Carvana: Predicting “Kicked” Purchase of Used Cars for An Online Dealership

DNSC 6279 Data Mining

May 5, 2018

Sahar Sohofi

Xuan Yang

Yinlu Wu

Problem Description

Carvana is a technology start-up based in Phoenix Arizona. It is an online-only used car dealership and it allows people to buy and sell their used cars through the website. Assuming a customer would like to purchase a used car, on the website they are able to search for a car according to body style, price range, financing, amount of monthly payments, year, color, interior features, engine transmission, cylinders, drive, fuel type.

As a way to improve their service and help their customers make more satisfactory purchases, Carvana would like to use their data on previous purchases to find out if each purchase of used car is a good or a bad purchase through supervised learning, and train the model built to predict good/bad purchases for future deals. A bad purchase, or a "kick", refers to used vehicles with serious issues (e.g. tampered odometers) that prevent them from being resold to customers or requires heavy repair work. Kicked cars are costly to both dealerships and end buyers considering transportation cost, repairmen cost, and loss in market share.

Thus, we need to conduct predictive analysis using supervised learning techniques such as Logistics Regression, Decision Trees and Bootstrap Forest Model, etc. Then select the best model built to help Carvana better care for their customers. Besides, according to the output of the best model, we could realize other interesting observations, for instances, overall likelihood of a kick, any correlation between predictors, and which variables contribute most to our target variable.

Data Selection

The dataset was obtained from Kaggle as a past competition. Our target variable is "IsBadBuy", a binary variable showing whether the purchased car is a "kick". On Kaggle website, there are two datasets, one training set with 72,983 records and a test set with 48,707 records . Because only training dataset has the target variable "IsBadBuy", we only focus on this dataset for building predictive models. Test set is for Kaggle participants to run models on and compete for scores. Besides that, there are 32 predictors, including 16 numerical variables and 16 categorical variables, describing vehicle age, make and model type, reference prices, auction information and other key information. A data dictionary describing all the variables is provided in the appendix.

Data Preprocessing

1. Correct data types as necessary: we found there were lots of predictors false classified. For example, our target variable should be categorical but in the original dataset it was recognized as continuous, so we needed make sure all variables are correctly identified.
2. Missing values: In the original dataset, only the variable “trim” has missing values, but we later discovered that in other variables there are many observations coded with “NULL”. We identified them as missing values. Since SAS JMP supports “informative missing” in decision trees and neural networks, we are not worried about the existence of missing values except for logistic regression models.
3. Corrected typing errors: we only corrected errors when we are certain that they are merely due to typing, such as uppercase vs lowercase.
4. Splitted text for data: in the original dataset, variables “Model” and “Submodel” are long text containing information on engine, volume, number of doors, wheel drive, etc. We used *Excel* to extract such information and created new variables. Unfortunately, since “Model” and “Submodel” are capped at 20 characters, many records are missing such information, making the new variables not reliable. Therefore, we still used the original “Model” and “Submodel” for predictive analysis. The training set with extracted variables are provided as an attachment.
5. Variable selection: we removed some variables which were obviously not related to target variable, like Buyer ID, and also removed variables with repeated information, like WheelTypeID vs WheelType, vehicle_year vs vehicle_age. Besides, we removed variables which were too complicated to model while not adding much predictive power, like Zip code and Purchase date.

Below is the data summary after pre-processing.

Columns	N	N Missing	N Categories	Min	Max	Mean	Std Dev
IsBadBuy	72983	0	2	.	.	.	.
**PurchDate	72983	0	517	.	.	.	.
Auction	72983	0	3	.	.	.	.
*VehYear	72983	0	.	2001	2010	2005.3430524917	1.7312516057818
VehicleAge	72983	0	.	0	9	4.1766438759711	1.7122102878797
Make	72983	0	32	.	.	.	.
**Model	72983	0	1064	.	.	.	.
**Trim	70623	2360	134	.	.	.	.
**SubModel	72975	8	863	.	.	.	.
Color	72881	102	15	.	.	.	.
Transmission	72974	9	2	.	.	.	.
*WheelTypeID	69809	3174	3	.	.	.	.
WheelType	69809	3174	3	.	.	.	.
VehOdo	72983	0	.	4825	115717	71499.995916857	14578.913128203
**Nationality	72978	5	4	.	.	.	.
Size	72978	5	12	.	.	.	.
**TopThreeAmericanName	72978	5	4	.	.	.	.
**PRIMEUNIT	3419	69564	2	.	.	.	.
**AUCGUART	3419	69564	2	.	.	.	.
*BYRNO	72983	0	.	835	99761	26345.842155022	25717.351218728
*VNZIP1	72983	0	.	2764	99224	58043.059945467	26151.640414915
VehBCost	72983	0	.	1	45469	6730.934326213	1767.8464352309
IsOnlineSale	72983	0	2	.	.	.	.
WarrantyCost	72983	0	.	462	7498	1276.5809846129	598.84678820937
Prin1	72668	315	.	-7.767877407126	29.558634404353	0.0011367955648	2.7156883170312
Prin2	72668	315	.	-4.372561241603	4.0153932335083	0.0014192650965	0.5424591535496
Prin3	72668	315	.	-10.07727880101	6.8346784931143	-0.000676700531	0.5002597639041
WheelDrive	22516	50467	5	.	.	.	.
Cylinder	48242	24741	8	.	.	.	.
Volume	40994	31989	.	1.5	8.1	3.3096770259064	0.9580790808608
Doors	63322	9661	4	.	.	.	.
Type	68772	4211	13	.	.	.	.

