Bill Driscoll: kolyyu22
David Nelloms: nelloms

CS 221 FINAL PROJECT: Classifying Kicked Cars

Abstract:

We classified cars purchased in auctions into two categories given a set of features. We used a Naïve Bayesian Classifier and a Least-squares Regression Classifier. Both were able to produce results that were a slight improvement over the baseline. LSR was slightly more effective than NBC at classifying cars, with 89.49% accuracy to Bayes' 88.37%. In the future we hope to clustering information to improve results.

Introduction:

Our task was to identify "kicked" cars based on a set of features describing the car in question. "Kicked" means the car was purchased at an auction and had unforeseen problems such as mechanical issues or a tampered-with odometer. This is important because each kick can cost thousands of dollars to the purchaser, and identifying kicked cars in advance of a purchase can provide real value. We obtained the data from the website www.kaggle.com and it was provided by CarVana.com. The data included thirty-three features potentially relevant to identifying kicks such as purchase date, vehicle year, make, model, transmission, and odometer reading. One of the challenges we faced in taking on this project was that the data was unbalanced—that is, there were far more examples of cars that were not kicked than cars that were kicked. We had to extract as much information as we could from the few positive examples that we had. There was also no guarantee that the features in the data set corresponded in any way to whether or not a car was kicked. Our biggest challenge was dealing with incomplete data—many of the features were null or unknown and we had to develop a framework to handle this. There was previous work done on the problem by participants in the contest hosted on Kaggle, but neither their code nor results were public.

Approach:

Baseline:

Our model for the baseline was the probability of the occurrence of an outcome in the training set. To classify, we predicted the most-likely outcome (in this case "not kicked") every single time. To train, we took the number of kicked cars vs non-kicked cars in the data, and selected which one was larger. For test data, we predicted the more frequent outcome for every example. This was a reasonable approach because with little information and next to no computation we could make fairly accurate predictions (well over 50% in this case). It also worked well as a baseline because any performance worse than this could be immediately discarded as not useful. Its main strength was that it was trivial to compute and not complex; its main weakness that it did not utilize the information stored in features.

Naïve-Bayes:

Naïve Bayes models the problem as the joint probability of features and outcome, assuming independence of features. In testing, given a set of features, we computed two conditional probabilities, one of which is the probability of the features we provided given that the car was a kick, and the other the probability of the features given that the car was not a kick. From there, we predicted the outcome that generated the higher probability, which is maximizing the likelihood of the features.

Bayes is a well-practiced method of classification with a sound theoretical background. It is often used as the first beyond-the-baseline approach to solving classification problems. Our data fit the model as it was mostly discrete.

One weakness of Bayes is that it assumes independence of features, which in most cases is not completely true. There is also the issue of underflow to contend with because Bayes involves multiplying numerous small probabilities.

Because Bayes requires numerous counts of occurrences to generate probabilities, we stored our data in a SQLite Database to efficiently complete such computations. We used the aggregate “count” function to quickly determine the number of occurrences of different features and feature-output combinations in real time. We used PHP because of its well-supported integration with SQLite. We generated two multidimensional associative arrays, one of which stored the probability of a feature, and the other of which stored the joint probability of the feature and a kicked output. Because of the numerous missing and null values, as well as the sparseness of kicks, we used LaPlace smoothing and experimented with different values of lambda to get optimal results. With many features and many more values that each feature could take on, we summed the logarithms of probabilities instead of multiplying the probabilities to contend with the underflow problem.

Least-squares Regression:

Least-squares Regression directly models the probability of the output given the features (as opposed to the joint probability of features and output in Bayes). We extracted features from the training data and used stochastic gradient descent to update a set of weights corresponding to each feature with regards to a loss function. When predicting an output given a set of features, we took the dot product of the features and their weights to produce a predictive value that could be mapped to a “kick” or “not kick” prediction.

