Sassafras

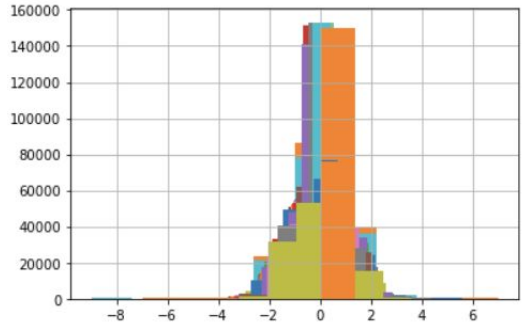
Introduction

In this module we are given 203,287 of simulated data items of X, with each data item consisting of 75 covariates divided into 9 groups (columns of X) and one target (Y). Based on the simulations of X, we are going to predict the Y values. Also, the data contains a certain portion of missing values recorded as either "?" or "NaN". Had we learned the simulation is an X→Y mapping, we used multiple models to make the prediction, but eventually decided to include random forest and Neural Network in this report. By comparing the final prediction results, we found out that the Neural Network model is more proficient in predicting ability.

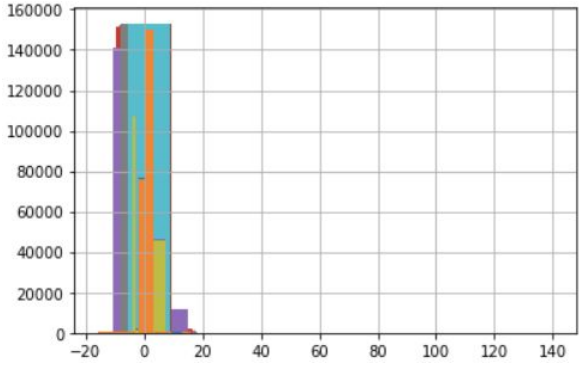
Feature Engineering

Before fitting the dataset, we needed to modify some of the entries. In the datasets, it was clear that all explanatory variables as well as response variables are all numerical data. The first thing we did is to deal with the missing/NAN/infinite values in the dataset. In that case, we replaced the NAN values with the mean of each column, respectively, which was calculated without these NAN values. This step cleaned out our dataset quite a bit, since in some columns the missing values take nearly 10% of the data. At the same time we tried to ensure data's consistency and avoid possible data loss in the data cleansing phase.

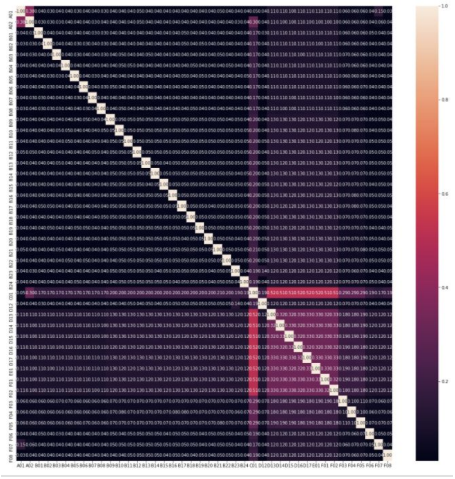
Next, we plot the distribution of the training data in order to visualize the pattern of the data items' distribution.



To standardize the data, we normalized all datasets in training as well as dataset.



Then we plot the correlation heatmap in order to help us identify the highly correlated variables, which may help in future feature selection stage.



Data Modelling

We used several regression models to fit our data. These models were elasticnet, XGBoost, Adaboost , random forest model and Neural Network models. Overall speaking, the Neural Network model performed the best with the smallest RMSE of 1.13. Therefore, we used the Neural Network model as the final model that we submitted to kaggle. However, we are including the details of the modeling process of both models below:

Why selecting these two models?

Since we were dealing with a very large dataset in this module, choosing a more robust and advanced model will help us better in prediction. Although we did try out some simple models such as linear regression, ridge regression etc., the results turned out not quite desirable. In addition, since there were lots of correlated variables, it was important for us to test the models that are naturally better at modelling correlated datasets. Random Forest, for instance, will always choose the most important feature to split on, hence it would reduce the influence of multicollinearity. Therefore, we chose to use the random forest model and neural network model.

Random Forest

We applied random forest to construct the model, as well as select important variables for objects using feature importance. We chose the parameters n_estimators to be 100, and the min_samples_leaf to be 3, to achieve the optimal result. Since there were 14 response variables, we fit a model with each response variable and all training datasets, to obtain a single Z value. We repeated 14 times to get the full responses. The model was further improved by selecting feature importances, which gave us a clear idea of which variables are important in the random forest model. We then fit the data using only important features that are presented, which is able to present a slight improvement to our prediction. After fitting the data, we predicted the data on the test dataset, and obtained a RMSE of 1.17. This is not as good as our neural network model,

therefore we didn't select the random forest model to be our final model.

```
d = {}
Y_train['Z01']
for i in range(1, 15, 1):

    d["m" + str(i)] = RandomForestRegressor(n_estimators=100, min_samples_leaf=3, max_features='sqrt',
                          n_jobs=-1)
    col = 'Z' + '0' + str(i) if i<10 else 'Z'+ str(i)
    print(col)
    d["m"+str(i)].fit(X_train, Y_train[col])
```

Neural Network model

We now apply Neural Network to construct the model. We first use all the variables to fit the dataset. We then fit the data by using selected variables. By using only important variables, we do not observe an improvement on our prediction. After fitting the data, we predicted the data on the test dataset, and obtained a RMSE of 1.13. The final result is better compared to the random forest model. Therefore we select Neural Network as our final model.

```
sgd = tf.keras.optimizers.SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='mean_squared_error', optimizer=sgd)
```

```
callbacks = [tf.keras.callbacks.EarlyStopping(patience=5, min_delta=1e-2)]
```

```
DNN_dict = {}
for col in y_columns:
    model = tf.keras.Sequential([
    tf.keras.layers.Dense(100, activation='relu',input_shape=X_train.shape[1:]),
    tf.keras.layers.Dense(60, activation='relu'),
    tf.keras.layers.Dense(10, activation='relu'),
    tf.keras.layers.Dense(1),
    ])
    sgd = tf.keras.optimizers.SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
    model.compile(loss='mean_squared_error', optimizer=sgd)
    model.fit(X_train,df_train_y[col],batch_size=200,epochs=50,shuffle=True,verbose=0,validation_split=0.3, callbacks=c
    DNN_dict[col] = model
```

```
D:\Anaconda3\lib\site-packages\tensorflow\python\keras\callbacks.py:1234: RuntimeWarning: invalid value encountered i
n less
  if self.monitor_op(current - self.min_delta, self.best):
```

```
for index, row in X_test.iterrows():
    p = make_predict(row.values.reshape(1,-1),df_test_y.iloc[index]['Class'],DNN_dict)
    df_test_y.loc[index,'Value'] = p
df_test_y[['Id','Value']].to_csv('DNN_dict.txt',header=True,index=False)
```

Conclusion

After the comparison of the two models' corresponding RMSEs, the Neural Network model's prediction proficiency has been proved with its RMSE being 1.13. Therefore, we choose to use the Neural Network model. Overall, the Neural Network model seems to perform quite well, but we were also aware that collinearity is still an issue presented. Although we tried some variable selection techniques as well as dimension reduction techniques to minimize the effect of collinearity, no better result was observed.