STANFORD UNIVERSITY, CS229 - MACHINE LEARNING

# Final Project Milestone - Kicked Car Prediction

Albert Ho, Robert Romano, X. Alice Wu

November 16, 2012

## 1 Introduction

When you go to an auto dealership with the intent to buy a used car, you want a good selection to choose from and you want to be able to trust the condition of the car that you buy. Auto dealerships purchase many of their used cars through auto auctions with the same goals that you have: they want to buy as many cars as they can in the best condition possible. The problem that these dealerships often face is the risk of buying used cars that have serious issues, preventing them from being sold to customers. These bad purchases are called "kicks", and they can be hard to spot for a variety of reasons. Many kicked cars are purchased due to tampered odometers or mechanical issues that could not be predicted ahead of time. For these reasons, car dealerships can benefit greatly from the predictive powers of machine learning. If there is a way to determine if a car would be kicked ahead of time, car dealerships can not only save themselves money, but also provide their customers with the best inventory selection possible.

The following paper is split up into 3 main sections describing our approach to solve this problem: Early Interpretation, Data Parsing, and Machine Learning Algorithms. First we identified possible trends by simply graphing data. Next, we came up with strategies for dealing with missing data and data presented in a text format. Finally, we applied various machine learning algorithms to the parsed data set in an attempt to predict which cars would have the highest risk of being kicked.

## 2 Early Interpretation

We obtained our data set from the Kaggle.com challenge "Don't Get Kicked" hosted by Carvana. Our first step was to graph the data we had to gain some initial insight into which features would be the most important. The features we were provided along with their descriptions are shown in Table 1.

## 3 Data Parsing

Many of the features we were looking at came in a text format, so we had to decide how to properly bin them. Furthermore, some data entries were missing, so we had to make some choices for how to replace them.

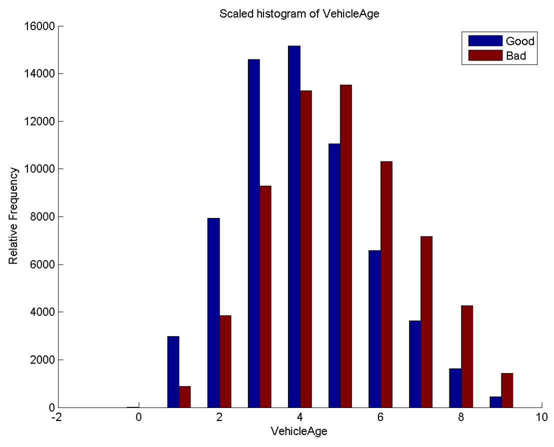
### 3.1 Word Bins

For data such as the name of the vehicle's model, manufacturer, and color, we had to assign unique identifiers to specific strings in the feature space. This was straightforward for a feature like transmission since we could assign 0 for Auto and 1 for manual. The process became more involved with features such as the car submodel. We decided that even though there were many different submodels, categorizing them with unique identifiers rather than grouping them was the more conservative option.

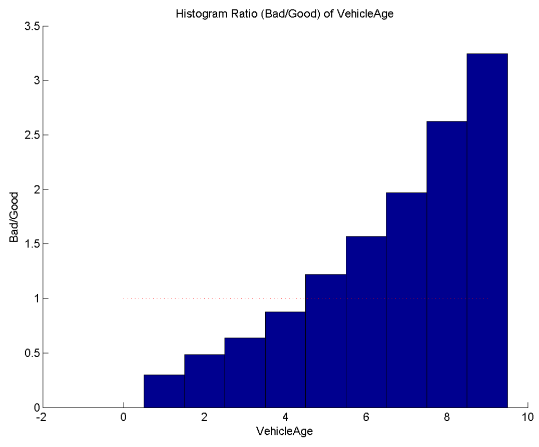
### 3.2 Missing Features

For many of the samples, particular features were missing. We had the option of throwing out the sample completely, but we believed that it would be a waste. We decided to implement the following rules: if the feature was represented with a continuous value, we would replace the missing value with the average of the feature over the other samples and if the feature was represented with a discrete value, we would create a new value specifically to identify missing data.

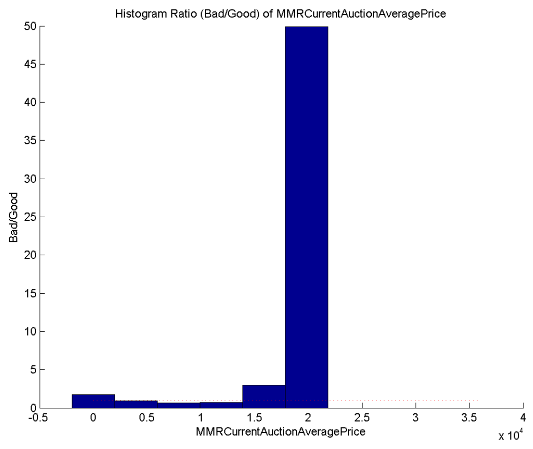
## 4 Data Visualization



(a) Ratio of Scaled VehicleAge



(b) Ratio of VehicleAge



(c) Ratio of MMRCurrentAuctionAvgPrice

Figure 1: blehbleblehblehbleh

## 5 Machine Learning Algorithms

With the data parsed and some initial insights to guide us, we started to apply some machine learning algorithms that would identify where we needed improvement and what strategy would be most effective.

### 5.1 Support Vector Machine

First, we tested our data with an SVM. We used the liblinear package v. 1.92. Our initial runs thus far have only been on raw numerical data (not on all post-processed data), so there is lots of room for improvement. We submitted our solution to Kaggle and placed 526 out of 571 teams on our first run.

### 5.2 Logistic Regression

Since the output of our data is binary, we decided to fit two logistic regression models as a fist pass. Adopting code from the course website and assignment 1, we implemented logistic gradient ascent and Newton's method. Both of the algorithms are run on the numerical training data only. When possible, a 70/30 cross validation was performed. The generalization error is defined as:

$$\varepsilon(h) = \frac{\sum |y_{true} - y_{predicted}|}{n} \tag{5.1}$$

#### 5.2.1 Logistic Gradient Ascent

Using logistic gradient ascent with a learning rate of alpha = 0.0001 and 500 iterations. We were not able to generate any log-likelihood data past the first iteration. The first One possible hypothesis we have about this method is that the data may not be convex

#### 5.2.2 Newton's Method

We first used Newton's method in a 70/30 cross validation scheme. We found that the algorithm converged after 7 iterations, with a relative difference of -1.6007e-010. Although at this point the algorithm was only run on the the numerical data from the total data set, it yielded a generalization error of 0.1178, which is highly promising. However, when the data was visualized