CMPE 239 Project Report

Kaggle competition - Don't Get Kicked!

Predict if a car purchased at an auction is a bad buy

Submitted by:

Haritha Peyyeti [010061519]

Yi Chou [005954273]

Submitted to:

Dr. Magdalini Eirinaki

Date of Submission: 30/11/2015

Table of Contents

Chapter 1 Introduction

1.1 Motivation

One of the biggest challenges faced by car dealers in purchasing used cars at an auction is the risk of buying cars with serious issues that prevent them from being sold to customers. These bad cars are called as "kicks". Kicked cars are often due to various problems such as tampered odometers, mechanical issues that could not be a priori, or some other unforeseen troubles. Buying kicked costs a lot to dealers after transportation cost, throw-away repair work, and market losses in reselling the vehicle. A model that can identify which cars have a higher risk of being a kick provides a real value to dealers in reducing their losses and also providing their customers with the best inventory selection possible.

1.2 Objective

The goal of this project is to develop a prediction model that dealerships can utilize when buying used cars at auctions. This model would enable dealerships to minimize the risk of buying kicked cars by predicting whether a car is kicked or not based on some features.

Chapter 2 System Design

We plan to apply various classification algorithms to predict if the car purchased at an auction is a good or bad buy. The goal is to compare and learn how each of these algorithms perform individually as well as ensemble.

2.1 Algorithms

2.1.1 Logistic Regression

The outcome of our problem is reconstructed to estimating the probability of the car to be bad given the independent variables. Replacing the outcome variable of the logistic function with a linear combination of dependent variables we intend to use for regression, we arrive at the formula for logistic regression. This method is chosen because it gives the car dealers the flexibility of classifying a car as good or bad based on his risk tendency. This algorithm helps in choosing predictors that are statistically significant in estimating the outcome of our purchase.

2.1.2 Logitboost

Logitboost is an additive Logistic regression that employs boosting approach when building a logit model. Here, we use regression method as weak classifier. The advantage of using this method is that it is easier to interpret since it is in the form of an equation.

2.1.3 Support Vector machines

From our literature research, we found out that Support vector machines is most appropriate for high dimensional data and when the predictor variables are non-linearly separable. This makes us chose this algorithm for our analysis.

2.1.4 Naïve Bayes

Naive Bayes algorithm is a family of simple probabilistic classifiers by applying Bayes Theorem, conditional probability and Maximum-likelihood training, which requires a strong independence assumption between the explanatory variables.

After the process of feature selection, transformation, and combination of similar features, it is reasonable to make independence assumptions to the features of our cleaned data set. In addition, Naive Bayes is a very fast learning, testing, and low storage requirements algorithm. Therefore, we choose this as one of the algorithms to be evaluated.

2.1.5 Classification and Regression Tree (CART)

CART is one of the nonparametric decision tree learning technique based on the idea of recursive partitioning that produces classification or regression trees. For categorical dependent variables, it produces classification trees, for numerical dependent variables, it produces regression trees.
Since the data set evaluated contains both categorical and numerical variables, CART is a good choice. In addition, CART is not sensitive to outliers in predictive modeling, which makes it a good alternative to generalized linear models. However, CART often tends to overfit the data. We will prune the trees by tuning some parameters in order to prevent overfitting problems.

2.1.6. Gradient Boosting

Gradient boosting machine is a flexible non-parametric technique for classification. This procedure is very useful for heterogeneous data. Also, The size of each tree can be controlled either by changing the *tree depth.* Over fitting can also be minimized by changing the *shrinkage* parameter.

2.1.7. Adaptive Boosting (AdaBoost)

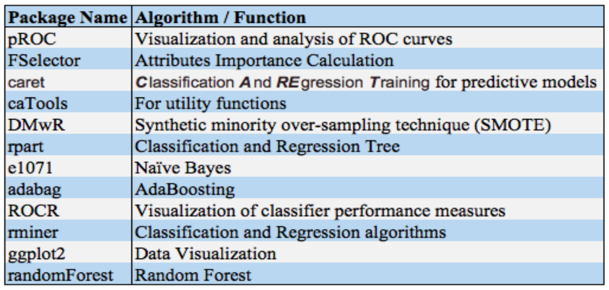
Adaboost uses training set re-weighting method instead of resampling method. Adaboost constructs a strong classifier as a linear combination of weak classifier, and the final classification decision will be based on the weighted vote of the weak classifiers.
Adaboost algorithm is sensitive to noisy data (outliers). Since our cleaned data set is normalized, the outliers are taken care of. Also due to the fairly good generalization of result, we choose Adaboost as one of the algorithms to be evaluated.

2.1.8. Random Forest

Random Forests is an ensemble learning method for classification, regression and other tasks. It generally often used to help correct the overfitting problem of decision trees. It does splitting decision with randomly selected variables on bootstrap sample from the training set. We are interested in comparing the result of Random Forest algorithm and the CART pruned tree on this data set. Random Forests are more robust to overfitting.

2.2 Technologies Used

We use different packages in R for data manipulation, visualization, and analysis.