Data Exploration

Summary and Visualization

As we explored the data we noticed that there is a small proportion of cars labeled as a bad purchase. In the below graphs these cars are highlighted. You can see that these cars even though scattered all over they are mostly aged 4-6 years old. Looking at the ‘Make’ distribution we can see which brands of cars are mainly to be labeled as a bad buy. Some of these cars are Dodge, Chrysler, Ford and Chevrolet. Looking at the Vehicle Cost column distribution we notice how much are most of these bad purchases valued at. These values mostly vary between $4,000-$10,000.

Distributions

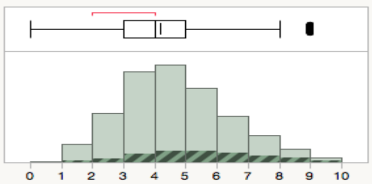
IsBadBuy

Frequencies

Level	Count	Prob
0	64007	0.87701
1	8976	0.12299
Total	72983	1.00000

N Missing 0
2 Levels

***VehicleAge**



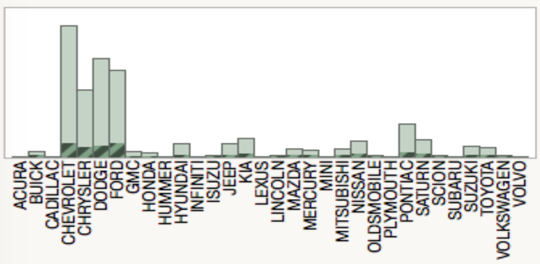
Quantiles

100.0%	maximum	9
99.5%		9
97.5%		8
90.0%		7
75.0%	quartile	5
50.0%	median	4
25.0%	quartile	3
10.0%		2
2.5%		1
0.5%		1
0.0%	minimum	0

Summary Statistics

Mean	4.1766439
Std Dev	1.7122103
Std Err Mean	0.0063379
Upper 95% Mean	4.1890662
Lower 95% Mean	4.1642216
N	72983

Make

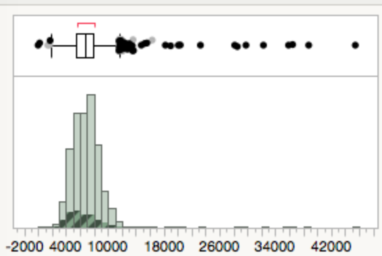


Frequencies

Level	Count	Prob
ACURA	33	0.00045
BUICK	720	0.00987
CADILLAC	33	0.00045
CHEVROLET	17248	0.23633
CHRYSLER	8844	0.12118
DODGE	12912	0.17692
FORD	11305	0.15490
GMC	649	0.00889
HONDA	497	0.00681
HUMMER	1	0.00001
HYUNDAI	1811	0.02481
INFINITI	42	0.00058
ISUZU	134	0.00184
JEEP	1644	0.02253
KIA	2484	0.03404
LEXUS	31	0.00042
LINCOLN	97	0.00133
MAZDA	979	0.01341
MERCURY	913	0.01251
MINI	24	0.00033
MITSUBISHI	1030	0.01411
NISSAN	2085	0.02857
OLDSMOBILE	243	0.00333
PLYMOUTH	2	0.00003
PONTIAC	4258	0.05834

N Missing 0

VehBCost



Quantiles

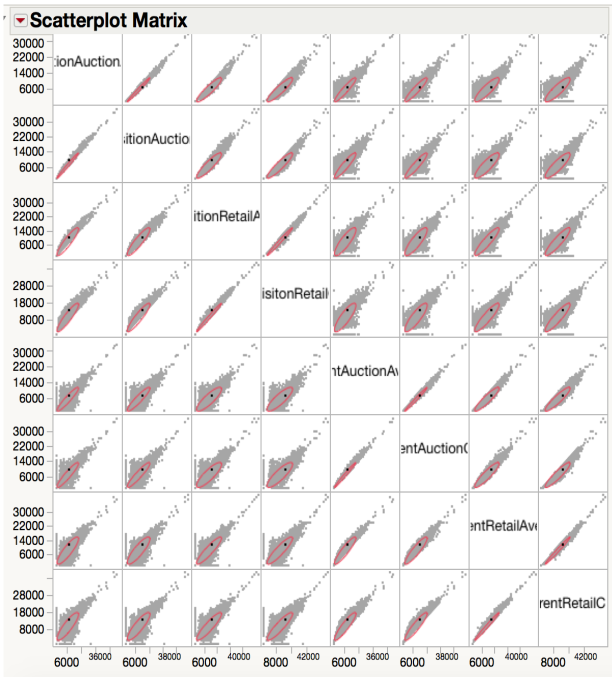
100.0%	maximum	45469
99.5%		11325
97.5%		10362
90.0%		8980
75.0%	quartile	7900
50.0%	median	6700
25.0%	quartile	5435
10.0%		4415
2.5%		3700
0.5%		3100
0.0%	minimum	1

Summary Statistics

Mean	6730.9343
Std Dev	1767.8464
Std Err Mean	6.5438552
Upper 95% Mean	6743.7603
Lower 95% Mean	6718.1084
N	72983

Multicollinearity Issue and PCA

The 8 variables related to auction price and retail price in the dataset are highly correlated as you can see below. A principal component analysis shows that the first PC alone explains 92% of the variation, and the first 3 components explain 99% of total variation. As a result we used the first three principle components instead of the 8 original variables for model building.

