Recognising future Credit Card Defaults through Multilayer Perceptron's and Support Vector Machines

Description and motivation of the problem

The ability to recognise when a customer is likely to default on a payment is of great importance to lenders and customers alike, it allows the lender to identify and contact customers to ensure they can come to a payment arrangement, minimizing the cost of risk, and improving customer relations. For customers, the added support is vital for mental and physical wellbeing of customers, providing support so that arrears are kept in check, minimising the damage to customer credit profiles and reducing the mental and physical distress that is put on the customer.

The purpose of this paper is to evaluate two models designed for a binary classification task, recognising when a customer is going to default on their next payment based on a pattern of 23 attributes. The models we will be using for this task are Feed-Forward Multilayer Perceptron's (MLP) and Support Vector Machines (SVM).

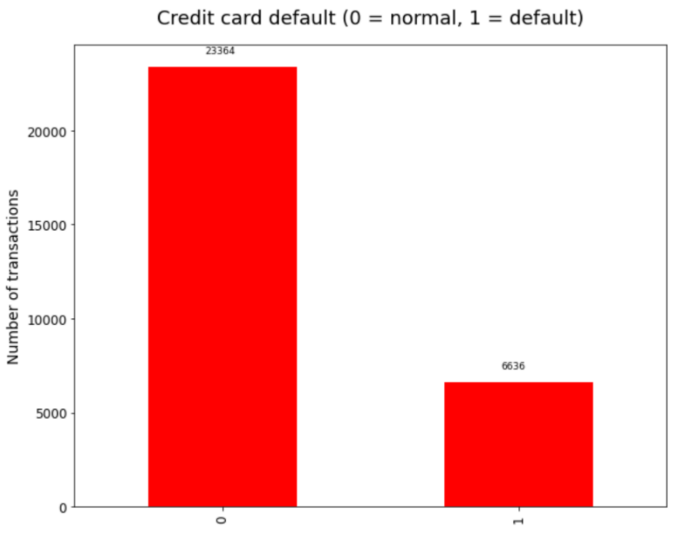
Dataset

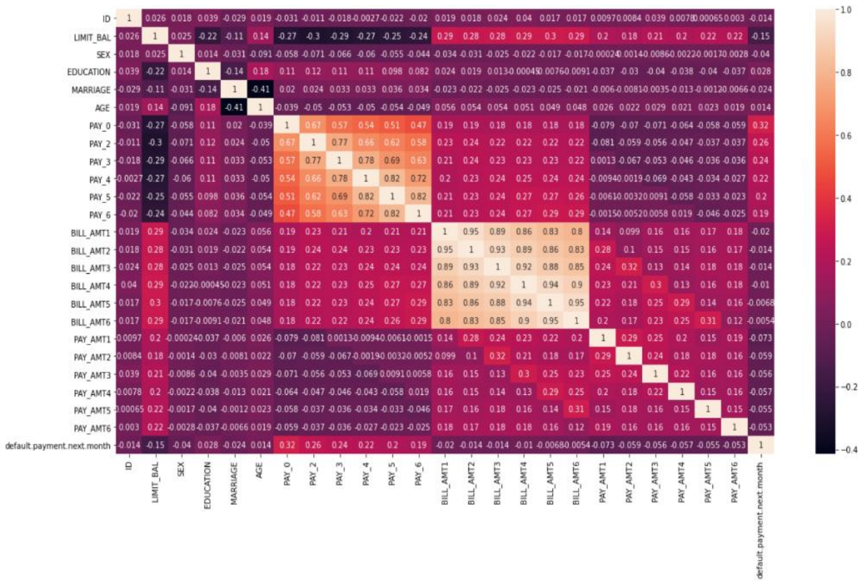
The dataset being used is provided by UCI Machine Learning and can be downloaded from Kaggle. The data contains 23 attributes and 30,000 samples with information on: default payments, demographic factors, credit data, history of payments, and bill statements of credit card clients in Taiwan from April 2005 to September 2005.

- ### Initial Data Analysis

As the table below shows there is an imbalance between the 'target' variable of whether payment has defaulted or not. Due to this imbalance, it was decided that it would be appropriate to use the SMOTE technique to balance the data and aid classification.

