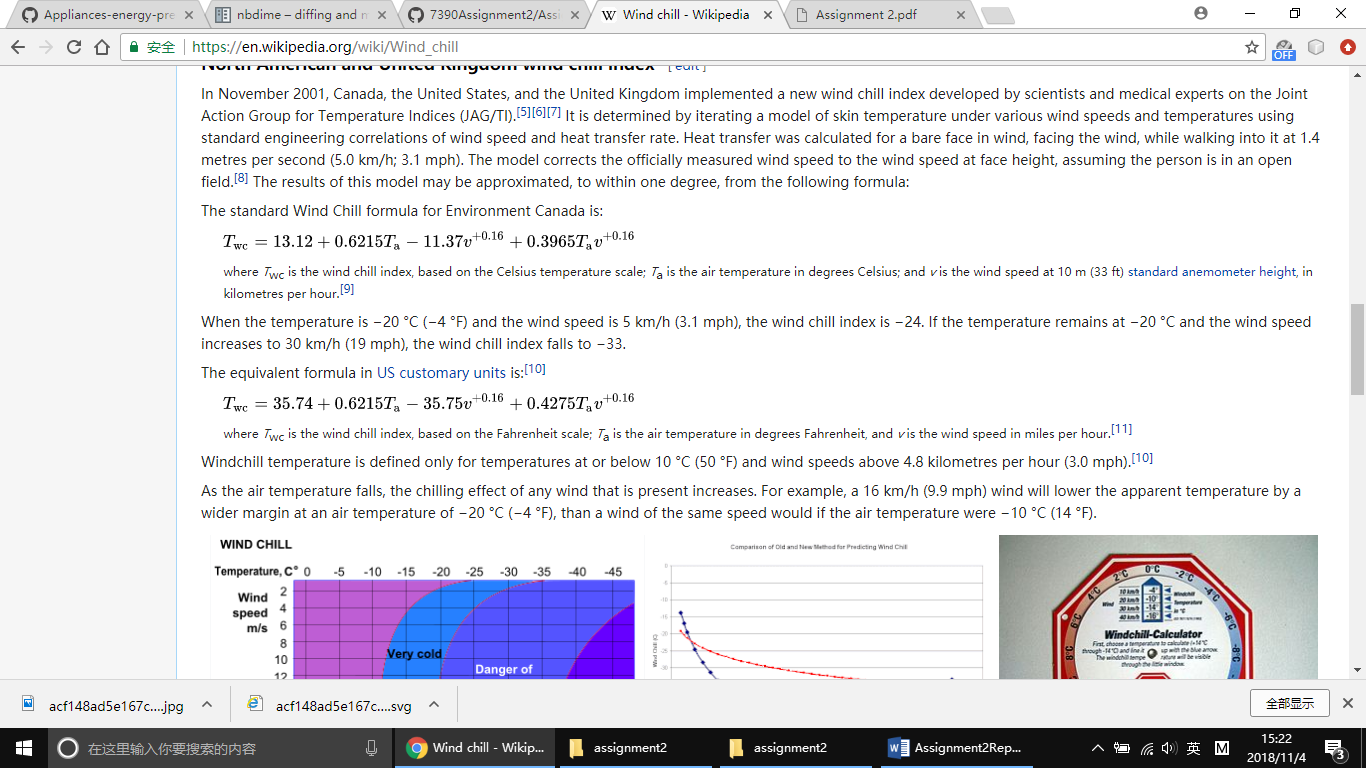
# Part3

In this dataset, there are 29 columns. Apart from appliance-target features, we still have 28 features. When describing the dataset, we found column data has format of data. In order to get more detailed information, we decided to generate new features below: day-of-year, month, minute, day-of-week, hour and day. In this assignment, we divided column date into those parts. Furthermore, we calculate minute-of-day. We found there are lots of data of temperature so we introduced the concept of wind-chill. Wind-chill is the lowering of body temperature due to the passing-flow of lower-temperature air. By using this, we can combine several columns of data into one column without any loss. Here is calculation of wind-chill:

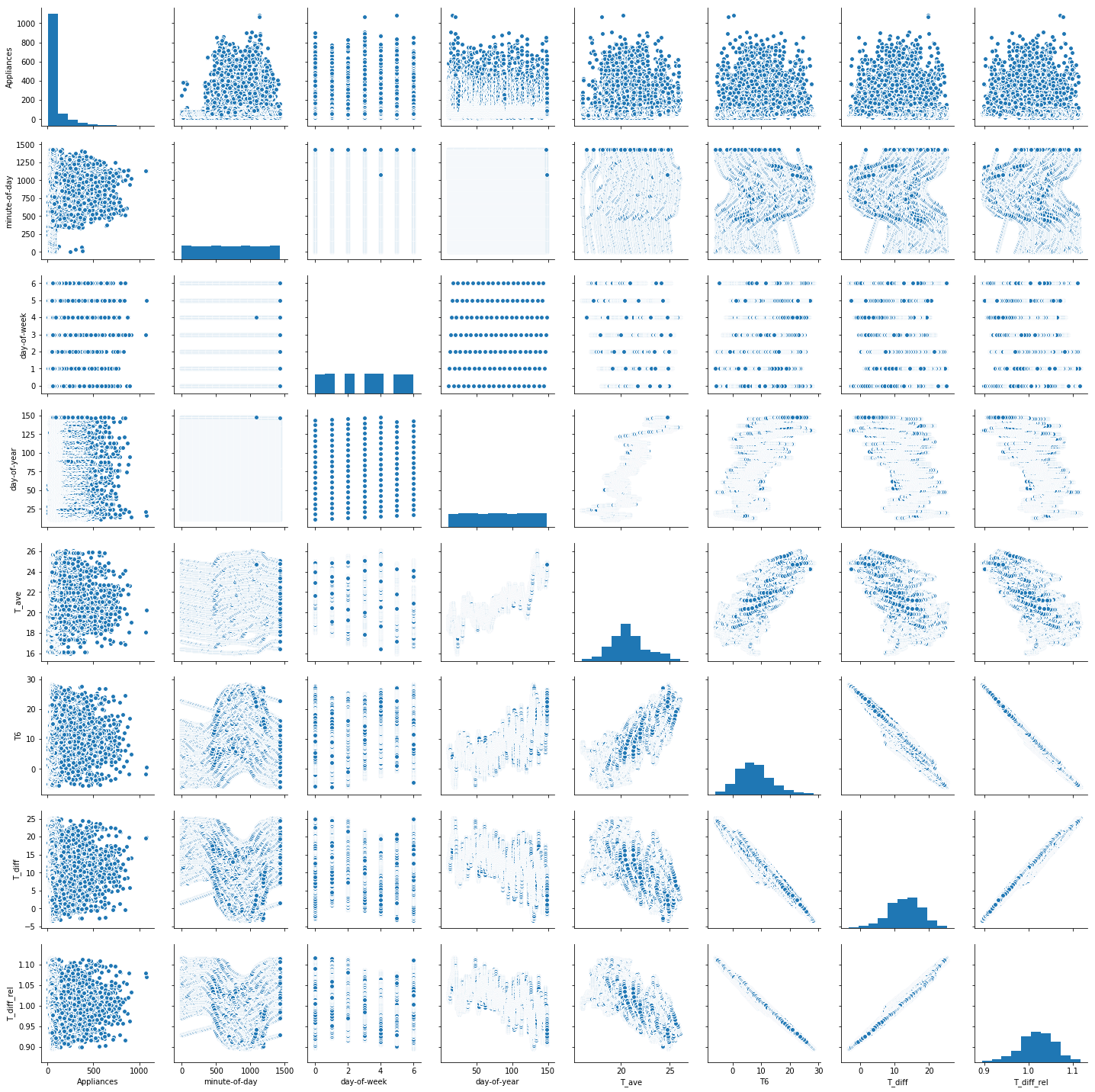


Here is the pair plot of features:

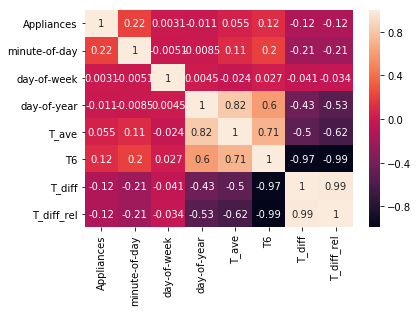


Also, we calculated average temperature of temperature columns except temperature outside because we already had wind-chill.

Here is pair plot of average temperature and other features:



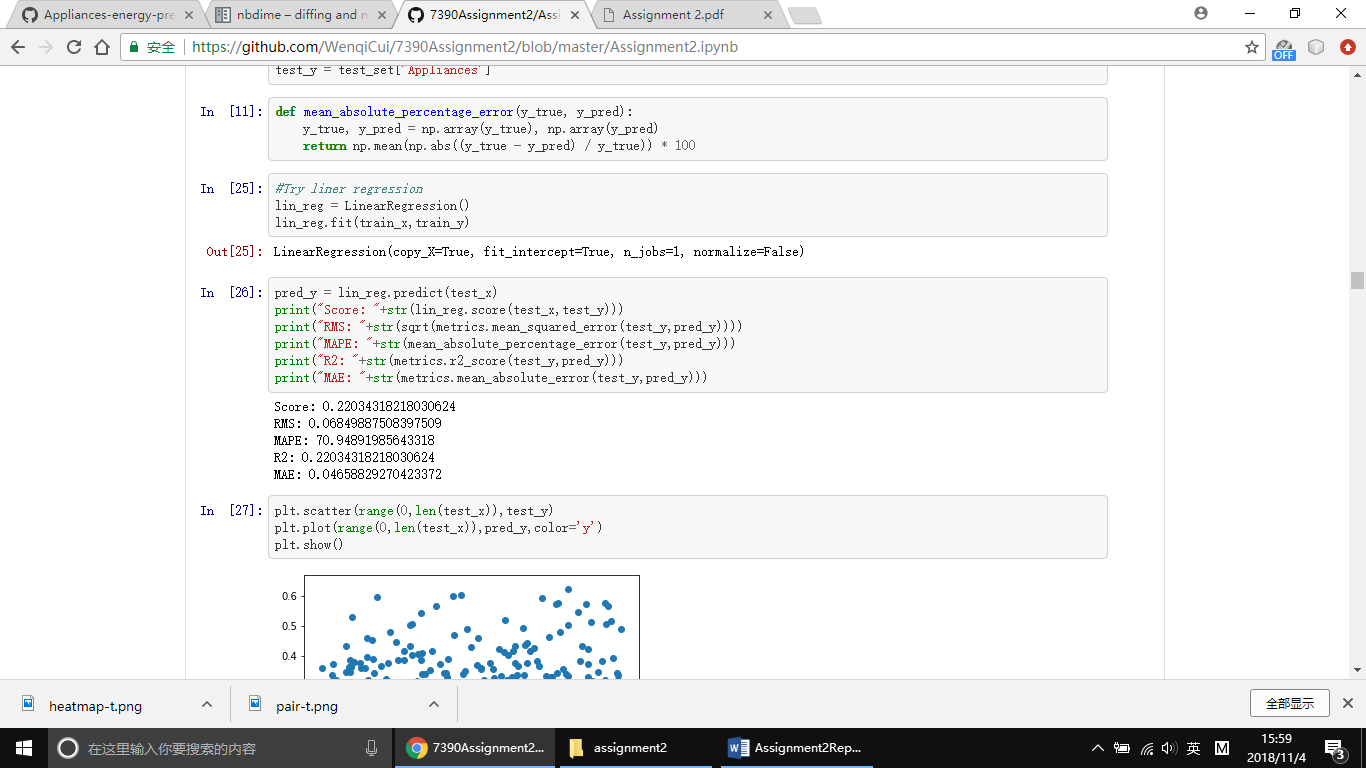
Here is heat map of those features:

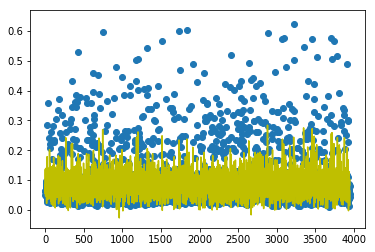


# Part4

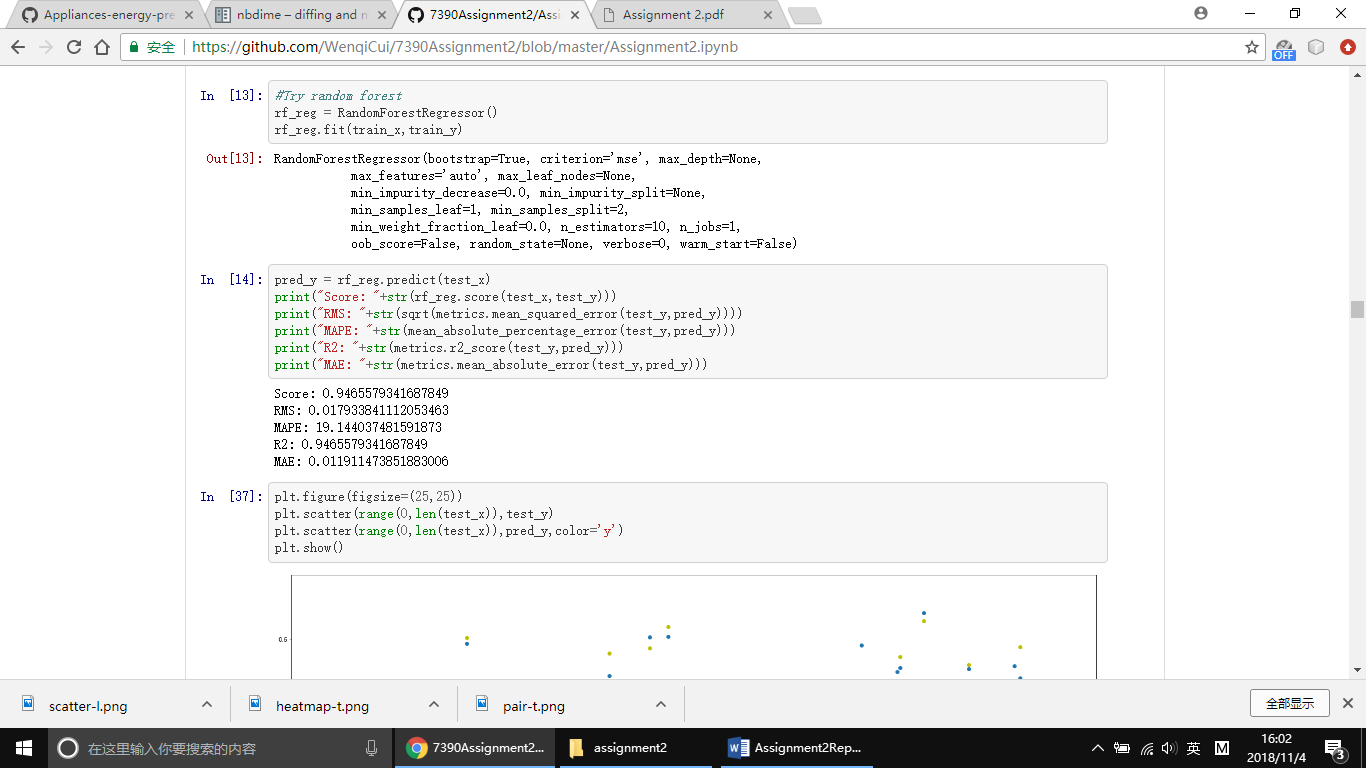
We combined the best performing algorithms into an ensemble, like algorithms with the best r-squared: Polynomial, Random Forest and decision trees The R2 (or R Squared) metric provides an indication of the goodness of ﬁt of a set of predictions to the actual values. In statistical literature this measure is called the coeﬃcient of determination. This is a value between 0 and 1 for no-ﬁt and perfect ﬁt respectively

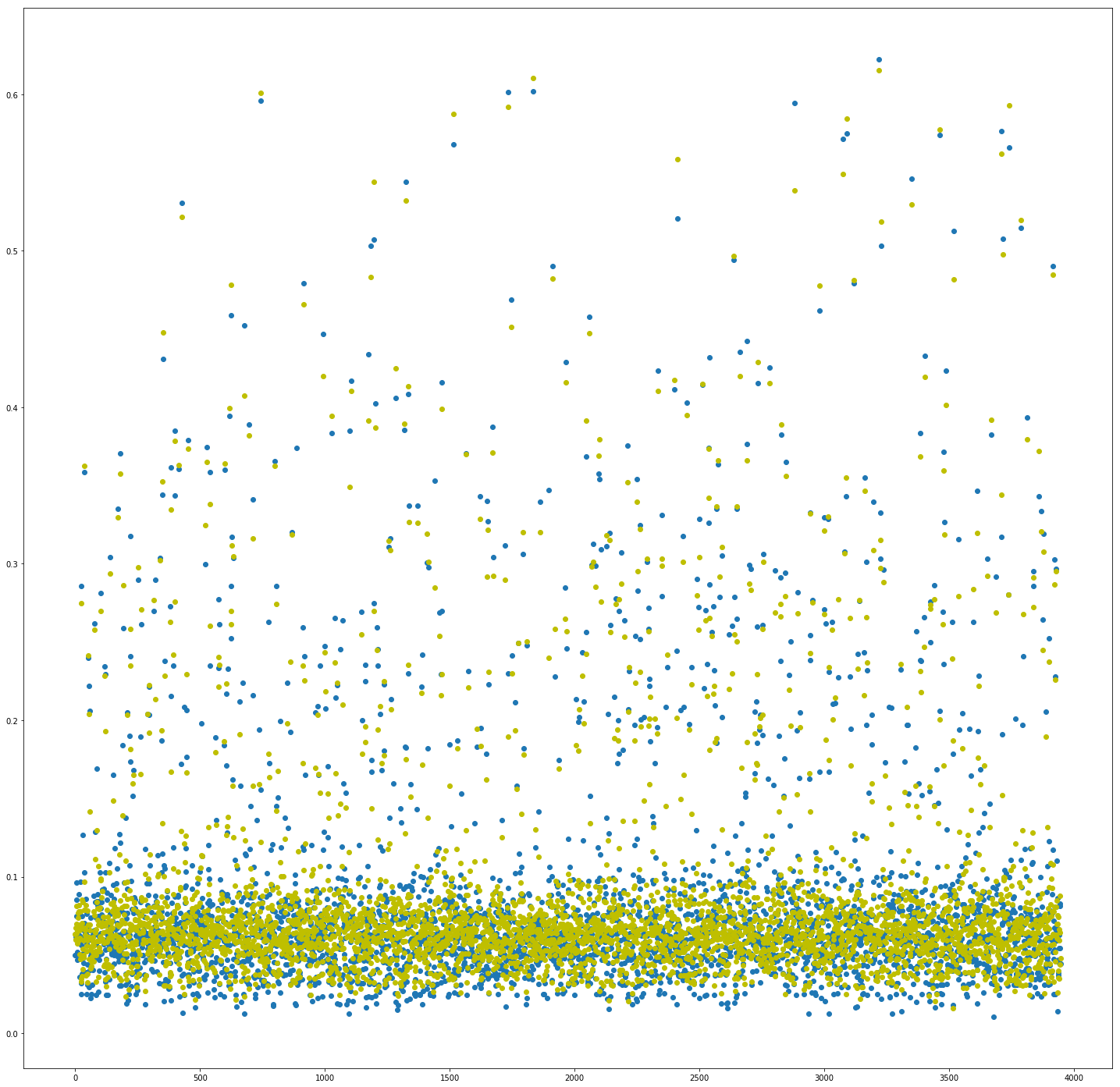
For Linear regression model, here are output results and scatter:



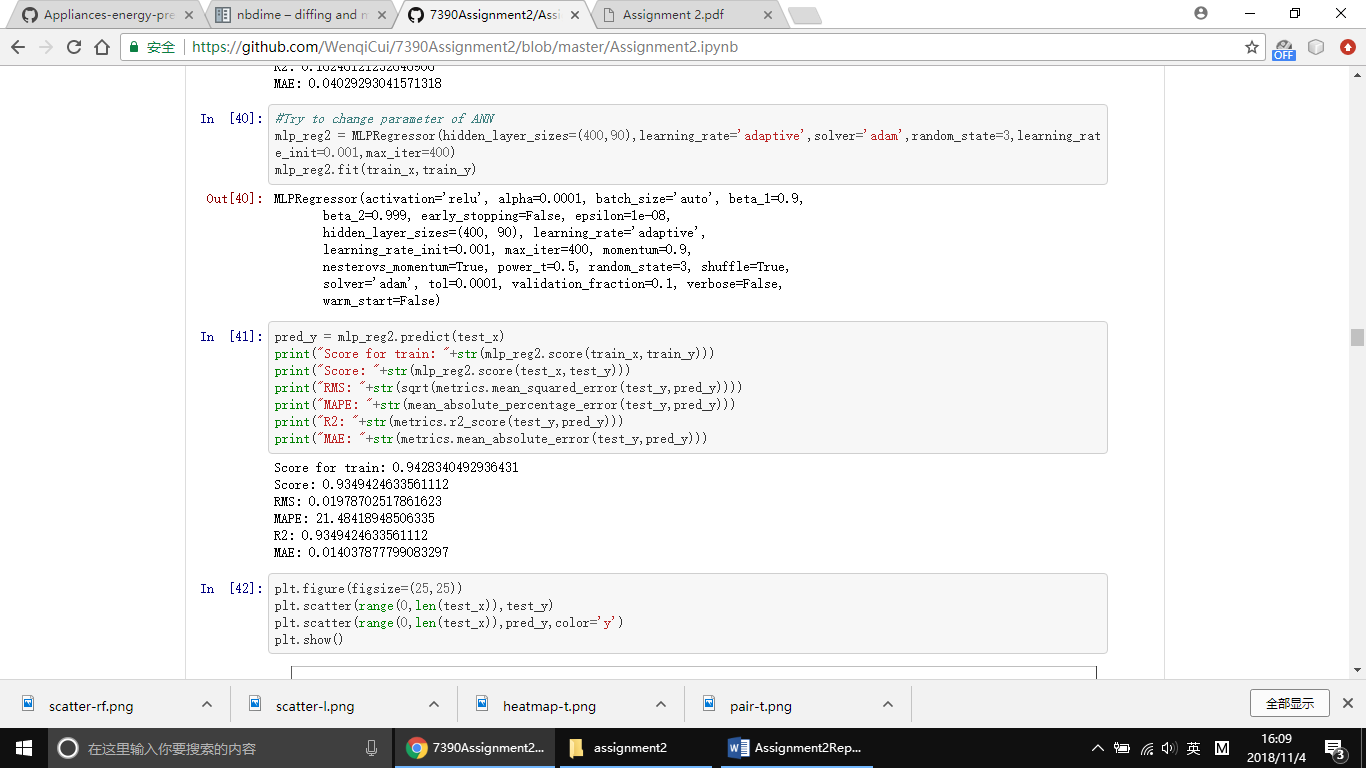


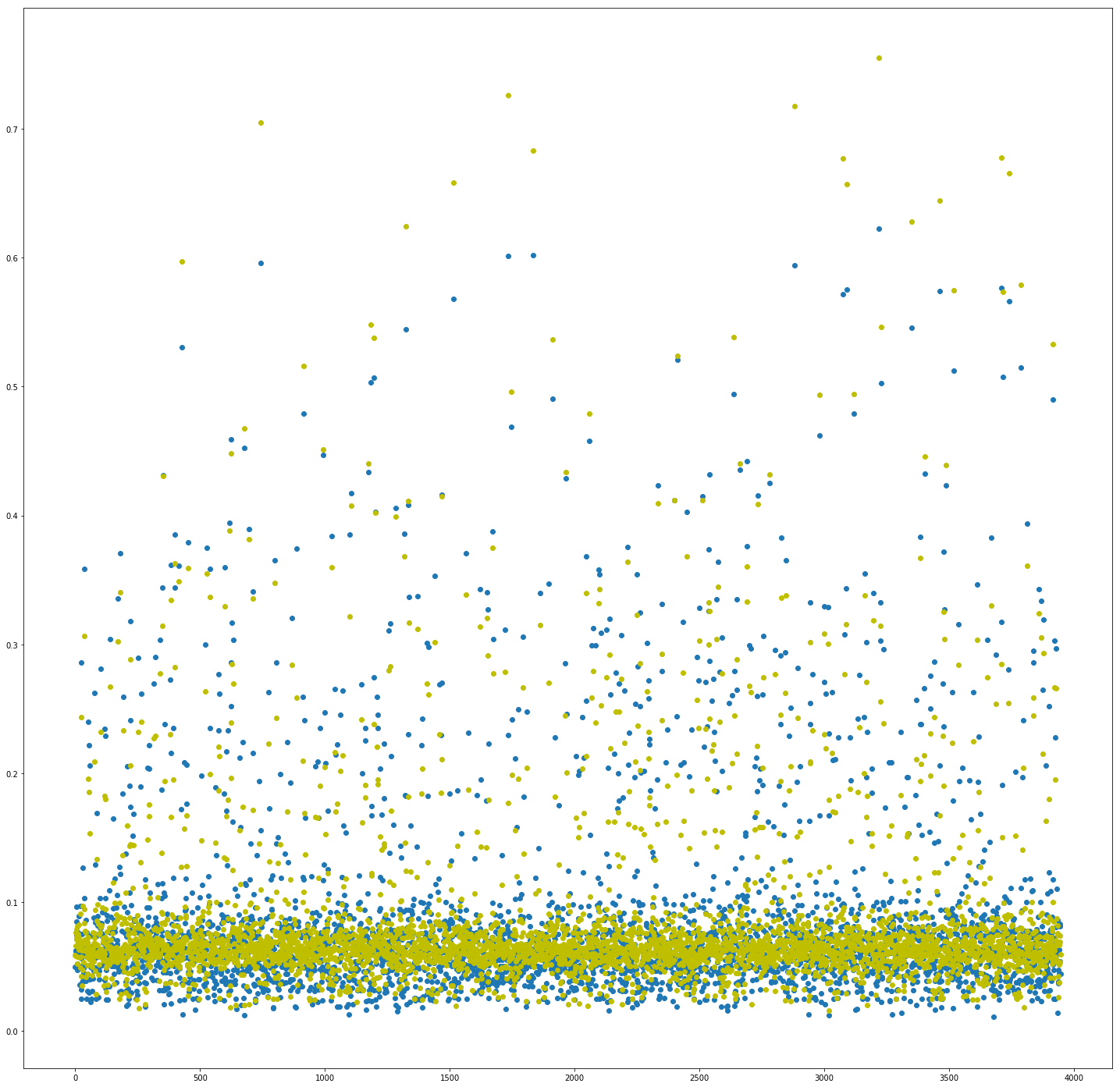
For random forest model, here are output results and scatter:





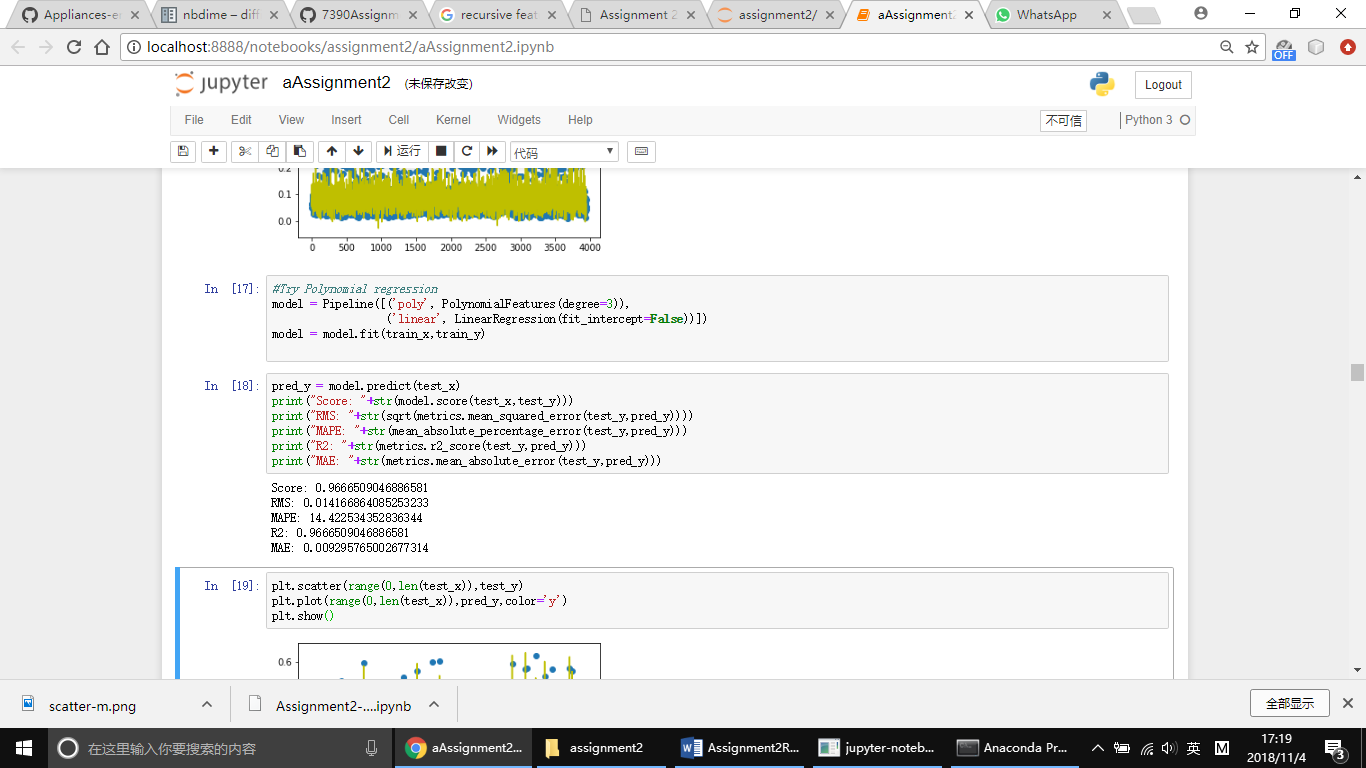
For neural network model, here are output results and scatter:

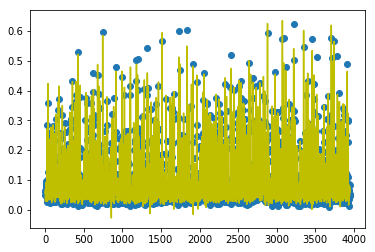




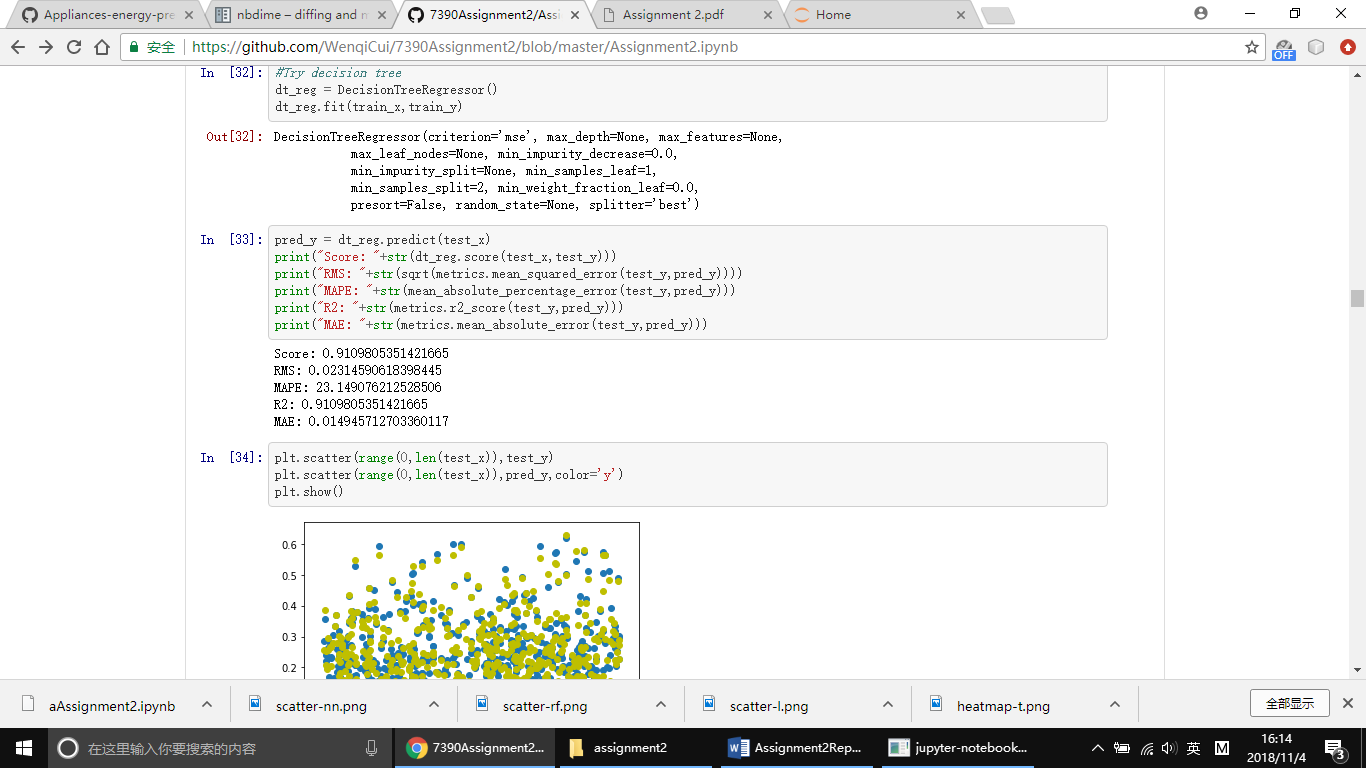
We also try out polynomial regression and decision tree model.

