AirBnB

What factors make for a profitable listing

INSY 5378 Big Data Project Report

John Abernethy

Qaisar Butt

Rathna Sindura, Chikkam

Swati Singhal

INTRODUCTION

The travel industry has exploded over the last few years, fueled by the growth of the internet and availability of online tools for anyone to book flights, cruises, hotels, etc. One does not need to be a rocket scientist to make use of these resources and plan their next up and coming adventure. Most of us that travel fairly frequently have run into problems while trying to book a hotel. For instance, the hotel you want to stay has no available rooms, or you can't find one acceptable that is close to where you need to be. This project is about a company that offers an alternative to traditional hotels for travelers' lodging needs: AirBnB. Today, AirBnB allows people to list their homes on the AirBnB website for travelers to rent. They can stay for one to several nights, depending on their needs and the stipulations of the host.

A brief introduction to AirBnb

According to an article published in the Business Insider issue from February 2016, AirBnB started as a way for three young men from San Francisco to turn their loft into a vehicle for helping them to pay the rent back in 2007. Not only did they offer a place to crash, but promised breakfast to their guests, as well. It did not take long for the trio to realize that they were onto something big. Now AirBnB is a multi-billion dollar company. According to the March edition of the online publication Recode, AirBnB claimed its valuation to be at $120 USD per share as it went to take over another rental provider named HotelTonight. This is a manifestation of the AirBnB leadership's desire to offer more accomodation options to their growing customer base.

Analysis Goals

The intent of our team is to determine what dimensions are important in predicting the price that an AirBnB host can charge based on those dimensions. We have two different methods we wish to use to attack the problem. One approach will use feature selection amongst the many available features in the dataset, and then run classical multiple regression as well as other algorithms, including multilayer neural networks, to predict the price. Another approach will be based solely on image data from URL links in the dataset, and feeding those as the feature set to a convolutional neural network (CNN) for predicting price. We will then report on the accuracy of both determine which method is more accurate.

About the Data

The dataset we decided on for analysis is a dataset containing AirBnB listings in the United States. The data was downloaded from the following link: https://public.opendatasoft.com/explore/dataset/airbnb-listings/export/?disjunctive.host_verifications&disjunctive.amenities&disjunctive.features&refine.country=United+States. Upon initial inspection we see the dataset is fairly sizeable, but perhaps not outside of

the abilities of a single machine to handle. Here are the dimensions:

Number of columns: 89, Number of rows: 224091

The variable of interest here is the one named “Price”. Note that “Price” has 96,509 missing values, which is 42.8% Missing values. Of course these missing values will have to be dropped, since it makes no sense to impute missing values of the target variable just to use the same model to predict them.

There are a number of columns that we can tell right off the bat will add nothing to the model, for instance ‘ID’, ‘Listing URL’, etc. Feature selection for initial model will be covered in the following section.

Data Cleaning and preprocessing

Initial feature selection: Dealing with dimensionality (regression problem)

Initially the data was quite messy, it required some wrangling to get it into useable form. We began by reading the data in as an “rdd” type spark object because we noted that some rows were shifted over due to having an “extra column”. We used Spark map and reduce functions to normalize feature names and filter out rows with extra columns. After we did that we had only 88023 rows remaining. That’s 32% of the original data. After converting the dataset to a spark SQLContext DataFrame, it was further processed by checking for “empty” values, which were then filled with the mean value for that column. Aside from the identifying variables and the ones containing urls mentioned above, we dropped some others that were either redundant or added nothing to the model, they are as follows:

['last_scraped','experiences_offered','host_name','host_neighbourhood','host_location', ‘street’,‘neighborhood’,’neighborhood_cleansed’,'city','smart_location','Country', 'country_code','latitude','longitude','geolocation','Calendar_updated', 'calendar_last_scraped']

This is the shape of the dataframe now: Shape of dataframe is 86552 x 51

Further cleaning

Of course there is still more cleaning to do. All the data types are as type string. Columns are separated into DateType, DoubleType, and StringType. Then date columns were converted to an integer value, number of days from that date to today. Furthermore, we kept some columns containing text. The text has been replaced with word counts. These are:

['name','summary','space','description','neighborhood_overview','transit','access',

'interaction','house_rules','host_about','host_verifications','market','amenities','features']

