**2. Data Understanding**

To begin our analysis of this business problem, our team obtained a data set via the Kaggle competition site. The dependent target variable in the data obtained was *isBadBuy,* which is a binary variable categorizing data instances as a bad buy or not a bad buy. Each data instance was a car that was sold at an auction, access to which was provided through the historical training set. The data set was obtained in the form of a .csv file, making it convenient to read and analyze using tools like Microsoft excel, python and R. The entire set contained 72,983 instances, including 32 independent attributes. The data set did, however, contain missing values. The base rate for the target variable *isBadBuy* in the training set was (8976/72983) = 12.298%.

A list of each attribute included in the original data set and its description is included below, as obtained from Kaggle. Each data instance, or each car that was purchased at the auction, was given a unique reference id number. Apart from this there were several features related to the vehicle specifications, including the make, model, submodel, year, trim, color, transmission, nationality and size. Two variables attempted to disclose the demand level for the car, by indicating whether it was manufactured by on of the top three American manufacturers and whether it was a prime unit. Purchase date, acquisition type, and purchase zip code/state were the variables relating to information about the purchase itself. The data also included a unique identifier number for the buyer that purchased the vehicle. The variable called *AUCGUART* indicated the level of guarantee that the auction provided for the vehicle, ranging form a green light to a yellow light and a red light. For this feature, a green light indicated the car had a guaranteed value, yellow indicated potential for issues, and red indicated it was sold as is. The feature warranty cost indicated how much the customer paid for a warrantee when purchasing the vehicle. Lastly there were a series of numeric features indicating the acquisition cost of the vehicle at both the current time and the time of purchase in good or above good condition in the retail market.

In using this data set, it is important to note that each instance in the set provides information about a vehicle that the dealership actually ended up purchasing at the auction. Since it is historical data, the information is only available to us because the car was actually deemed promising enough for the dealership to have actually purchased it. Once it was purchased, details about the car itself were made available, and we got access to information about whether or not the car ended up being a lemon. Thus, there is to some extent an innate bias in our data set. For this reason it is extremely important that, as mentioned before, any data-mining model developed be used along with domain expertise in order for it to have any useful meaning. Once the dealership has made its preliminary decision about the quality of the car, the model can be used to further assist in assessing the risk inherent in the purchase.

**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. A lot of what we learned in class became immediately applicable as we began clean, factorize and analyze our data set. With this in mind, we began our the first step by removing instances in our data set that contained null values, which resulted in removing 3,488 cars out of the 70,000 given. Then we proceeded to factorize the variables into numerical values form character values. This meant turning the make “Mazda” into 10, for example and the make “Honda” into 5, since the scikit-learn has a difficult time with character values.

The next and perhaps most important step in the process, was 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:

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 and gain ratio of each variable using an R library called FSelector. We then ranked each variable from the highest information gain, to the lowest information gain and got the following chart:

|  |  |  |
| --- | --- | --- |
| **Feature** | **Gain Ratio Score** | **Info Gain Score** |
| VehYear | 0.00787 | 0.018113 |
| VehicleAge | 0.007294 | 0.020037 |
| VehBCost | 0.006925 | 0.012475 |
| MMRAcquisitionAuctionAveragePrice | 0.006593 | 0.013462 |
| MMRAcquisitionAuctionCleanPrice | 0.006542 | 0.013039 |
| MMRCurrentAuctionAveragePrice | 0.006455 | 0.013288 |
| MMRCurrentAuctionCleanPrice | 0.006297 | 0.012587 |
| AUCGUART | 0.006277 | 0.001811 |
| PRIMEUNIT | 0.006238 | 0.001792 |
| MMRCurrentRetailAveragePrice | 0.005222 | 0.009843 |
| MMRCurrentRetailCleanPrice | 0.004724 | 0.00916 |
| MMRAcquisitionRetailAveragePrice | 0.004651 | 0.008428 |
| MMRAcquisitonRetailCleanPrice | 0.004407 | 0.0082 |
| Model | 0.003969 | 0.03138 |
| SubModel | 0.003521 | 0.022779 |
| WarrantyCost | 0.003479 | 0.007718 |
| VehOdo | 0.003435 | 0.006185 |
| Trim | 0.002216 | 0.009885 |
| VNZIP1 | 0.001898 | 0.01173 |
| WheelTypeID | 0.001859 | 0.001996 |
| WheelType | 0.001859 | 0.001996 |
| Auction | 0.00171 | 0.0024 |
| BYRNO | 0.001671 | 0.008678 |
| Make | 0.001247 | 0.004265 |
| Size | 0.001001 | 0.002748 |
| TopThreeAmericanName | 0.000961 | 0.0022 |
| VNST | 0.000881 | 0.003431 |
| Transmission | 0.00051 | 0.000116 |
| IsOnlineSale | 0.000265 | 0.000044 |
| Color | 0.000163 | 0.000524 |
| Nationality | 0.000071 | 0.000057 |
| PurchDate | 1E-10 | -0.008304 |