Synthetic Minority Over-sampling Technique (SMOTE) is used when there is a dominant feature that will complicate results from a learning algorithm. For example, in this case it is possible for our model to have a high degree of accuracy just by correctly predicting the dominant feature, even if it has completely failed at predicting the minority feature. SMOTE essentially oversamples the minority class by synthesizing duplicate values.





This correlation matrix can provide insight into variables which are likely to have the greatest/least impact as predictors of default payments. Pay_0 – Pay_6 are variables which indicate the repayment status of the customer indicating if the customer is paying on time or if there are any payment delays, this feature also has the highest positive correlation with customer defaults and are therefore likely to have the greatest effect on our model. However, as one of the key factors of neural networks is that it does not require much feature engineering, we will be keeping these for the time being.

Algorithms

○ Multilayer Perceptron (MLP)

Multilayer Perceptron's are essentially supervised learning neural networks which consist of an input layer, one or more hidden layers and an output layer. What makes MLP's especially interesting is using these 'hidden' layers, which during learning will extract features and assign weights based on the coefficients of the input layers.

The input layer of a MLP will pass the input values to the hidden layers, these input values are modified based on the weights that are assigned to the hidden neurons, who then determine the output. This output value is then compared with the desired target output, producing a loss function which is then back propagated back through the network adjusting the weights. After several runs the network will be trained to produce an output depending on the input values.

Multilayer Perceptron's are useful in detecting with non-linear relationships between features, adjusting the weights and activation functions applied in every hidden layer, which in the past has been very useful for computer vision (which has now been almost completely replaced by Convolutional Neural Networks). However, the algorithm can be seen as difficult to interpret due to its complexity and as neural networks can be prone to overfitting it generally requires large amounts of data to perform well.

○ Support Vector Machines (SVM)

SVM's are supervised learning algorithms that can be used for classification and regression problems, and although it is technically not regarded as a neural network it is indeed held in high regard. Support Vector Machines in the context of a binary classification task will find and maximise the margin distance between two classes. Utilising the kernel trick allows SVM's to solve non-linear classification problem by introducing another dimension so that the data is then able to be divided by a hyperplane. The algorithm performs especially well on smaller datasets, datasets with sample sizes larger than 10,000 will find It will require a lot of computational power. SVM's are also not able to handle noisy data with overlapping classes as affectively as other algorithms, however, unlike MLP's it does not get stuck in local minima's, always producing a global minima.

Hypothesis statement

There are a lot of studies about the prediction of credit card defaults and similarly credit card fraud, which typically includes a binary classification problem solved by SVM's and MLP's. In the *Egyptian Informatics Journal, E.A. Zanaty's Support Vector Machines versus Multilayer Perception in data classification* suggests that when it comes to data classification SVM's will perform better when compared to MLP's as they are able to 'generate near the best classification as they obtain the optimum separating surface which has good performance on previously unseen data'.

Initial post-SMOTE models show accuracy of 61.5% for MLP and 68.6% for SVM, therefore it is expected that SVM will perform better than MLP once model parameters are refined but likely doing so marginally. It is also expected that training SVM models will take significantly longer to train when compared to MLP models, this is largely due to the fact that as per scikit-learn documentation SVM algorithms are based on libsvm and therefore *'the fit time scales at least quadratically with the number of samples and may be impractical beyond tens of thousands of samples'*.

Methodology

To ensure that the data is suitable for our models we first had to do some pre-processing. Columns that were not necessary were removed, checks indicated there were not missing values that needed removing and finally the data was split into input features and our target outputs. As the values in our data varied so drastically it was important to bring them to an appropriate scale.

Now that the data is set to an appropriate scale the data is split into training and test sets with the test set accounting for 20% of the data. For the process of model selecting a grid search approach was implemented to adjust hyperparameter selection for both MLP and SVM models. For validating our models, a 10-fold cross-validation method is used, this essentially divides the training and test data by the same number of samples with the appropriate proportion of the target class. Cross-validation is usually seen as the preferred method as it allows the model to train on multiple 'splits' of the data, an average of the cross-validation accuracies is then taken to aid model selection.