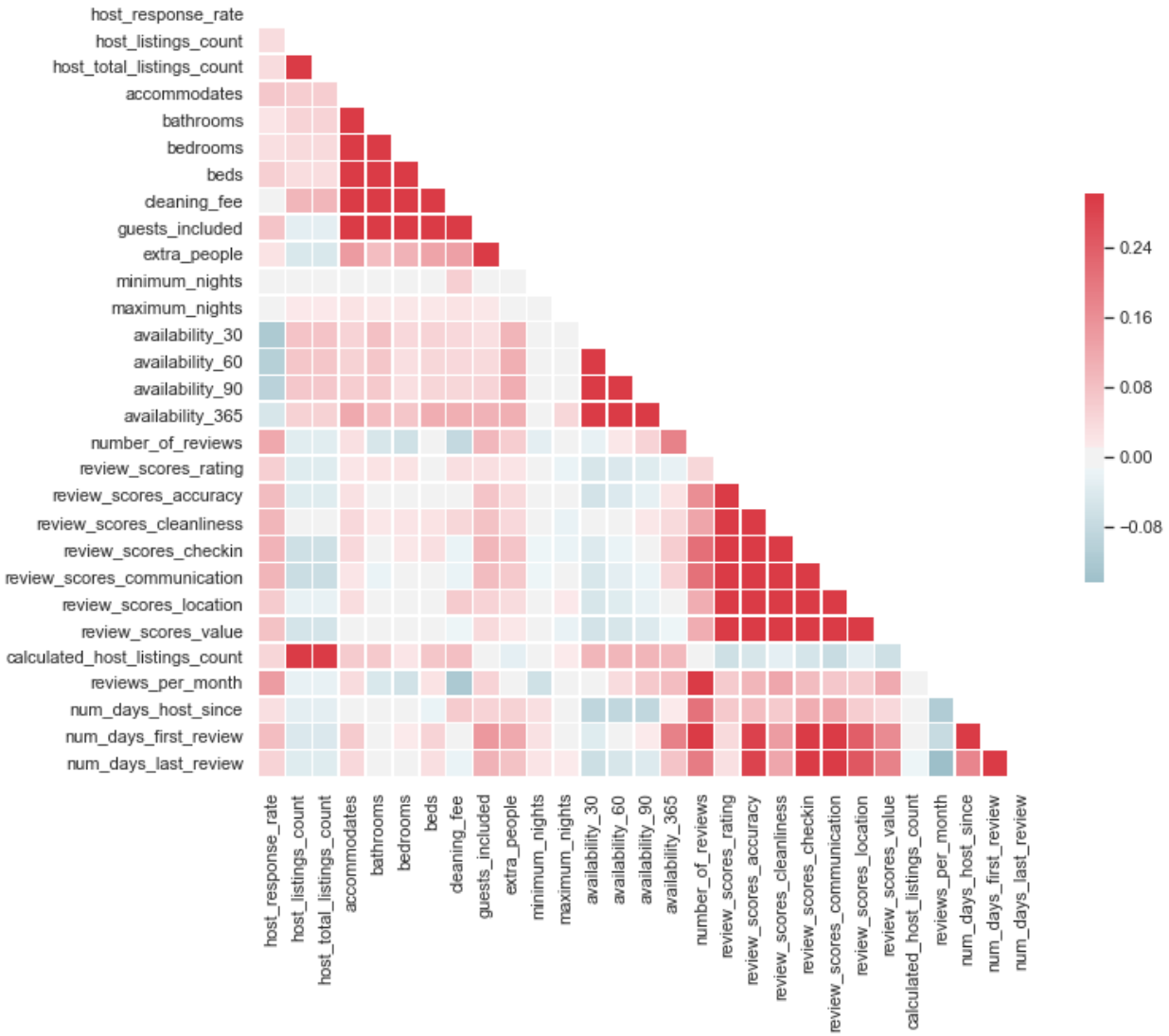
Another problem we ran into was that some of the zip codes were of non-standard length. Most had a length of 5 but there were a few of length 7 and 9, as well. After zip code lengths were normalized to 5, we have 898 distinct values for zip code in the analysis.

