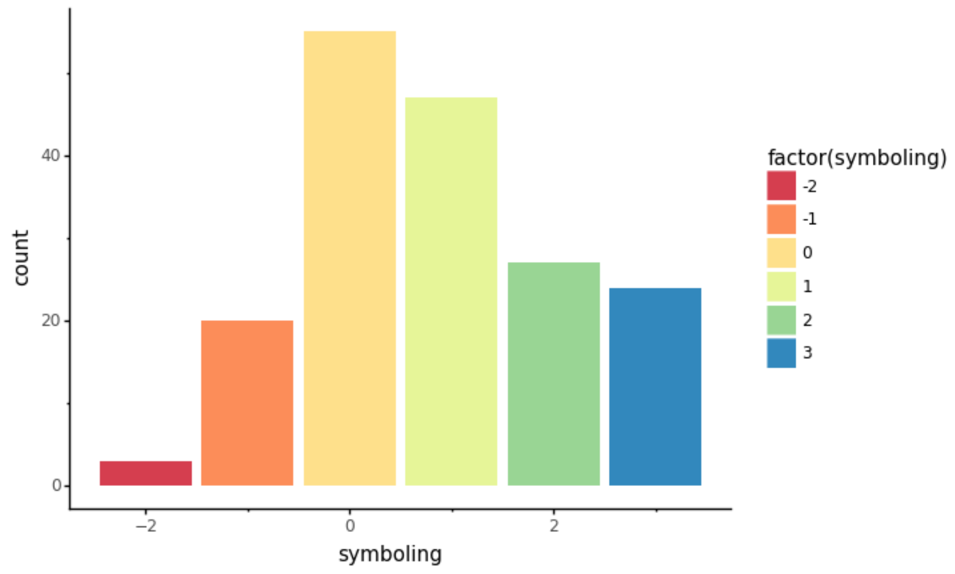
The primary task of this project is to predict the risk factor of a car. This feature is referred to as "symboling" throughout the data. The term symboling in this context corresponds to the degree to which the car is riskier than its price indicates. This factor ranges from discrete values of -3 to +3 in which a symboling of +3 would be considered risky and a symboling of -3 would be considered safe. In addition, the attribute is split up into "safe" and "risky" to allow for binary classification, being termed "riskiness" for simplicity. Being able to predict whether a car is a risky investment provides useful information when purchasing a new car. If a consumer wants to sell their car ever in the future, they will be more inclined to buy a car that is a safe investment. Furthermore, it can be used to help predict whether the price of insurance will be high or low for a particular car. Lastly, I thought it would be interesting to predict the make of a car. While it does not have many practical applications, the attribute is spread and diverse making it more of a challenge to predict.

Data

While conducting my exploratory data analysis I found there were 205 rows and 26 columns. Out of the 26 columns, 16 were a numerical data type and the remaining 10 were strings. Additionally, there existed 59 question marks ('?') in the data to represent null values.

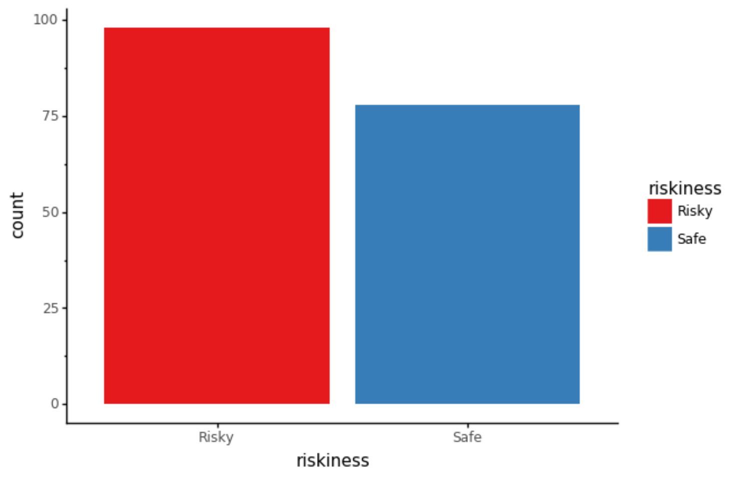
My first and foremost question is to be able to predict the symboling of a car. As shown in Figure 1.1, the symboling factor is skewed to the left meaning cars are more often than not considered a risky investment. Due to the skew in the data, it's probable that the models may have trouble predicting lower values since most of the data consists of larger values.

