Homework 2: Exploratory Data Analysis in Python

Deven Shah 111482331 dsshah@cs.stonybrook.edu

I implemented Linear Regression, Lasso Regression, Support Vector Regression, and K-Nearest Neighbor Regression. I predicted log error using the above 4 models using the same attributes. I found that Lasso Regression and Support Vector Regression gave lesser Squared Mean Error than Linear Regression and K-Nearest Neighbor Regression.

My favorite models out of these is Linear Regression.

I implemented Linear Regression in the following way.
First I decided to remove the columns which contained categorical data like architecturalstyletypeid, airconditionertypeid etc. Upon deleting those I was left with 31 columns. Then I thought of deleting the NaN values in those columns. I checked the length of columns if the NaN values are deleted. Then I chose the columns having around 89,000 records after their NaN values are deleted. I was left with 8 columns. I created a new dataframe having these 8 columns. I deleted the records even if atleast one of the column values contained NaN values. After this I was left with around 88,000 records. I split the dataset into training and testing set, the size of testing set used is 5% of the entire dataset. Using these 8 columns I made a linear regression model to predict the logerror. I removed the attribute columns one by one whose coefficients were very low in the range of e-10. Also, I kept a look at the squared mean error value. Similarly, I kept on reducing the columns. Eventually, I was left with 4 feature columns namely, calculated finished square feet, fips, tax value dollar cnt, and land tax value dollar cnt. The squared mean error value was the least for the combination of these 4 features which was 0.0169875170064. I ran the test cases which the Zillow Prize Challenge provided and submitted my predictions. I got a score of 0.0648942 and was ranked 2002 on the leaderboard of the Zillow Prize Challenge.

For predicting the records for the parcelid in the submission file, I did inner join of submission file with the main file which consists of the entire data. I replaced the NaN values in a particular column with the mean of the column.

After implementing the Linear Regression Model, I implemented Lasso Regression, SVM and KNN. Lasso Regression gave a bit lesses squared mean error value of 0.0169827951958 and Support Vector Regressor gave a bit higher squared mean error value of 0.0175178710114 than Linear Regression Model on the testing set. However,

upon submitting the predictions made by these models, the score obtained was very less particularly 0.0691738 for Lasso Regression and 0.0652292 for Support Vector Regressor. R2 score for Lasso Regression was positive 0.00340191595686 though close to zero which was higher than that for Linear Regression, 0.00312482685843. However, for Support Vector Regressor r2 score came out to be negative particularly -0.0279978345822.

I also implemented the K-Nearest Neighbor Regressor. The mean square error was 0.0170341793884 and r2 score was 0.000386547326443 close to 0. I found these after keeping the number of neighbors parameter equal to 1080. I was surprised to see that the r2 score below 1080 or above 1085 negative.

Apart from these 4 implementations in Python, I also implemented Bayesian Regularised Neural Network in R language. I got higher squared mean error value of 0.0258193 than the other models. I submitted the predictions made by this model, and the score I obtained was 0.0648916 while for Linear Regression I received the score of .0648942. This model improved my rank by 10 places making me stand at 1998 rank.

The independent and the dependent variables are normalized in the Bayesian Regularised Neural Network so that the input space and the output lies between [-1, 1]. This is required because it uses Nguyen Widrow algorithm for initialization of coefficients of features for which the input space should lie in the range [-1, 1]. The formula for normalizing the variables is :

$$z = 2 \times (x - base) / (spread - 1)$$
$$where: \ base = min(x)$$
$$spread = max(x) - base$$

The total number of parameters to be estimated by a neuron are 1 + 1 + number of independent variables. The first one is for the weight of the sigmoid or activation function and the second one is for the bias of sigmoid or the activation function for a hidden neuron. Also each neuron has to estimate the weight of each independent variables. It uses

Nguyen Widrow algorithm applied to initialise the parameters to be estimated.
Initially random values are assigned to the parameters to be estimated for each neuron with given mean and standard deviation, in our case it is -0.5 and 0.5 respectively.
Then, scaling factor is calculated using formula

$$scaling\ factor = 0.7 \times (number\ of\ hidden\ layer\ neurons)^{1/n}$$
$$where,\ n : number\ of\ predictors$$

the input space is divided into equal intervals and each neuron is allocated a particular input space. A sequence is generated from -1 to 1 having intervals equal to number of hidden layer neurons.

Normalization factor for every neuron is calculated using the estimated weight of each independent variable or feature using the formula:

$$normalization\,factor = sqrt(sigma(weight^2))$$

The estimated weights are updated using formula:

$$wi = scaling\,factor \times wi / normalization\,factor$$

bias of the sigmoid function is also updated using the formula,

$$biasi = scaling\,factor \times sequence\,number\,of\,ith\,neuron$$

After the sigmoid weight, bias and the weights of the independent variables are initialized, it is then passed to the neural network to predict the log error. After the log error is predicted, the sum of squares error(Ed) and sum of network parameters(Ew) are calculated using formula,

$$Ed = \Sigma (y_{pred} - y)^2$$

$$Ew = (weight\,of\,the\,sigmoid)^2 + (bias\,of\,the\,sigmoid)^2 + \Sigma(w_i^2)$$

$$where\ w_i\ is\ the\ weight\ of\ each\ independent\ variable\ or\ feature$$

The formula for prediction is:

$$y_i = g(x_i) + e_i = \sum_{k=1}^{n} w_k g_k(b_k + \sum_{j=1}^{p} x_{ij}\beta_j) + e_i$$

$$where:\ g(x)\ is\ the\ activation\ function\ of\ each\ neuron$$

$$g(x) = (e^{2x} - 1) / (e^{2x} + 1)$$

The cost function is defined as

$$F = \beta \times E_w + \alpha \times E_d$$

$$where\ \alpha\ and\ \beta\ are\ cost\ function\ parameters\ defined\ as:$$

$$\beta = (number\,of\,independent\,variables - number\,of\,parameters\,to\,estimate) / (2 \times sum\,of\,squares\,error)$$

$$\alpha = (number\,of\,parameters\,to\,estimate) / (2 \times E_w)$$

To reduce this cost function gradient descent is used until the gradient becomes minimum or for the last three iterations the gradient doesn't change or until it completes all its iterations.
For each iteration prediction of dependent variables is done, sum of square error and sum of network parameters are calculated, using which cost function is updated.

I found the Bayesian Regularised Neural Network quite interesting though it only improved my rank on leaderboard by 10 places.

Interesting things I learned

I learned a many new things in the homework. Apart from various modelling techniques I learned how to clean the data. I realised the importance of data visualization. On plotting various graphs, I came to know about the outliers because of which my model did not perform well. I encountered upon removing just one outlier, my model performed better than previously when it had the outlier. I was surprised to see some models gave less square mean error value but perform bad while submitting on Kaggle challenge while models giving more squared mean error value performed better.

Summary

1. Features used :
    a. Calculated finished square feet
    b. Fips
    c. Tax value dollar cnt
    d. Land tax value dollar cnt

Regression Model	**Squared Mean Error Value**	**R2 score**
Linear Regression	0.0169875170064	0.00312482685843
Lasso Regression	0.0169827951958	0.00340191595686
Support Vector Regressor	0175178710114	-0.0279978345822
K-Nearest Neighbors Regressor	0.0170341793884	0.000386547326443
Bayesian Regularized Neural Network	0.0258193	