Correlational analysis for feature selection

See below the Seaborn heatmap for correlations of the features, note the target variable 'price' is not included. Thirteen features were dropped for being highly correlated with other features, they were:

['cleaning_fee','guests_included','availability_60','availability_90','availability_365', 'number_of_reviews','review_scores_accuracy','review_scores_cleanliness','review_scores_checkin', review_scores_communication','review_scores_location', 'review_scores_value', 'num_days_first_review']

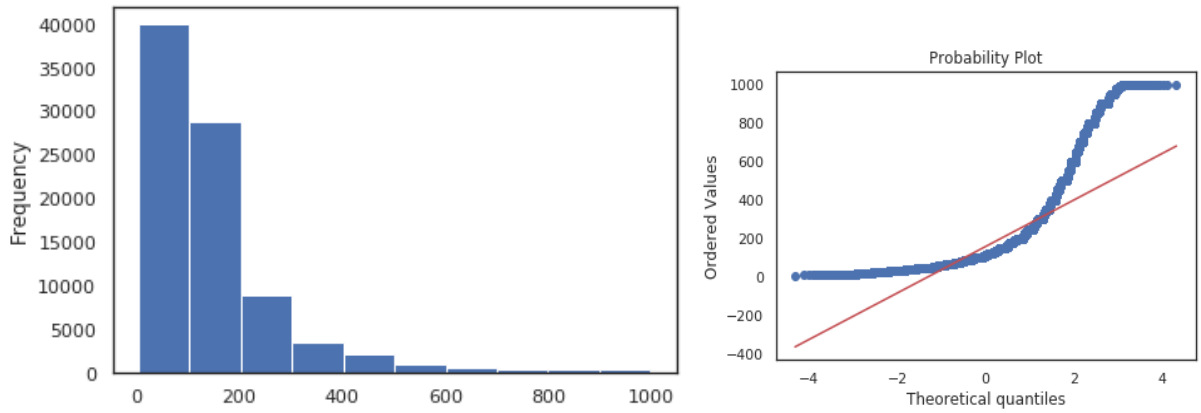
There were also three columns with lists of different items/services offered by hosts, which were ['host_verifications','amenities','features']. They were expanded into separate columns for each value in their respective lists. So, after dropping some columns and adding more, we now have 174 columns.



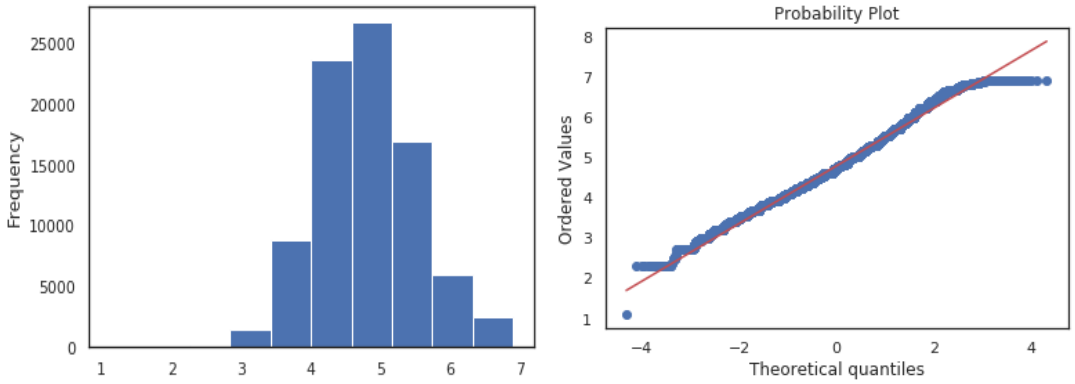
Data preprocessing

At this point it is now a question of how to transform the variables so that they render the best possible model. The problem we are looking at here is a regression problem. Not one of those easy textbook problems they give you in undergraduate stats, but one with traps and pitfalls that we need to be mindful of. There are plenty of numerical variables in the model, most of them differ enough in magnitude and range that they need to be scaled. Of particular angst was the variable "maximum_nights", with a range from 0 to 2147483647. That maximum value that looks like a Dallas telephone number is the maximum number of days that the AirBnB host would allow a renter to stay on the property. Since the average lifespan is somewhere around 29000 days, such a number is nonsensical. The question then becomes, do we do anything with it, or do we just scale and move on?

