Predicting House Prices in Ames Iowa, Project 1 for Big Data & Analysis

Henry Stuklis (a1706223)

August 29, 2019

Abstract

In this report I introduce several regression models for predicting the sale price of houses in Ames Iowa. I firstly explain what a Kaggle competition is and how the participants are ranked against one another. I then detail my workflow for solving the Kaggle competition problem given the training data. I describe the data, talk about the types of regression that I fit to the given data and why these types of regression models suit this data. I describe the actions taken to clean the data and finally apply the models. I discuss the results of the regression models and find that Lasso, Elastic Net and Gradient Boosting Regression give the least error in prediction.

1 Introduction

This project for Big Data & Analysis required us to complete a Kaggle competition. Kaggle is a website that lists interesting data sets along with problems to solve using that data. Mainly these problems involve building a statistical model to perform prediction. On the Kaggle website you are allowed to submit your statistical models predictions and see how much error it has in predicting the true known values. Kaggle then displays a leaderboard of all submitted models and the main aim of competing in these Kaggle competitions is to try and rank as high as possible. The less error present in a model the better and more robust the model is in prediction. So the workflow pipeline for a Kaggle competition is similar to as follows.

- Clean and pre-process the base data (given as training and testing data)
- Complete some feature engineering (transforming, dropping or creating variables)
- Pick a reasonable statistical model to predict the required values
- Train the statistical model on the training data
- Analyse the error in predicting the testing data (either by submitting on Kaggle or calculating the error yourself)

My detailed and commented Python code that goes through this entire problem pipeline can be found attached separately to this report in house-prices-code.ipynb.

In our case we are given that we need to split the provided train.csv into 70% training data and 30% testing data. So the error will need to be calculated separately. Also as we are effectively reducing the amount of data that our statistical model will be trained on I have chosen to not submit on Kaggle. As my model wil have less training data than all the other submissions on Kaggle and will not be able to be ranked on the same scale due to this.

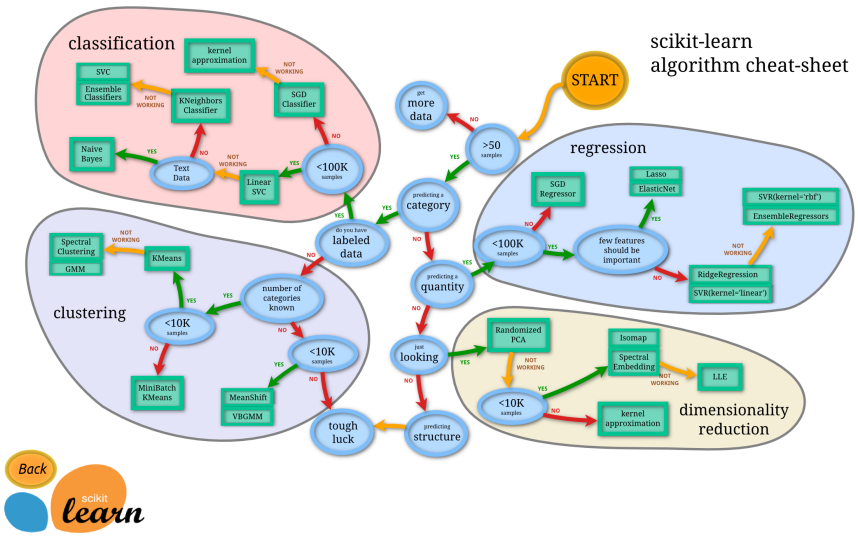
As listed on this Kaggle competitions webpage, the purpose of this project is to develop a predictive model to determine the house sale price based on the provided data set. The data set contains 80 different variables of 1460 different house sales in the township of Ames in the USA state of Iowa. These 80 variables range from physical numerical aspects of the house like the total squared footage of the basement of the house to categorical variables such as what type of electrical power setup the house has. So in effect based of the data set we are creating a statistical model for the sale price of a house restricted to the township of Ames Iowa. The data as provided by Kaggle comes in the form of unclean data in that it has not

already been pre-processed and there may be inconsistencies that need to be remedied. For a full description of how the data was recorded we can refer to data_description.txt as provided alongside the data.

The type of statistical model required to solve this problem is known as a regression model. As we are predicting a numerical house price which could potentially take any value from 0 to the highest house sale price and every number in between. The following Methodology section will discuss the choice of regression models used to price this sale price.

2 Methodology

scikit-learn is a Python package that provides a large number of regression models for use on Python data-frames. From the very start of my code I planned to implement a regression model from this package but the huge number of options overwhelmed me. So I referred to a great flowchart I found on the scikit-learn documentation website. This machine learning flow chart helped me implement the best algorithm I could for my data.