- **MLP**

Pytorch was used to design the architecture and parameters used for our MLP models. The initial model was designed one input layer (there is no theoretical reason to use more than two) with 23 neurons to take the 23 sets of input features, one hidden layer containing 16 neurons and one single output, this was essentially as we are looking to do a binary classification task, we will want the outputs to be either 1 or 0 (default or no default). The activation function used is ReLU as it is generally more computationally efficient than sigmoid functions and it generally performs better. Dropout was also considered; this method essentially randomly drops neurons and is a good tool for preventing overfitting.

To avoid networks being stuck in local minima MLP networks will start with random weights each time they are trained, therefore performance will differ each time the model is trained.

Skorch is a neural network library that wraps the PyTorch model and is used to train our network. To train our MLP models we are using Skorch's NeuralNetBinaryClassifier to wrap our model, the training runs for a maximum of 50 epochs, we say maximum as we are using early stopping so that training is stopped should the valid loss not improve for 5 consecutive epochs, this will make the training more computationally effective as well as preventing overfitting. An initial learning rate of 0.1 and an optimizer momentum rate of 0.8 was selected.

- **SVM**

Our Support Vector Machine (SVM) model has been created using Scikit-learn’s SVM module where we have tuned and adapted its architecture and hyper-parameters on our training data to identify the model that best classifies our target variable and is generalisable to unseen data. Unlike our MLP models, SVM does not start with random weights, therefore does not encounter the issue of entering and random local minima and do not have as many training parameters available as our MLP models do. Also, unlike MLP models SVM has a very slow training time especially on large training data samples, one method used to speed up computation is the use of LinearSVC which incorporates the linear kernel and implementing it in a liblinear rather than a libsvm method which scales better to a larger number of samples. However, the linear kernel is only one of the hyper-parameters we are looking to identify the best SVM model, the hyper-parameters chosen to evaluate are SVM’s kernel functions, Gamma and C.

Evaluation

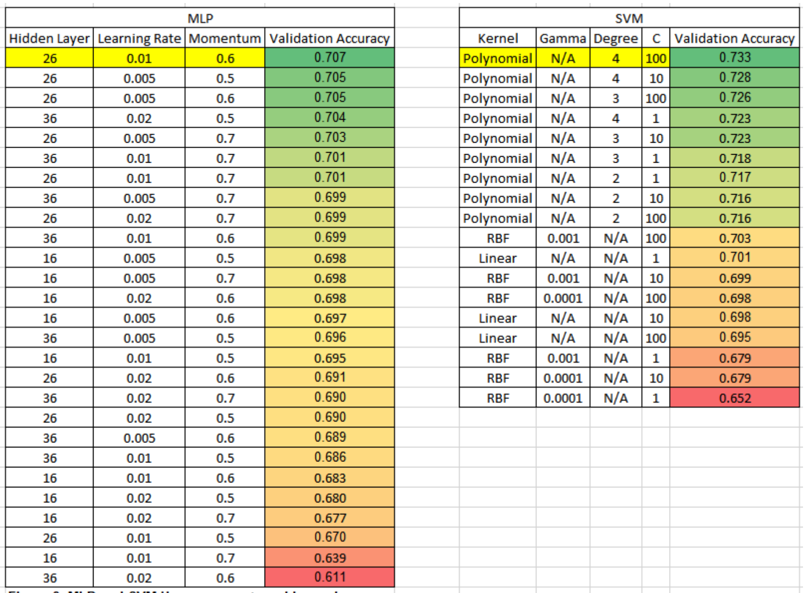
Given that the dataset is imbalanced it is important to evaluate the models appropriately, prior to utilising SMOTE on the training data the MLP model was showing an accuracy of 81.5%, however after plotting a confusion matrix it was clear that there was an imbalance in the classification with the minority class being drastically mis-classified.

Once the minority class was oversampled using SMOTE accuracy drastically dropped to 61.5%, however, the classification of the target classes was more balanced. As only the testing data has not been oversampled it is more important to evaluate the models based on the True Positive and False Positive rate of classification. To evaluate and compare the best models a variation of the following hyperparameters were chosen and implemented in a grid search fashion: the size of the hidden layers, learning rate and momentum for our MLP models and kernel functions, Gamma and C for our SVM models.

- **Model Selection**

The table below shows the grid search results conducted on both our MLP and SVM models, allowing us to select the model with the highest accuracy but also the model which best classifies our target variables using cross validation.

The MLP grid search is testing the validation accuracy for a combination of hyper-parameters, focusing on the number of neurons in the hidden layer (16,26,36), the learning rates (0.005, 0.01, 0.02) and the momentum applied (0.5,0.6,0.7).



MLP			
Hidden Layer	Learning Rate	Momentum	Validation Accuracy
26	0.01	0.6	0.707
26	0.005	0.5	0.705
26	0.005	0.6	0.705
36	0.02	0.5	0.704
26	0.005	0.7	0.703
36	0.01	0.7	0.701
26	0.01	0.7	0.701
36	0.005	0.7	0.699
26	0.02	0.7	0.699
36	0.01	0.6	0.699
16	0.005	0.5	0.698
16	0.005	0.7	0.698
16	0.02	0.6	0.698
16	0.005	0.6	0.697
36	0.005	0.5	0.696
16	0.01	0.5	0.695
26	0.02	0.6	0.691
36	0.02	0.7	0.690
26	0.02	0.5	0.690
36	0.005	0.6	0.689
36	0.01	0.5	0.686
16	0.01	0.6	0.683
16	0.02	0.5	0.680
16	0.02	0.7	0.677
26	0.01	0.5	0.670
16	0.01	0.7	0.639
36	0.02	0.6	0.611

SVM				
Kernel	Gamma	Degree	C	Validation Accuracy
Polynomial	N/A	4	100	0.733
Polynomial	N/A	4	10	0.728
Polynomial	N/A	3	100	0.726
Polynomial	N/A	4	1	0.723
Polynomial	N/A	3	10	0.723
Polynomial	N/A	3	1	0.718
Polynomial	N/A	2	1	0.717
Polynomial	N/A	2	10	0.716
Polynomial	N/A	2	100	0.716
RBF	0.001	N/A	100	0.703
Linear	N/A	N/A	1	0.701
RBF	0.001	N/A	10	0.699
RBF	0.0001	N/A	100	0.698
Linear	N/A	N/A	10	0.698
Linear	N/A	N/A	100	0.695
RBF	0.001	N/A	1	0.679
RBF	0.0001	N/A	10	0.679
RBF	0.0001	N/A	1	0.652

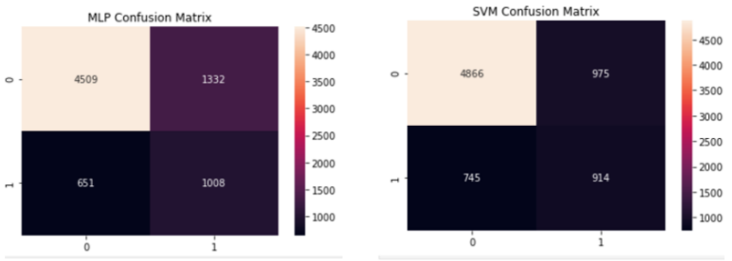
Figure 3: MLP and SVM Hyper-parameter grid search

For our SVM grid search we are primarily seeing how the different kernels; Radial Basis Function (RBF), Linear and Polynomial affect classification of the target variables. These kernels have then been tested with the following range of parameters; Regularisation parameter ‘C’, Degree (for the polynomial kernel) and the kernel coefficient ‘Gamma’. Due to the computation time taken to run SVM grid searches it was decided to mainly focus on Polynomial kernels as initial models suggested this would be the most effective.

- **Model Comparison**

The two best models applied to the post-SMOTE training data have been selected based on their validation accuracy (Figure 3) and their performance has been further evaluated through confusion matrices, precision, recall, f1-Scores and Receiver Operating Characteristics Curves (Figure 4).

