Dataset 2: Default of Credit Card Clients Dataset

Classification Problem: This dataset contains information on default payments, demographic factors, credit data, history of payment, and bill statements of credit card. Problem to predict whether the client will default the payment next month. Class 0 (No Default): 23365 observations, Class 1 (Default): 6637 observations

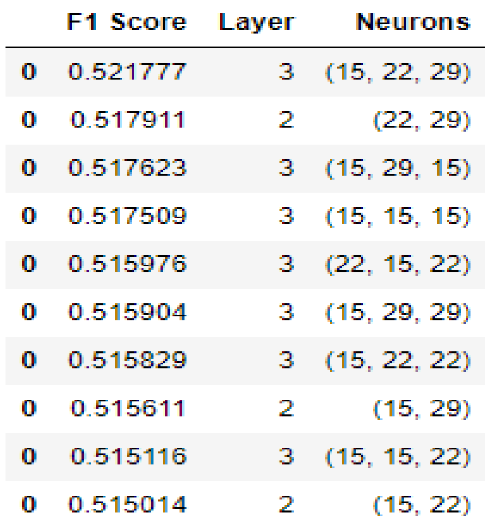
Exploratory Data Analysis:

- It is a highly imbalanced dataset with Class 0 contributing to 78% of the observations and Class 1 with 22%
- Grouping the less frequently occuring class under Other for the features Education and Marriage.
- Creating Dummy variables for the categorical features Sex, Education and Marriage.
- SMOTE sampling the target class for K-Nearest Neighbor algorithm and undersampling the majority class for Artificial Neural Networks Algorithm
- Due to imbalance of the classes in the dataset, F1 score is being considered as the metric for evaluation (F1 score is a function of both precision and recall and we need to know how well the model is predicting the Default Class 1)
- As the data points are in different scales, feature scaling using StandardScalar is performed before the implementaion of both Artificial Neural Networks and K-Nearest Neighbors algorithms.

Artificial Neural Networks (Training is all performed on undersampled data with 60:40 ratio of the classes)

Finding the best combinations of layers and neurons

Parameters: Epoch = 100, Batch_size = 200, optimizer = Adam, learning_rate=0.005, Activation: Sigmoid



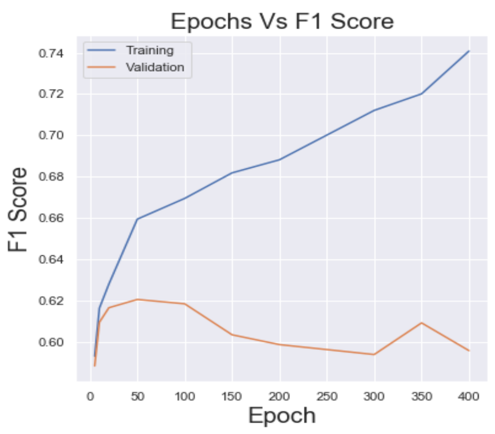
	F1 Score	Layer	Neurons
0	0.521777	3	(15, 22, 29)
0	0.517911	2	(22, 29)
0	0.517623	3	(15, 29, 15)
0	0.517509	3	(15, 15, 15)
0	0.515976	3	(22, 15, 22)
0	0.515904	3	(15, 29, 29)
0	0.515829	3	(15, 22, 22)
0	0.515611	2	(15, 29)
0	0.515116	3	(15, 15, 22)
0	0.515014	2	(15, 22)

Inputs: Layers = [2,3], Neurons = [15,22,29]

36 Combination of neurons-layers were formed based on the above inputs. For each combination of neurons and layers, neural network model was trained and evaluated on the actual test data. Table shows the combinations that had the top 10 F1 Scores on the testing data. But there were very small changes in the F1 scores for all the combinations minimum of 0.49 and maximum of 0.52. These results give us an estimate of the number of neurons and the layers that can used in the future experiments. We can choose **3 Layers with (15, 22, 29) neurons in each of the 3 layers**. Cross Validation is not performed here as it is computational expensive considering there would be 36 neural network algorithms to be run for all the combinations

Hyperparameter Tuning to find the optimal number of epochs

Parameters: Batch_size=200, optimizer=Adam, learning_rate=0.01, Activation: Sigmoid, Layers = 3, Neurons = (15,22,29)