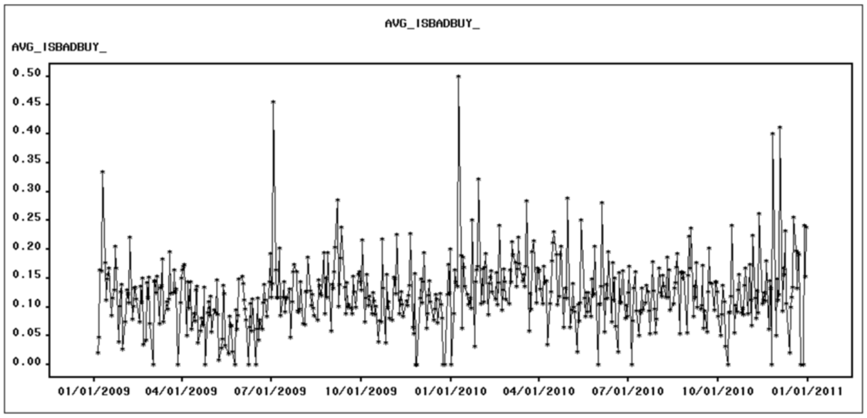


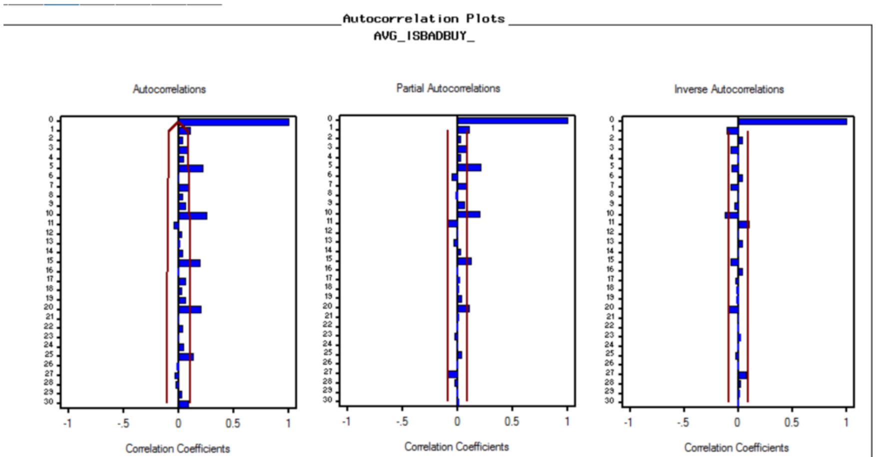
Eigenvalues

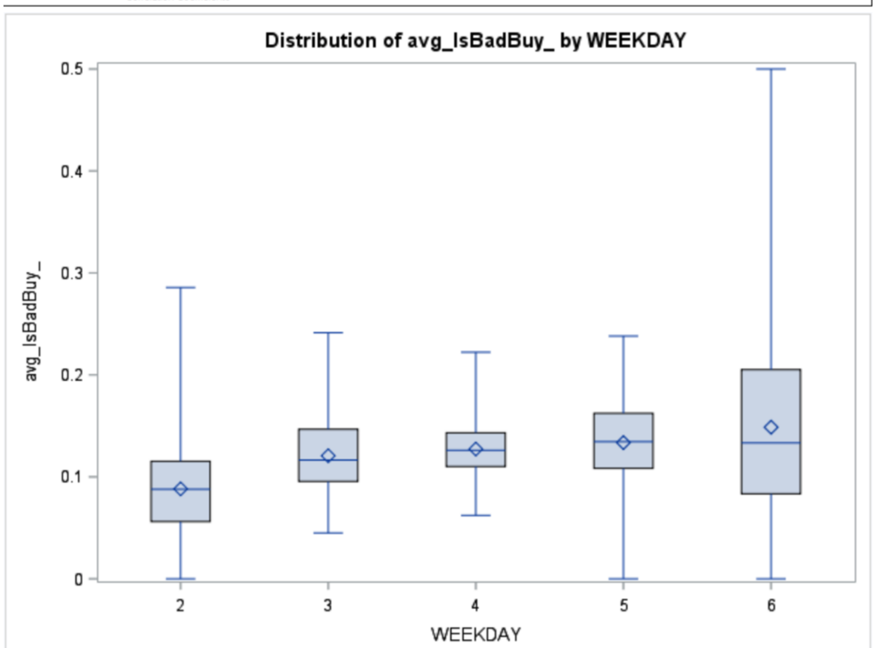
Number	Eigenvalue	Percent	20 40 60 80	Cum Percent
1	7.3735	92.168		92.168
2	0.2943	3.678		95.846
3	0.2501	3.127		98.973
4	0.0437	0.547		99.520
5	0.0289	0.361		99.880
6	0.0060	0.075		99.956
7	0.0023	0.029		99.985
8	0.0012	0.015		100.000

Time Series Pattern

We are interest in whether purchase date has any impact on the likelihood of a kick, and whether time series modeling is necessary. We used R to aggregate the data into daily proportion of bad buys for each day in the dataset, and performed time series analysis using SAS forecasting system. The plot of the series does not show any obvious trend of daily kick rate. There is a strong autocorrelation at lags of multiples of 5. Since the dataset mainly consists of workdays, strong autocorrelation at lag 5 means a weekly pattern. A boxplot by weekday also suggests our suspect of weekly seasonality. Specifically, bad purchases are least likely to be found on Mondays. Since this is beyond the scope of this course and dramatically complicates modeling, we will consider including time series analysis in the next stage. R code and SAS code are included in Appendix B and the aggregated daily kicked rates are included as attachments.


