**3. Data Preparation**

While data preparation is often an overlooked step in the entire data mining process, it is perhaps the most time consuming and fundamentally important as it is the building block for all other steps that follow. It is crucial to note that “often the quality of the data mining solution rests on how well the analysts structure the problems and craft the variables” (Provost 2012). With this in mind, we began our the first step of analyzing the available attributes and finding the ones that presented the most information gain and importance in solving our problem.

It is also important to note that while this section and the others that follow are written in a linear fashion, the process itself was not at all. There was constant rethinking of ideas, methods and conclusions that took place in real time, however for the sake of the reader, we shall present it as a linear process.

*3.1 Attribute Selection*

The first step was taking a look at the available attributes and converting numeric attributes into nominal ones using the following function in R:

Figure 3.1:

Test\_data$Attribute <- factor(Test\_data$Attribute)

After converting many of the variables to the desired format, we then proceeded to take a look at the information gain using a ranker method in WEKA. It should be noted that numeric variables were discretized on the optimal splitting points that gave the highest information gain.

Figure 3.2:

|  |  |  |
| --- | --- | --- |
| **Rank** | **Attribute** | **Information Gain** |
| 1 | MMRCurrentRetailCleanPrice | 0.1591972 |
| 2 | MMRAcquisitonRetailCleanPrice | 0.1525066 |
| 3 | MMRCurrentRetailAveragePrice | 0.1521401 |
| 4 | MMRAcquisitionRetailAveragePrice | 0.1475890 |
| 5 | MMRCurrentAuctionCleanPrice | 0.1378067 |
| 6 | MMRAcquisitionAuctionCleanPrice | 0.1352096 |
| 7 | MMRCurrentAuctionAveragePrice | 0.1276983 |
| 8 | MMRAcquisitionAuctionAveragePrice | 0.1272369 |
| 9 | WheelTypeID | 0.0637566 |
| 10 | WheelType | 0.0637026 |
| 11 | Model | 0.0319518 |
| 12 | SubModel | 0.0240122 |
| 13 | VehicleAge | 0.0196533 |
| 14 | VehYear | 0.0179228 |
| 15 | VehBCost | 0.0131958 |
| 16 | VNZIP1 | 0.0126275 |
| 17 | PurchDate | 0.0117275 |
| 18 | WarrantyCost | 0.0092531 |
| 19 | BYRNO | 0.0072054 |
| 20 | Trim | 0.0070979 |
| 21 | VehOdo | 0.0056278 |

The above table gives an idea of the top 21 most relevant variables when trying to predict the target variable *IsBadBuy*. Although the attribute *Model* seems to show a relatively high information gain, it does not tell the whole story. The variable *Model*, has over a thousand unique variables, making its information gain highly likely to be a case of overfitting, which is why we did not include in the following model. The same is true for *SubModel*, *Trim, VNZIP1 and BYRNO,* all of which represented a case of overfitting.

It is also important to realize that the market price of the car represented by the six attributes that begin with *MMR*, all take into account the Model, Trim and Submodel of the car as well as many of the unique features since the market price is unique to each car[[1]](#footnote-1). However it should also be noted that the market price of a car in absolute terms should have little predictive ability since some cars are worth more than others. With this in mind, the market price of a car is only useful when it is used in relative terms.

In terms of the information gain given by *VehBCost*, we could not include it since it represents a case data leakage[[2]](#footnote-2). The reason this is the case, is because the cost of the vehicle (price paid) is not known when looking to bid on a car. You only know the price paid for the vehicle once the bidding has ended. For this reason it could not be used.

The variable *PurchDate* was also not included in the model for intuitive purposes. The past date of purchase of car has little, if not zero, predictive ability when looking at future cars in an auction.

By looking at the information gain given by each attribute on the target variable of *IsBadBuy*, we were able to grasp what variables to include and what variables to use in feature creation process that was to follow.

*3.2 Feature Creation*

Using the information gain of each variable and the domain knowledge and intuition to remove features that represented leakage, overfitting and other potential data mining obstacles, we proceed to use our understanding of the data to come up with new features that could possibly have higher information gains combined, rather than separate.

We first identified the top attributes that represented the car’s market price at current day and when it was first bought. As we noted before, the stand alone price of the car says nothing without it being compared to a relative term. With this in mind, we constructed 4 attributes that represented the difference between the auction and retail market price of the car when it was first purchased and the auction and retail market price of the car in current day adjusted for the age of the car.

We also created one more variable to represent the fact that a pure odometer reading for the car says little without accounting for how old it is. To better illustrate this, think of a car that was bought and then driven form the east coast to the west coast in a year and then sold, versus a car that was driven from the east cost to the west coast in 3 years. Their odometer would read the same, however the average miles driven would be drastically different. Each of these cars should be treated differently in terms of the quality of the car, which is why we created an *AdjVehOd* that took into account this information. For this fact we created an adjusted odometer attribute that represented the average miles driven per year rather than a pure odometer reading.