This kind of distribution is not normal, so a z-score transformation, $z_a = (x_a - \mu)/\sigma$, will not work. It is not exponential, so a log transformation will not work on it, either. We will probably proceed with this one in place and place a cap on it at the (3rd quartile * 1.5 IRQ) point. Note that this will change the high leverage threshold but leveraging outliers will not be nearly as bad. The class variable "price" does roughly follow a classical exponential distribution. It practically screams for a log transformation. Note the quantile and histogram plots:



And after a log-transformation:



We can visually determine that the transformed data is roughly normal. This step is important. In classical regression, if normality and equal variance stipulations are violated, then the model is compromised and perhaps will not make the best possible predictions. As far as the regressors in the model, wherever scaling can be applied, it should be sufficient to scale them and move on, according to this article from "Towards Data Science" found here:
https://towardsdatascience.com/scale-standardize-or-normalize-with-scikit-learn-6ccc7d176a02

Columns with multiple lists

As mentioned briefly at the beginning of this section, there were three fields with lists of values, they are ['host_verifications','amenities','features']. It turns out that there is too much information lost by doing a word count of these, because obviously the lists will contain different entries. For example, a small snippet of sample output from “amenities”:

amenities

0 wireless internet,air conditioning,kitchen,fre...
1 tv,wireless internet,air conditioning,pool,kit...
2 tv,wireless internet,air conditioning,pool,kit...
3 internet,wireless internet,kitchen,free parkin...
4 tv,wireless internet,air conditioning,pool,kit...
5 wireless internet,air conditioning,kitchen,fre...
6 tv,cable tv,internet,wireless internet,air con…

And the lists range in size from 0 to 68. The story of the other two is something similar. So what we did instead of word count was to create a column for each element in a sublist and represent its presence for an entry with a “1”, or its absence with a “0”. This is how we grew out to 174 columns. The number of columns going into the regression model grew to over a 1,000 after one-hot-encoding, thanks to all the zip codes (over 800) in the data set.

Data preparation for the image processing with deep learning model

Data from the top 6 cities in the USA was downloaded and cleaned to maintain authenticity of the listing, and to facilitate the matching of entries with their images. First image downloads were done for only NYC, to test our approach. It took 6-7 hrs to download all the 15537 images. We identified 469 unique prices for the 15537 images from 28682 initial rows of data. We kept ‘listing ID’, ‘picture URL’, and ‘Price’. Converting images to attributes we come out with 299x299 = 89401 attributes with which to predict price. We then dropped unlabeled rows, duplicate rows, null and improbable rows. We know the best way to deal with image data is with convolutional neural networks, but the question is how.

Analysis

As earlier mentioned this is a regression problem, so that is the first analysis approach we will try. In our case the regression has to be performed with a log-transformed response variable "price" and scaled predictor variables, where applicable. Having to log-transform the response poses a problem in using the some of the ML-algorithms in pyspark.ml, like the standard regressionEvaluator package. It was necessary to correct model violations like non-normality and possible heteroskedasticity. Since the model estimations have to be exponentiated to get actual predicted values, we have to do this manually, and then create our own function to calculate MSE and RMSE. R-squared should be calculated from the log regression model.

We used several algorithms, regression, and a handful of others based on regression. They were Random Forest Regression, Gradient Boosting Regression (GBT Regression), and LASSO regression.

Multiple regression

In multiple regression, $\hat{Y} = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n X_n + \epsilon_i$, the idea is to minimize prediction error terms $SSE = \Sigma_{i=1}^{n} (\hat{y} - y_i)^2$. The algorithms handle this with matrix algebra.

GBT regression

Gradient boosting is a regression tree based algorithm that builds an additive model in a forward stepwise manner (scikit-learn.org).

Lasso regression

Lasso regression works by L1 normalization. This adds a penalty to the absolute value of the magnitude of the coefficients. As a result, some of the variable coefficients can be shrunk to exactly zero (Statistics How to).

Random Forest Regressor

Here the estimator is a random forest that fits classifying decision trees on bootstrapped samples from the data. Samples are drawn with replacement, i.e. one entry could be drawn multiple times (scikit-learn.org).