AVG_ISBADBUY_
AVG_ISBADBUY_
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
01/01/2009 04/01/2009 07/01/2009 10/01/2009 01/01/2010 04/01/2010 07/01/2010 10/01/2010 01/01/2011


Autocorrelation Plots
AVG_ISBADBUY_
Autocorrelations
Partial Autocorrelations
Inverse Autocorrelations
-1 -.5 0 0.5 1
Correlation Coefficients
Correlation Coefficients
Correlation Coefficients


Distribution of avg_IsBadBuy_ by WEEKDAY
avg_IsBadBuy_
0.5
0.4
0.3
0.2
0.1
0
2 3 4 5 6
WEEKDAY

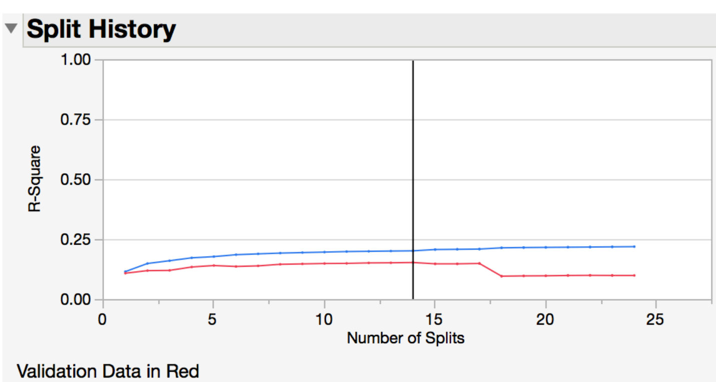
Data Mining Techniques selected

We tried 7 different types of supervised data mining techniques which we share here the significant results we got through some of these techniques. The model comparison section of this document goes into details as to which of the models are better and how. We partitioned the training set of 72,983 records into 60% training and 40% validation. Unless otherwise specified, all modeling is done using SAS JMP. The seven methods are:

- Logistic Regression
- Decision Tree — Partition
- Neural Networks
- Bootstrap Forest
- Boosted Tree
- K Nearest Neighbors
- Naïve Bayes

Decision Tree

The decision tree we got ended in 14 splits. The fit details show a misclassification rate of less than 10% and an RMSE of 0.29. We are focus on predicting IsBadBuy=1 and this model predicts IsBadBuy=0 very well.



Fit Details

Measure	Training	Validation	Definition
Entropy RSquare	0.2043	0.1555	1-Loglike(model)/Loglike(0)
Generalized RSquare	0.2690	0.2080	(1-(L(0)/L(model))^(2/n))/(1-L(0)^(2/n))
Mean -Log p	0.2978	0.3131	∑ -Log(ρ[j])/n
RMSE	0.2901	0.2960	√ ∑(y[j]-ρ[j])²/n
Mean Abs Dev	0.1683	0.1723	∑ |y[j]-ρ[j]|/n
Misclassification Rate	0.0995	0.0996	∑ (ρ[j]≠ρMax)/n
N	43873	29110	n

Confusion Matrix

Training

Actual IsBadBuy	Predicted Count 0	1
0	38244	203
1	4163	1263

Validation

Actual IsBadBuy	Predicted Count 0	1
0	25405	155
1	2744	806

Column Contributions

Term	Number of Splits	G^2	Portion
WheelType	1	3861.52314	0.5758
**Model	3	1748.00388	0.2606
Auction	1	394.82038	0.0589
VehicleAge	4	340.286521	0.0507
VNST	2	187.829799	0.0280
**PRIMEUNIT	2	101.892624	0.0152
Prin3	1	72.0613985	0.0107

Neural Networks

After trying different layers and nodes we reached a more significant result by applying 2 layers and 3 nodes in each layer. Below shows the confusion matrix we got in the results. This model predicts true negatives really well and if we were interested in our model predicting IsBadBuy=0 this would have been a good model but we are focusing on a true positive prediction which is IsBadBuy=1.

Model NTanH(3)NTanH2(3)

Training

IsBadBuy

Measures	Value
Generalized RSquare	0.2294358
Entropy RSquare	0.1721546
RMSE	0.3000068
Mean Abs Dev	0.1811845
Misclassification Rate	0.1166093
-LogLikelihood	13590.389
Sum Freq	43873

Confusion Matrix

Actual	Predicted Count	
IsBadBuy	0	1
0	37044	1403
1	3713	1713

Confusion Rates

Actual	Predicted Rate	
IsBadBuy	0	1
0	0.964	0.036
1	0.684	0.316

Validation

IsBadBuy

Measures	Value
Generalized RSquare	0.1440841
Entropy RSquare	0.105782
RMSE	0.309788
Mean Abs Dev	0.1875273
Misclassification Rate	0.1250429
-LogLikelihood	9652.0408
Sum Freq	29110