Figure 3.2 (Python Script in Appendix A)



After creating these attributes we then ran another information gain (“IG) test to compare out newly created features and their IG versus their original features. As you can see in the chart below, each of the created variables had a higher information gain than their original counterparts.

Figure 3.3

|  |  |  |
| --- | --- | --- |
| **Rank** | **Attribute** | **Information Gain** |
| 1 | WheelTypeID | 0.0639567 |
| 2 | WheelType | 0.0639020 |
| 3 | Model | 0.0320760 |
| 4 | DifAquiredAvg\_CurAvgAuction | 0.0248049 |
| 5 | SubModel | 0.0238734 |
| 6 | DifAquiredAboveAvg\_CurAboveAvgAuction | 0.0234160 |
| 7 | DifAquiredAvg\_CurAvgRetail | 0.0204563 |
| 8 | VehicleAge | 0.0197949 |
| 9 | DifAquiredAboveAvg\_CurAboveAvgRetail | 0.0189444 |
| 10 | VehYear | 0.0179685 |
| 11 | MMRAcquisitionAuctionAveragePrice | 0.0141110 |
| 12 | MMRCurrentAuctionAveragePrice | 0.0139565 |
| 13 | MMRCurrentAuctionCleanPrice | 0.0137191 |
| 14 | MMRAcquisitionAuctionCleanPrice | 0.0134356 |
| 15 | VehBCost | 0.0131512 |
| 16 | VNZIP1 | 0.0125862 |
| 17 | AdjVehOd | 0.0125713 |
| 18 | PurchDate | 0.0118711 |
| 19 | MMRCurrentRetailCleanPrice | 0.0116824 |
| 20 | MMRCurrentRetailAveragePrice | 0.0116251 |
| 21 | WarrantyCost | 0.0098552 |
| 22 | MMRAcquisitionRetailAveragePrice | 0.0093032 |
| 23 | MMRAcquisitonRetailCleanPrice | 0.0088381 |
| 24 | BYRNO | 0.0071587 |
| 25 | Trim | 0.0071261 |
| 26 | VehOdo | 0.0056366 |

After examining the results and using our domain knowledge we decided that we would use the 7 most relevant attributes to build our model which were the folding:

*WheelType, VehicleAge, AdjVehOd, DifAcquiredAvg\_CurAvgAuction, DifAcquiredAboveAvg\_CurAboveAvgAuction, DifAcquiredAvg\_CurAvgRetail, DifAcquiredAboveAvg-CurAboveAvgRetial.*

**4. Model**

As we began the modeling phase of our project, we were focused on building a model that returned class membership probability. In our case this meant the probability that the car would be either a good or bad buy. Because our problem was strictly oriented towards having the mode accurate probability estimates, we were able to narrow our search for models. Logistic Regression has long been considered “the most common procedure” for estimating class probability so that seemed to be a natural starting point (Provost 2012).

Although logistic regression is the most popular in terms of class membership probability estimate, it is by no means the only option and definitely not the best universal learner. The No-Free-Lunch Theorem clearly states that “any learning algorithm has a limited scope of phenomena that it can capture, or an inherent inductive bias, and there can be no universal learner” (Ben-David, et al, 2011). For this reason we also chose to ensemble our Logistic Regression model with a Naïve Bayes model to improve the overall predictive ability of our model.

We however decided not to use a decision tree models since class membership probability is often not as accurate with unbalanced data, sets such as ours, even with smoothing methods such as Laplace. Additionally, probability from decision trees often is skewed towards 0 and 1 (Chawla and Cieslak). For this reason we chose not to include in an ensemble model.

Another model that we considered was a Support-Vector Machine. SVMs are very powerful and although not quite suited for probability estimation due to their loss functions, there have been many academics papers, which discuss calibrating the results to produce sufficient probability estimates (Drish). Nevertheless, SVMs are very computationally demanding and require more memory in training and use than we are able to support. For this unfortunate reason we could not use it.

Before we begin analyzing our model-building phase, it is useful to note that our data and model was being tested against a GINI score, that was used in the Kaggle competition as the scoring metric[[3]](#footnote-3). For this purpose, when evaluating our models, we used the GINI score and AUC metric to chose our model.

*4.1 Logistic Regression Model*

For our model-building phase we used R exclusively. For building our logistic regression model (“Log Model”) we used the *glm* package[[4]](#footnote-4) since it is the most robust and popular in the R ecosystem. We first ran a simple logistic regression on the target variable *IsBadBuy* with all 7 of our selected attributes.

