ABOUT THE DATA:

I took the data from a Kaggle competition, which I am working on here. I never used google prediction API to analyze the data, so thought of using it for that purpose. In this competition, participants are asked to combine historical usage patterns with weather data in order to forecast bike rental demand in the Capital Bikeshare program in Washington, D.C. We were provided hourly rental data spanning two years. For this competition, the training set is comprised of the first 19 days of each month, while the test set is the 20th to the end of the month. We must predict the total count of bikes rented during each hour covered by the test set, using only information available prior to the rental period.

FEATURE ENGINEERING:

There was column including the datetime feature which I divided into various other features such as year, month, day of week, and hour in the anticipation that they will serve as very important features. I used pandas dataframes to store these training values. With ranking being determined from Root Mean Squared Logarithmic Error (RMSLE), our aim becomes to accurately guess the natural logarithm of bike demand at different times (actually demand count plus one, in order to avoid infinities associated with times where demand is nil). To facilitate this, we add the logarithm of the casual, registered, and total counts to our training DataFrame.

Do you think these predictions are good?

The google prediction API gave a Classification accuracy of 98%, but I feel I would like to have more control over the parameters in order to decide how my model worked on the dataset. I have found many instances from various blog posts that other API's have outperformed google's API by a great margin. I also feel that the predicting for a test file with huge data is difficult with this API as it took hours for me to get it completed (maybe I am doing it the wrong way as I don't have much experience on this API).

Therefore I used python's Machine Learning Libraries from sklearn to build my model which I was more comfortable with. I used python's sklearn ensemble api to run Random Forest and Gradient Boosting classifier to train my model and got a RMSLE loss of 0.42211 which ranked among the top 14% of the teams in the competition.

The below are the various parameters which I got in order to see if the model predicted well or not.

The best 3-fold cross validation accuracy on training set by using Random Forest on Casual Users: 83.19%

The best 3-fold cross validation accuracy on training set by using Random Forest on Registered Users: 84.00%

The best 3-fold cross validation accuracy on training set by using Gradient Boosting on Casual Users: 85.44%

The best 3-fold cross validation accuracy on training set by using Gradient Boosting on Registered Users: 87.19%

Mean Absolute Error when predicting using Random Forest on Casual Users: 0.275778

Mean Absolute Error when predicting using Random Forest on Registered Users: 0.125512613491

Mean Absolute Error when predicting using Gradient Boosting on Registered Users: 0.191177

Mean Absolute Error when predicting using Gradient Boosting on Casual Users: 0.339482519544

My solution to this above problem may seem to be naïve and simple, as I am starting out in this field. I have less than 4 months experience doing this kind of stuff and I feel with time I will improve a lot given the right opportunity.

Note:- Some of my code are inspired from Kaggle forums and Utah Data Science meetups.