Confusion Matrix

Actual	Predicted Count	
IsBadBuy	0	1
0	24464	1096
1	2544	1006

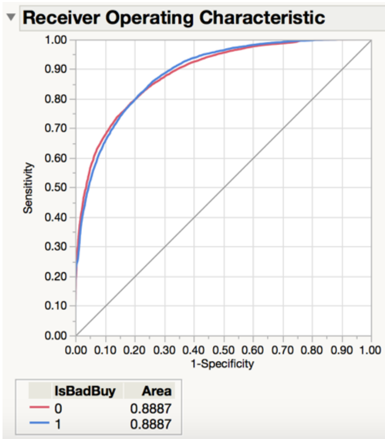
Confusion Rates

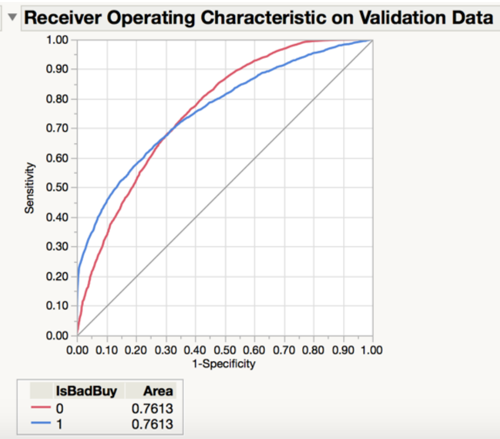
Actual	Predicted Rate	
IsBadBuy	0	1
0	0.957	0.043
1	0.717	0.283

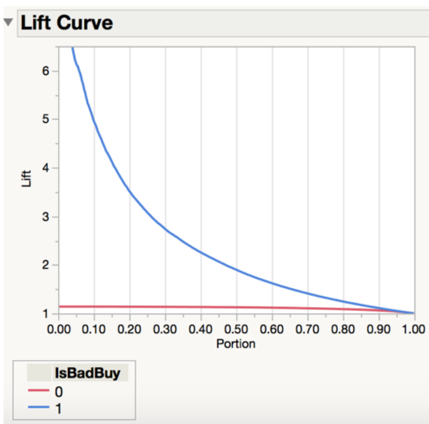
Neural Network does not give us details on column contributors however the fit details are shown below: We can see an RMSE of 0.31 on the validation dataset and a misclassification rate of 12%. The drawback of neural networks is that it does not provide column contribution, and the classification result varies from time to time.

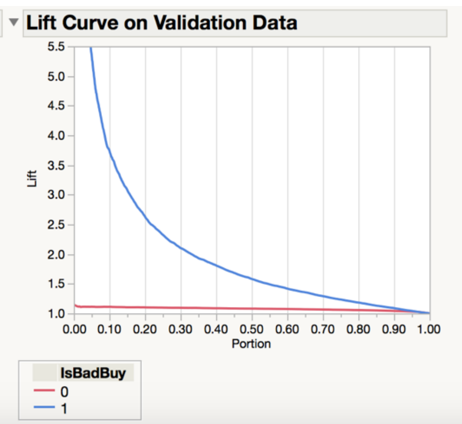
Bootstrap Forest

As you can see in the model details below the confusion matrix results are even worse than the confusion matrix we got in Neural Networks however, the fit details show a low RMSE of 0.29 on the validation data and a misclassification rate of 10% also on the validation dataset. We can also see a good shape in the ROC curve. The ROC curve looks much better on the training data than it does on the validation set. Looking at the lift curve we understand that in this model, for instance, the top 10% does 3.5 times better than a random dataset. This result is obtained with 100 iterations and 5 splits per tree.









Specifications

Target Column:	IsBadBuy	Training Rows:	43813
Validation Column:	Validation	Validation Rows:	29170
		Test Rows:	0
Number of Trees in the Forest:	23	Number of Terms:	17
Number of Terms Sampled per Split:	4	Bootstrap Samples:	43813
		Minimum Splits per Tree:	10
		Minimum Size Split:	72

Overall Statistics

Measure	Training	Validation	Definition
Entropy RSquare	0.2811	0.1743	1-Loglike(model)/Loglike(0)
Generalized RSquare	0.3599	0.2317	(1-(L(0)/L(model))^(2/n))/(1-L(0)^(2/n))
Mean -Log p	0.2683	0.3074	∑ -Log(ρ[j])/n
RMSE	0.2787	0.2949	√ ∑(y[j]-ρ[j])²/n
Mean Abs Dev	0.1679	0.1779	∑ |y[j]-ρ[j]|/n
Misclassification Rate	0.0985	0.1012	∑ (ρ[j]≠ρMax)/n
N	43813	29170	n

Confusion Matrix

Training

Actual	Predicted Count	
IsBadBuy	0	1
0	38203	214
1	4103	1293

Validation

Actual	Predicted Count	
IsBadBuy	0	1
0	25365	225
1	2728	852

Below we can see the details on our column contributors. Wheel type plays a significant role in our model and the second most significant column is Model. Overall 17 predictors contribute to our Bootstrap Forest model.