Figure 4.1 (full output in Appendix B)

**GINI Score: 0.21503**

**AUC: 0.7279**

|  |  |  |
| --- | --- | --- |
| **Logistic Regression Model 1** | | |
| **Coefficients** | **Estimate** | **Std. Error** |
| (Intercept) | -2.08E+00 | 4.73E-02 |
| VehicleAge | 1.67E-01 | 5.98E-03 |
| WheelTypeCovers | -3.83E-02 | 1.42E-02 |
| WheelTypeNULL | 1.85E+00 | 2.58E-02 |
| WheelTypeSpecial | 8.18E-02 | 5.98E-02 |
| DifAquiredAvg\_CurAvgAuction | -3.41E-04 | 1.21E-04 |
| DifAquiredAboveAvg\_CurAboveAvgRetail | 6.05E-05 | 7.15E-05 |
| DifAquiredAvg\_CurAvgRetail | 1.49E-04 | 7.53E-05 |
| DifAquiredAboveAvg\_CurAboveAvgAuction | 1.15E-04 | 1.10E-04 |
| AdjVehOd | 3.19E-06 | 1.13E-06 |

After building the first logistic regression model we then decided to start optimizing the parameters and testing which model produced the highest GINI score. Below is a graph that shows the GINI score of altered models, ultimately resulting in our original modeling testing the highest.

Figure 4.2

From the above chart, our original feature selection process seemed to prove successful since, the original model recorded the highest scores in GINI and AUC. It should be noted that our original model ranked 300 out of 571 models submitted on Kaggle[[5]](#footnote-5). While these initial results seemed promising, we decided we could do better and went forward with building a Naïve Bayes probability estimator.

*4.2 Naïve Bayes*

While Naïve Bayes models are known for making the often-erroneous assumption that all attributes are conditionally independent of each other which often skews probability estimates towards one and zero, we decided to use its value nonetheless since it could possible correct some of the biases that a Logistic Regression model often makes. Additionally, the naïve bayes model helps correct a flaw in logistic regression, which is the fact that it cannot output a probability if data is missing. A naïve bayes model does not suffer this same problem and thus can correct instance where a piece of information is missing. With this in mind, we constructed a naïve bayes model separately from the logistic regression model before stacking them. Do create the naïve bayes model, we used the e1071 package in R.

The first model was built using all seven attributes and achieved a GINI score of 0.21774, actually higher than our original logistic regression model, which was surprising. We then tested one more naïve bayes model using Laplace correction, however there was no change since our data set was so large. Given our success of both models, we decided to average their probability estimates in an attempt to mitigate each of the model’s biases.

*4.3 Logistic Regression and Naïve Bayes Stacked Model*

After taking the probability estimates and averaging the two equally, we were able to harness both of the model’s predicting power while simultaneously reducing their biases. The following graph shows the GINI scores of weighting the Naïve Bayes and Logistic Regression Model’s predictions differently:

Figure 4.3

Given the above finding, we decided the optimal weighting would be 40% logistic regression and 60% naïve bayes since we got the highest Gini score of 0.22389.

*4.4 Model Conclusion*

Taking a step back, it is important to ask the question, “how does this model help the firm?” It is crucial to always have the end goal in mind of solving your original problem, which is predicting whether a car will be a good buy in the future or a bad buy in the future. The above stacked model helps a buyer at an auction by outputting a probability that the said car will in fact either be good or bad in the future. By doing this, we do not solve the exact problem, but we aid the buyer in his decision and give him an anchoring point.

It is useful to think about this in two examples. Lets say that a buyer is new to the job. He is looking at a used Ford Fusion that is 3 years old and has such and such attributes. He is fairly certain that there is not a high risk that this car is bad, and he in fact is thinking about placing a bid. He then puts the info of the car into our model and sees that in fact there is 73% chance that the car is in fact worthless. With this new piece of information, his thoughts on bidding have changed and we have perhaps saved him from losing on average $6000 from buying a bad car. Additionally, the model could be used as a conformation tool, when a buyer thinks that the car is either bad or good and can thus confirm it with our model, making him more confident in his decision. While our model cannot claim to *solve* the problem at hand (nor do I believe anyone can), we can confidently say that our model can aid the buyer in his decision, which is exactly what is being asked of us.

**5. Evaluation:**

Evaluating a model is a key step in any data mining process. While we have constantly been evaluating our model throughout each process, this section will focus on previous evaluation techniques and choices as well as how to evaluate the model in the future as it is deployed.

*5.1 Pre-Deployment Evaluation*

Evaluating different models and parameter settings is key to selecting the right model. We have constantly been doing that throughout the process, however this section will go through why we chose to use the Gini score as our main metric for evaluation.