Although validation accuracy is an important metric, when dealing with imbalanced data it is more important to examine how accurately each model is correctly identifying the target classes. As shown in figure 4 the MLP model is better at classifying the default class (1), correctly classifying 1,008/1,659 = 61%, where the SVM model was only able to correctly classify 914/1,659 = 55%. However, where the MLP model struggles is correctly classifying the non-default majority class (0), only correctly classifying 4,509/5,841 = 77%, compared to the SVM models 4,866/5,841 = 83%. F1-Scores are also marginally better for the SVM model, this is attributed to the higher recall of the non-default class and the marginally higher precision score for the default class. Although the SVM model would appear to perform better, it is important to consider the goal of the model, it may be more important for banks to have a higher recall rate, being more important to correctly predict more defaulters if it means incorrectly predicting non-defaulters. Hence, as the MLP algorithm performs better at detecting the minority class it may be more appropriate for this kind of problem.



```
              precision    recall  f1-score   support

         0.0       0.87      0.77      0.82      5841
         1.0       0.43      0.61      0.50      1659

    accuracy                           0.74      7500
   macro avg       0.65      0.69      0.66      7500
 ighted avg       0.78      0.74      0.75      7500
```

```
              precision    recall  f1-score   support

           0       0.87      0.83      0.85      5841
           1       0.48      0.55      0.52      1659

    accuracy                           0.77      7500
   macro avg       0.68      0.69      0.68      7500
weighted avg       0.78      0.77      0.78      7500
```

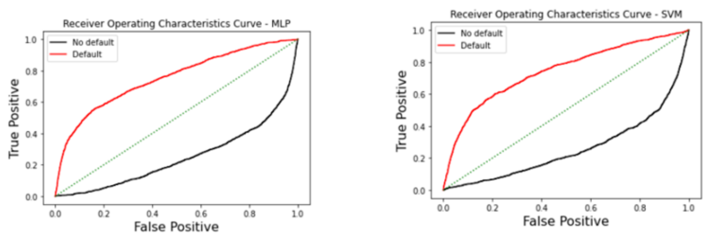


Figure 4: Confusion Matrix, Precision, Recall, F1-Score and ROC Curve.

Conclusion

The purpose of the study has been to determine how accurately two trained models (MLP vs SVM) can predict future credit card payment defaults based on unseen data. This is no easy task and there is a significant amount of research on imbalanced data, including fraud and cancer detection. Throughout the study, 45 models were created and through grid-search the best MLP and SVM models were chosen to further analyse. Although the SVM model had an overall better accuracy rate, it is important to consider the application of the models, to credit card issuers it is more expensive to incorrectly predict the non-default class (false negative). Therefore, it may be more appropriate to utilise the MLP model, which is more suited to predicting the minority class (defaults).

Due to computational limitations, we have learned that SVM algorithms do not scale well with larger data sets, we were able to speed up computations by increasing the cache sizes used, however this was only marginally effective, and it is more important to narrow your search, which is why once it was determined that Polynomial kernels showed better performance, we decided to limit our grid search.

For future work it would be interesting to do some further feature engineering, perhaps remove each input feature one at a time to determine its affect on the model to determine which features to keep and which to remove. Due to computational limitations, we were also only able to utilise 3 stratified k folds for our validations, which is usually seen as the bare minimum, ideally it would be best to 10. Ultimately testing a wider variety of hyper-parameters for our models would likely be the most effective at discovering better models.

References

Zanaty, E., 2012. Support Vector Machines (SVMs) versus Multilayer Perception (MLP) in data classification. *Egyptian Informatics Journal*, 13(3), pp.177-183.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, E. (2011) "Scikit-learn: Machine Learning in Python", *Journal of Machine Learning Research*, 12, pp. 2825--2830.

sklearn.svm.LinearSVC — scikit-learn 0.24.1 documentation (2021) *Scikit-learn.org*. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html#sklearn.svm.LinearSVC (Accessed: 5 April 2021).

Yildirim, S. (2020) "Hyperparameter Tuning for Support Vector Machines — C and Gamma Parameters", *Towards Data Science*. Available at: https://towardsdatascience.com/hyperparameter-tuning-for-support-vector-machines-c-and-gamma-parameters-6a5097416167 (Accessed: 5 April 2021).

Christmann, G. (2019) *LinearSVM vs PolySVM - Taiwan Credit Card Defaults*, *Kaggle.com*. Available at: https://www.kaggle.com/guichristmann/linearsvm-vs-polysvm-taiwan-credit-card-defaults (Accessed: 3 April 2021).

Heaton, J. (2017) *The Number of Hidden Layers*, *Heaton Research*. Available at: https://www.heatonresearch.com/2017/06/01/hidden-layers.html (Accessed: 1 April 2021).

Brownlee, J. (2021) *How to Use ROC Curves and Precision-Recall Curves for Classification in Python*, *Machine Learning Mastery*. Available at: https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classification-in-python/ (Accessed: 6 April 2021).

Multilayer Perceptron (MLP) vs Convolutional Neural Network in Deep Learning (2021) *Medium*. Available at: https://medium.com/data-science-bootcamp/multilayer-perceptron-mlp-vs-convolutional-neural-network-in-deep-learning-c890f487a8f1 (Accessed: 5 April 2021).

Cruz, R. (2021) *SVM using scikit learn*, *Data Science Stack Exchange*. Available at: https://datascience.stackexchange.com/questions/989/svm-using-scikit-learn-runs-endlessly-and-never-completes-execution (Accessed: 3 April 2021).

Osuna, E., Freund, R. and Girosi, F. (1997) "Support Vector Machines: Training and Applications", *MASSACHUSETTS INSTITUTE OF TECHNOLOGY ARTIFICIAL INTELLIGENCE LABORATORY and CENTER FOR BIOLOGICAL AND COMPUTATIONAL LEARNING DEPARTMENT OF BRAIN AND COGNITIVE SCIENCES*.

Fernández, A., Krawczyk, B., García, S., Galar, M., Herrera, F. and Prati, R. (2018) *Learning From Imbalanced Data Sets*. 1st ed. Springer, p. 126.

Appendix

1. Glossary

Binary Classification	The process of assigning a series of elements into one of two groups.
Hyperplane	A boundary which helps to classify data.
Gamma	Determines the level of influence of samples selected by the model.
C	Determines by how much you want to avoid misclassification. Large values of C will lead to a smaller-margin hyperplane leading to higher accuracy.
Degree (Polynomial)	Determines the flexibility of the hyperplane boundaries.
Linear Kernel	Kernel used in SVM for classification problems, it is the fastest of all the kernels and is used when the data is linearly separable.
Polynomial Kernel	Kernel used in SVM for classification problems, represents training data over polynomials of the original data, used for non-linearly separable data.
Radial Basis Function (RBF)	Kernel used in SVM for classification problems, calculates the distance between points to determine the optimal hyperplane.
Local/Global Minima	Local Minimas of a function are points where the function value is the smallest in comparison to nearby points, however it can still be larger than more distant points. Global Minimas of a function are where the function value is the lowest in comparison to all other points.
Cross-Validation	Used to evaluate machine learning models by splitting the training and test data into several groups(folds) and testing each fold to determine the average performance.
Epoch	A hyper-parameter that determines the number of times an algorithm works through the training data.
Learning Rate	A hyper-parameter that determines the step size of each epoch while moving towards a loss function minimum.
Rectified Linear Unit (ReLu)	Activation function used in algorithms that does not activate all neurons at the same time, seen as more efficient compared to sigmoid functions.
Optimizers	Methods used to change hyper-parameters of networks such as weights and learning rates to reduce the loss function minimum.
Optimizer Momentum	Accelerates gradient descent to assist in finding the global minima.

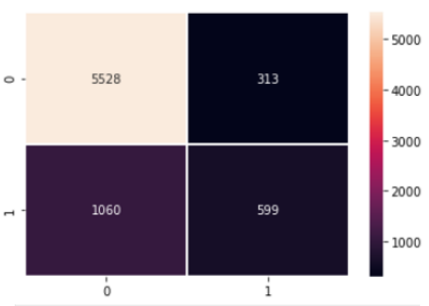
2. Implementation details

After pre-processing the dataset to ensure there were no missing values and that the data was on an appropriate scale it was important to set a baseline for our model performance, therefore we began by creating a simple feed forward multilayer perceptron network with a single hidden layer and this network was applied without up sampling the minority just to see how the network performed.