Column Contributions

Term	Number of Splits	G^2	Portion
WheelType	461	2113.90643	0.3785
**Model	359	1163.28391	0.2083
VNST	707	372.342552	0.0667
VehBCost	636	280.531774	0.0502
*VehicleAge	710	271.727404	0.0487
Color	849	220.985423	0.0396
Make	617	193.472277	0.0346
Auction	517	192.382025	0.0344
WarrantyCost	866	182.694439	0.0327
Prin1	408	132.790766	0.0238
Size	562	128.411466	0.0230
Prin3	423	108.916939	0.0195
VehOdo	451	107.574215	0.0193
Prin2	378	83.9806732	0.0150
**Nationality	198	23.6918742	0.0042
Transmission	56	5.6594442	0.0010
IsOnlineSale	35	2.73955646	0.0005

Boosted Tree

We move on to Boosted Tree in hopes of a better result. As shown below the confusion matrix seems similar to Bootstrap Forest confusion matrix. Fit details show a misclassification rate of 10% and RMSE of 0.29. The lift curve also seems similar and it shows that the top 10% of our model does 3.4 times better than a random sample. ROC curve on the training dataset shows a much poorer performance than ROC curve in Bootstrap Forest.

Boosted Tree for IsBadBuy

Specifications

Target Column:	IsBadBuy	Number of training rows:	43813
Validation Column:	Validation	Number of validation rows:	29170
Number of Layers:	50		
Splits per Tree:	3		
Learning Rate:	0.1		
Overfit Penalty:	0.0001		

Overall Statistics

Measure	Training	Validation	Definition
Entropy RSquare	0.1851	0.1657	1-Loglike(model)/Loglike(0)
Generalized RSquare	0.2454	0.2210	(1-(L(0)/L(model))^(2/n))/(1-L(0)^(2/n))
Mean -Log p	0.3041	0.3106	∑ -Log(ρ[j])/n
RMSE	0.2922	0.2963	√ ∑(y[j]-ρ[j])²/n
Mean Abs Dev	0.1766	0.1797	∑ |y[j]-ρ[j]|/n
Misclassification Rate	0.0996	0.1026	∑ (ρ[j]≠ρMax)/n
N	43813	29170	n

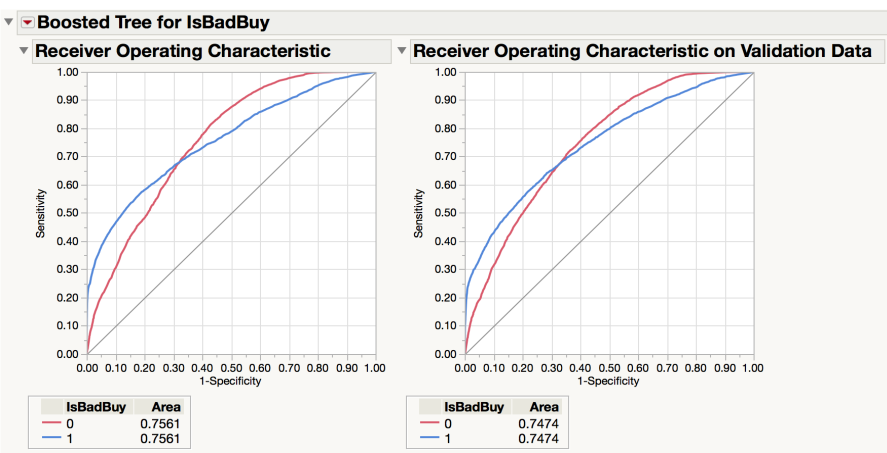
Confusion Matrix

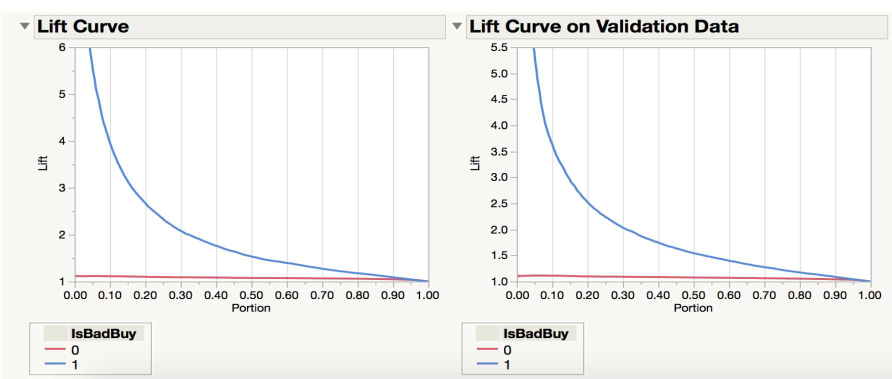
Training

Actual IsBadBuy	Predicted Count 0	1
0	38183	234
1	4128	1268

Validation

Actual IsBadBuy	Predicted Count 0	1
0	25369	221
1	2773	807





The column contributor seems somewhat different from our previous model. There are 11 contributors, the most important column is still wheel type but the second most important is no longer model, it is now vehicle age/year.

Column Contributions

Term	Number of Splits	G^2	Portion
WheelType	28	1230058.3	0.4093
VehYear	29	951455.132	0.3166
VehBCost	24	176419.068	0.0587
WarrantyCost	18	165425.099	0.0550
**Model	20	160411.237	0.0534
VNST	7	99379.8272	0.0331
Auction	11	98322.2086	0.0327
**TopThreeAmericanName	2	59508.7091	0.0198
Make	1	32765.2078	0.0109
Prin3	9	16174.181	0.0054
VehOdo	1	15635.317	0.0052
Color	0	0	0.0000
Transmission	0	0	0.0000
**Nationality	0	0	0.0000
Size	0	0	0.0000
IsOnlineSale	0	0	0.0000
Prin1	0	0	0.0000
Prin2	0	0	0.0000

K Nearest Neighbors

K Nearest neighbors is extremely slow to run and the classification result is not as satisfactory as bootstrap forest and boosted trees.