We chose LSR because it is a very common method of classification with many resources and theoretical underpinnings. More so than Bayes, it helps to highlight the important features in classifying the cars. One weakness is the possibility of overfitting to the training data. Another is that it relies on the data being linearly separable. And finally, it takes longer than Naïve Bayes: the optimization process is more complex as there is no closed-form solution to generate optimal weights.

We implemented the algorithm in Python. One of the reasons for this was that there was a previous assignment that provided useful structures for such a task. We made extensive use of the Counter data structure for maintaining weights and features. We used an L2 regularizer to avoid overfitting. We initially planned on using a hinge-loss function, but switched to least-squares because it is not a one-sided loss function.

Data and Experiments:

Bayes:

We evaluated the effectiveness of the algorithm based on what percentage of cases it could correctly classify compared to the baseline. We modified the parameter lambda to simulate having seen certain kick-feature combinations multiple times. In our results, the minimum lambda value of 1 produced the best results. Increasing lambda decreased accuracy in a roughly linear fashion. With the optimal value of lambda, we were able to correctly classify 88.374% of cars, compared to a baseline value of 87.693%. While a 0.681% increase was not as high as we expected, it still represents an appreciable increase given that incorrectly predicting a single kick is a matter of thousands of dollars.

We also generated a table that displayed the probability of the car being a kick given a single feature, for every feature, determined using our distributions from the training data. While this was somewhat revealing, it was also heavily skewed by LaPlace smoothing.

The runtime of this algorithm was overall determined by the number of values that the features could take on and the size of the testing data set. Building the probability distributions was accomplished in linear time. In practice, we could run the full size test cases much quicker using Naïve Bayes than with Least-squares Regression.

Least-squares Regression:

We evaluated the rough effectiveness of regression in the same way that we did for Bayes. The results were slightly better—for the optimum configuration of parameters, it predicted the correct output 89.49% of the time, compared to the same baseline of 87.693%, representing a 1.797% improvement. We also used a more fine-grained evaluation of performance by tracking recall and specificity. Using precision and recall, we were able to compute an F1 score which better reflects the algorithm's overall performance.

As with Bayes, we attempted to discern which features were most relevant in determining whether a car was a kick or not a kick, this time by looking at the weights associated with each feature. The results were more meaningful this time, and we were able to ascertain that the two most important features were the age of the car and, somewhat surprisingly, the information about its tires. Having no information about the car's tires was the single feature that most strongly implied that the car was a kick. A close second was the car having a high age. Likewise, having information about the wheel type most strongly implied that the car was NOT a kick, followed by the car being of a low age. We were also able to look at which makes of cars were the "best" and "worst" based on how strongly they correlated with being kicks. Based on our results, the best-made cars were Hondas, and the worst-made were Suzukis.

We also experimented with different parameters and discovered that if the initial step size was too large the predictive power decreased although recall increased. We came to the conclusion that the large step size did not have time or granularity to settle at the minimum of the loss function. We found that combination of parameters that gave the best results was 0.1 step size, 0.5 step size reduction, 20 training rounds, and a regularization factor of 1, although with the exception of step size and step size reduction the predictions were fairly insensitive to changes in parameters.

Because the training process must update the weight for each feature in every sample multiple times, this algorithm took significantly longer to run. Initial passes on the full data before optimization took up to 30 minutes to complete. We cut down on the amount of data by a scaled amount to compensate for this.

