Description and Analysis of HW 5

“In the first part, you will explain in detail what you have tried and what got you your best performance. You will describe the difficulties you encountered and how you solved them. The format of this small report does not need to be fancy. (10 points)”

This homework assignment was for the most part available for the students to select among various machine learning algorithms and to gain experience with which bag they chose to select. For my submission under advisement of Dr. Benveniste I focused the majority of my effort on to Sklearn Toolkit, with GridSearchCV/GradientBoostingClassifier. Afterwards I moved onto XGBoost, a sleek more effective utility for “regressive” modeling.

However, in order to have these toolkits be useful, it was necessary to make the data available to be understood by the learning algorithms. For this small subgroup, and introductory exercise the recipe remain mostly the same. We began by naturally reading in the data sets; Test and Train, into the Jupyter environment, where it was possible to “preprocess” the data. First by Normalizing the categorical variable into a weighted basis. After separating the target values into its own column, we could then One Hot Encode both frameworks to separate the functional data with its standalone partner as to not destroy the original set. After some research, it should be noted here, that the data within a given size and in assembly for machine learning are majorly preprocessed in similar steps, with the variety within pertaining to the final information sought. This being said most, if not all data is preprocessed.

Finally, with our hot encoded train, test and resultant y. We can pass the set into our chose algorithms, firstly I attempted Xgboost, which without much knowledge was difficult as I spent many hours on attempting workarounds for the package. Under suggestion, I quickly proceeded to Sklearn, who’s familiar form was easier to understand. Here I found that the majority of the work in this assignment was involved in the tuning of parameters of these ML functions. Working individually through the parameters of the GradientBoostingClassifier (GBM). Optimal parameters were found for the Porto Seguro Safe Driving within areas of the highest mean as the classifier places it against the others, in some cases the defaults were sufficient. Here it occurred to me (with the help of Piazza) that at the moment efficient coding was not the most effective way to tackle the prediction for the dataset, many people had already tuned similar parameters on Kaggle. Investigating, I found mostly that they converged to a group, given that most went about it in different ways.

Here after some work I was able to construct a workable version the XGBoost package from Github. XGBoost had a very similar language schema to GBM, so integrating in the outstanding code was quick and simple. Again, parameterizing and reading documentation took the majority of the time, however it was mechanical and not to frustrating.

As I have submitted multiple versions of my XGboost code that are awaiting marking, so far the brute force of Sklearn has been the most effective.

Output was a simple write script with included ID and Prediction columns, to csv for submission into the professor for analysis.

Other candidates in this arena included works of H20 and TensorFlow, but in a way more interesting is to see, as we have two working classifiers, what the impact would be of layering these methods, the weights would most assuredly be different but it what way, graphically this would be interesting to see.