Report: Final Project

Muhammad Fahad Asghar

Data Processing

Data was loaded into the notebook. In order to prepare the data, we start off by doing and **OLS fit** and look at the **R-sqr** value to get a general idea of our original data. In order improve our data we try getting **significant columns** from our data set by adding only the columns that have a **p-value** less than **0.5.** But we see that there is a decrease in R-sqr value. Probably because now we are left with less data points and are either overfitting or there is greater distance between our fit and the actual points. Letting this go we try and find interaction terms in the data set. This is done by using **5-fold cross validation** to set up a base value for R-sqr and then finding the top **10 interactions**. From these we add the top 2 and to the OLS fit again. And it is seen that the R-sqr value has increased. Being greedy, I try to find more interactions and add them to the model. After adding two more interactions and doing an OLS fit again I see a significant increase in the R-sqr. Now even though I feel there was no need to do so and would have probably saved me from the **over fitting** problem I faced later, I again in order to get better data again try to remove the columns with p-value less than 0.5 and seeing that there was not much compromise on the R-sqr value, I decide to go with it. Following this I split my data into test and training sets using **cross validation**.

Regression Models

I start off with the **Linear Regression** model. Using the processed data, I train the model using training data and then find predictions for training and test data. The training data prediction is around **50%**, which is promising but the test prediction seems to be quite low at this point. Since there is not much that can be done, I move on to my next model

The next Model is **Lasso regression**. Before performing Lasso Regression, I choose the parameters using **Lasso Cross Validation technique**. I create a range of alphas and carry out **10-fold with 100000 iterations.** Finding the **optimal value lambda**, I use it as my parameter and do a **Lasso fit**. It can be seen that the plot does not help much in getting the optimal value of the alpha. Hence, we had also calculated the optimal value by **Lasso Cross Validation**. Using that value, we carry out Lasso fit. Later, the MSE and the correlation is also calculated. MSE is although low but the correlation is somewhat less than what we had gotten with linear regression.

Following this I move onto my third regression model – **Random Forest Regressor**. With high hopes for this model I take great care to get as good parameters as I can. So first I fit the model with the **defaul**t values and immediately get a train correlation of around **50%.** This was quite promising since I was yet to improve my model. I do it in two way **– Cross Validation and Bagging**. But I will only talk here about **bagging** however, the code for cross validation is included in the file. First, I try to find the optimal value for **n_estimators**. I check it from **10 till 350** and find it to be optimal around **50**. Using this value, I continue further and find the optimal values for **max_depth** and **max_features**. Excited to get better results than before, I fit my model and find the test and train correlations. The train correlation was around 90% which got me excited, but the test correlation gave me a value close to **0**. Which I believe happened as a result of **over-fitting** the data. However, I tried improving my model in several ways by tweaking my initial data, but it gave me no promising result

One thing that surprisingly worked, but I did not include because it didn't make any sense. When I did **df1.append(df1),** essentially doubling by number of rows, I saw a drastic increase to a correlation of about **60%** with random forest regression. But since, including this did not make any sense, I did not go with is and declared **Linear Regression** to be the best model.

Classification Models

Next, for the classification model I start off in the same way with the processed data. I also **encode** the values we want to predict to 1 and 0. The first model that I use is **Logistic Regression**. I start off by doing a **count plot** for the values we want to predict to get a general idea of the trend. After this I fit my model and look at the correlation. To my dismay my train correlation is around **20%** but my test correlation is even **less than 0**. In a desperate attempt to improve my results for this model I try doing logistic regression with **10-fold cross validation**. We see a definite **increase** in the correlation and from recall from the **classification report**, it can be seen that the upcoming prediction is that of price **going up**. There is definite increase in the train correlation and also in train correlation, but sadly train correlation is still really low.

After this I try **SVM** model **with optimum parameters for the gamma and C** and after looking at the model score for training and test (both being above 50), I am convinced that this might be the **best model to use** here.

Moving on, I try **Random forest classifier** just as I did for the regression part. Following the same path, I arrived at the same end – **bad results**.

Hence, after this I was convinced that **SVM** here is the best model to go about.

Note: I added the two functions and they worked perfectly when I ran them with my df1 and df2 i.e just X_train and y_train as inputs. But when I tried adding some X_test, I kept getting dimensionality errors. If this happens, kindly try to run the model with just first two inputs and see that the functions do work. I just wasn't sure how to inculcate the third input. I assumed that it will come from the TAs when the program was tested. But I am not sure.