Figure 1.1 Distribution of Symboling



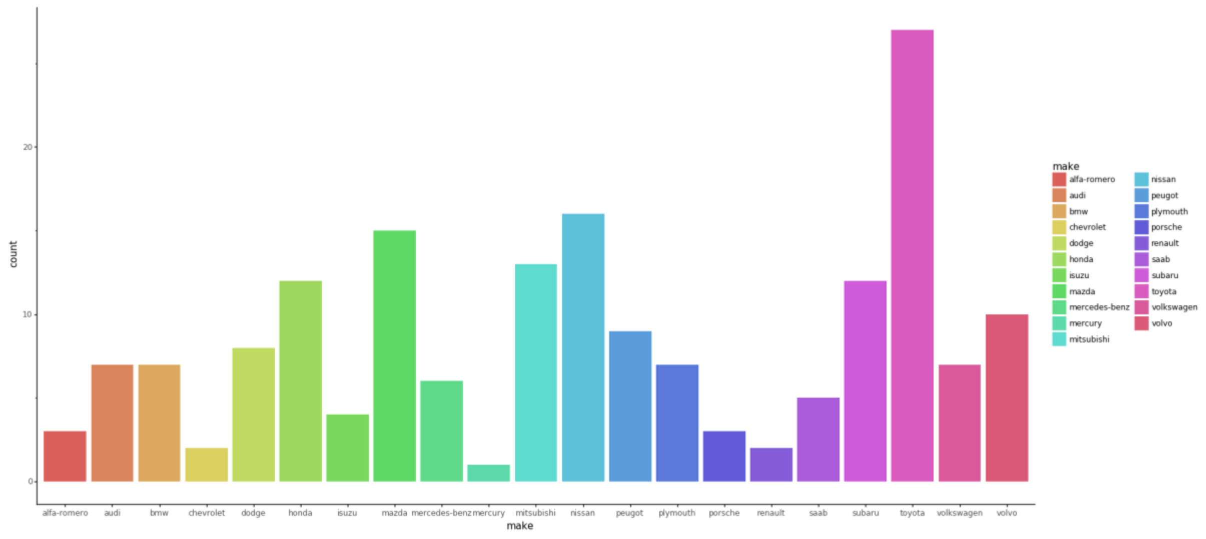
Next, I decided to create a binary classification for symboling called "riskiness" which assigns a symboling less than or equal to zero as "Safe" and a symboling greater than zero as "Risky." After categorizing the symboling factor, Figure 1.2 shows that the data is distributed fairly well allowing for greater accuracy during the prediction process. However, there is some loss of information such as the degree to which a car is risky or safe (which may or may not be relevant).

Figure 1.2 Distribution of Riskiness



Lastly, I attempted to predict the make of a car. Due to the number of possible makes as seen in Figure 1.3, a lower accuracy is to be expected. Additionally, the data has a varying distribution meaning the algorithms may be slightly biased towards the makes that exist in larger quantities.