Evaluating a model is different in every scenario and is dependent upon not only your data, but the model’s objective. Accuracy, true positive rate, precession and the like are almost always concerned with a binary outcome of either a correct or false classification. However, the model we have built and the underlying logistic regression and naïve bayes models output not a pure classification, but the probability of membership to a class. The reason that this is relevant is that all the mentioned evaluation metrics are subject to a cutoff point, meaning we could manipulate the metrics within a range by changing what the necessary probability needs to be to be classified as either good or bad. These metrics are irrelevant however since we are not concerned with predicting whether it is good or bad, but simply how good or how bad the car might be, so we cannot use these traditional evaluation metrics.

Instead we needed to use a metric that takes into consideration the moving thresholds, such as AUC or Gini index. Both of these metrics are closely related to each other as shown below.

Figure 5.1



It should also be noted that the Kaggle competition used a Gini index score, which influenced our evaluation metric since we wanted to compare our model. We could have used AUC, with no difference in results. Our final Gini score placed us in the top 47% of competitors and only 0.04 away from top place.

*5.2 Post-Deployment Evaluation*

Evaluating a model after it is in use is a constant process. While metrics and numbers can be used in the building phase, it can be much harder to use metrics after its deployment depending upon the nature of the model. Because our model outputs a probability estimate, it is hard to measure the accuracy of it as we noted before without setting a threshold, which is impractical in reality. Therefore it is hard to predict an expected improvement by using the model exactly, however we have proposed two alternative methods.

One way to evaluate how effective the model is in the use phase is by taking a look at the baseline rate of buying unsellable cars, which is 12.3%. A business case could be developed by taking a look after using the assisting model for one year (in order to have a large enough sample size) and comparing the baseline rate of unsellable cars to the rate of unsellable cars bought at the auction. While this is not a perfect evaluation metric, it could identify if there is any improvement by buyers in identifying which cars are unsuitable. This metric however could be skewed by that particular time period, selling and buying habits, and perhaps the buyers themselves. Nevertheless it would be more useful than not evaluating the model at all.

Another important metric for evaluating the success of the model is often overlooked and much simpler than people realize. Buying cars at auctions is an art in itself, and many buyers have been doing it for their entire lives. A crucial evaluation metric is the buyer’s opinion of the model. By testing out our model, and receiving their input we could greatly improve it or perhaps realize how useless it is. While this metric is obviously more qualitative in nature as opposed to quantitative, it could still prove useful in understanding the effectiveness of the model.

**6. Deployment**

The deployment phase of the model is when it all comes together and can be seen as one entire product. That doesn’t mean the process is over, since it constantly needs to be evaluated and improved as some attributes become relevant other become irrelevant and new data presents itself. Nevertheless it is still an exciting time for us as data scientists.

*6.1 Use Phase*

As we have reiterated through out our analysis, our deployment phase will allow buyers, while they are at auctions, to receive a quick probability estimate that the car they are looking at is in fact worthless. Below we have included an example iPad GUI of what we believe the model would look like from the front-end user perspective.

Figure 6.1



The following is a mere representation of what I could possibly look like and is more useful as a visualization of the end product. High probabilities would be highlighted in red, while low probabilities would be highlighted in green. Additionally, there would be an import feature for retrieving the market price of the car so it would not have to be manually entered.

*6.2 Associated Risks*

While there are no glaring risks that our model presents since it is a guide to buyers and not an outright decision based model, it is nonetheless useful to identify some of the possible risks. One risk that Carvana might need to consider is how likely buyers are on relying solely on the model’s estimate. It should be well explained and understood by buyers that is it is an aid and worth consulting, but should not be used as a pure decision tool since the dynamics of used car auctions are so complex that a model cannot perfectly fit it. In order to mitigate this risk, a day of training with a demo session would need to take place so all buyers could understand how to use the application and what the limitations are.

Another risk that Carvana might want to consider is the complexity of the model itself. While we may be able to understand the underlying model, the end user most likely will not. While the front end should be designed so there is as little confusion as possible, buyers might also not use the application if they do not know what is happening behind the front end. This risk could be addressed through a possible information session, but most likely it will take time for them to trust the application and feel comfortable with an outside opinion while making their decisions.

1. http://www.manheim.com/help/mmr#mmraccess [↑](#footnote-ref-1)
2. Data leakage occurs when data that is unattainable at the time of prediction is used as an attribute in training the data. [↑](#footnote-ref-2)
3. https://www.kaggle.com/c/DontGetKicked [↑](#footnote-ref-3)
4. http://cran.r-project.org/web/packages/glmnet/index.html [↑](#footnote-ref-4)
5. https://www.kaggle.com/c/DontGetKicked/leaderboard [↑](#footnote-ref-5)