Project1 Report

Wang Chengming 20450392

November 10, 2017

1 Data Preprocessing

In this project, the dataset is pretty big in that it has 57 features and 3220 samples. All the features are denoted as numbers so we have no idea the meaning of each of them. It becomes inappropriate to delete or combine some of the features. But still, there's some preprocessing we can do to get a good classification result.

```
In [14]: data.describe().iloc[:, [0, 1, 2, -3, -2, -1]].applymap(lambda x: round(x, 1))
```

```
Out[14]:             0       1       2      54      55      56
         count  3220.0  3220.0  3220.0  3220.0  3220.0  3220.0
         mean      0.1     0.2     0.3     5.2    50.1   278.6
         std       0.3     1.2     0.5    33.1   126.5   523.7
         min       0.0     0.0     0.0     1.0     1.0     1.0
         25%       0.0     0.0     0.0     1.6     7.0    36.0
         50%       0.0     0.0     0.0     2.3    15.0    95.0
         75%       0.0     0.0     0.4     3.8    44.0   274.0
         max       4.5    14.3     5.1  1102.5  2204.0  9088.0
```

The table above is the general information of the dataset. I only show the first and last three columns. It is pretty clear to see that these features have huge difference in mean and standard deviation values. So it is necessary to scale the data to make all features have mean 0 and standard deviation 1.

2 Model Selection

Then we can move to select an appropriate model to classify the dataset. Here I chose between two models, MLP and random forest. Why I chose these two? I think nowadays deep learning has shown its power in digging out hidden features. In this project there are so many features, I think with MLP, with deeper and larger neural network hidden features will be digged out and be used to get great classification result. For random forest, I have seen that many winners in classification competitions on Kaggle used random forest as their model. So I think it will be a perfect choice here. I used a validation set to tune the hyperparameters of each model. In MLP, I tuned the hidden layer size and learning rate. In random forest, I tuned the depth. Below are performances of each classifier on the same validation set. I only show the best ten of each.

```
In [23]: # The performance of MLP
         acc.sort_values('accuracy')[-10:]
```

```
Out[23]:     learning_rate hidden_layer_size  accuracy
         36           0.14           (8, 32)  0.947205
         19           0.08               100  0.947205
         39           0.14       (8, 16, 32)  0.947205
         43           0.16               100  0.947205
         27           0.10               100  0.947205
         48           0.20                70  0.950311
         12           0.07           (8, 32)  0.950311
         4            0.05           (8, 32)  0.950311
         20           0.08           (8, 32)  0.950311
         8            0.07                70  0.956522
```

```
In [29]: # The performance of Random Forest
         acc2.sort_values('accuracy')[-10:]
```

```
Out[29]:     depth  accuracy
         22     24  0.939746
         24     26  0.940358
         14     16  0.940673
         23     25  0.940990
         16     18  0.941304
         25     27  0.941603
         17     19  0.941915
         20     22  0.943151
         19     21  0.943156
         18     20  0.945951
```

From the above results, we can see that in general MLP and random forest both perform well enough. MLP is slightly better than random forest. So I will choose MLP as my classifier. The learning rate will be 0.07 and hidden layer size will be 70. There will only be one hidden layer.

Now I have got the model. The final step will be to apply it to the test set and get the result.

```
In [32]: clf = MLPClassifier(solver='lbfgs', alpha=0.07,
                             hidden_layer_sizes=70, random_state=1)
         clf.fit(train_data, train_labels[:, 0])
```

```
Out[32]: MLPClassifier(activation='relu', alpha=0.07, batch_size='auto', beta_1=0.9,
                beta_2=0.999, early_stopping=False, epsilon=1e-08,
                hidden_layer_sizes=70, learning_rate='constant',
                learning_rate_init=0.001, max_iter=200, momentum=0.9,
                nesterovs_momentum=True, power_t=0.5, random_state=1, shuffle=True,
                solver='lbfgs', tol=0.0001, validation_fraction=0.1, verbose=False,
                warm_start=False)
```

```
In [35]: preds = clf.predict(testdata)
         preds = pd.DataFrame(preds).astype('int')
         preds.columns = ['Prediction']
         preds.to_csv('project1_20450392.csv', index=False)
```