CS434 Final Project Report

Danny Barnes, George Crary, Griffin Gonsalves

1 Feature formulation and preprocessing

1.1 Features

What are the features you feed to your learning algorithm? Did you simply flatten the 7 rows into a vector for features? Did you transform or aggregate the given data to engineer your own features?

We used every feature provided in the raw data, and flattened into a vector. We decided to remove the time of day feature for the sake of cleaning up the data. We decided to go off of the base features and not generate any new ones.

1.2 Preprocessing

Did you pre-process your data in any way? This can be for the purpose of reducing dimension, or reducing noise, or balancing the class distribution. Be clear about what you exactly did. The criterion is to allow others to replicate your works.

We decided as a group that removing the time of day and exclusively using the 24-hour time would be acceptable as the feature is redundant. In kmeans, we normalized the data by using numpy's scale function in order to balance the distribution for better results.

2 Learning algorithms

2.1 Algorithms explored

Provide a list of learning algorithms that you explored for this project. For each algorithm, briefly justify your rationale for choosing this algorithm.

Random Forest regressor, q-learning, kmeans, SVM, knn, Naive Bayes,

We were interested in ensemble learning due to the limitations on the training and test data provided to us. Random Forest seemed like a good candidate for one of our models due to the fact that it creates higher variance by bagging a subset of the features provided and generating a defined number of trees to then average into a final predictive model.

Q-learning was an unsupervised method we explored but found that it would be more suited to a live, or real-time predictive model based on a built history of user data. Naive Bayes was also another point of discussion for our group as we thought the classifier might be able to form good predictions, but as it turns out the data does not lend itself to the algorithm as we have different forms of data than documents or large quantities of text.

For Kmeans we explored various cluster counts and distances to isolate anomalies. We used euclidean distance. Point further than 3 std deviations were considered but was ineffective unfortunately.

SVM seemed to be a good algorithm when we discussed it in class. A small amount of research suggested that some other researchers had good results on similar problems with this algorithm.

SVM/KNN Exploration here.

2.2 Final models

What are the final models that produced your submitted test predictions?

SVM & Random forest & Kmeans (no output generated for kmeans)

3 Parameter Tuning and Model Selection

3.1 Parameter Tuning

What parameters did you tune for your models? How do you perform the parameter tuning?

Random Forest regressor implementation in scikit learn had no option to try voted selection, though we were interested in testing out the different results. . The implementation we used had an option to enable bootstrapping for better training of the model. Initially chose 100 trees to be generated by the algorithm in order to verify that the algorithm was running properly but upscaled to 1000 when we saw the values looked good.

SVM regression params - Only adjusted prediction threshold for the boolean prediction

3.2 Model selection

How did you decide which models to use to produce the final predictions? Do you use cross-validation or hold-out for model selection? When you split the data for validation, is it fully random or special consideration went into forming the folds? What criterion is used to select the models?

For Random Forest we used cross validation

SVM - Cross validation

4 Results

Do you have any internal evaluation results you want to report?

Nope