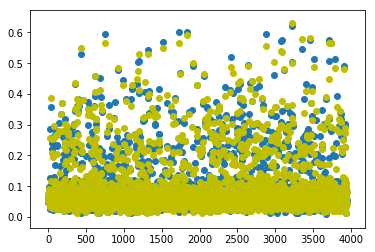
For neural polynomial regression, here are output results and scatter:





For neural decision tree, here are output results and scatter:

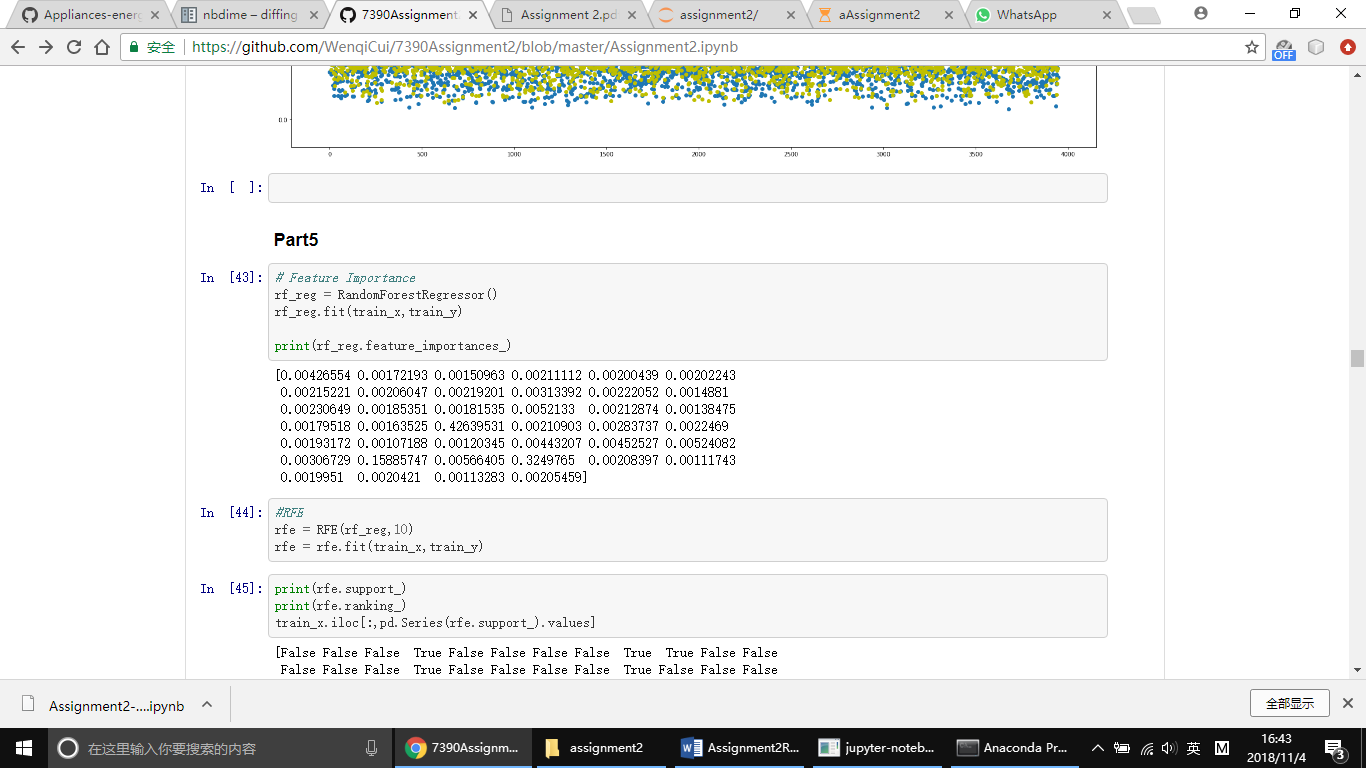




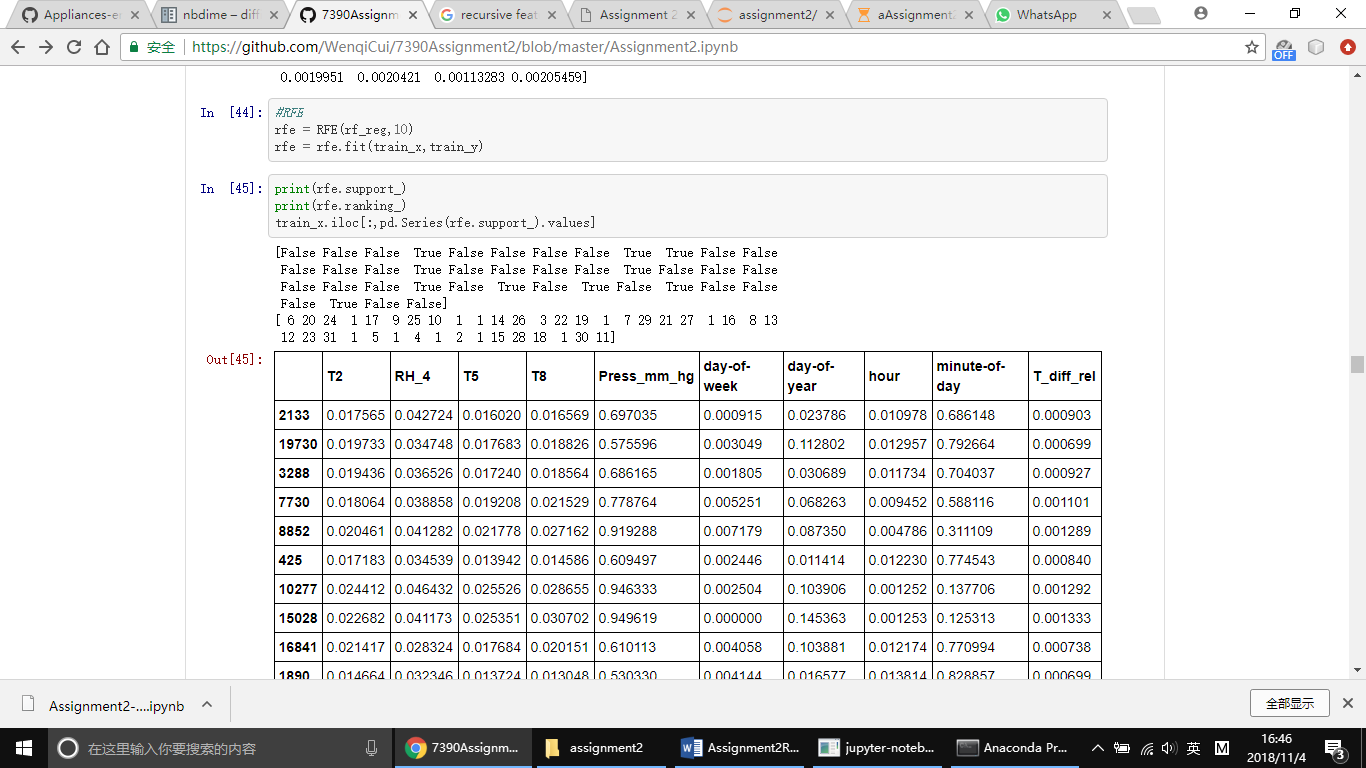
By comparing those output results and scatters, we found that the best model for this dataset is random forest model.

# Part5

By using RandomForestRegressor, we can calculate importance values for each features. The results follows:

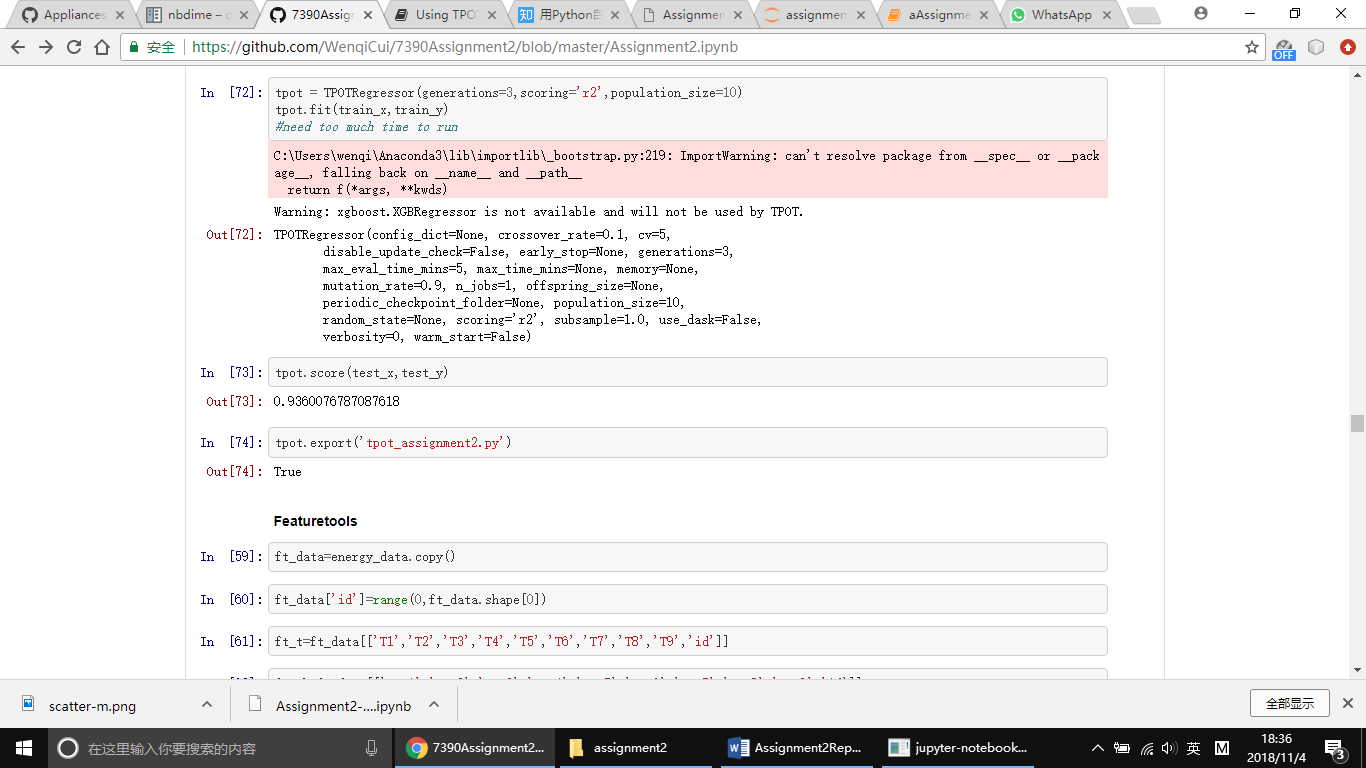


And recursive feature elimination can also provide Boolean value and rank order for each features:



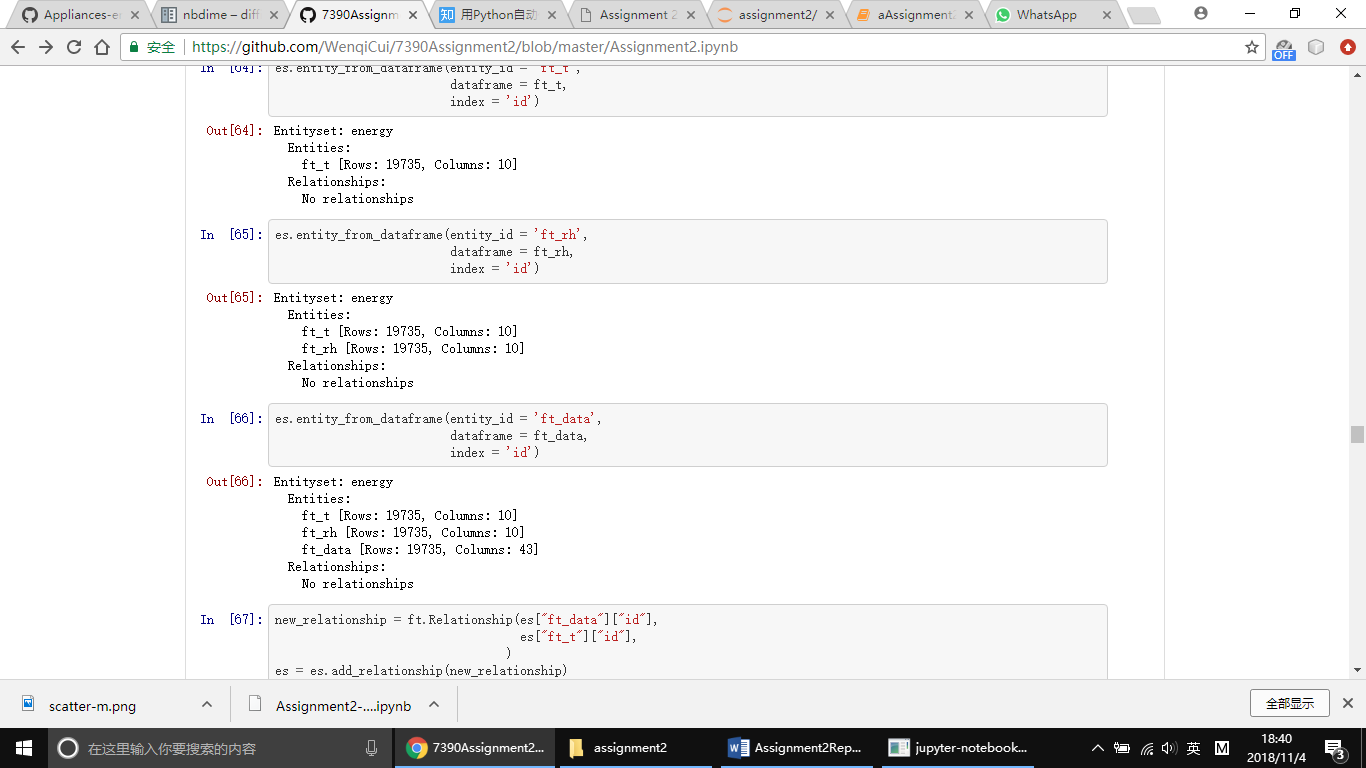
Tpot:

The fit function initializes the algorithm to find the highest-scoring pipeline based on average k-fold cross-validation. Then, the pipeline is trained on the entire set of provided samples, and the TPOT instance can be used as a fitted model. Result:



Featuretools:

Featuretools excels at transforming temporal and relational datasets into feature matrices for machine learning. By using featuretools, we can get relation between each feature.

