A Comparative Study of Support Vector Machines and Multilayer Perceptron for Predicting Direct Marketing Response in Banking

Tareq Galala

Abstract

This paper presents a comparative study between two algorithms to predict whether customers will sign to a long-term deposit with the bank. In our work, we used Support Vector Machine (SVM) and Multilayer Perception Neural Network (MLP). A grid search was conducted for both algorithms to optimize the hyperparameters. The best models were evaluated against a test set and critically evaluated by their F-Measure and ROC Area values (AUC). Results show that MLP achieved the highest value of F-Measure and AUC.

1. Introduction

Bank marketing campaigns are very useful finding customers willing to subscribe to new or existing products, but this sometimes poses a challenge in finding them as they are likely to reject the offers or new products being proposed. Marketing campaigns are important in to increase financial goals and competitive edge for any bank. Typically, under 1% of the entire populace will have a positive reaction to the mass campaign [1]. Selecting prospective customers to subscribe to a direct marketing campaign based on their characteristics is a challenge.

The aim is to critically evaluate and compare two models, to classify and predict whether the client will subscribe to a term deposit or not. This classification task will use namely Support Vector Machines (SVM) and Multilayer Perceptron (MLP). Many models will be evaluated in the training stage by conducting Bayesian optimization and several random searches followed by a grid search for both algorithms, finding the optimal models with the highest area under the curve metric.

The remainder of this paper is organized as follows. This section explores our two algorithms along with the pros and cons. Section 2 describes the dataset used for training and testing the models. Section 3 outlines our hypothesis. Section 4 discuss the methods and techniques on training, and validating as well as describing the hyperparameters used for building our models. Section 5 presents the experiments, evaluation and results. Finally, section 6 concludes and outlines future work.

1.1. Support Vector Machines (SVM)

SVMs are a useful technique for classification problems, which maps an input vector and finds a hyperplane which maximizes the margin distance between two classes. Two parallel hyperplanes are constructed on each side of the hyperplane that separate the data. The separating hyperplane is the hyperplane that maximize the distance between the two parallel hyperplanes [2]. An assumption is made that the larger the margin or distance between these parallel hyperplanes the better the generalization error of the classifier will be [3]. SVM generalizes very well with fewer observations. When the optimality is rounded, SVM can produce a unique solution. This forms a fundamental difference of SVM and Neural Networks, which produce multiple solutions based on local minima [4]. However, a common disadvantage is the lack of transparency in results.

1.2. Multilayer Perceptron (MLP)

A multilayer perceptron (MLP) is an artificial neural network. They are composed of an input layer, an output layer that makes a prediction about the input, and in between those two, a number of hidden layers that are the true computational engine of the MLP.

The main advantage is they can be used for difficult to complex problems. However, they need more computational power and are prone to overfitting. Table 1, presents some of the advantages and disadvantages of SVM & MLP that we encountered in our experiments.

Model	PROS	CONS
Support Vector Machines (SVM)	• High accuracy on small datasets • Wide range of kernel functions • Convex optimization problem & memory efficient • Effective in high dimensional spaces	• Time consuming for large datasets • Less effective with overlapping classes • Kernel complexity • No probabilistic explanation for classification
Multilayer perceptron (MLP)	• High predictive power • General function approximators • Single layer computation time is very fast • The hidden neurons act as feature extractors	• Large number of parameters • Problems of convergence to local optima • Hard to explain output (Black-Box) • Learning rate can be difficult to adjust

Table 1. Pros & cons for support vector machines (SVM) & multilayer perceptron (MLP)

2. Dataset

The data is related with direct marketing campaigns of a Portuguese banking institution retrieved from the University of California Irvine (UCI) Machine Learning Repository. We choose the dataset containing (4521) instances with (17) attributes without missing values. The marketing campaigns were based on phone calls. Often, more than one contact to the same client was required, in order to access if the product (bank term deposit) would be ('yes') or not ('no') subscribed.

2.1. Data Pre-processing

Figure 1 shows the dataset response variable 'y', which clearly indicates imbalanced classes in our dataset. The class **no** has 4000 samples and class **yes** has 521 with a 11.5% class ratio. We applied SMOTE technique on the training data only to balance the classes. The test set remained unseen for a true test without SMOTE.

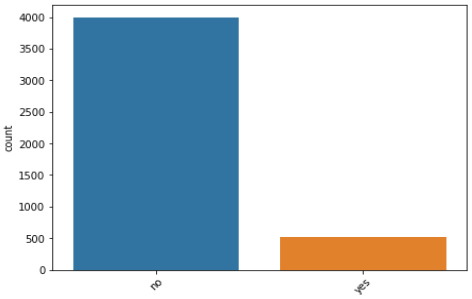

Fig 1. Response variable 'y' class count

	age	balance	day	duration	campaign	pdays	previous
mean	41.17	1422.65	15.91	263.96	2.79	39.76	0.54
std	10.57	3009.63	8.24	259.85	3.1	100.12	1.69
min	19	-3313	1	4	1	-1	0
max	87	71188	31	3025	50	871	25

Table 2. Statistic Summary of the Numerical (continuous) Variables of the Dataset

	job	marital	education	default	housing	loan	contact	month	poutcome	y
count	4521	4521	4521	4521	4521	4521	4521	4521	4521	4521
unique	12	3	4	2	2	2	3	12	4	2
freq.	969	2797	2306	4445	2559	3830	2896	1398	3705	4000

Table 3. Statistic Summary of the Categorical (discrete) Variables of the Dataset

As part of the analysis we separated the numerical variables and categorical variables and performed the statistics accordingly. The categorical values were encoded with a technique called label encoding. Label encoding is simply converting each value in a column to a number. We also performed univariate and bivariate analysis and visualization to our data and checked for correlations between our numerical variables.

