**CREDICXO ASSIGNMENT REPORT**

**Abstract**

The data provided was mainly numerical (the features). The 166 feature columns were chosen as inputs to a Multilayer Perceptron Model built using Keras. Certain pre-processing steps were performed before feeding the data into the model. Highly correlated features were dropped using two methods – manual selection considering correlation values and using Principal Component Analysis (PCA). Hence, the conclusion is presented in the form of a comparison which indicates that a comparable performance was obtained using both the methods even when the number of PCA selected features is less than manually selected ones. Accuracy in both the cases comes out to be around 99%.

**Pre-processing**

1. Selected 166 feature columns from the given data
2. Feature selection (manual)
   1. Plotted a correlation heatmap of the features
   2. Dropped all features having a correlation value greater than 0.9
3. Feature selection (PCA)
   1. Selected 100 features by performing principal component analysis
4. At this stage, I had two sets of data – PCA data and manually filtered data
5. Performed train and test set split on both pieces of data separately
   1. Train-Test split ration – 80:20
6. Performed Standard- Scaling – since the input feature values varied considerably, standard scaling was needed so as to eliminate the possibility of the model thinking that a feature with higher value holds more weight/importance, thereby giving skewed results

**The model – Multilayer Perceptron**

Given the kind of data – MLP felt like the best choice in this case. There was enough data and enough number of features for a deep learning model like MLP to be trained on. CNN and RNN were not chosen because CNN is mostly used for image data and RNN is for capturing sequential dependence.

The same model was used for both the PCA data and manually selected feature data.

**Description of the model:**

1. Implemented using- Keras API, Tensorflow backend
2. No. of layers: 3 hidden layers
3. Nodes: Most of the time, the number of nodes in the first hidden layer is 2/3 of the input size. This guided the selection of number of nodes.
4. Activation – Relu
5. Output layer activation – Sigmoid (Since classification task was being performed, hence probabilities were needed)
6. Loss function – Binary Cross- entropy (for binary classification)
7. Optimizer – Adam
8. Batch size - 1
9. Epochs – 15
10. Weight initialisation – from Random initialisation from normal distribution

\*Final performance scores – Validation loss, Loss, Accuracy, Validation accuracy, F1 score, Precision, Recall – for both the cases are provided in the included jupyter notebook.

\*Graphs are provided separately – for both cases

\*Graphs are also included in the jupyter notebook