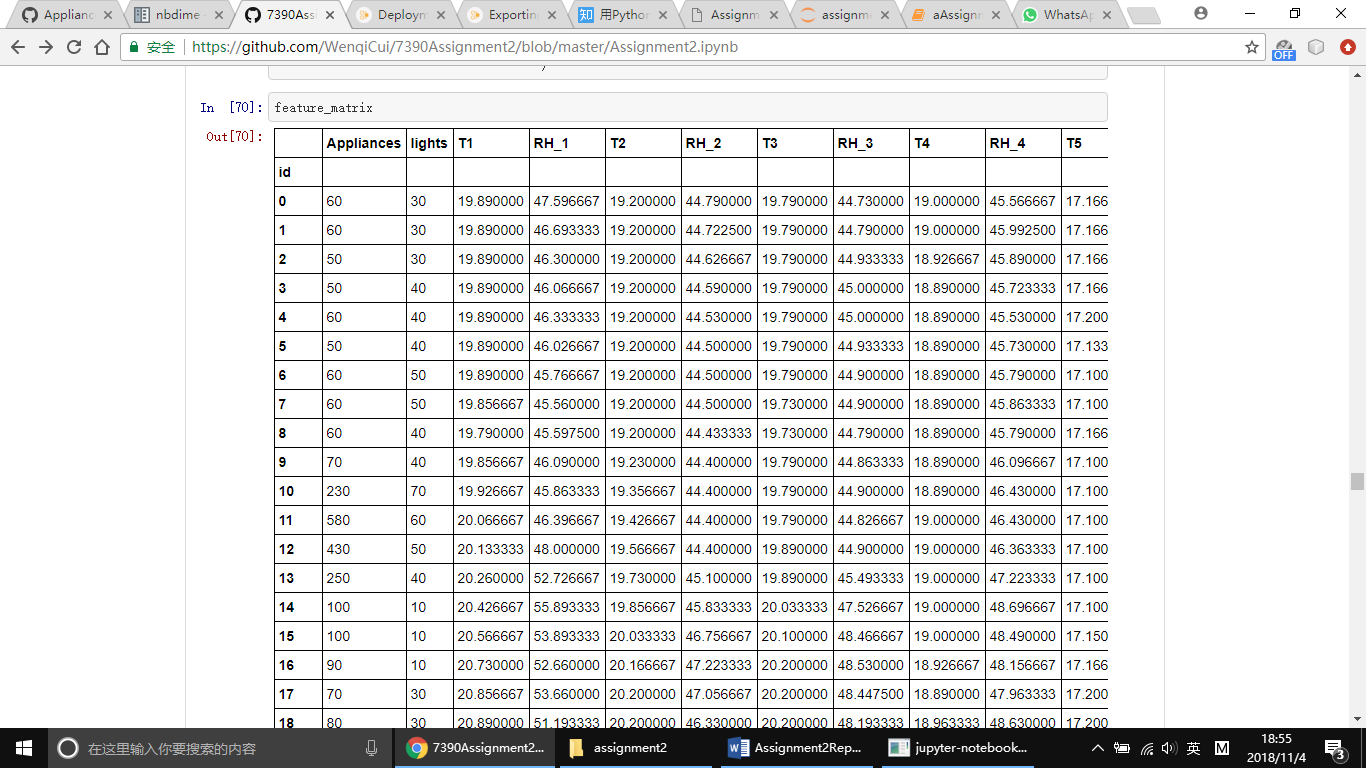


Deployment of machine learning models requires repeating feature engineering steps on new data. In some cases, these steps need to be performed in near real-time. Featuretools has capabilities to ease the deployment of feature engineering.

We built some features definitions using DFS. Because we have categorical features, we also encode them with one hot encoding based on the values in the training data.

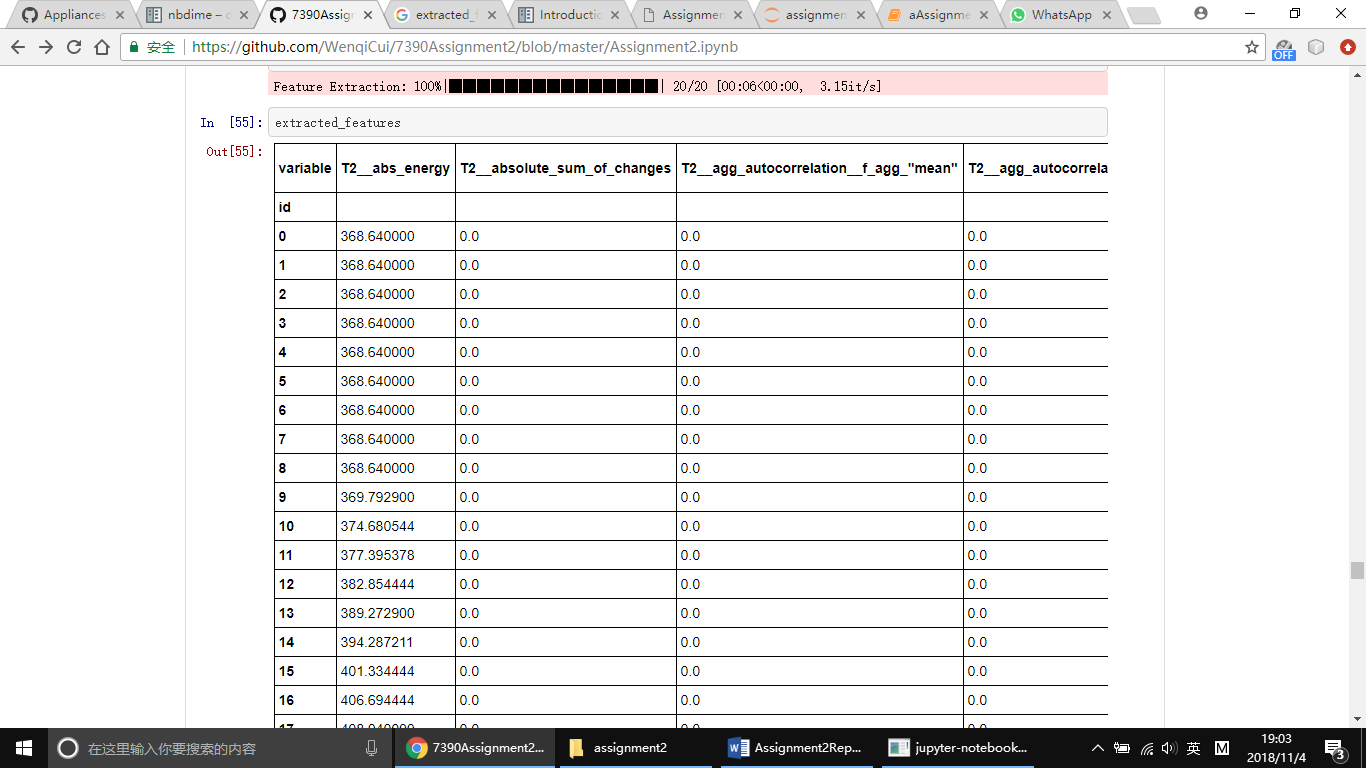
We have the exact same features as before, but calculated on using our test data.

Here is the result:

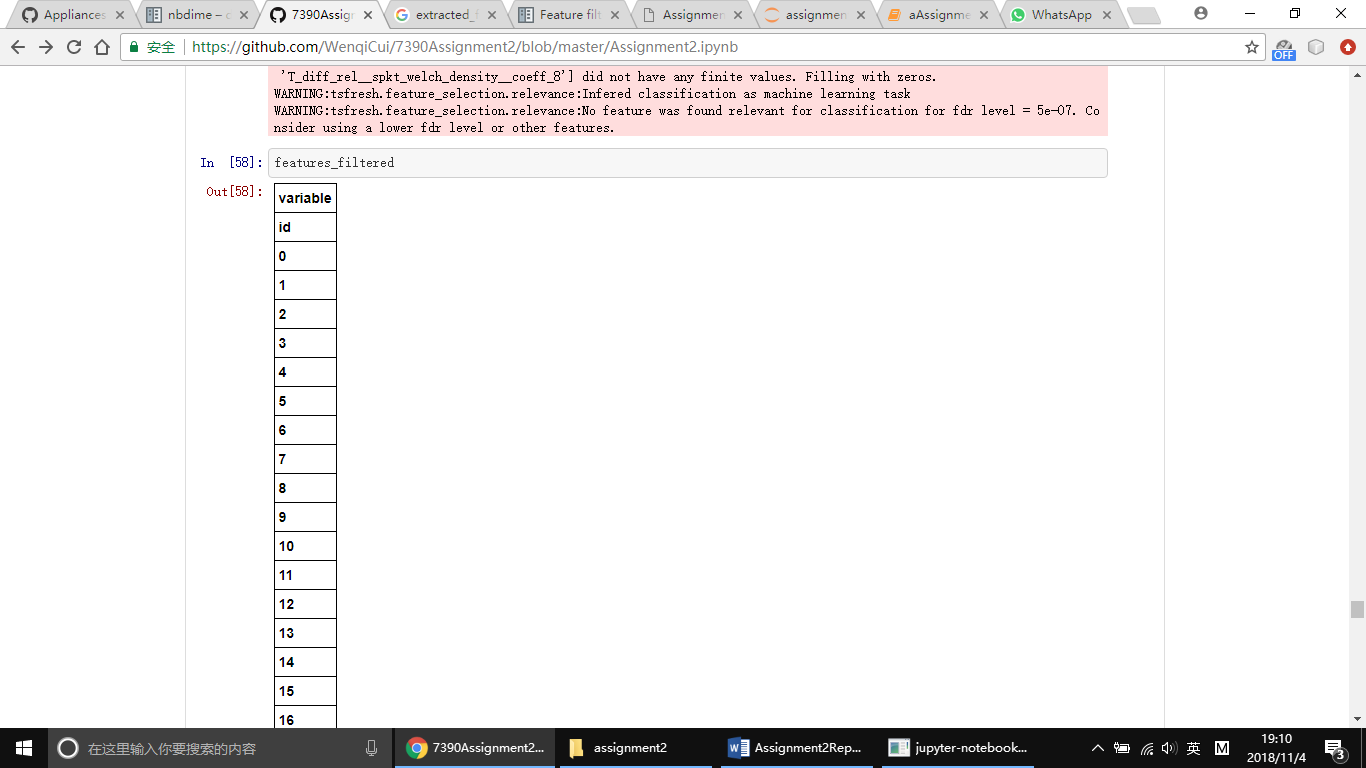


Tsfresh:

Tsfresh is used to extract characteristics from time series. When we want to calculate different characteristics like the maximal or minimal temperature, tsfresh can calculate those features automatically. The extracted features can be used to describe time series based on the extracted characteristics. Further, they can be used to build models that perform regression tasks on the time series. Results as follows:

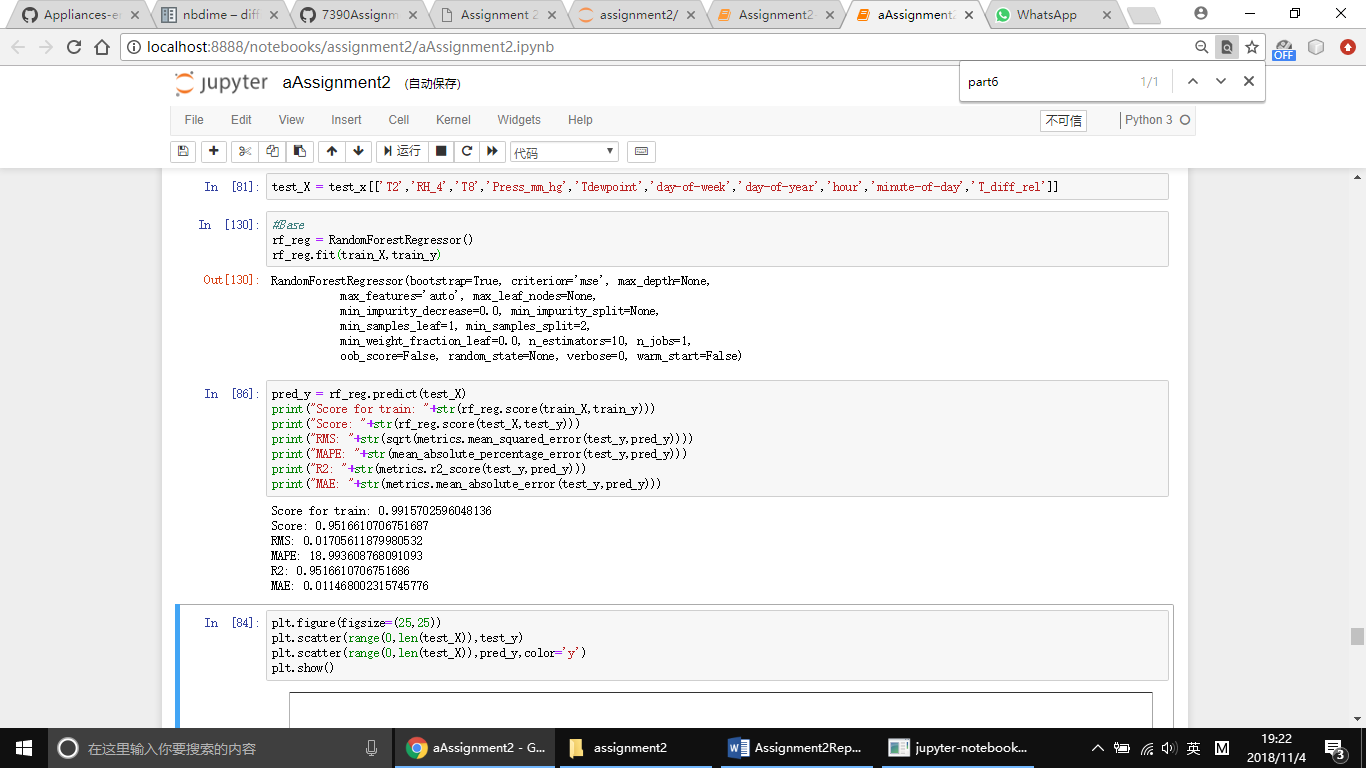


To limit the number of irrelevant features, tsfresh deploys the fresh algorithm.



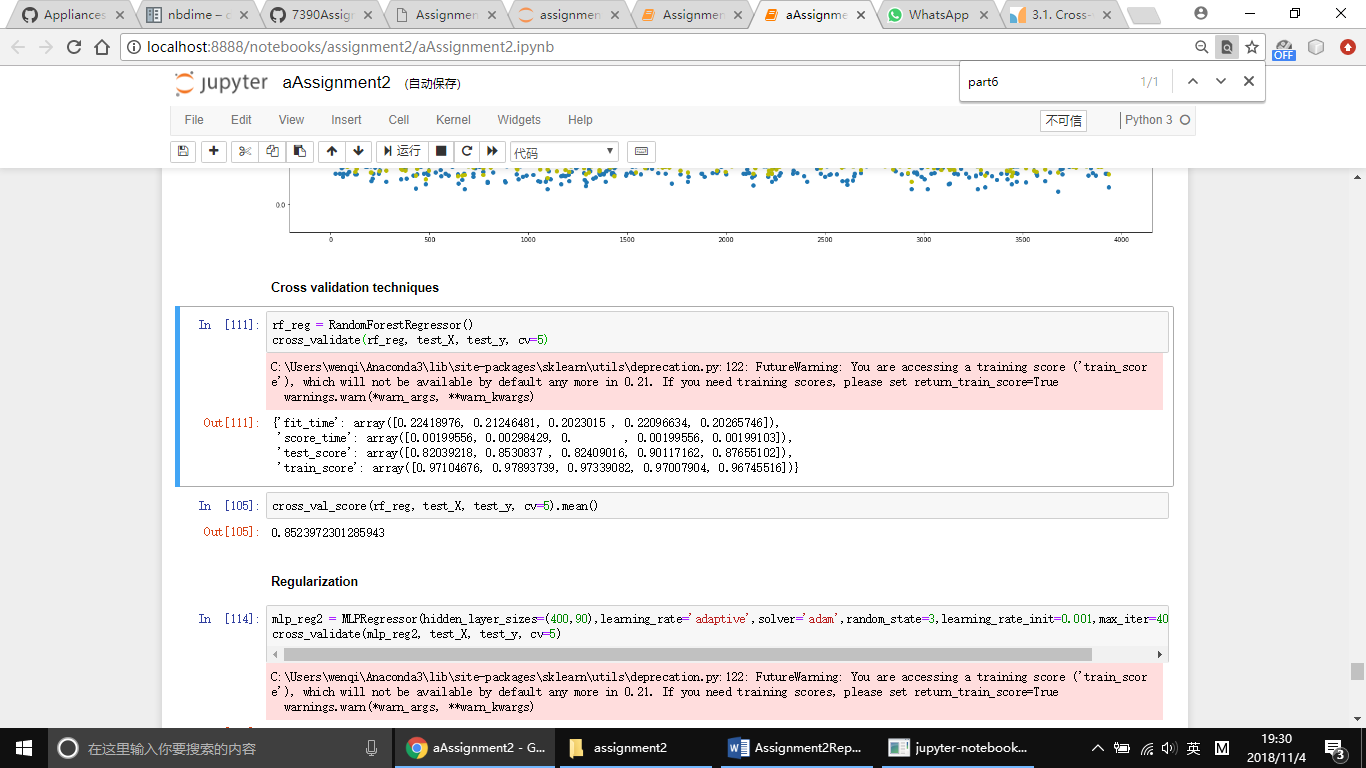
# Part6

To begin with, we have RandomForestRegressor as the base:



Cross validation techniques:

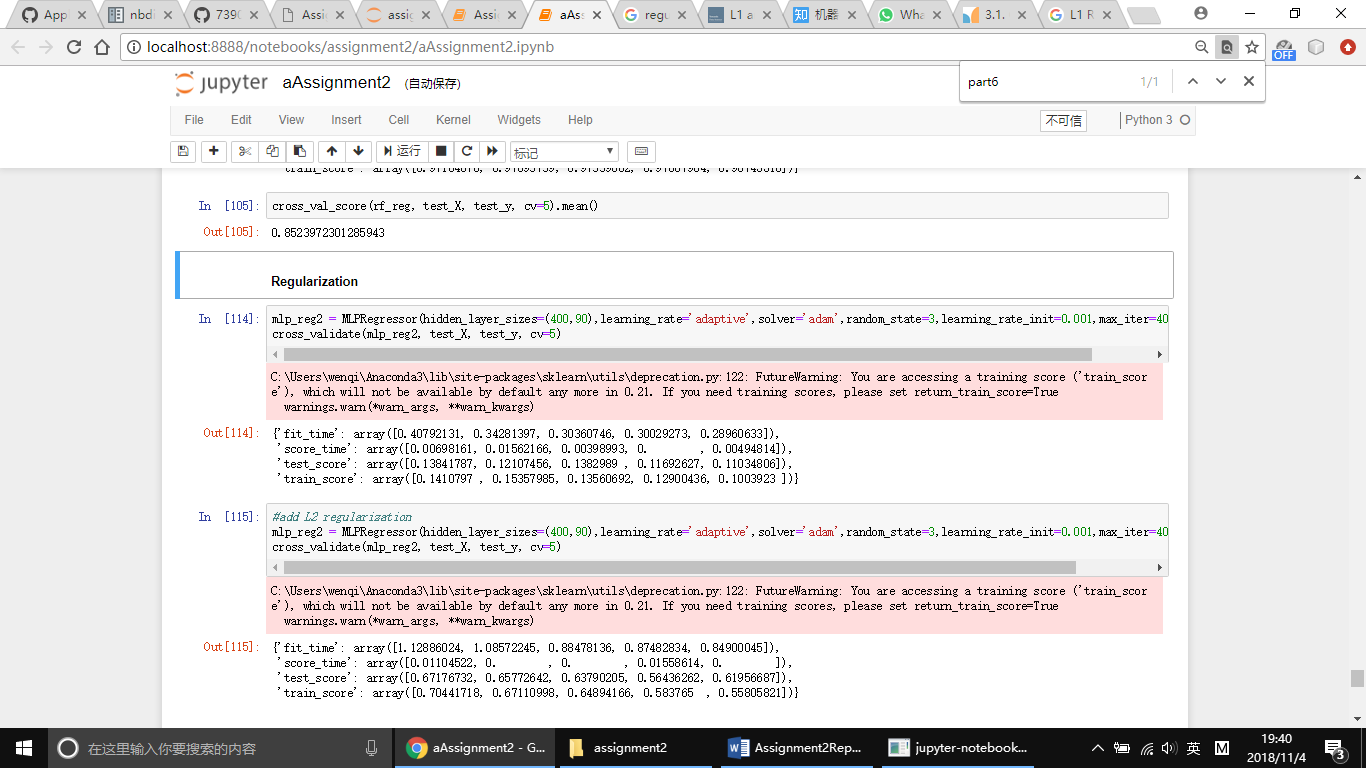
We need to make sure our model get most of the patterns from the data correct and low on bias and variance. A solution to this problem is a procedure called cross-validation. A test set should still be held out for final evaluation, but the validation set is no longer needed when doing CV. In the basic approach, called k-fold CV, the training set is split into k smaller sets. The performance measure reported by k-fold cross-validation is then the average of the values computed in the loop.