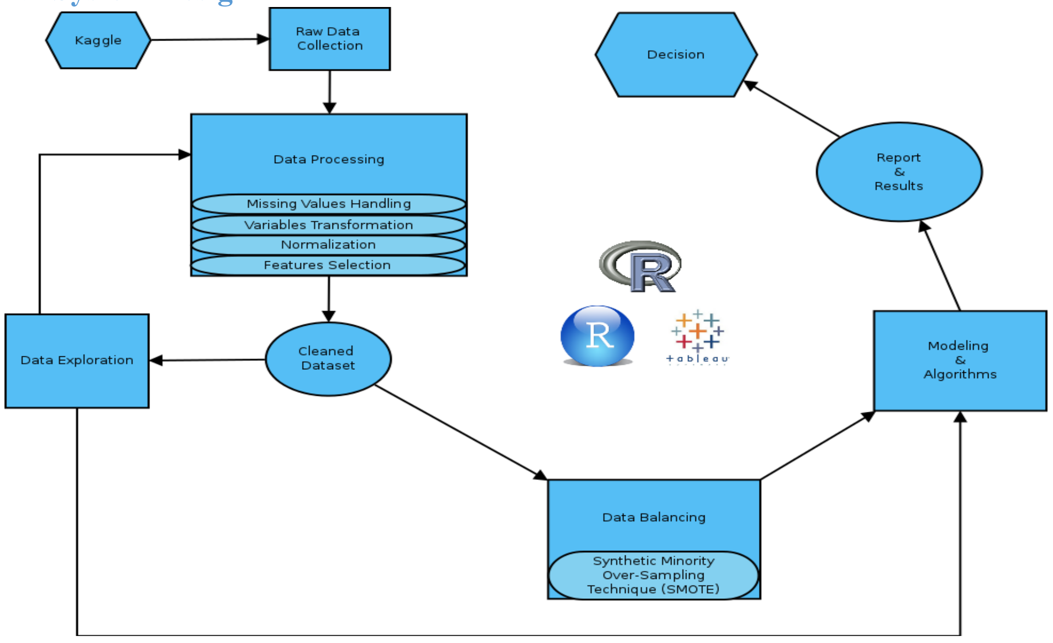


Package Name	Algorithm / Function
pROC	Visualization and analysis of ROC curves
FSelector	Attributes Importance Calculation
caret	***C***lassification ***A***nd ***RE***gression ***T***raining for predictive models
caTools	For utility functions
DMwR	Synthetic minority over-sampling technique (SMOTE)
rpart	Classification and Regression Tree
e1071	Naïve Bayes
adabag	AdaBoosting
ROCR	Visualization of classifier performance measures
rminer	Classification and Regression algorithms
ggplot2	Data Visualization
randomForest	Random Forest

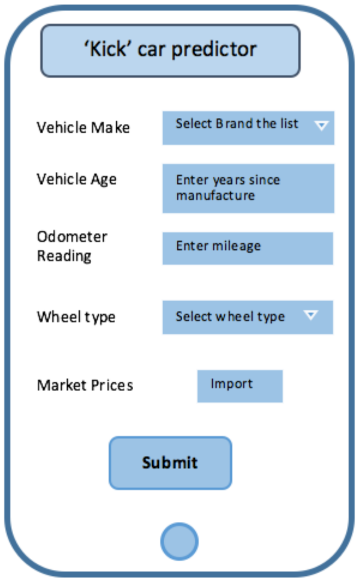
2.3 Architecture related decisions

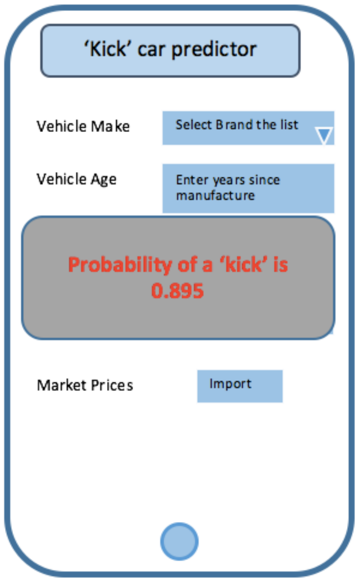
We chose R as the primary programming language for this project because of the abundance of packages that suits our project needs and the ease of data visualization and exploration. Also, R is the best choice for analyzing the statistical aspects of our data. After our initial model building efforts in R that failed due to the memory issues on our local systems, we created an Amazon EC2 micro instance on the server. When the memory issues were the same on the EC2 instance too, we realized that it was the large factor levels (around 1100 classes) in some of the independent variables that is causing this trouble. We handled this problem by reducing the dimensionality of our variables, thus able to run it locally on our system.

2.4 System Design



2.5 Use Case





The deployment of this project will allow car dealers to predict if the car being auctioned is in fact worthless. To demonstrate how our predictive model would be deployed, we created a visualization from the user-end perspective. It is only a representation of how this model can be used. Since, our model returns the probability, it gives flexibility to the dealer to suit their risk attitude.

Chapter 3 Implementation Details

3.1 Data Description

We obtained our data set from the Kaggle competition "Don't Get Kicked" hosted by Carvana. The training data set contained 32 features with 72,983 samples with a labeling of 0 for good car purchases and 1 for "kicks. The predictors included information about the specifications of the vehicle, information specific to the auction, and information about price/cost measures specific to the purchase as well as from market research (refer to appendix for the list of features and their descriptions.) Just as in any other fraud detection cases where positive examples are few in number, the number of bad cars were under-represented in the data set, representing 12.3% of the total sample size.