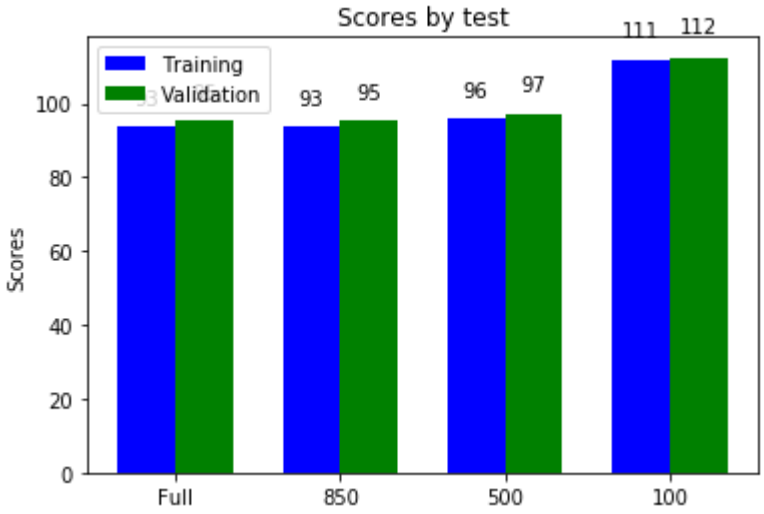
Analysis Results

Let's look at the regression models that were run:

First, the data was split as follows:

	Counts	
Training	51935	60.03977
Validation	17381	20.09341
Test	17185	19.86682
Total	86501	

Training and validation regression results for the four feature sets were:



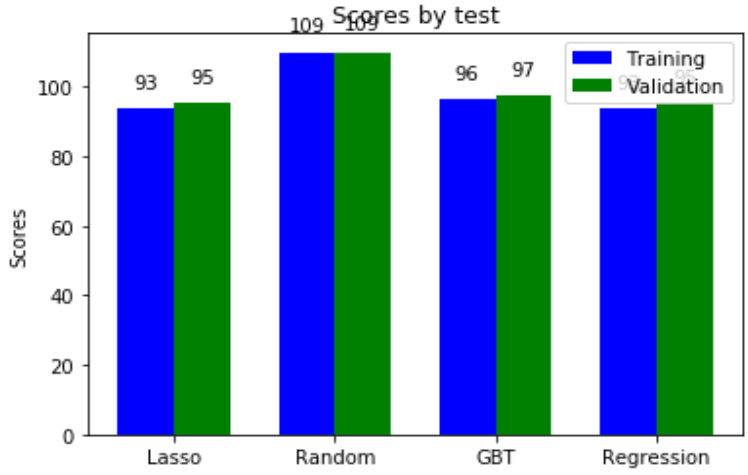
	MAE	RMSE
Regression train	50.13	93.698
Regression val	50.96	95.21
850 Train	50.32	93.87
850 val	50.93	95.27
500 Train	51.92	96.01
500 Val	52.35	97.29
100 Train	63.83	111.56

100 Val	63.67	112.38

We have tried several models for regression. Listed here are the full feature set after feature selection and one-hot-encoding were completed, plus regressions run with 850, 500 and 100 features feature sets that were selected by the Chi-square feature selector. The Chi-square feature selector works by determining variable independence. Variables determined to be independent from the model are discarded.

Note that the full feature set was the best performer, with an RMSE of 92.92. This is a surprisingly good value, considering how "dirty" the data was before processing. The mean was 158.95, the standard deviation was 140.98, and the validation RMSE was 95.93, which is well within one standard deviation of the Training set. Also note that the set of variables in the full regression model explain 69% of the variation in the response variable "price".

Results from other algorithms



	MAE	RMSE
Regression Train	50.13	93.698
Regression Val	50.96	95.21
Lasso Train	49.92	92.927

Lasso Val	50.931	94.932
GBT Train	53.53	96.28
GBT Val	54.81	98.92
RF Regression Train	59.98	109.46
RF Regression Val	60.20	109.83

A couple of things stand out from the results. First, we note that the best performers on the validation data were the multiple regression and the Lasso regression. It is interesting to note, however, that while lasso did better with predicting on the training data as well as the validation data, it overfits slightly less than the multiple regression did, as well. It is possible that since the L1 normalization can make variable coefficients to be exactly zero, the reduced number of predictors in the Lasso model reduces variance, or increases bias and thus reduces overfitting.