K Nearest Neighbors

IsBadBuy

Training Set

K	Count	Misclassification Rate	Misclassifications	
1	43873	0.18786	8242	
2	43873	0.18866	8277	
3	43873	0.14346	6294	
4	43873	0.14492	6358	
5	43873	0.13231	5805	
6	43873	0.13231	5805	
7	43873	0.12705	5574	
8	43873	0.12737	5588	
9	43873	0.12484	5477	*
10	43873	0.12497	5483	

Validation Set

K	Count	Misclassification Rate	Misclassifications	
1	29110	0.18392	5354	
2	29110	0.18643	5427	
3	29110	0.14452	4207	
4	29110	0.14394	4190	
5	29110	0.13030	3793	
6	29110	0.13026	3792	
7	29110	0.12635	3678	
8	29110	0.12679	3691	
9	29110	0.12436	3620	*
10	29110	0.12436	3620	

Confusion Matrix for Best K=9

Training Set

Actual IsBadBuy	Predicted Count 0	1
0	38183	264
1	5213	213

Validation Set

Actual IsBadBuy	Predicted Count 0	1
0	25364	196
1	3424	126

Naive Bayes

Results of our Naive Bayes model show a higher misclassification rate of 15% however the confusion matrix shows a higher level of sensitivity meaning identifying the true positives.

Naive Bayes

IsBadBuy

Training Set

Count	Misclassification Rate	Misclassifications
43859	0.17486	7669

Validation Set

Count	Misclassification Rate	Misclassifications
29102	0.18318	5331

Confusion Matrix

Training Set

Actual IsBadBuy	Predicted Count 0	1
0	34578	3860
1	3809	1612

Validation Set

Actual IsBadBuy	Predicted Count 0	1
0	22826	2732
1	2599	945

Model Comparison

Below is a comparison of all the seven data mining technique we tried and their key measurements. We include here misclassification rate because it is the most commonly used measurement for classification problems. Sensitivity is included because we are more concerned about target level IsBadBuy=1 and need to capture this portion of data as much as possible. AUC is also included because it is related to the measurement used in the Kaggle competition.

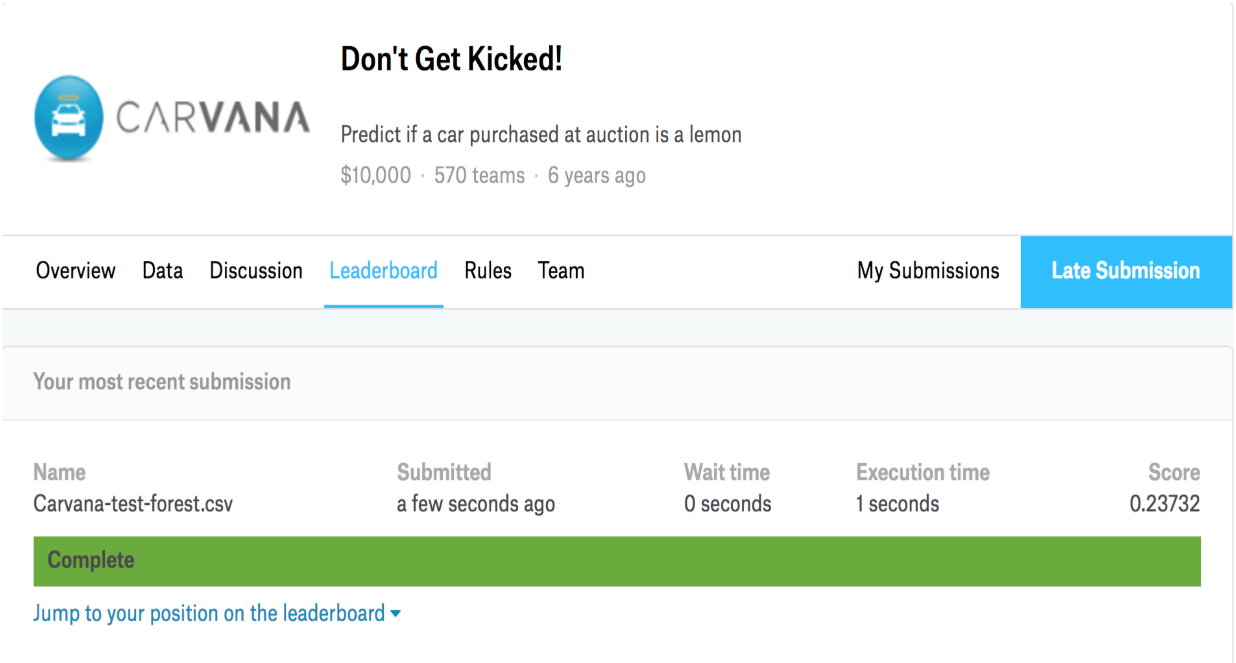
	Mis-classification	Sensitivity	AUC	Top Contributors
Logistic Regression	0.1073	0.2332	0.7383	**WheelType, Model, VehAge,** VNST, VehOdo, VehBCost
Decision Tree	0.0996	0.2270	0.7275	**WheelType**, **Model,** Auction, **VehAge**
Neural Networks	0.1250	0.283	0.7270	N/A
Bootstrap Forest	0.1012	0.24	0.7611	**WheelType, Model**, **VehAge**, VNST
Boosted Tree	0.1004	0.2251	0.7560	**VehAge, WheelType**, Primeunit, Auction, Top3American
K Nearest Neighbors	0.1244	0.0355	N/A	N/A
Naive Bayes	0.1596	0.3581	N/A	N/A

As shown above, there is no universal "best" model with best performance in all measurements. The best misclassification rate is provided by Decision tree, while best sensitivity is provided by Naïve Bayes and best AUC provided by Bootstrap forest. We need to further clarify the business goal to decide which measure to rely on and which data mining model to choose.