The above table gives an idea of the most relevant variables when trying to predict 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 contain an excessive amount of unique values thus representing a case of overfitting.

Although it would have been beneficial to the outcome of our model in terms of the competition, from an academic perspective we had to remove *VehBCost*, since it represented a case of data leakage. The cost of the vehicle (price paid at the auction) is not known when looking to bid on a car, so it cannot be used in training our model since it will not be available to us when predicting our target variable. You only know the price paid for the vehicle once the bidding has ended. For this reason it could not be used.

There were also 4 other variables that represented data leakage that we had to remove, which were the current day prices of each car. For obvious reasons, we can only use the price of the car at acquisition day to gain any insight, since knowing the future is not possible when predicting a value at the auction. For this reason we had to remove those features as well.

The variable *PurchDate* was also not included in the model for intuitive purposes. The past date of purchase of the car has little, if not zero predictive ability when looking at future cars in an auction. The same is true for the variable *VehYear*, since it is the year the car was made and really is the same metric *VehAge*, which we could use since it will be available in the future for predictions.

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.

The first feature we created was an adjusted odometer that we labeled *VehOdoAdj*. We did this because we believed that a pure odometer for the car says little without taking into account how old the vehicle 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 a *VehOdoAdj* 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.

The *MMRAcquisitionAuctionAveragePrice*, *MMRAcquisitionAuctionCleanPrice*, *MMRAcquisitionRetailAveragePrice* and *MMRAcquisitonRetailCleanPrice* features all went through the same feature creation process as well. We adjusted them by the age of the vehicle in order to gain more insight into how these prices reflected against the age of the actual vehicle.

Show adjustments here:

After we created these new features, we ran another information gain and gain ratio test in R to see how these new features compared to the unadjusted variables we had before. Below is the result:

|  |  |  |
| --- | --- | --- |
| **Feature** | **Gain Ratio Score** | **Info Gain Score** |
| VehYear | 0.00787 | 0.018113 |
| AdjMMRAcquisitionAuctionAveragePrice | 0.007586 | 0.018731 |
| AdjMMRAcquisitionAuctionCleanPrice | 0.007356 | 0.019094 |
| VehicleAge | 0.007294 | 0.020037 |
| AdjMMRAcquisitionRetailAveragePrice | 0.007047 | 0.017363 |
| VehBCost | 0.006925 | 0.012475 |
| AdjMMRAcquisitonRetailCleanPrice | 0.006718 | 0.017638 |
| MMRAcquisitionAuctionAveragePrice | 0.006593 | 0.013462 |
| MMRAcquisitionAuctionCleanPrice | 0.006542 | 0.013039 |
| MMRCurrentAuctionAveragePrice | 0.006455 | 0.013288 |
| MMRCurrentAuctionCleanPrice | 0.006297 | 0.012587 |
| AUCGUART | 0.006277 | 0.001811 |
| PRIMEUNIT | 0.006238 | 0.001792 |
| MMRCurrentRetailAveragePrice | 0.005222 | 0.009843 |
| VehOdoAdj | 0.004961 | 0.012621 |
| MMRCurrentRetailCleanPrice | 0.004724 | 0.00916 |
| MMRAcquisitionRetailAveragePrice | 0.004651 | 0.008428 |
| MMRAcquisitonRetailCleanPrice | 0.004407 | 0.0082 |
| Model | 0.003969 | 0.03138 |
| SubModel | 0.003521 | 0.022779 |