The Lasso regression turned out some interesting data on the variables and their relative importance to the model. For example, there are 36 out of the 1333 that L1 Normalization reduced to exactly zero. 32 of those were zip codes, one was a state, we also had 'tub_with_shower_bench', 'sentiment_market', and one of the many property types. On the high positive end of coefficient values we have another property type, room_type, bedrooms, and a lot of zip codes. Actually the top one was a zip code. On the high negative end of that, we have mostly zip codes. This tells us that the old real estate cliche "the three things that matter in property are location, location, location" is likely to be true.

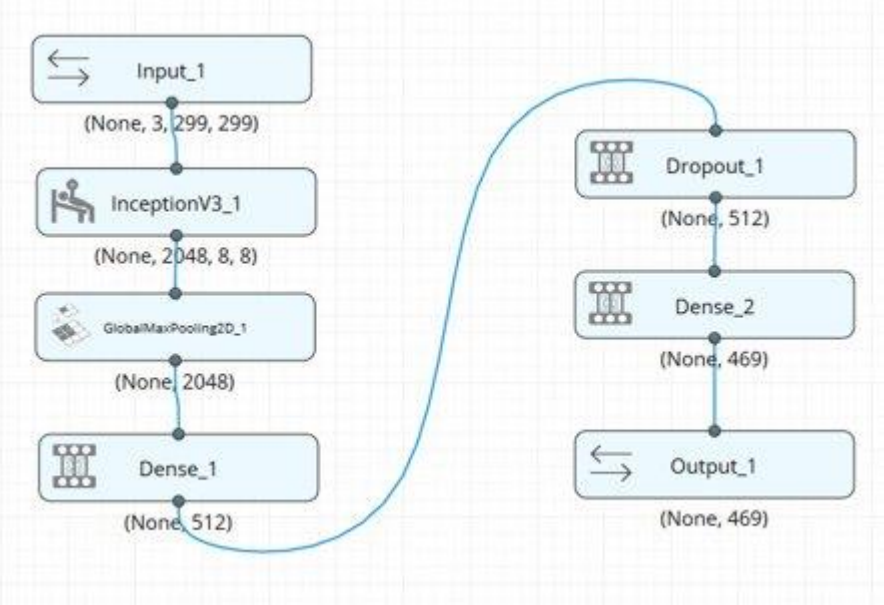
Results from CNN model with image data

We ran into a few challenges with the image processing. A stand alone PC running in pseudo distributed mode cannot handle the load. The Databricks community edition lacks file storage for all those images. There were a few things we tried to analyze the image data:

- Using Spark's deep learning package that uses Keras to create a few regression models using convolutional neural networks.
- Using deep Cognitions's Deep Learning Studio to build the CNN model for a dataset consisting of 15000+ records.

We looked into sample projects online for Deep Learning Studio. It cannot support a regression model with a dynamic numeric range of outputs. Furthermore, each image will be downscaled to acceptable Keras based transfer learning models like Inception-V3, ResNet, etc., resulting in 299x299, or 224x224 sized images, where actual image information can be lost. This could hamper the model's accuracy. A solution could be to turn it into a classification problem. Since Deep Learning Studio supports uploading

of customized data sets to create various deep learning models, we used Pandas scripts to create the “images based on price’ folder structure needed for the images to be uploaded to the Deep Learning Studio. The New York City data contains 15537 records with 469 unique values for price. Here is the model from Deep Learning Studio:



The problem was we would get unexpected shape error regardless of the dropout rate we used.

Conclusions

AirBnB is a way for people to turn their unused rooms into a bit of money. It also is a good source of data to have fun with. Our adventure began by downloading the nearly 1GB size file. We faced several challenges along the way in gaining insight from the data. One could say the data was structured, or tabulated in a neat comma delimited format. But it was not. Most of the rows had more elements than there were elements in the header. It took us quite a while to realize that was what was going on with all the missing values and data type errors while trying to change the schema of the dataframe.

What we came up with was a regression problem with a continuous dependent variable [‘price’]. The distribution was non-normal and possibly exhibited unequal variance, or heteroskedasticity. Either of these will invalidate a regression model, which is why we had to do a log transform of the dependent variable [‘price’]. As it turns out, working with this was a bit tricky with the pyspark mllib, until we figured it out, that is.

We ran regressions with three different sets of features along with the full feature set. The features were chosen through feature reduction with a chi-square algorithm. The full feature set was the best

performer. Then we performed three other tests with the full feature set. Of all the tests, the best overall performer was the Lasso regression. It did better on the training set, the validation set, and it performed better on overfitting, as well.