The Kaggle competition used Gini index as the scoring criterion, and the formula of Gini Index is:

Gini Index = (AUC - 0.5) * (1 - probability(IsBadBuy=1))[3].

Since our bootstrap forest gave the highest AUC, we ran bootstrap forest on the test set and submitted to Kaggle as a late submission. We received a score of 0.23732, ranking 129 among the total 570 submissions, or 22% in place, and the highest score is 0.26719. The test set pre-processed using the same methods, and the test set submission, are also attached to this report.



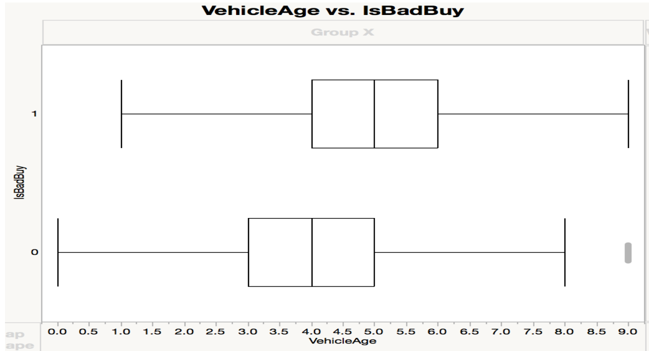
Conclusion

For the purpose of the Kaggle competition, our best model is bootstrap forest with AUC of 0.76. In reality, we expect businesses to care more about misclassification results and sensitivity rate. Since the cost of misclassifying a Kick and buys it outweighs the cost of misclassifying a good car and misses it, we recommend our Naïve Bayes model to Carvana, which has the highest sensitivity of capturing kicked purchases. However, even the highest sensitivity (36%) is not good enough to make improvement in business, and it is achieved by sacrificing misclassification rate.

All our data mining models agree on the below list of variables that contribute most to identify a bad buy: Wheel Type, Vehicle Age, Model, State, Purchase Cost. Based on the below tabulates, we recommend Carvana when making used car purchases:

- Avoid vehicles with no Wheel Type information, and old vehicles!
- Be cautious in certain states
- Be cautious with cheap deals
- Be cautious with certain model types

	IsBadBuy			
	0		1	
WheelType	Row %	N	Row %	N
Alloy	88.95%	32065	11.05%	3985
Covers	91.96%	30349	8.04%	2655
Null	29.52%	937	70.48%	2237
Special	86.89%	656	13.11%	99



The biggest challenges in this data mining project are:

- Messy data, with large proportion of missing values, cropped model names, and missing important vehicle features like engine information.
- Some models are not stable because of randomness in nature: neural networks provide different results when the model is run repeatedly, even with identical settings. Bootstrap forest and boosted tree provides different column contribution information when run repeatedly.
- Trade-off between different measurements: since these measurements do not agree on the same model, choosing the most important measurement is crucial to business.
- Slow algorithms: logistic regression is extremely slow when one or more of the categorical variables contain too many levels, like our “model” variable with 1000 levels. K-nearest neighbors method is extremely computationally intensive, which took us 5 hours to run in SAS JMP.

In our nest stage, we expect to further improve our models by:

- Trying different cut-off probabilities to balance between misclassification rate and sensitivity rate.
- For records with cropped model name, search for extra data and supplement with information like engine, volume, etc.
- Perform time series analysis using purchase date.

Appendix A: Data Dictionary

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 vehicle's 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 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

AUCGUART	The level guarantee provided by auction for the vehicle (Green light - Guaranteed/arbitratable, Yellow Light - caution/issue, red light - sold as is)
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 B: Time Series Codes

R code:

```
car <- read.csv(file='/Users/florencewu/Desktop/DNSC 6279 Data Mining/Project/Carvana - clean and
PCA 4-12.csv')

ts <- read.csv(file='/Users/florencewu/Desktop/DNSC 6279 Data Mining/Project/IsBadBuy-Time.csv')

library(sqldf)

ts$PurchDate = as.Date(ts$PurchDate, "%m/%d/%y")

ts1 <- sqldf("select avg(IsBadBuy), PurchDate from ts group by PurchDate")

ts3 = ts(ts1)

write.csv(ts3, file='/Users/florencewu/Desktop/DNSC 6279 Data Mining/Project/IsBadBuy-TSW.csv')
```

SAS Code:

```
DATA NEW;
SET WORK.CAR;
TIME= _N_;
WEEKDAY=WEEKDAY(PURCHDATE);
RUN;
PROC SORT DATA=NEW;
BY WEEKDAY;
RUN;
PROC BOXPLOT;
PLOT AVG_ISBADBUY_ * WEEKDAY;
RUN;
```

Reference

[1] https://www.kaggle.com/c/DontGetKicked/data

[2] https://www.carvana.com/

[3] https://www.kaggle.com/c/DontGetKicked/discussion/925