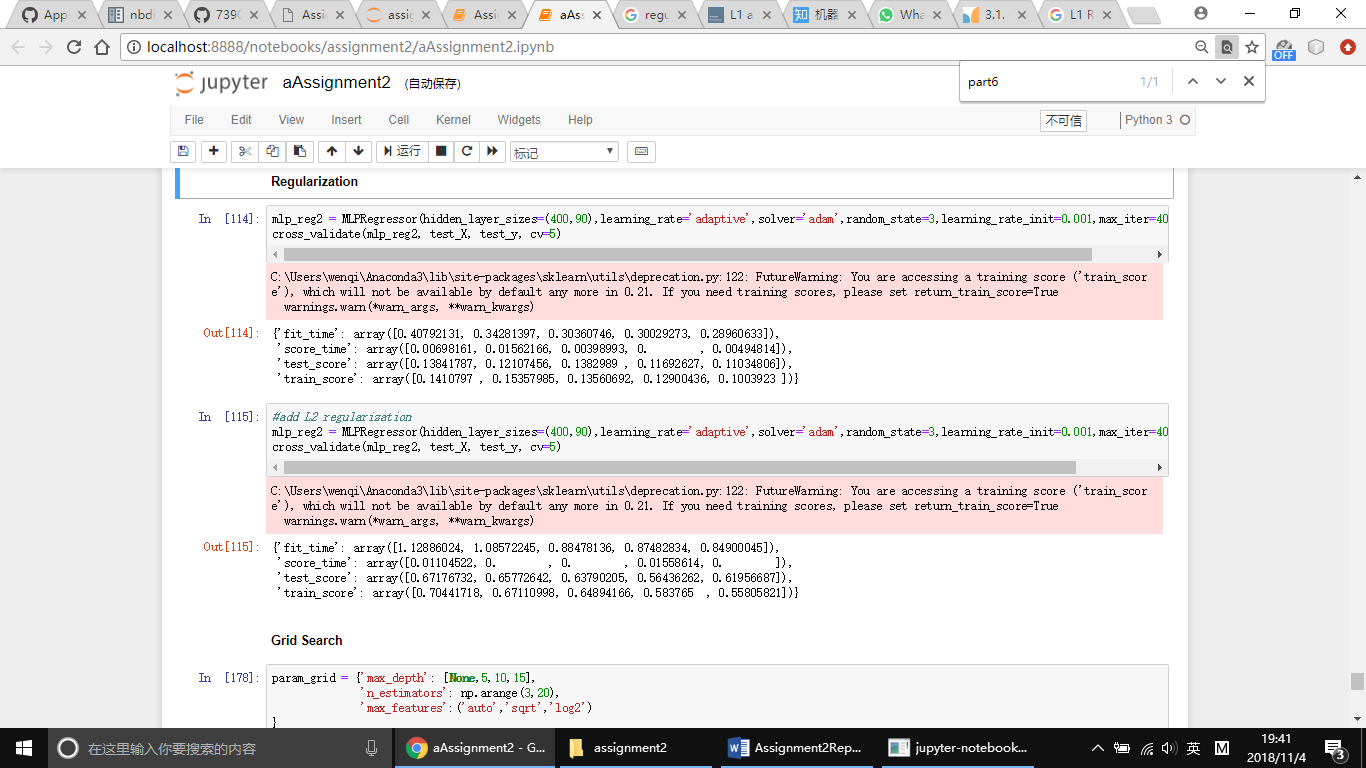
Regularization:

Regularization basically adds the penalty as model complexity increases. Regularization parameter penalizes all the parameters except intercept so that model generalizes the data and won’t over fit.

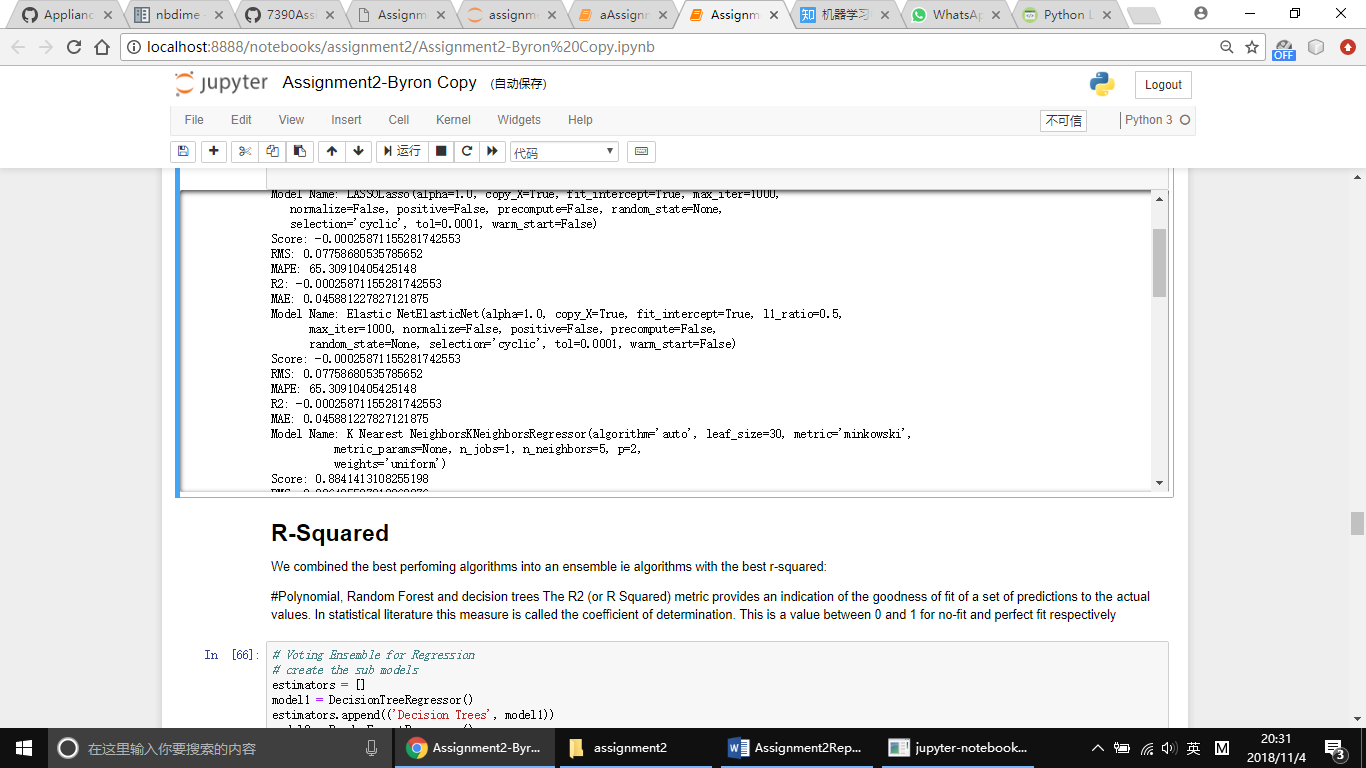
L1 Regularization:



L2 Regularization:

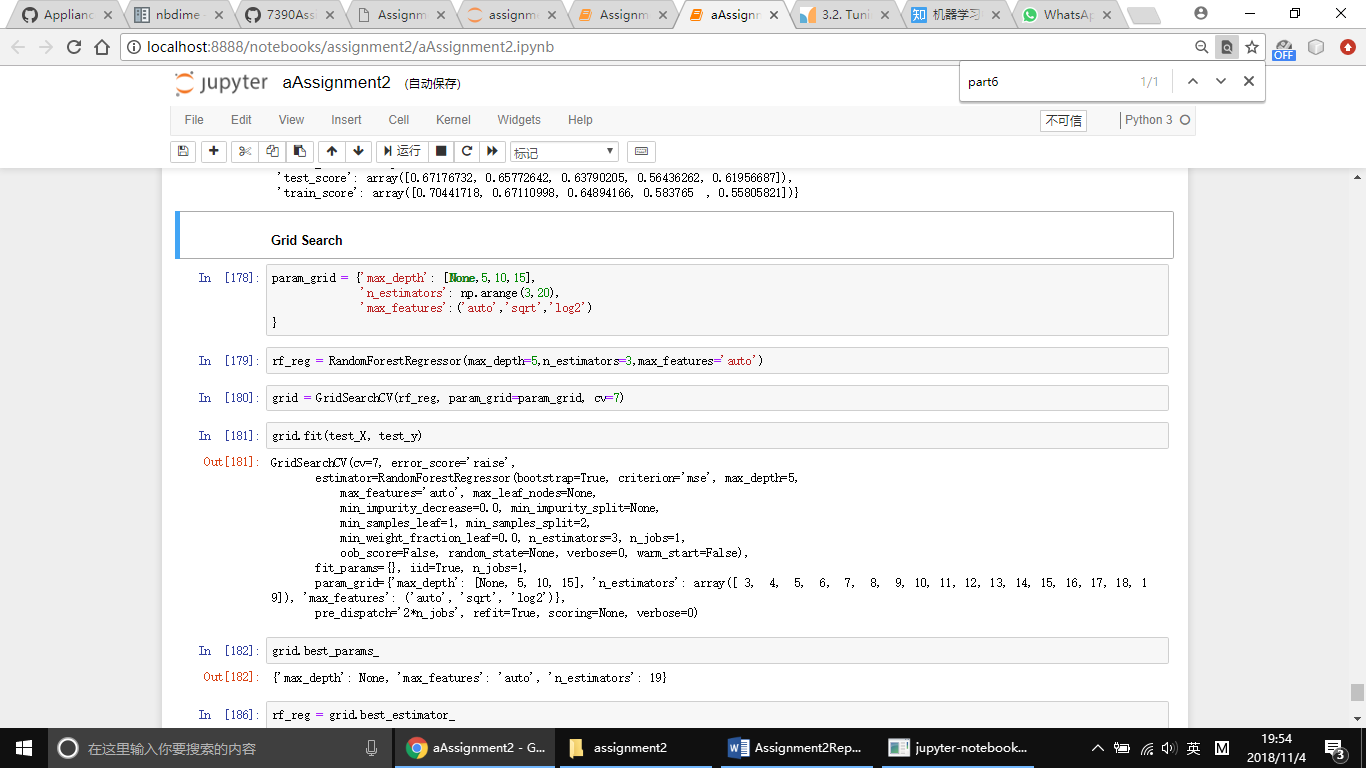


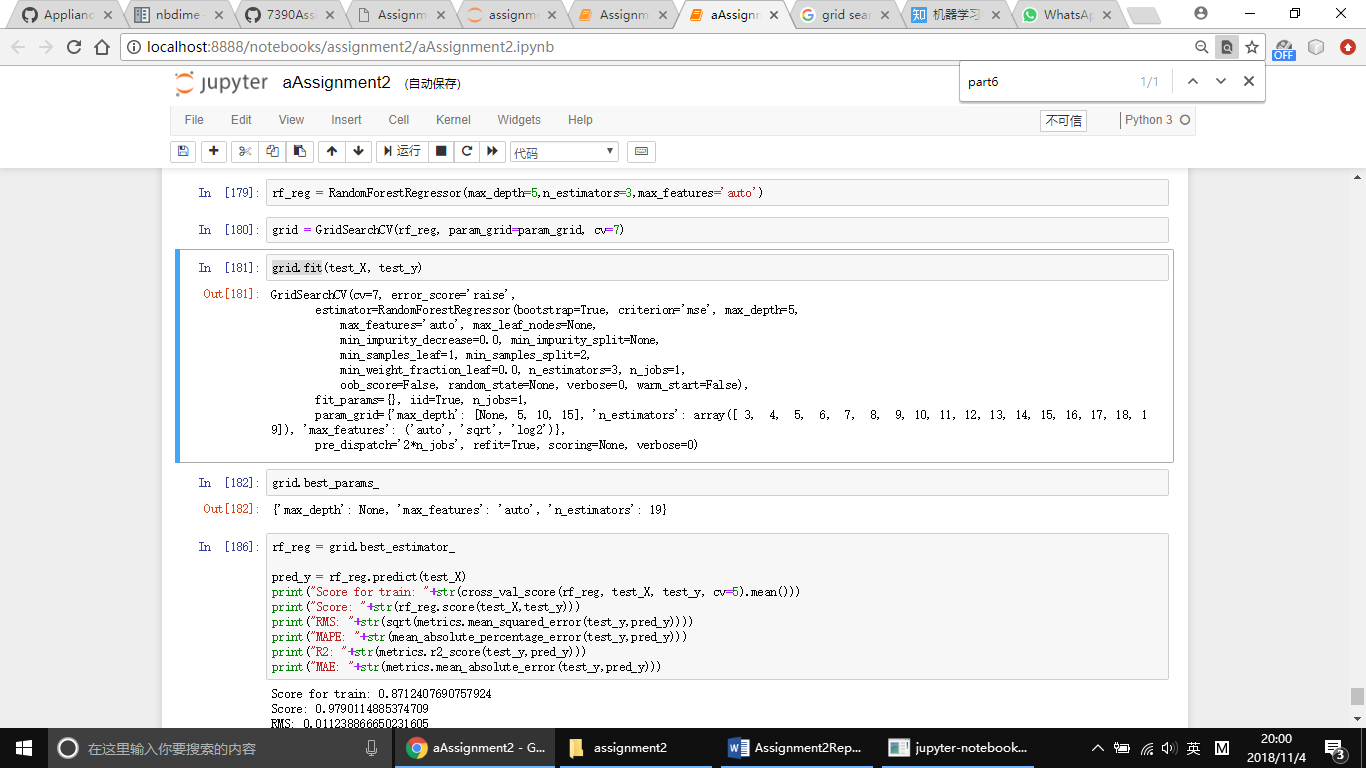
Elastic net:



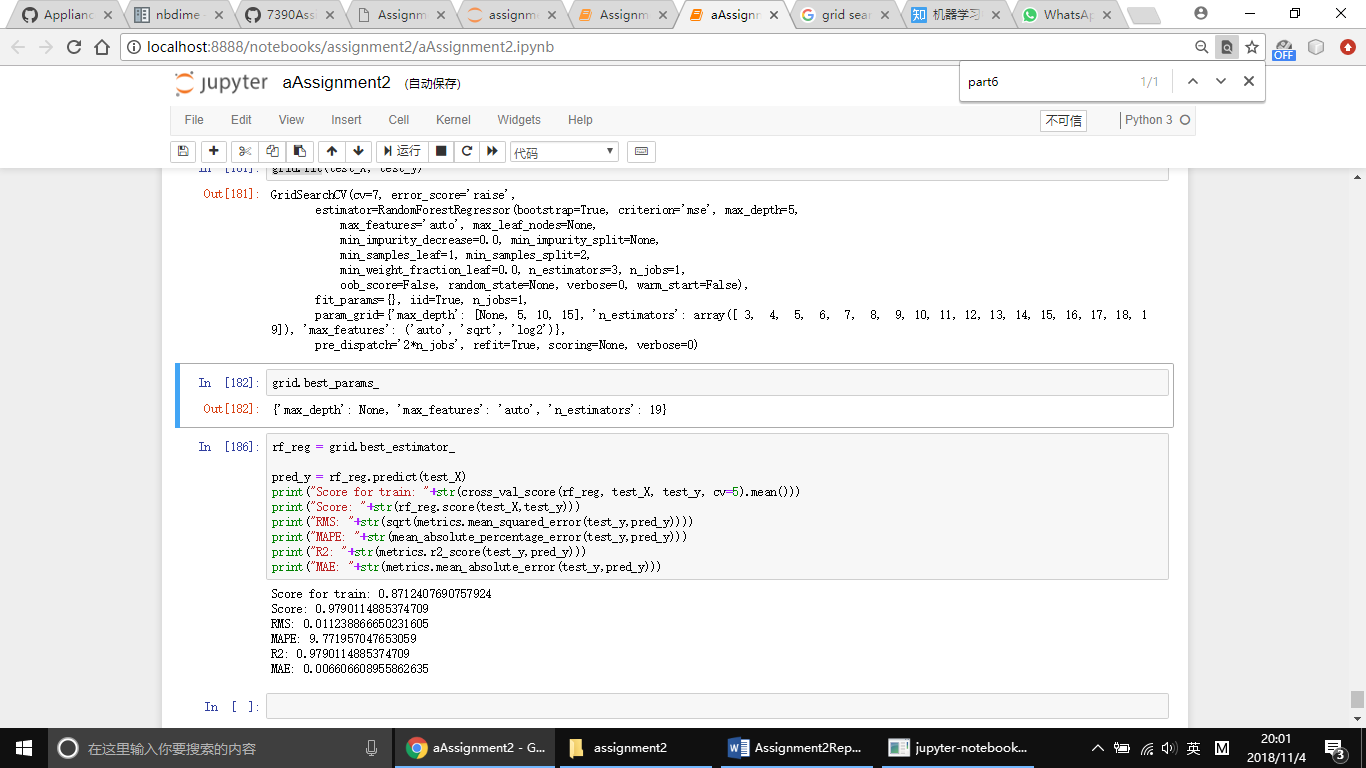
Grid Search:

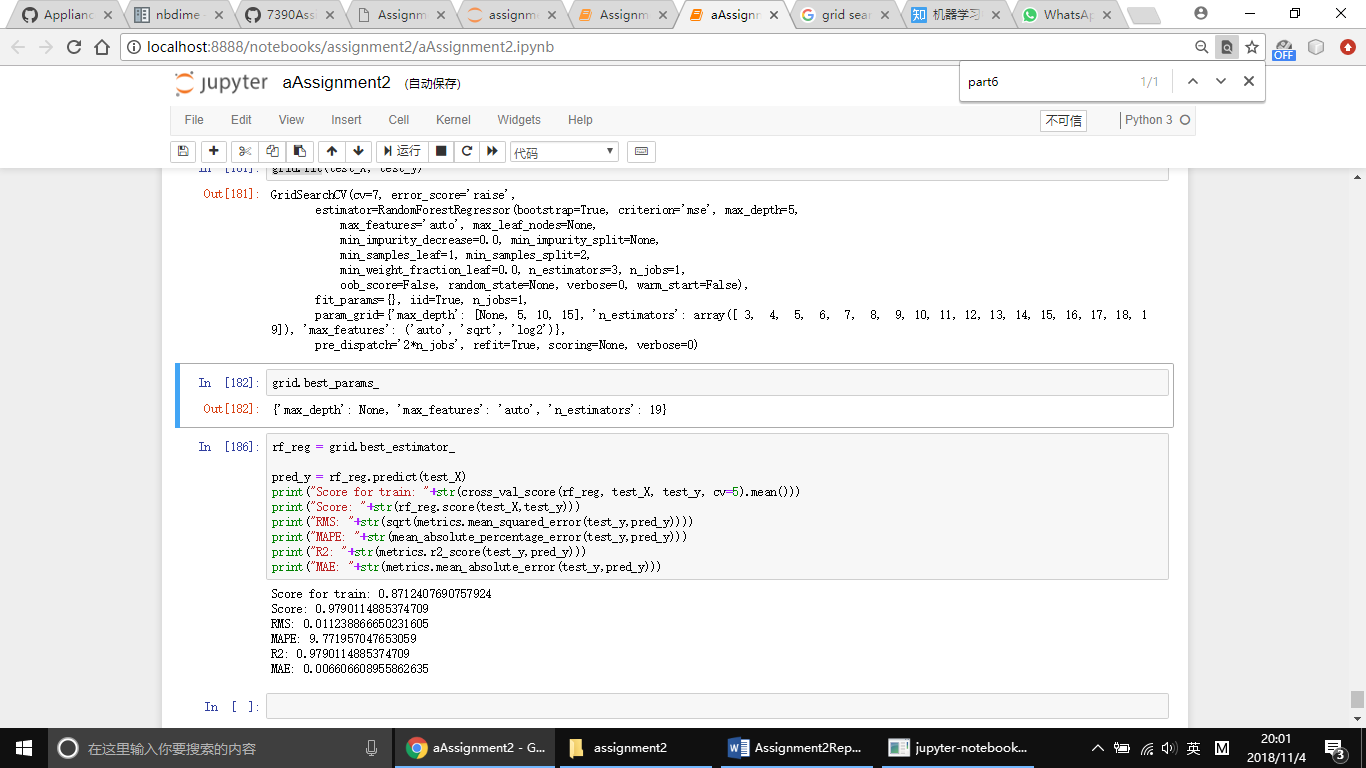
Grid search exhaustively generates candidates from a grid of parameter values specified with the paramgrid parameter:





The result of best parameters as follow:



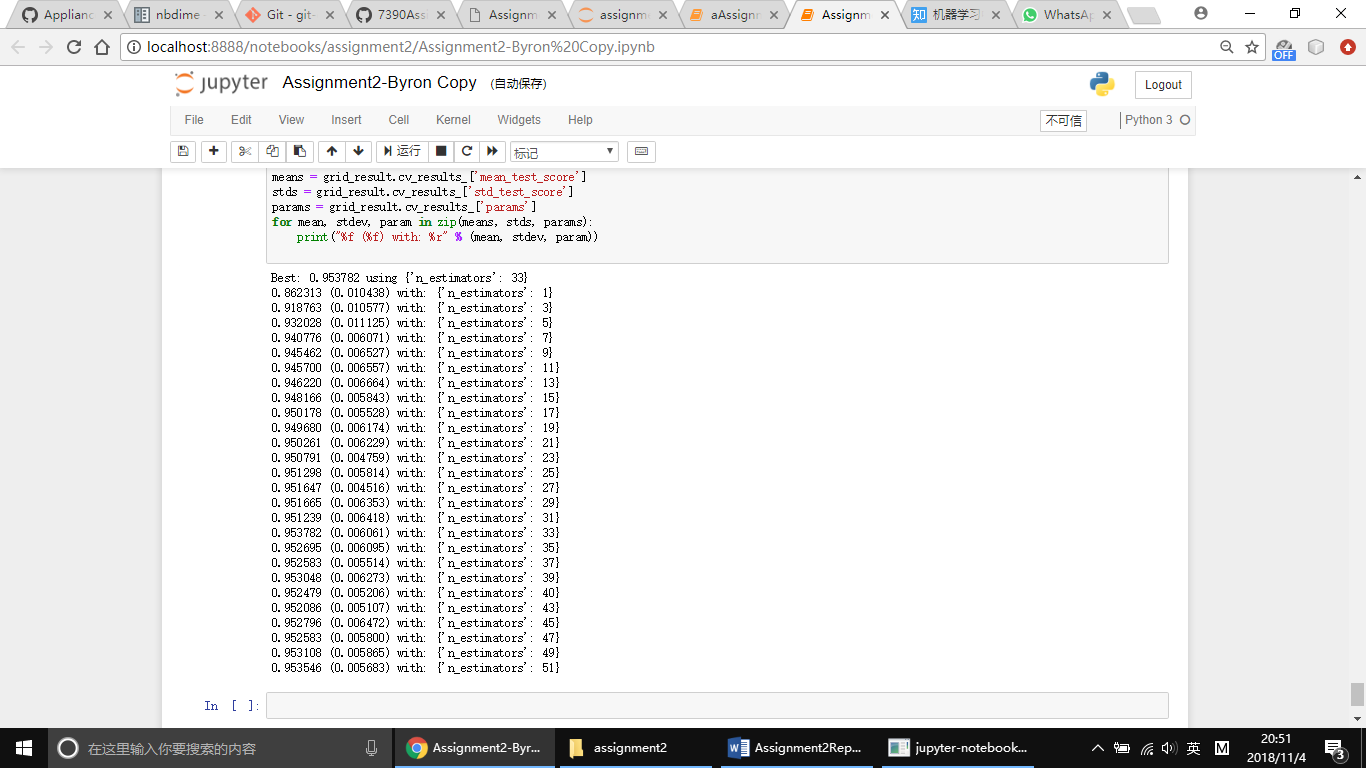


# Part7

In this part, we have three steps.

The first step is to create feature union. The second step is to create pipeline. As we mentioned before, random forest is the best model for this case. The last step is to evaluate pipeline. Result showed as follow: 0.9704315913681134

We know from the results in the previous section that Random Forest algorithm achieves good results on the dataset. But can it do better. The default value for the number of neighbors in Random Forest is 10. Here we can use a grid search to try a set of diﬀerent numbers of neighbors and see if we can improve the score. The below example tries odd k values from 1 to 50, an arbitrary range covering a known good value of 10. Each k value (n neighbors) is evaluated using 10-fold cross-validation on a standardized copy of the training dataset.



# Part8

Nbdime provides tools for diffing and merging Jupyter notebooks. Nbdime provides “content-aware” diffing and merging of Jupyter notebooks. It understands the structure of notebook documents. Therefore, it can make intelligent decisions when diffing and merging notebooks.

We can use command nbdiff-web Assignment28.ipynb Assignment2-ByronCopy.ipynb

Result like follow:

