CS434 Final Project Report

Tsewei Peng

1 Feature formulation and preprocessing

1.1 Features

The features I fed to the learning algorithms are all the features except for the hour of day. To do this, I simply flatten the 7 rows of features into one row like the format of the final test instances, and then slice the data so the hour of day feature is stripped from feeding into the learning algorithms, which means a total of 56 features are needed for one prediction. I've tried taking out the "slope" feature, or the meal count features including: "morning", "afternoon", "evening", "night"; however, the error rates are higher when those features are excluded. I did not aggregate the data to engineer my own features, but I did do a principal component analysis to reduce the dimension of the training data set.

1.2 Preprocessing

I pre-processed the data by first balancing the class distribution, and then reducing dimension and finally normalizing the data. The training data set is highly unbalanced, with most of the data set having y value of 0. To balance the class distribution, I generated samples from a normal distribution based on the mean and standard deviation of the original data with y value of 1; after the sampling, the class is balanced with ratio of 1:1. For dimension reduction, I used principal component analysis, with a variance retention of 95%, which generates around 18 to 19 components during hold-out validation. After that, I normalized the generated training data set using built-in scaler in sklearn library.

2 Learning algorithms

2.1 Algorithms explored

1. Linear Regression. It is a simple algorithm worthing trying.

2. Logistic Regression. This is also another simple algorithm we learned from class.

3. Stochastic Gradient Descent Classifier. The built-in gradient descent algorithm with a lot of choices of loss functions to try.

4. Passive Aggressive Classifier. Something I learned about in the sklearn library documentation, and tried it since it is easy to implement using the library.

5. Naive Bayes Classifier. A classification algorithm learned in class. Tried it but was not expecting anything, since the feature in training data is not conditionally independent.

6. Support Vector Classifier. A classifier using Built-in svm with default rbf kernel function.

7. kd-tree. I was hopeful this would work really well since the artificially-generated samples could mean clusters of data, which would can potentially lead to more accurate classification.

8. Decision Tree. Just another algorithm learned in class that I thought was worth trying.

9. Multi-layer Perceptron. Hopeful on this one as MLP can reveal some interesting patterns in data that is sometimes not attainable by other models.

10. K-mean clustering. A clustering algorithm we learned in class. I was hopeful this one would work as the artificially generated samples can potentially generate 2 clusters for better classifications.

2.2 Final models

Final models that produced my submitted test predictions: 1) Linear Regression, 2) Logistic Regression, 3) Stochastic gradient descent classifier.

3 Parameter Tuning and Model Selection

3.1 Parameter Tuning

For all of the models tried, I tried to tune them by trying out different number of iterations, changing class weight for penalizing mis-classifications, and the weight of L2 regularization. Also, I tested the accuracy by using hold-out validation by leaving one training set out for validation, which means several accuracies are produced for each algorithm. The accuracies produced as numbers of true positives, false positives, false negatives, true negatives.

For kd-tree, I've tried to adjust the number of k. For MLP, I've tried different amount of hidden nodes, as well as 1 and 2 hidden layers. For k-mean, I've tried different amount of k after unsuccessful attempt of classification at $k = 2$. The learning rate of algorithms are default tuned to optimal, so I did not spend a lot of time experimenting different values of learning rate.

For stochastic gradient descent classifier, I ended up using loss function of log loss after experimentations with different loss functions.

3.2 Model selection

Since the data are heavily imbalanced, with most data samples having y value of 0, it is hard to find a model that have high accuracy even after data sampling and model tunings. Most of the algorithms without tuning produce very unsuccessful results, with most of them producing near 0 true positives. After tuning, some models produce more true positives. But, this comes with a cost, the number of false positives greatly increases as well. During model selection, I used hold-out validations. Also, I use a loop to run the algorithm several times to ensure all of the data sets are being held-out once.

Since the competition is graded based on the F1 measure and the area under ROC curve, this means even with sacrifice of increasing amount of false positives, number of true positives cannot be 0. The final models selected to generate test predictions are the ones that performed the best during hold-out validations.

4 Results

I'm amazed that some of the algorithms that I thought would work, in the end did not work. For example, MLP, k-means, kd-tree, etc. I think it's because the data sampling causes the regressions to generate a overly large boundaries, which means the validation results in high amount of true positives and low amount of false negatives, but at the same times generates a lot of false negatives. I don't think I successfully found a model to find the pattern in the data sets, but it is a learning experience.