3.2 Data Preprocessing

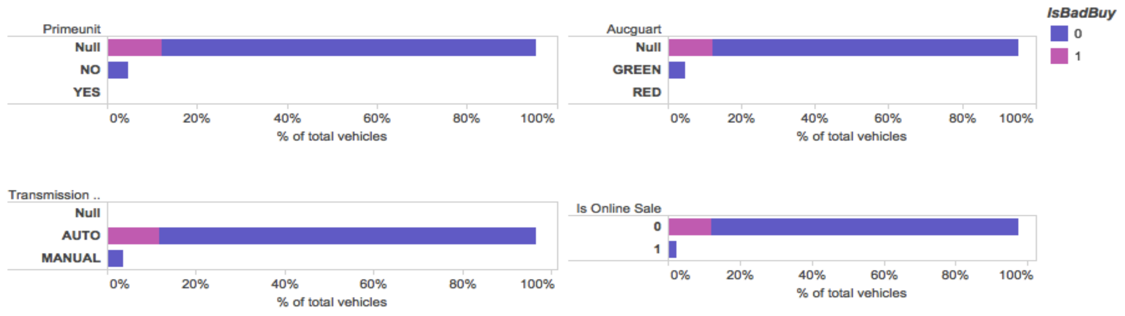
The dataset contained redundant variables, poor quality variables, and NULL fields for several features. The data was pre-processed and cleaned before starting the analysis. For both the training and test datasets, we performed following transformations:

a) **Missing value treatment**: Missing values for numeric variables are replaced with the median value of all the samples, while a separate category called 'NA' is created for categorical variables.

b) **Redundant variables**: *VehAge* which represents years elapsed since the year of manufacture provide gives the same information as the difference between *PurchDate* the date of purchase at auction and *VehYear* the year of manufacture. Since one variable is a linear combination of the other two variables, we retained VehAge and removed the other two variables. Also, *WheelTypeID* gives the id of *WheelType* that describes the wheel type (alloy or covers). We filtered out WheelTypeID.

c) **Poor quality variables**: *Primeunit,* a binary variable that identifies if a vehicle has higher than standard demand has missing values in 95% of instances. Hence this feature is dropped.

d) **Near Zero variance variables**: Near zero variance diagnoses for predictors that have very few unique values relative to the number of samples and the ratio of the most common value to the second most common value is large. Such variables add little or no value to the model and can be dropped. *Transmisssion, IsOnlineSale, AUCGUART* have a unique value for more than 95% of samples and are hence dropped.

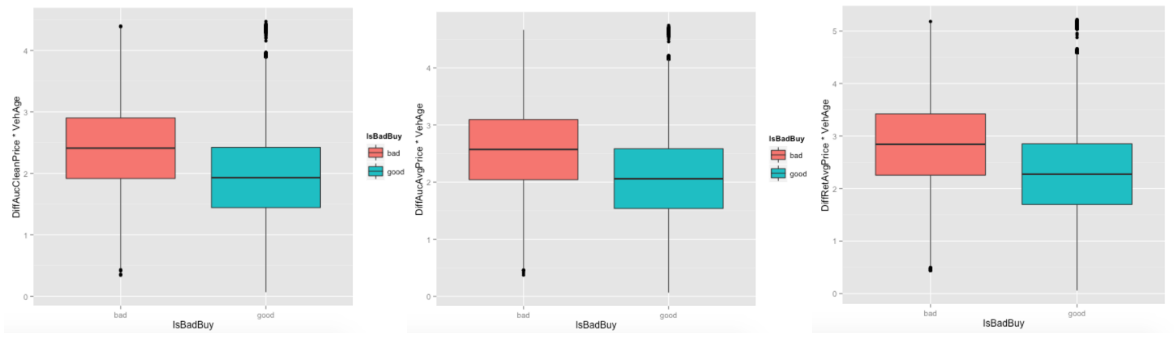


e) **Transforming variables**: Various market prices that begin with *MMR* take into account the *Make, Model, Sub-model* of the vehicle. We realized that these absolute market prices have little predictive ability, and hence created new features that represent the relative prices between the retail market price and auction price adjusted by the age of the vehicle.

f) **Using String distance to handle large text factors**: There are 1130 distinct vehicle models in the dataset. We used Jaro-Winkler distance from strndist package in R that measures the distance between two text strings for approximate string matching. Though it is easier to deal with lesser levels, the downside of this process is that the new clusters created are difficult to interpret and not all may be meaningful.

g) **Data Balancing**: In the original data set, only 12.3% of the instances are bad cars, which implies that the prevalence of the classes is far from being equal. From our literature research on the techniques used for balancing the data, we realized that down-sampling the majority class, or up-sampling the minority class (with replacement) or hybrid methods are the possible alternatives available. We performed Synthetic Minority Over-Sampling Technique (SMOTE) from DMwR package in R which down-samples the majority class and synthesizes new data points in the minority class. We over-sampled 5 times more extra bad instances and under-sampled 1.2 times less good instances (0.6 times of good instances after oversampling process). Our resulting balanced data now includes equal proportion of good and bad cars. We chose this method because this is claimed to have better classifier performance compared to the other methods.