Figure 1.3 Distribution of Make



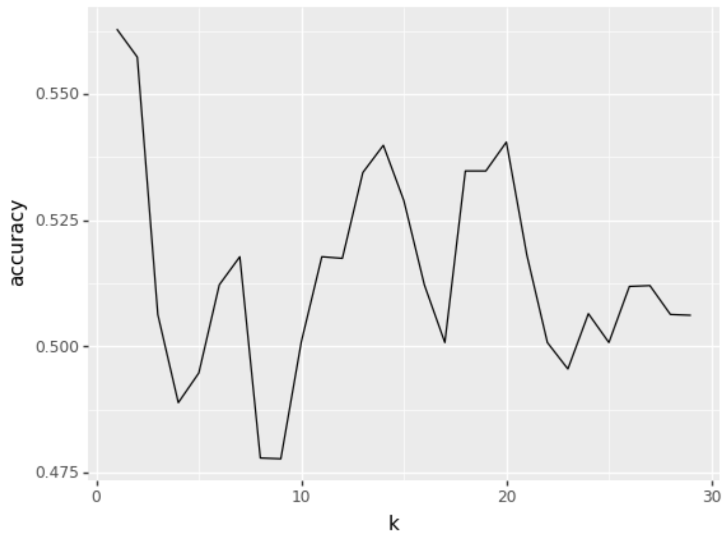
Methodology

Before running any models on the data, I first had to clean the data up. My first issue was how to deal with the question marks in the data. To handle this, I first removed all question marks in the data and replaced them with null. Next, I had to change the column type from an object (this was due to the fact that there were question marks in the data) to a float for all numerical attributes. I then replaced each null value with the average of that particular numerical attribute so that the integrity of data remained intact. Finally, there were only two categorical values with a null value which I proceeded to drop from the table. In order to remove any outliers in the data, I found the z-scores of each row and removed any rows with a z-score of three or above which are considered unusual. This reduced the number of rows that existed in the data by 27, leaving 176 rows left in total in the table. Lastly, I performed a Principal Component Analysis on the data to get the three most relevant features in the data.

For each predicted attribute, I used three different machine learning algorithms using both PCA and all of the numerical attributes for comparison. Each model is run 50 times validated using k-fold Cross Validation (with the value of k equal to five) to ensure greater accuracy. In addition, the data is split into an 80% training set and a 20% testing set.

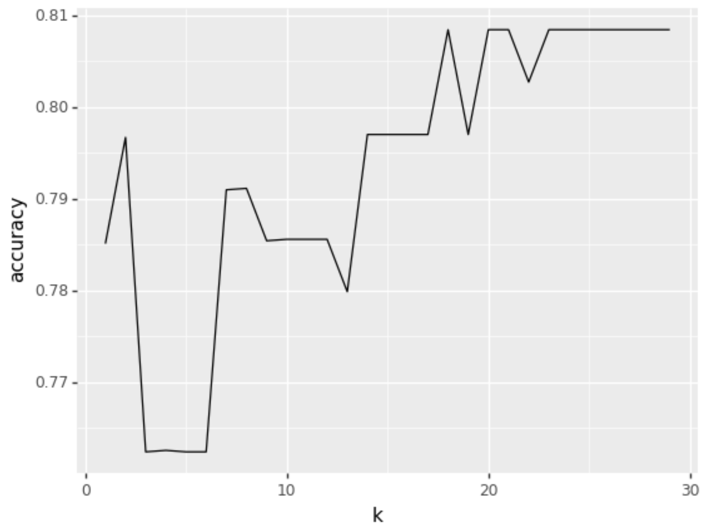
In order to predict symboling, I ran linear regression, logistic regression, and k-nearest neighbors. Running linear regression was simply to fit the model against the training set and validate using the testing set. For logistic regression, the respective model is created and fit. In addition, a confusion matrix is generated to visually display the accuracies. Finally, in order to predict symboling using k-nearest neighbors the optimal value of k needs to be computed. To do this, I graphed the accuracy vs. the value of k as shown in Figure 2.1. As seen in the figure, the optimal value of k is 1 for predicting symboling.

Figure 2.1 Optimal k-Value of Symboling



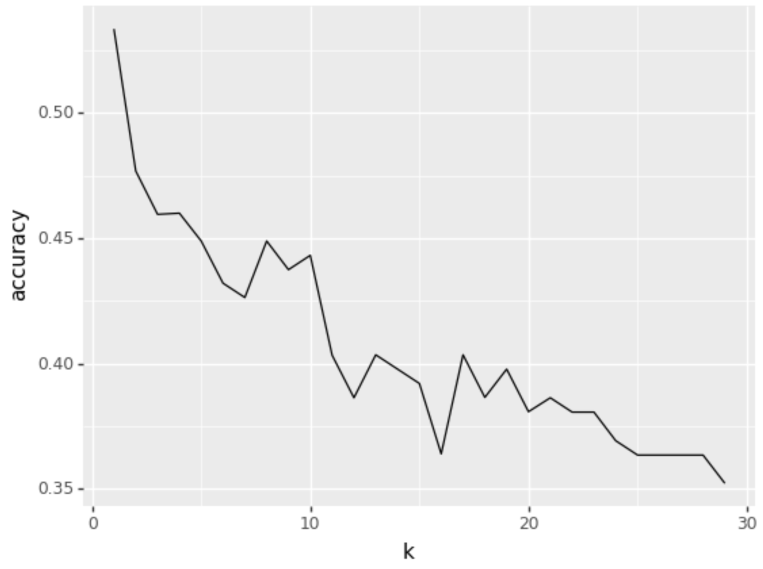
I then ran logistic regression, k-nearest neighbors, and decision trees to predict riskiness. The methodology for running logistic regression and k-nearest neighbors remains the same. As seen in Figure 2.2, the optimal value of k is 20 for k-nearest neighbors in predicting riskiness. Lastly, decision trees ran fairly similar in that the data was split, model fitted, and accuracy score produced and averaged out.