Above you can see that our features that we created based on an adjusted for the vehicle’s age out performed their unadjusted counterparts. The *VehOdoAdj* doubled information gain as compared to *VehOdo* when it is not adjusted. After removing the all of the features that represented data leakage, poor information gain, we produced the follow 15 features that we would then use for our model:

VehicleAge, WheelTypeID, VehOdoAdj, WarrantyCost, VNZIP1, AUCGUART, PRIMEUNIT, AdjMMRAcquisitionAuctionAveragePrice, AdjMMRAcquisitionAuctionCleanPrice, AdjMMRAcquisitionRetailAveragePrice, AdjMMRAcquisitonRetailCleanPrice, MMRAcquisitonRetailCleanPrice, MMRAcquisitionRetailAveragePrice, MMRAcquisitionAuctionCleanPrice, MMRAcquisitionAuctionAveragePrice

**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 that were able to achieve this result. With that being said, we built 7 independent models in order to find the most optimal GINI score.

While building the different models there is often a sense that the model is a black box, so reading the documentation became a critical, although time consuming part of the process. In order to tweak each model it quickly became obvious that we could not achieve optimal results without understanding the functionality of each parameter. With that being said, we decided it would be useful to give a little insight in to each model that we didn’t cover in class.

*Random Forest*

The random forest model essentially allowed us to create a specified number of trees during the training of the model based of a random number of features that we had in our data set. By using a number of trees that are randomly generated, it creates a random “forest” of trees, hence the name. This model was critical because it is known to neutralize biases and variances in data sets like ours. When predicting the probability for a car, it actually creates as many predications as there are trees and then averages all of those classifications, divided by the total number of trees.

*Naïve Bayes*

The Naïve Bayes model is one of the most fundamentally simple models that is around in data mining since it based off of Bayes Theorem. Bayes theorem is statistical rule that is based around the idea of conditional probability that states: The probability of A given B occurs, is equal to the probability of B given that A occurs multiplied by the probability of A occurring and then divided by the probability of B occurring. One of the underlying assumptions that allows this theorem to hold true, is that the events A and B have to be independent. The reason the model is a “naïve” bayes, is that it makes the naïve assumption that all the features of the model are independent, which is often not true and is why its considered naïve. However naïve bayes are often some of the best probability estimators of al models, regardless of its naïve assumption of independence.

*Ada Boosting*

Ada Boosting is short for Adaptive Boosting of models. The reason this model is concerned adaptive in its learning, is that is builds multiple classifiers in the training phase through iterations. After each iteration, it assigns a weight to each example in the training set based on whether or not the classifier got it right or wrong. After each round, the weights of each incorrectly classified example are increased, and the weights of each correctly classified example are decreased, so the new classifier focuses on the examples, which have so far eluded correct classification. While this model can be sensitive to noisy data ,it proved well with our data.

*Gradient Tree Boosting*

Gradient Tree boosting is another method of boosting, similar to ada boosting. While Ada boosting attempts to correct misclassification of examples through each of its iterations, gradient tree boosting attempts to correct its self based on a loss-function that is specified during training. The loss-function that we deicide to use in order to boost the trees was the least squares loss-function.

As you can see from the graph above, the Gradient Boosting Model had the highest individual GINI score, followed by the Random Forest, Naïve Bayes and Ada Boosting. Surprisingly, the Logistic Regression model did much worse than we had expected since logistic regressions are often an optimal strategy when computing class probabilities. Nevertheless, we were not satisfied with just having a “good” probability classifier so we went ahead and attempted what is called blending or stacking our models.

*Blended Model*

Blending models is a common process in data mining since every model has its own biases and flaws in how it computes the probability. While some models may be better than other, there is definitely no universal model that is by far the best. With that concept in mind, we decided to blend these models together in order to neutralize biases in each model and see if we could actually achieve a higher score with multiple models rather than a single one.

We achieved this step by taking the probabilities from each model and weighting them evenly to come to a final probability for each car. While blending was a relatively easy step in terms of time and code, we found that the process of finding the optimal weights for each model was much more complicated. In the end we evenly weighted the predictions between the Random Forest, Naïve Bayes, K-NN, Random Forest, Ada Boosting and Gradient Tree Boosting model to come to our final model that had the highest GINI score results.