3.3 Data Visualization

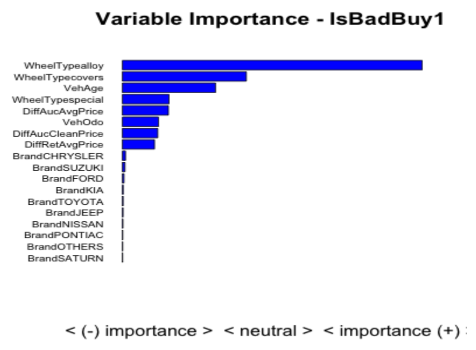


According to the box plots, we can see that the difference of auction / retail price for bad cars are generally higher than for good cars. These imply that for bad cars, either the retail price is higher, or the auction price is lower, which is reasonable. This shows that it is a good indicator for predicting bad cars.

Chapter 4 Experiments

4.1. Methodology

4.1.1 Feature Selection



The features used for the predictive models are selected using Chi-sq., Information gain and step-wise logistic regression. We did not use the attributes specific to the purchase such as *VehBcost (*price paid for the vehicle), *BYRNO* (buyer ID) that are unattainable at the time of prediction.

4.1.2 Model building

We trained the classifiers using both unbalanced and balanced datasets to compare their performance for each of the algorithms. We used the training set to build the model using 5 fold cross-validation and use the best model after adjustments to make predictions for test set data.

Unbalanced data: We split the data randomly where the outcome of the car is known to training and test datasets in the ratio of 70:30 ensuring that the prevalence the response variable is equal in both the datasets. The prevalence of bad cars is around 12% in both the datasets. Size of Training set: 51,088 samples; Size of Test set: 21,895 samples

Balanced data: We first balance the training data using the technique presented in the previous section which results in a dataset with equal prevalence of good and bad cars. The prevalence of bad cars is now around 50% in the dataset used for training the models. Size of Training set: 107,712; Size of Test set: 21,895

For Classification and Regression Tree (**CART**), we first grow the full classification tree, then prune the tree by tuning the complexity parameter (CP) value. Then estimate the prediction and area under curve for both full and pruned trees. For **Naive Bayes**, we perform both the original algorithm (without tuning parameter) and the one with Laplace Smoothing Correction. For **Random Forests**, after doing the implementation with and without tuning the number of predictors selected, we compute variable importance.

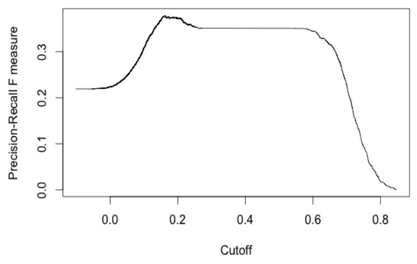
4.1.3 Performance Evaluation Measures

As mentioned in the earlier sections, due to the underrepresentation of bad cars in our data, it is more appropriate to look at precision and recall here rather than only accuracy. The car dealers would be interested more interested in reducing the false negatives (FN) which directly relates to the losses incurred than the false positives (FP) which relates to the opportunity cost.

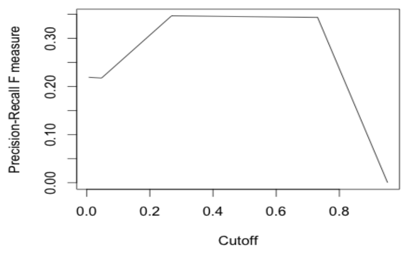
$$Precision = \frac{TP}{TP+FP} \qquad Recall = \frac{TP}{TP+FN}$$

In practice, since there is a trade-off between precision and recall, we included F-measure as one of the performance metrics of our classifiers. Also, through our literature study we found out that Area Under the Curve (AUC) is a good metric because it takes into account sensitivity (recall) and specificity.

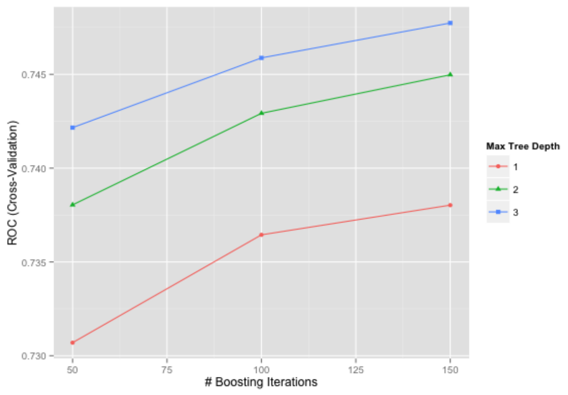
4.2 Graphs / Plots for Different Parameters / Algorithms Evaluation



For Logistic regression, the plot suggests that the F-measure is maximum when the threshold for a car to be classified as a 'kick' is set around 20%.

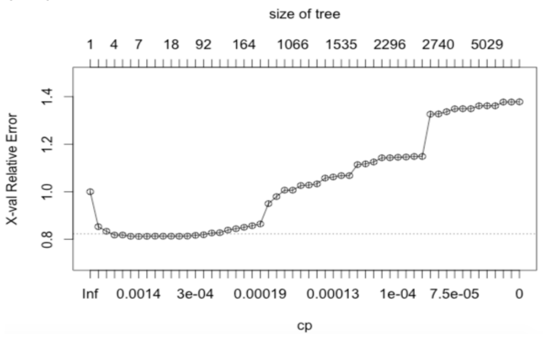


For Logitboost, the plot suggests that the F-measure is maximum when the threshold for a car to be classified as a 'kick' is anywhere between 30 - 70%, depending on the risk-intake attitude of the dealers.