Figure 2.2 Optimal k-Value of Riskiness



To predict make I used logistic regression, k-nearest neighbors, and decision trees. The methods involved were the same in predicting riskiness. One note is that the optimal value of k found was 1 as seen in Figure 2.3.

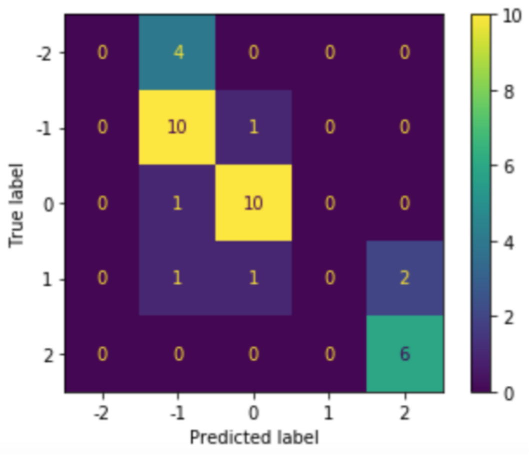
Figure 2.3 Optimal k-Value of Make



Results for Symboling

In addition to finding the accuracies for finding symboling, I computed the confusion matrix for logistic regression as seen in Figure 3.1.

Figure 3.1 Confusion Matrix for Symboling



	-2	-1	0	1	2
-2	0	4	0	0	0
-1	0	10	1	0	0
0	0	1	10	0	0
1	0	1	1	0	2
2	0	0	0	0	6

The final accuracies for predicting symboling are shown in Table 1.1 below.

Table 1.1 Accuracy Scores for Symboling

Algorithm	**Domain-Expert**	**PCA**
Linear Regression	0.332	0.637
Logistic Regression	0.393	0.558
k-Nearest Neighbors	0.438	0.562

Based on the table, linear regression produced the best accuracy of 0.637. The advantage of using linear regression is that it is fast and easy to implement. However, the downside to linear regression is that it cannot predict categorical data meaning it cannot be used in order to predict the riskiness attribute which is easier to predict in comparison to symboling.

Results for Riskiness

A sample confusion matrix for predicting riskiness is shown in Figure 3.2. The recall, precision, and accuracy.

Table 1.2 Confusion Matrix

Algorithm	**Values**
Recall	0.913
Precision	0.913
Accuracy	0.889

Figure 3.2 Confusion Matrix for Riskiness

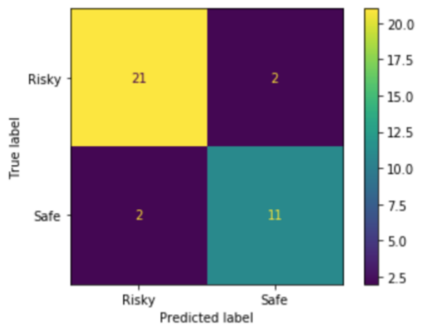
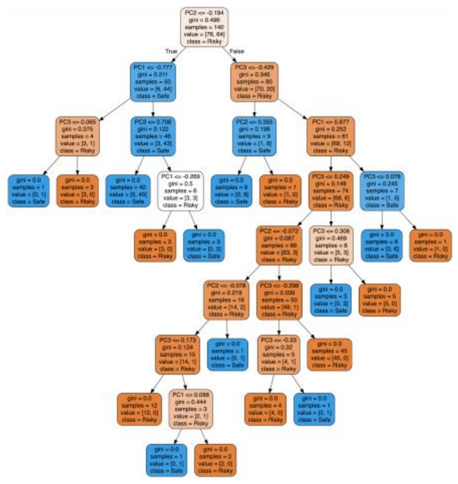


Figure 3.3 Decision Tree for Riskiness



Accuracies for riskiness is shown in Table 1.2 below.