So from the start we travel down into the predicting a quantity bubble. We have a lot less that 100k samples so we travel into the few features should be important bubble. As we are predicting house sale price we expect a small number of the recorded variables to be important for prediction. So I chose to implement these two listed algorithms: Lasso and Elastic Net. I also chose to compare these with the other algorithms comping from the no answer to the few features bubble: Ridge Regression, SVR (both linear kernel and rbf) and an Ensemble Regressor called Gradient Boosting Regressor. The three of these implementations that had the best performance in the resultant

error were Lasso, Elastic Ridge and Gradient Boosting. So I will discuss these three implementations and how they work.

Lasso is actually an acronym that stands for Least Absolute Shrinkage and Selection Operator. Lasso is a type of regression that encourages a sparse regression model to be built. Meaning that it is one that is effected by only a few significant predictors. This makes the model well suited for complex models with a high amount of dependence/correlation and multi-colinearity. As the sparseness minimises the effect of the predictors that have high correlation with the most important predictors. So Lasso is a great model that we can just throw all of our data into and it largely does the model selection (selecting the most important predictor variables) on its own. This makes Lasso a great algorithm for this data set as we can expect things such as the total above ground squared footage and the number of rooms present in a house to be very correlated. So Lasso will simplify our data greatly to perform better prediction of sale price.

Elastic Net is a regression model that combines the properties of Lasso with another form of regression called Ridge Regression. Ridge Regression is very similar to ordinary least squares estimation (linear regression where we find a line of best fit). Just that the coefficients are restricted in size. Elastic Net is something like a weighted average of a Lasso model and a Ridge Regression model. So it shares many of the good properties that we have already discussed about Lasso regression.

Gradient Boosting Regression is an ensemble regression method. This is a quite complex style of regression where the regression model has a learning algorithm imposed on itself. It employs the use this learning structure by constructing several independent estimators and then repeating the same process over the learning algorithm resultant estimators. In particular Gradient Boosting is an ensemble method that builds these estimators sequentially eventually constructing a powerful regression model by constantly improving a weak regression model.

The following experiment section will go through the implementation of the above algorithms on the processed data and analyse the resultant error.

3 Experiment

Firstly, I loaded the data contained in train.csv into Python in the form of a pandas data-frame. I then checked the column labels and discovered that the variable `Id` was doubling up on the indexing of the rows as a pandas data-frame. So I completely removed it from the data.

Then I checked for the percentage of missing data in each of the recorded variables. I found that a large number of the columns contained Python reserved 'NA' values. In Python 'NA' is an alias for null or empty so I had to go through each categorical variable that had been labelled with an 'NA'

and change each to a different label. As for the missing data present in the `LotFrontage` variable I decided that in order to minimise the impact of these missing variables I would replace each missing value with the median `LotFrontage` in the data. This was to ensure that the `LotFrontage` would not skew the regression model too heavily. As for the other missing numerical variables I replaced each with zero value as in each case it represented what the data was describing with a missing value. As for missing categorical variables that did not have an 'NA' variable recorded I decided to fill in the missing values with the most common level of that categorical variable. I also changed the labelling of some categorical variables that had numerical levels present to ensure coherence when I would later fill each categorical level with dummy indicator variables.

Next I decided it would be best to normalise all numerical variables (x) present in the data by applying a $\log(1+x)$ transformation onto each. This would ensure a better prediction as each of the discussed regression models requires approximately normal numerical variables for best performance.

I then inserted dummy indicator variables for each level of each categorical variable. This is done to allow every single variable present in the data to be accepted into a Python regression model. As it requires that all data be numeric.

I then sliced the data in train.csv into the required 70% training data and 30% testing data.

Finally I fitted each of the six chosen regression models on the training data and evaluated the root mean squared log error (the error type used on the Kaggle leaderboard) of each of the six models. This resulted in the Lasso, Elastic Net and Gradient Boosting Regression performing best in terms of error. The exact errors can be found at the end of my attached code.

4 Discussion & Conclusion

If I take my best error and compare with those errors listed on the Kaggle leaderboard I rank in around the top 2000. I think that given I have only used 70% of the data they used to train their models on the leaderboard that this is quite a good result. If I used the 30% testing data as training data I think I would perhaps make it into the top 1000. Around this area on the leaderboard the differences between ranks is extremely small, usually a four or five decimal point difference. So I think that my model ranks around the majority of those good models submitted on Kaggle.

I believe there is still a great amount of improvement that could be applied to my regression model. I did some initial experimenting with a regression model called XGBoost which is separate from those included in scikit-learn. It has a great deal of parameters that control the boosting

method and I think that chosing enough correct parameters could improve the accuracy of the regression quite a bit. There are also lots of different types of model stacking and averaging techniques that I could apply to several regression models to create a new regression model.

So I believe that the regression model was successful in predicting the sale price of houses in Ames Iowa to a relatively high accuracy. There is still huge room for improvement in my model but it does a great job with the limited training data.