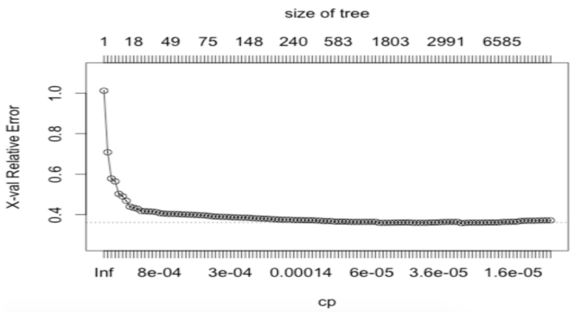


Generalized Boosting is an ensemble of weak prediction models. The ROC is higher for more number of boosting iterations and for the tree depth of 3. As we grow the tree larger than 3 nodes, the ROC is found to be decreasing.

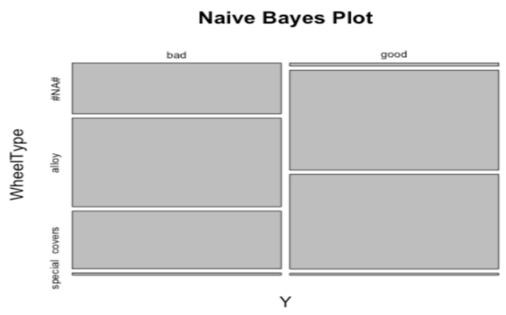
CART: The plots of complexity parameter (CP) value vs. size of tree vs. X-value relative error are generated. We pick the CP value that corresponds to smallest relative error and with smallest size



Unbalanced Data: The best CP value among is 0.0003979, providing number of splits = 30 with relative error = 0.8128.

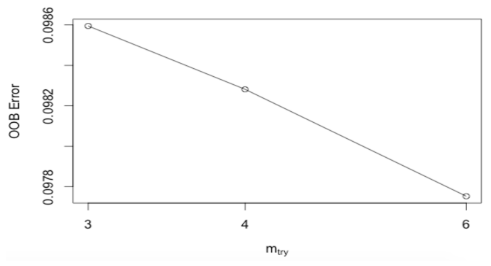


Balanced Data: Although the error keeps decreasing, in order to prune the tree without overfitting (with smallest trees (that is, number of splits) possible), we pick the "elbow" where the CP value is 0.003634, with number of splits = 16 with relative error 0.4391.

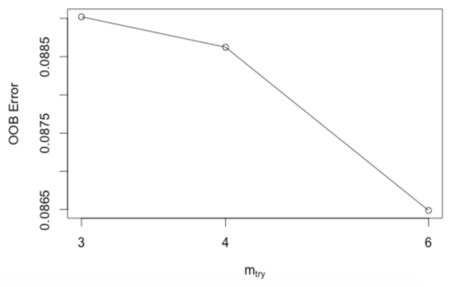


The Mosaic plot of Naive Bayes Algorithm. For the variable wheel type to the outcome variable. Due to many observation has a numerical probability close to zero for all classes (warning message given by R), we add smoothing term Laplace = 3 to add 3 counts to each cells.

Random Forest: We tune the number of variables selected (mtry) for optimal result, by implementing tuneRF function, the following plots show mtry to the Out-of-Bag (OOB) error. For both unbalanced and balanced data, the tuning suggestion based on OOB are similar. Using OOB to estimate the mtry to optimize result has a possibility of bias.

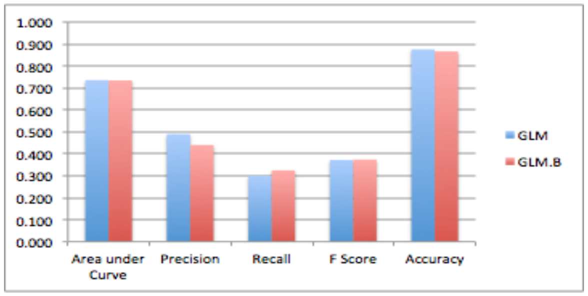


Unbalanced data

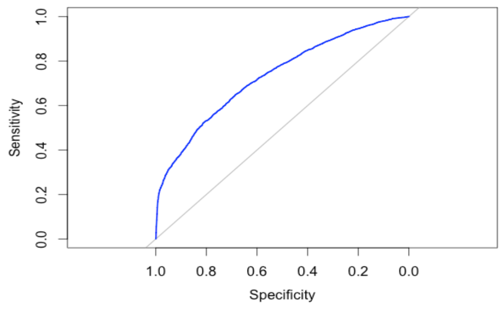


Balanced data

4.3 Data Analysis

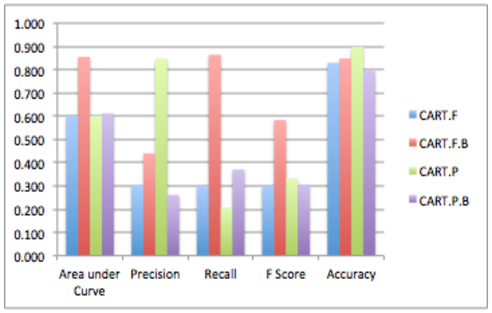


By comparing the F Score or AUC of logistic regression with balanced and unbalanced data, we notice that balancing the data did not impact the model performance. Hence we decide to choose the model with unbalanced data to compare with other models.