Initially it appeared that the network performed exceptionally well, reporting an accuracy rate of 82%, however this was only due to the high level of accuracy classifying the majority class. The minority class was only correctly classified 599/1,659 = 36%, this was too low for our use case.

That initial pre-processing had already identified that there was an imbalance in the data, with the minority target class (defaulters) accounting for 28.4% of the data to 71.6% for majority target class (non-defaulters), this was common among this type of dataset, such as credit card fraud data and medical classification tasks therefore balancing the data using oversampling techniques was the next logical step.

MLP - Confusion Matrix (No SMOTE)



```
              precision    recall  f1-score   support

         0.0       0.84      0.95      0.89      5841
         1.0       0.66      0.36      0.47      1659

    accuracy                           0.82      7500
   macro avg       0.75      0.65      0.68      7500
weighted avg       0.80      0.82      0.80      7500
```

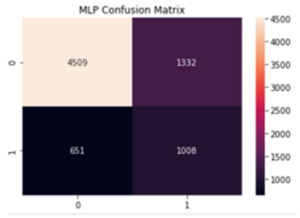
Therefore, the next step was up sampling the minority class, the method implemented was Synthetic Minority Oversampling Technique (SMOTE) which essentially duplicates the examples in the minority class, so the model has more examples available during training.

What is important to remember is that the initial model trained without SMOTE was not the best model identified by performing the grid search. The figure below illustrates the performance of both best models from MLP and SVM identified by a grid search on data that has not been over sampled.

Interestingly, our best MLP models seem to be practically unaffected by the effects of SMOTE, when the sample is imbalanced MLP models perform significantly better than SVM, as stated in *Alberto Fernandez et. al. Learning from Imbalance Data Sets,* SVMs *'will favour the majority class, as concentrating on it will lead to a better trade-off between classification error and margin maximization, this will come at the expense of the minority class'*. Performance of SVM's on imbalanced datasets can potentially be further improved by increase the values of C which determine the tolerance of error for its margin.

Best Models (SMOTE)

MLP Confusion Matrix

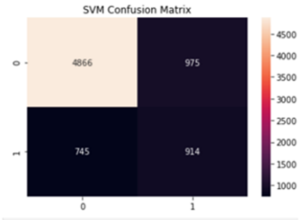


```
              precision    recall  f1-score   support

         0.0       0.87      0.77      0.82      5841
         1.0       0.43      0.61      0.50      1659

    accuracy                           0.74      7500
   macro avg       0.65      0.69      0.66      7500
ighted avg       0.78      0.74      0.75      7500
```

SVM Confusion Matrix



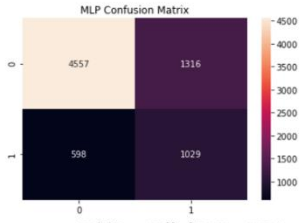
```
              precision    recall  f1-score   support

           0       0.87      0.83      0.85      5841
           1       0.48      0.55      0.52      1659

    accuracy                           0.77      7500
   macro avg       0.68      0.69      0.68      7500
weighted avg       0.78      0.77      0.78      7500
```

Best Models (No SMOTE)

MLP Confusion Matrix

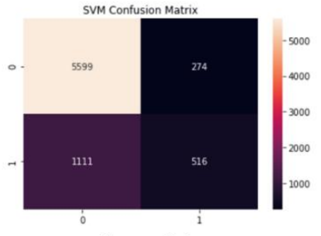


```
              precision    recall  f1-score   support

         0.0       0.88      0.78      0.83      5873
         1.0       0.44      0.63      0.52      1627

    accuracy                           0.74      7500
   macro avg       0.66      0.70      0.67      7500
weighted avg       0.79      0.74      0.76      7500
```

SVM Confusion Matrix



```
              precision    recall  f1-score   support

           0       0.83      0.95      0.89      5873
           1       0.65      0.32      0.43      1627

    accuracy                           0.82      7500
   macro avg       0.74      0.64      0.66      7500
weighted avg       0.80      0.82      0.79      7500
```