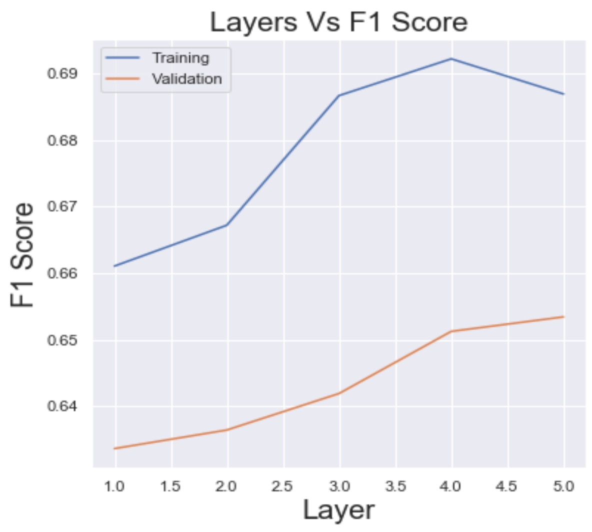
Inputs: Layers = 3, *Neurons = [15,22,29]*

Model implementation with cross validation to find the optimal parameter

With increase in the epochs, training F1 scores of the undersampled data are increasing upto 0.74 with 400 epochs. But, **validation F1 scores have reached its saturation with 50 epochs** and there is a decrease in the validation scores after 50 epochs. This means that the model is not generalising well after 50 epochs. We choose 50 epochs as the optimal parameter for our future experiments.

Hyperparameter Tuning to find the optimal number of layers

Parameters: Epoch: 50, Batch_size=200, optimizer=Adam, learning_rate=0.01, Activation: Sigmoid, Layers = 3, Neurons = (15,22,29)



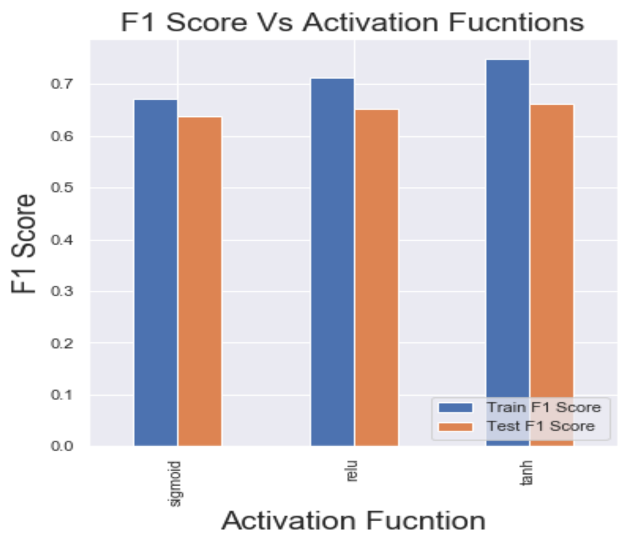
Layer: [1,2,3,4,5]

Model implementation with cross validation to find the optimal parameter

With increasing number of layers, the training and validation F1 scores on the undersampled data have an increasing trend. The model has optimised when the number of layers reached 4. **Both the training and validation accuracies are decreasing after 4 layers**. Cross Validation is helping us to ensure the model is generalising the data well and not having any overfitting issues.

Hyperparameter Tuning to find the optimal Activation Fuction

Parameters: Epochs = 50, Batch_size=200, optimizer=Adam, learning_rate=0.01, Activation: Sigmoid, Layers = 3, Neurons = (15,22,29)



Activation Functions: ['sigmoid','relu', tanh]

Model implementation with cross validation to find the optimal parameter

The training F1 score for tanh is better than sigmoid and relu activation functions. But the validation F1 scores are almost the same across the functions. Hence, **the model is approximately performing the same with any of the any activation functions**. For this dataset, the model is independent on the activation function and any of these fuctions can be used
Note: The activation function of the output layer is always kept as Sigmoid, as this a binary classification problem and we want the output to range from 0 to 1.