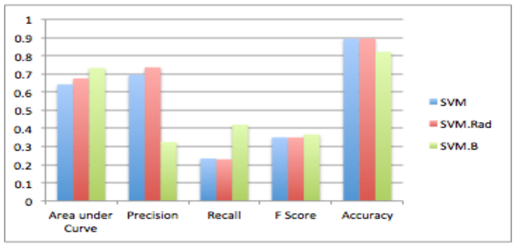


For unbalanced logitboost model, area under curve is 0.734, which is high compared to other models. Accuracy increased with increase in boosting iterations, having an accuracy of 0.89 for 30 boosting iterations. By comparing the F Score of Logiboost with balanced and unbalanced data, we decide to pick the model with unbalanced data as they are nearly the same.

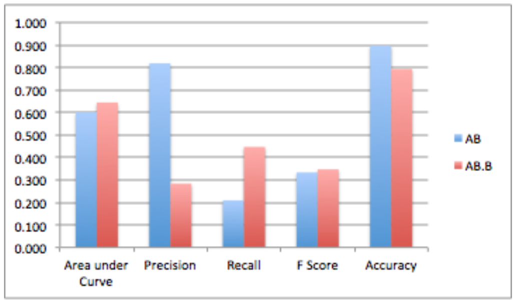
For the full grown CART model which based on the balanced data, the area under curve is 0.8549, which is very high when compare with other models. Due to the nature of overfitting problem that decision trees usually make, we consider that it might be the result of overfitting. For unbalanced data, the area under curve for full grown tree is 0.6016, and 0.6008 for pruned tree. For pruned tree with balanced data set, the area under curve is 0.6117.



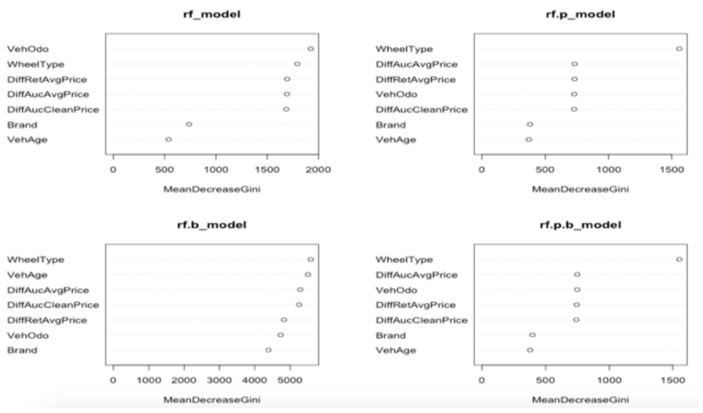
By comparing the F Score of CART with balanced and unbalanced data, we decide to pick the model of full tree with balanced data. But since there might be a problem of overfitting for full grown tree, we also pick the model of pruned tree with unbalanced data, which has second highest F score among the CART models.



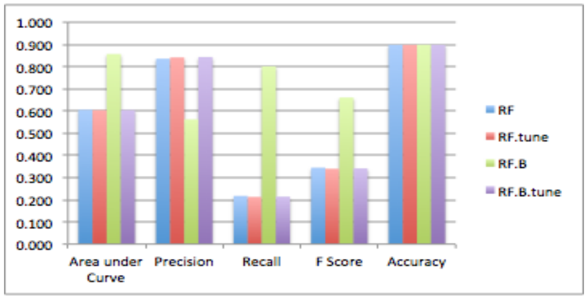
By comparing the F Score of SVM linear and SVM Radial with balanced and unbalanced data, we decide to pick the original SVM model with balanced data.



For unbalanced data, the area under curve for Adaboost is 0.6015. The area under curve for the one with balanced data is 0.6445, which is slightly higher compared to the ones built from other algorithms.
By comparing the F Score of Adaboost model with balanced and unbalanced data, we decide to pick the one with balanced data.

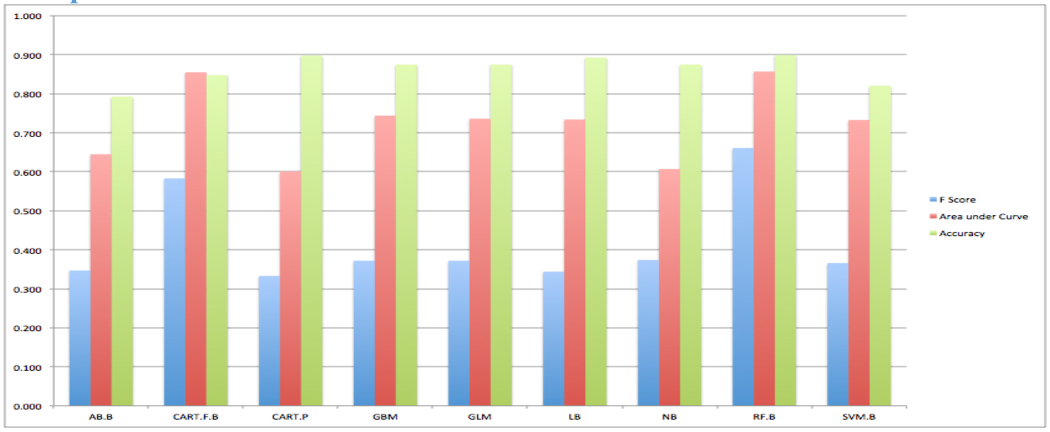


Random Forest: Three plots out of four suggest that WheelType is the most important variables, and one suggests that VehOdo is the most important one.
The performance of un-tuned Random Forest is very good, that its area under curve is 0.8569 and the F Score is 0.661, the highest among the models chosen. Due to the reason that Random Forests are more robust to overfitting, we consider this model performs well without the problem of overfitting.