Conclusion:

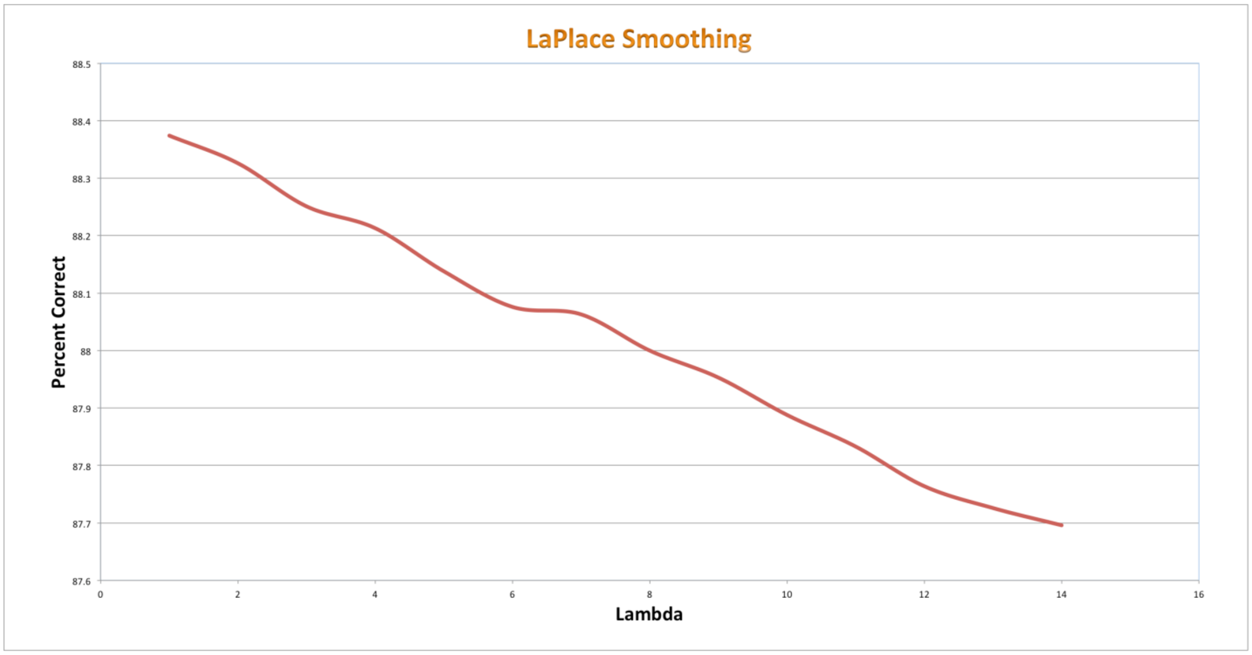
Although we were able to get meaningful results using NBC and LSR, we would have liked to attempt to use clustering to extract more meaning from our classified results. Another possible extension would be to try SVM because it is possible that the samples were not linearly separable.

Over the course of the project we encountered many difficulties: for example, the difficulty of working with a large amount of data (on the order of tens of thousands of samples). As mentioned, we had to reduce the amount of data in order to get our results in an acceptable amount of time. We also ran into many implementation bugs and spent the majority of our time debugging. We would most likely use a pre-written library in the future rather than coding almost everything ourselves. We also found dealing with incomplete data to be surprisingly difficult, as an ordinate amount of time was spent on conceptualizing how to represent null and previously-unseen values in our models. Finally, we learned that next time we are buying a used car, it might be best to stay away from Suzuki!

Bill Driscoll: kolyyu22
David Nelloms: nelloms

Appendices:

Appendix 1: Lambda values vs Correct classification percentage



Appendix 2: Probability of Kicked given Variable calculated from training distributions with lambda = 1

VARIABLE	P(KICKED | VARIABLE)
VehYear nothing	0.5
VehicleAge nothing	0.5
Make nothing	0.5
Color nothing	0.5
Transmission nothing	0.5
WheelTypeID nothing	0.5
WheelType nothing	0.5
Nationality nothing	0.5
VehSize nothing	0.5
TopThreeAmericanName nothing	0.5
VNST nothing	0.5
IsOnlineSale nothing	0.5
WheelTypeID 0	0.4
Make LEXUS	0.375
Make INFINITI	0.344827586
VehYear 2010	0.333333333