Table 1.3 Accuracy Scores for Riskiness

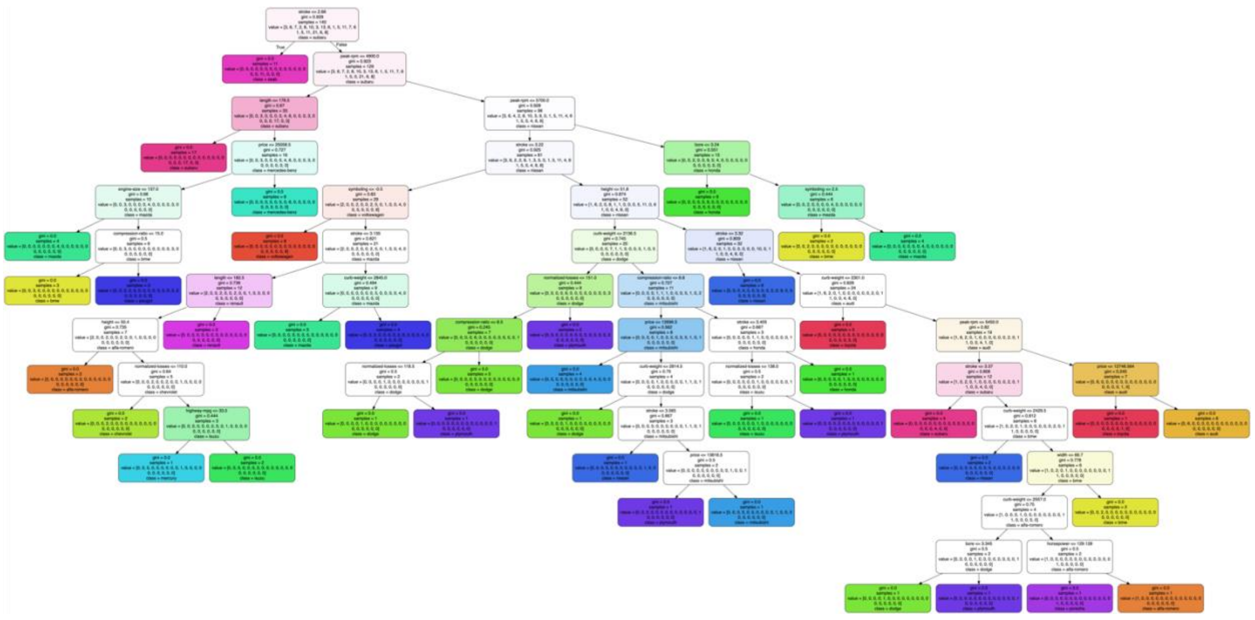
Algorithm	**Domain-Expert**	**PCA**
Logistic Regression	0.694	0.791
k-Nearest Neighbors	0.591	0.808
Decision Trees	0.616	0.750

Using Table 1.2, k-nearest neighbors yielded the best accuracy of 0.808. The benefits of using k-nearest neighbors is that it has no assumptions and is easy to implement. However, a disadvantage to using this model is that it can get slow very quickly.

Results for Make

Lastly, I computed the decision tree for make as seen in Figure 3.4, however, I did not compute the confusion matrix for logistic regression since there are 21 different makes (making the confusion matrix unhelpful and unreadable).

Figure 3.4 Decision Tree for Make



The accuracy scores for make can be found in Table 1.3 below.

Table 1.4 Accuracy Scores for Make

Algorithm	**Domain-Expert**	**PCA**
Logistic Regression	0.262	0.375
k-Nearest Neighbors	0.284	0.533
Decision Trees	0.689	0.519

As seen in the table, decision trees led to the best accuracy of 0.689. The reason for this is because decision trees are able to pick up on attributes in small quantities or over a diverse range. However, decision trees are also prone to overfitting. Due to the size of the tree as seen in Figure 3.4, this could very well be the case.

Conclusion

The final accuracies for each algorithm were as follows: symboling using linear regression - 0.637, riskiness using k-nearest neighbors - 0.808, and make using decision trees - 0.689. Based on the results, PCA performed better eight out of nine times in comparison to the domain-expert selection. The reason for this is because PCA brings out the best predictors' attributes in the data. While more attributes may provide more knowledge about the data, some attributes may actually make it harder to predict since they are given equal weight.

In terms of future applications, the prediction of symboling can be used for insurance agencies to determine what to charge based on the predicted result. From a consumer standpoint, it might not be necessary to know the exact risk factor symbol of a car. As such, being able to know if a car is a risky or safe investment would be more than sufficient. Furthermore, consumers can use this prediction to determine if the price of insurance for the vehicle will be high or low in comparison to the price of the car.