RF: With tuned number of predictors selected (mtry) based on Out-of-Bag error, the area under curve is 0.6036 with unbalanced data, and 0.6042 with balanced data. We can see that tuning the parameter based on OOB does not help improving the model.

Comparison across models:



Based on Area Under Curve and F-measure, we choose Random Forest as the best model across all models, followed by Support Vector Machine.

Chapter 5 Discussion & Conclusion

5.1 Decisions Made

We decided not to use purchase specific features in predicting the outcome as we want to generalize the use case of our project to dealers who want to predict if the car is a lemon rather than identifying it retrospectively.

5.2 Things that worked well

- We decided to use vehicle brand instead of the clusters we created from vehicle models. This helped us reduce the computation time and also made it possible to run our implement the model on our local system instead of using the Amazon EC2 server.
- Based on our literature research, we decided to balance our data. On comparing, we realized that balancing the outcome classes in the training data does not necessarily lead to an improved classifier performance.

5.3 Difficulties Faced

Computational Difficulties: Repeated cross validations for k-folds algorithm for Support Vector Machine, AdaBoost, and Random Forest are computationally expensive, which took long time for computing.

5.4 Things that didn’t work well

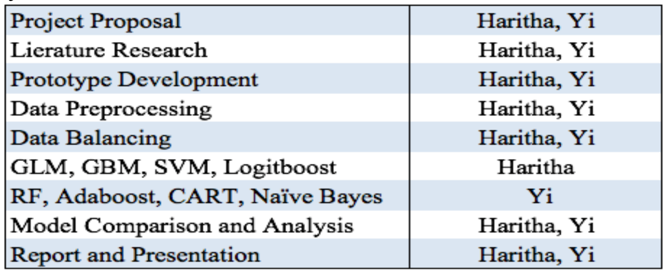
Naive Bayes Algorithm: Repeated cross validations for k-folds algorithm for Naive Bayes algorithm (caret package) generates the observation has a numerical probability close to zero for all classes for over 20 thousand of observations. Therefore, the result contains many missing / undefined classifications. We apply Naive Bayes algorithm by other implementation (the naiveBayes function from e1071 package). However, although no warning message occurred, the conditional probability of zero result problem is still occurred. The same problem occurred even with the tuned Laplace parameter.

5.4 Conclusion

We were successful in implementing various classification algorithms including ensemble methods. Random Forests and Support Vector Machine were better than the remaining models. Wheel type and Odometer reading or mileage are the most important variables to check when purchasing used cars.

Chapter 6 Task Distribution

Though we implemented four algorithms each, all the critical stages in the project were handled by both of us.



Project Proposal	Haritha, Yi
Lierature Research	Haritha, Yi
Prototype Development	Haritha, Yi
Data Preprocessing	Haritha, Yi
Data Balancing	Haritha, Yi
GLM, GBM, SVM, Logitboost	Haritha
RF, Adaboost, CART, Naïve Bayes	Yi
Model Comparison and Analysis	Haritha, Yi
Report and Presentation	Haritha, Yi

Appendix I. Data Features (From Kaggle Website)

Field Name	Definition
RefID	Unique (sequential) number assigned to vehicles
IsBadBuy	Identifies if the kicked vehicle was an avoidable purchase
PurchDate	The Date the vehicle was Purchased at Auction
Auction	Auction provider at which the vehicle was purchased
VehYear	The manufacturer's year of the vehicle
VehicleAge	The Years elapsed since the manufacturer's year
Make	Vehicle Manufacturer
Model	Vehicle Model
Trim	Vehicle Trim Level
SubModel	Vehicle Submodel
Color	Vehicle Color
Transmission	Vehicles transmission type (Automatic, Manual)
WheelTypeID	The type id of the vehicle wheel
WheelType	The vehicle wheel type description (Alloy, Covers)
VehOdo	The vehicles odometer reading
Nationality	The Manufacturer's country
Size	The size category of the vehicle (Compact, SUV, etc.)
TopThreeAmericanName	Identifies if the manufacturer is one of the top three American manufacturers
MMRAcquisitionAuctionAveragePrice	Acquisition price for this vehicle in the above Average condition at time of purchase
MMRAcquisitionAuctionCleanPrice	Acquisition price for this vehicle in the above Average condition at time of purchase
MMRAcquisitionRetailAveragePrice	Acquisition price for this vehicle in the retail market in average condition at time of purchase
MMRAcquisitonRetailCleanPrice	Acquisition price for this vehicle in the retail market in above average condition at time of purchase
MMRCurrentAuctionAveragePrice	Acquisition price for this vehicle in average condition as of current day
MMRCurrentAuctionCleanPrice	Acquisition price for this vehicle in the above condition as of current day
MMRCurrentRetailAveragePrice	Acquisition price for this vehicle in the retail market in average condition as of current day
MMRCurrentRetailCleanPrice	Acquisition price for this vehicle in the retail market in above average condition as of current day
PRIMEUNIT	Identifies if the vehicle would have a higher demand than a standard purchase
AcquisitionType	Identifies how the vehicle was aquired (Auction buy, trade in, etc)
AUCGUART	The level guarntee provided by auction for the vehicle (Green light - Guaranteed/arbitratable, Yellow Light - caution/issue, red light - sold as is)
KickDate	Date the vehicle was kicked back to the auction
BYRNO	Unique number assigned to the buyer that purchased the vehicle
VNZIP	Zipcode where the car was purchased
VNST	State where the the car was purchased
VehBCost	Acquisition cost paid for the vehicle at time of purchase
IsOnlineSale	Identifies if the vehicle was originally purchased online
WarrantyCost	Warranty price (term=36month and millage=36K)