Using the Optimal parameters to find the Best ANN Model

Optimal Parameters: Epochs: 50, Activation Function: Sigmoid, Layers: 4, Neurons: (15,22,29,15), Batch Size: 200, Learning Rate: 0.01, Loss: Binary_crossentropy, Optimizer: Adam

Classification Report

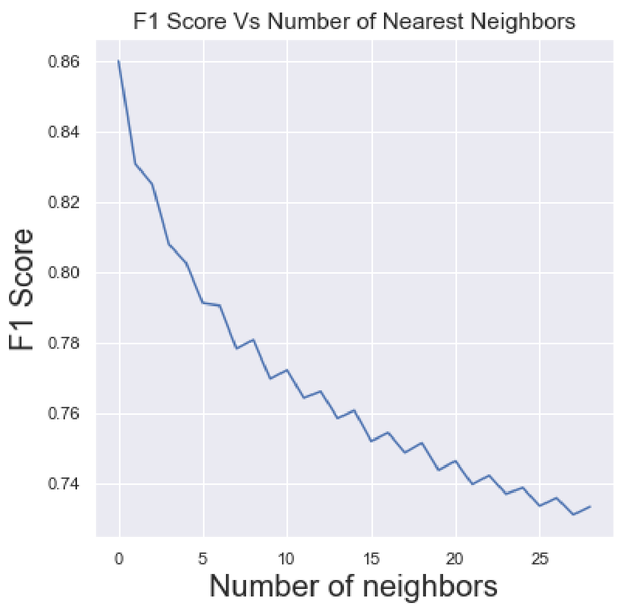
```
              precision    recall  f1-score   support

           0       0.86      0.87      0.87      7000
           1       0.53      0.51      0.52      2000

    accuracy                           0.79      9000
   macro avg       0.70      0.69      0.69      9000
weighted avg       0.79      0.79      0.79      9000
```

Confusion Matrix

	Predicted 0	Predicted 1
Actual 0	6095	905
Actual 1	978	1022

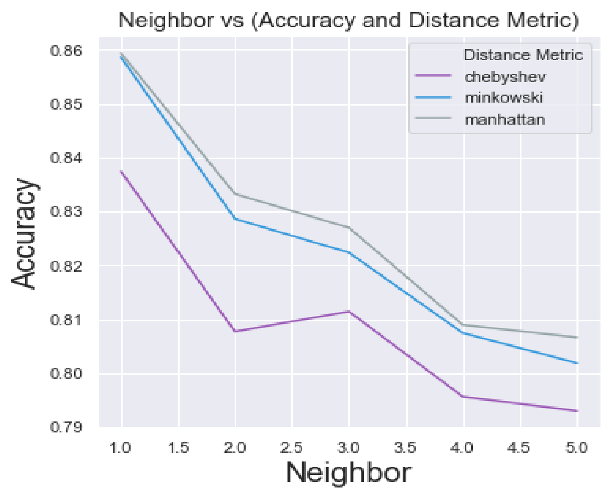
<u>K-Nearest Neighbors (Training is all performed on the SMOTE sampled data)</u>

<u>Hyperparameter Tuning to find the optimal number of neighbors</u>



Neighbors: 1 to 30
Distance Metric: Minkowski, Weight: Uniform
<u>Model implementation with cross validation to find the optimal parameter</u>
Validation accuracy implemented with cross validation on the SMOTE sampled data is continously decreasing with increase in the number of nearest neighbors. Although, the validation score is the highest with 1 neighbor, it may lead to overfitting in most cases and as the decision boundary will not be clearly defined with just 1 neighbor. So, we prefer **3 neighbors as the Optimal number of neighbors**. And as the number of neighbors is increasing, the model is trying to fit the noise more than generalising the data. For example, when the number of neighbors is equal to the number of observations in the dataset, the training accuracy will be one, but that is a case of overfitting as the validation accuracy will be very poor.

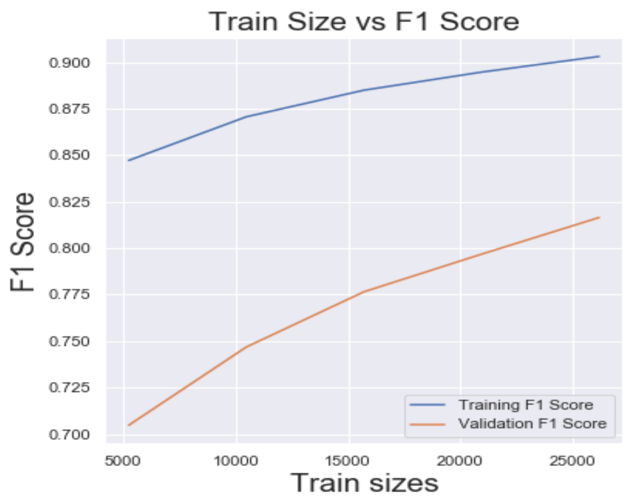
<u>Hyperparameter Tuning to find the optimal Distance Metric</u>



Neighbors: 1 to 10, Weight: Uniform
Distance Metrics: ['chebyshev','minkowski','manhattan']
<u>Model implementation with cross validation to find the optimal parameter</u>

Distance Metric Manhattan is performing the best for this dataset. Manhattan distance is the sum of the absolute difference between the data points. The default distance metric "minkowski" is also performing well with its validation accuracies very close to manhattan.