Bill Driscoll: kolyyu22
David Nelloms: nelloms

Make HUMMER	0.333333333
Make PLYMOUTH	0.333333333
Make TOYOTA SCION	0.333333333
VehicleAge 9	0.315649867
Make LINCOLN	0.314285714
Make ACURA	0.307692308
VNST AR	0.307692308
Make MINI	0.294117647
VehYear 2001	0.286358512
Make SUBARU	0.28
VehicleAge 8	0.268401487
VehicleAge 0	0.25
VehYear 2002	0.247109827
Color NOT AVAIL	0.225806452
VehicleAge 7	0.224025974
Make OLDSMOBILE	0.219354839
Make CADILLAC	0.2
VehYear 2003	0.192556634
VNST PA	0.181818182
VehicleAge 6	0.174440107
VehSize SPORTS	0.174291939
Make MAZDA	0.169491525
VehSize COMPACT	0.167090363
VNST MI	0.166666667
VNST NY	0.166666667
Make NISSAN	0.164915117
VNST VA	0.163326653
VehSize SMALL TRUCK	0.16
VNST IN	0.159010601
VNST MD	0.157650696
VehSize LARGE SUV	0.157024793
TopThreeAmericanName FORD	0.153961136
Make SUZUKI	0.153266332
Make MERCURY	0.152985075
Make FORD	0.152582507
VehYear 2004	0.152440996
Make JEEP	0.152259332
VNST IL	0.150537634

Bill Driscoll: kolyyu22
David Nelloms: nelloms

Make BUICK	0.150234742
Color YELLOW	0.148648649
WheelTypeID 3	0.148558758
WheelType Special	0.148558758
VehicleAge 5	0.145419355
VehSize MEDIUM SUV	0.144701783
Make VOLKSWAGEN	0.144444444
VNST SC	0.14395689
Make MITSUBISHI	0.142620232
Color GOLD	0.142030848
VNST LA	0.141463415
Color BEIGE	0.140243902
VehSize SMALL SUV	0.13992674
Color PURPLE	0.138655462
VNST TX	0.138476515
VNST NV	0.138028169
Nationality TOP LINE ASIAN	0.137078652
Make CHRYSLER	0.136304063
Color BROWN	0.134948097
TopThreeAmericanName OTHER	0.134612691
Color RED	0.134064758
Nationality OTHER ASIAN	0.133876358
Color OTHER	0.133802817
VNST CA	0.133395306
VehYear 2005	0.131624864
Nationality OTHER	0.130769231
Make SATURN	0.130668716
VNST AL	0.12962963
Color GREEN	0.128739316
Transmission MANUAL	0.128571429
Color WHITE	0.126866694
VehSize VAN	0.126764621
VNST UT	0.126195029
Make HYUNDAI	0.126058325
VNST NM	0.125
IsOnlineSale 0	0.123385301
Transmission AUTO	0.122783761
Color SILVER	0.122439843

Bill Driscoll: kolyyu22
David Nelloms: nelloms

VNST NJ	0.121052632
Nationality AMERICAN	0.12070612
VNST CO	0.120364346
TopThreeAmericanName CHRYSLER	0.1189562
Color BLUE	0.118789448
VNST NH	0.117647059
Make PONTIAC	0.117507886
VNST IA	0.115511551
VehSize CROSSOVER	0.115234375
VehSize MEDIUM	0.114573786
Make GMC	0.113989637
Color MAROON	0.113328013
Color GREY	0.113071201
VNST GA	0.112560055
VNST AZ	0.112244898
VNST NC	0.111243734
Make KIA	0.110663984
VNST FL	0.110647182
WheelTypeID 1	0.110112463
WheelType Alloy	0.110112463
VehicleAge 4	0.109800452
Color BLACK	0.109419186
VNST TN	0.108874657
Make HONDA	0.107255521
VehSize LARGE TRUCK	0.10723719
IsOnlineSale 1	0.107205624
TopThreeAmericanName GM	0.106355042
VNST MS	0.104166667
Make DODGE	0.102723735
Make TOYOTA	0.102071006
VNST ID	0.100840336
Make CHEVROLET	0.097071531
VehSize SPECIALTY	0.095818815
VNST MO	0.095343681
VNST OH	0.093541203
VehYear 2006	0.093088995
VNST WA	0.092105263
VNST WV	0.091463415