Appendix II. Variables Selection

The importance of features is ranked based on Information Gained (Entropy Based Filter) and the Chi-Squared Filter Methods, in the following tables:

Information Gained	
variable	**attr_importance**
WheelType	0.045204222
DiffAucAvgPrice	0.016175741
DiffAucCleanPrice	0.015420089
VehAge	0.014103892
DiffRetAvgPrice	0.013915313
VehYear	0.012857263
DiffRetCleanPrice	0.012505793
VehBCost	0.009571243
WarrantyCost	0.00466398
VehOdo	0.003903305
Brand	0.002886538
Size	0.002435935
RefId	0.002124252
PurchMY	0.001771126
TopThreeAmericanName	0.001520531
PurchMonth	0.001079427
Auction	0.000984706
Qtr	0.000407788
PurchYear	0.000311745
ColorNew	0.000197631
Nationality	0.000132384

Chi-Squared Filter	
variable	**attr_importance**
WheelType	0.38429959
DiffAucAvgPrice	0.18161128
DiffAucCleanPrice	0.17717884
VehAge	0.17344932
DiffRetAvgPrice	0.16870804
VehYear	0.16524054
DiffRetCleanPrice	0.1607572
VehBCost	0.15094882
WarrantyCost	0.10269095
VehOdo	0.08958764
Brand	0.07650924
Size	0.07026393
RefId	0.06382646
PurchMY	0.0592611
TopThreeAmericanName	0.05604212
PurchMonth	0.04628981
Auction	0.04526634
Qtr	0.02850156
PurchYear	0.0249433
ColorNew	0.01982279
Nationality	0.01622129

Appendix III. Confusion Matrices for Algorithms Applied

Classification and Regression Tree (CART)

CART - Unbalanced Full Tree		
	TRUE	
Prediction	Good	Bad
Good	17343	1885
Bad	1859	808

CART - Balanced Full Tree		
	TRUE	
Prediction	Good	Bad
Good	16239	366
Bad	2963	2327

CART - Unbalanced Pruned Tree		
	TRUE	
Prediction	Good	Bad
Good	19102	2136
Bad	100	557

CART - Balanced Pruned Tree		
	TRUE	
Prediction	Good	Bad
Good	16375	1695
Bad	2827	998

Adaptive Boosting (Adaboost)

Adaboost - Unbalanced		
	TRUE	
Prediction	Good	Bad
Good	19077	2129
Bad	125	564

Adaboost - Balanced		
	TRUE	
Prediction	Good	Bad
Good	16165	1489
Bad	3037	1204

Naive Bayes (NB)

Naïve Bayes - Unbalanced		
	TRUE	
Prediction	Good	Bad
Good	18337	1874
Bad	865	819

Naïve Bayes - Balanced		
	TRUE	
Prediction	Good	Bad
Good	6484	350
Bad	12718	2343

Random Forest (RF)

Random Forest - Unbal w/o Tuning		
	TRUE	
Prediction	Good	Bad
Good	19087	2108
Bad	115	585

Random Forest - Balanced w/o Tuning		
	TRUE	
Prediction	Good	Bad
Good	17521	535
Bad	1681	2158

Random Forest - Unbal w/ Tuning		
	TRUE	
Prediction	Good	Bad
Good	19094	2120
Bad	108	573

Random Forest - Balanced w/ Tuning		
	TRUE	
Prediction	Good	Bad
Good	19095	2117
Bad	107	576

Generalized Boosted Regression Models (GBM)

GBM - Unbalanced		
	TRUE	
Prediction	Good	Bad
Good	18361	1886
Bad	841	807

GBM - Balanced		
	TRUE	
Prediction	Good	Bad
Good	18361	1886
Bad	841	807

Generalized Linear Models (Logistic Regression)

GLM - Unbalanced		
	TRUE	
Prediction	Good	Bad
Good	18361	1886
Bad	841	807

GLM - Balanced		
	TRUE	
Prediction	Good	Bad
Good	18091	1819
Bad	1111	874

Logiboost (LB)

LogiBoost - Unbalanced		
	TRUE	
Prediction	Good	Bad
Good	18930	2078
Bad	272	615

LogiBoost - Balanced		
	TRUE	
Prediction	Good	Bad
Good	18790	2459
Bad	412	234

Support Vector Machines (SVM)

SVM - Unbalanced		
	TRUE	
Prediction	Good	Bad
Good	18927	2063
Bad	275	630

SVM - Balanced		
	TRUE	
Prediction	Good	Bad
Good	16844	1561
Bad	2358	1132

SVM.rad - Unbalanced		
	TRUE	
Prediction	Good	Bad
Good	18980	2075
Bad	222	618