<u>Learning Curves: Train Size Vs Accuracy</u>



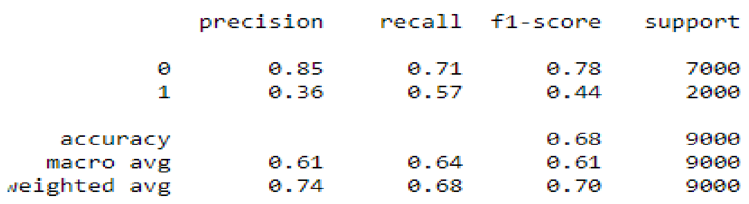
Neighbors: 3, Weight: Uniform
Distance Metrics: 'manhattan'
<u>Model implementation with cross validation to find the performance with increasing training sizes</u>

The model is continously learning and generalising the data well with increasing training sizes as we have both training and validation F1 score with a gradual upward trend. Looks like the model has not reached the saturation yet and the model may tend to learn more with even more data points.

Model implementation with the Best Parameters

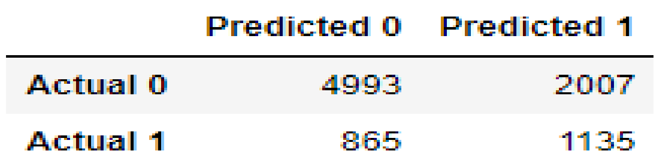
Optimal Parameters: Neighbors: 3, Distance Metric: Manhattan, Weight: Uniform

Classification Report



```
              precision    recall  f1-score   support

           0       0.85      0.71      0.78      7000
           1       0.36      0.57      0.44      2000

    accuracy                           0.68      9000
   macro avg       0.61      0.64      0.61      9000
weighted avg       0.74      0.68      0.70      9000
```

Confusion Matrix



	Predicted 0	Predicted 1
Actual 0	4993	2007
Actual 1	865	1135

Comparison of the model performances of all the implemented algorithms

Note: Data Cleaning and the metrics used to evaluate is the same across all the algorithms

Ranking based on the Test Recall scores for all the Algorithms

Why is Recall Score used here?

It is metric that determines how well the classifier was able to predict the specific target class. For this dataset, our class of interest is to determine and find the users that are going to Default in the next month. We do not worry much if user who is not going to default is classified as Default as its misclassification cost is very low. The goal of the model is to know how well the model can predict the Default Class 1.

Classifier	F1 Score	Recall Score
Boosted Decision Tree	0.50	0.60
Decision Tree	0.52	0.57
K-Nearest Neighbor	0.44	0.57
Artifical Neural Network	0.52	0.51
Radial SVM	0.52	0.49

Is ANN and KNN performing better than the other 3 algorithms?

Based on the recall scores, both ANN and KNN algorithms are not able to perform better than the Boosted Decision Tree. Recall score of KNN is same as the recall score of the Decision Tree.

Observations of all the 5 algorithms

- Overall, the performance of all the algorthims is not really great on this dataset as the maximum recall score of only 0.60 was achieved by the Boosted Decision Tree.
- This is problem of having an imbalanced dataset with a smaller number of significant features in the dataset.
- Sampling of the dataset helped the algorithms to perform better as there was approximately 10% increase in the F1 scores for all the algorithms implemented. But even after this increase, the model's performance does not seem be great.
- ANN algorithm was not able to improve its performance even its various combination of layers and neurons (i.e different network architechture).
- KNN algorithm seemed to have the least F1 score of 0.44 compared to the F1 scores of the other algorithms. And prediction of the target class was quite time consuming compared to the first dataset as this dataset had a total of 29 features.

How the Model Performance can be improved

- When more data with the minority class is available along with more relatable features present in the dataset.
- Performing some feature transformation may help improve the performance, athough it is not guaranteed. Feature transformation was tried on some of the features, but it did not help in improving the model performance