VehSize LARGE	0.090516432
VNST OK	0.089957163
VNST NE	0.086956522
Make SCION	0.085365854
VNST MA	0.083333333
VehicleAge 3	0.082496863
VehYear 2007	0.080715532
WheelTypeID 2	0.080004056
WheelType Covers	0.080004056
VNST KY	0.078014184
VNST MN	0.073170732
Color ORANGE	0.072519084
VehicleAge 2	0.065843621
VNST OR	0.062015504
VehYear 2008	0.056439942
VehicleAge 1	0.039374326
Make ISUZU	0.038961039
Make VOLVO	0.037037037
VehYear 2009	0.034136546

Bill Driscoll: kolyyu22
David Nelloms: nelloms

Appendix 3: Parameter configurations and their corresponding performance metrics for Least-squares regression

Parameter Variation and Performance of Least Squares Regression

Initial Step Size	Step Size Reduction	Number of Rounds	Regularization	Correctness	Precision	Recall	Specificity	F1 Score	True Positives	True Negatives	False Positives	False Negatives
Var												
1	0.5	10	1	0.732890319929	0.180033480444	0.329251322015	0.789539471114	0.232782369146	1183	20213	5388	2410
0.1	0.5	10	1	0.894841405768	0.709703287891	0.246312273866	0.985859927347	0.3654702479339	885	25239	362	2708
0.01	0.5	10	1	0.894870152155	0.709134615385	0.246312273866	0.985820866372	0.36526936583	885	25238	363	2708
	Var											
0.1	0.9	10	1	0.891107761869	0.70495049505	0.198163094907	0.309363458614	0.309363458614	712	25303	298	2881
0.1	0.5	10	1	0.894841405768	0.709703287891	0.246312273866	0.985859927347	0.3654702479339	885	25239	362	2708
0.1	0.1	10	1	0.894909912996	0.718567860117	0.240189256888	0.986797390727	0.360033375052	863	25263	338	2730
0.1	0.01	10	1	0.890011646299	0.649921507064	0.230448093515	0.982578805515	0.340250667762	828	25155	446	2765
		Var										
0.1	0.5	1	1	0.894870152155	0.709134615385	0.246312273866	0.985820866372	0.36526936583	885	25238	363	2708
0.1	0.5	5	1	0.894870152155	0.709134615385	0.246312273866	0.985820866372	0.36526936583	885	25238	363	2708
0.1	0.5	10	1	0.894841405768	0.709703287891	0.246312273866	0.985859927347	0.3654702479339	885	25239	362	2708
0.1	0.5	20	1	0.894909912996	0.710843373494	0.246312273866	0.985938049295	0.365853658537	885	25241	360	2708
0.1	0.5	50	1	0.894841405768	0.710040160643	0.246033954912	0.985898988321	0.365440264572	884	25240	361	2709
			Var									
0.1	0.5	10	1	0.894841405768	0.709703287891	0.246312273866	0.985859927347	0.3654702479339	885	25239	362	2708
0.1	0.5	10	1000	0.894841405768	0.709703287891	0.246312273866	0.985859927347	0.3654702479339	885	25239	362	2708
0.1	0.5	10	10000	0.894841405768	0.709703287891	0.246312273866	0.985859927347	0.3654702479339	885	25239	362	2708
0.1	0.5	10	40000	0.894019319038	0.735599622285	0.216810464793	0.989062927229	0.334909716251	779	25321	280	2814

Category Best	Global Best

Bill Driscoll: kolyyu22
David Nelloms: nelloms

Appendix 4: Variables with the highest and lowest weights in least-squares regression