In addition, the dataset was split into a stratified 80% training set and 20% test set and then SMOTE was applied only to the training set. The total training set after SMOTE was 5597 observations.

3. Hypothesis Statement

This paper aims to compare SVM and MLP. Our priors should capture our beliefs about a reasonable hypothesis for the two algorithms.

- Both models are expected to have high prediction accuracy and AUC, however, evidence suggests that MLP have better prediction results compared to SVM in classification tasks [6].

- We assume that we will have better results with Bayes optimization.
- It's faster to train an SVM than an MLP model.

4. Methods

In this section, we will describe both models' methodology and describe the methods used to train and validate our dataset. In addition, we will describe the architecture and hyperparameters.

4.1. Methodology

The methodology in our experiments is to hold-out 20% of the dataset for testing, to compare our best performing models for SVM and MLP. The rest of the data (80%) is used for training and model selection. We used stratified sampling in splitting the data to ensure that the train and test sets have approximately the same percentage of samples of each target class as the complete set. In the model selection, we used 10-fold stratified cross-validation method in our grid search to find the optimal hyperparameters. Since the dataset has an imbalance problem, we selected F-measure and AUC as our metric choice to find our optimal model for SVM and MLP.

Different optimization methods were used in our experiments. Firstly, we used a Bayesian Optimization method for both algorithms and secondly, a random search followed by further tuning with a five nested grid search loop to find the best hyperparameter combinations for both algorithms with cross-validation. In addition, the MLP model used the early stopping method to avoid overfitting and to lead to a better generalization.

In the test comparison section, our selected optimal models were then trained on the whole training set, the models get saved in mat format then loaded automatically and a prediction is made on the test set.

4.2. SVM architecture and hyperparameters

A random search followed by more tuned grid search for the optimal SVM was used in finding the following hyperparameters: Kernel function (Linear, Polynomial and RBF), in case of a polynomial, the polynomial order, Box Constraint (C) which is a parameter that controls the maximum penalty imposed on margin-violating observations, which helps to prevent overfitting (regularization). If increased, then the SVM classifier assigns fewer support vectors. However, increasing the box constraint can lead to longer training times [5]. A smaller value gives a more rigid model, less sensitive to overfitting.

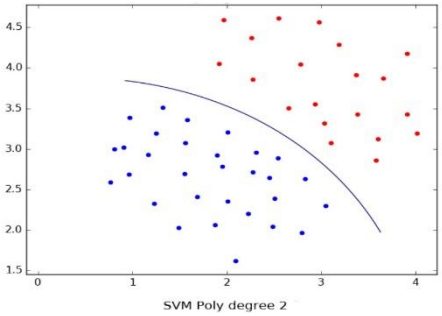

Fig 2. SVM poly degree 2, higher-degree polynomial kernels allow a more flexible decision boundary

4.3. MLP architecture and hyperparameters

The network that we used for our MLP is a two-layer feedforward network, with a hyperbolic tangent sigmoid (tansig) transfer function in the hidden layer, and a log-sigmoid (logsig) transfer function in the output layer with one neuron. Since we have a vector of [0,1] in our target variable then log sigmoid is best option. The performance function used was the cross-entropy which is the default loss function in our net. The number of inputs depends upon the input dataset.

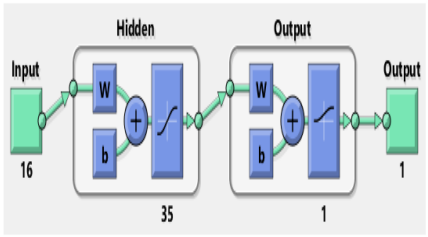

Fig 3. MLP network architecture with one Hidden layer with 35 neurons, one output and 16 inputs

A random search followed by a tuned grid search for the optimal MLP model was used to find the following hyperparameters:

Number of hidden neurons in a layer, training function (trainbfg, trainrp, trainscg, traincgp, traincgb), the learning rate, momentum to avoid local minima and the activation function of the hidden layer (logsig, tansig). The hyperparameters were optimized on the bases of performance and generalization of our model.

5. Results, Findings & Evaluation

5.1. Model Selection

In our model selection part, we used two methods. Firstly, the cross-validation Bayes optimization and secondly a random search followed by a further tuning with grid search. Our experiments found the latter have better results. Table 4 & 5 shows the results from our grid search for SVM and MLP after conducting multiple searches. The metrics shown here are the accuracy, F1-score which is the harmonic mean of precision and recall and Area Under the Curve (AUC) for each combination. The time recorded here is in seconds using parallel computations on two workers.

Support Vector Machine Hyperparameters Grid Search						
Kernel	Order	Box	Accuracy	F1	AUC	Time
polynomial	2	0.5	87.42	0.87	0.92	103.7
polynomial	2	0.75	87.65	0.87	0.92	112.2
polynomial	**2**	**0.8**	**88.37**	**0.88**	**0.93**	**113.3**
polynomial	3	0.5	84.12	0.83	0.87	91.7
polynomial	3	0.75	85.08	0.84	0.88	65.5
polynomial	3	0.8	84.24	0.84	0.88	88.7
polynomial	4	0.5	86.54	0.86	0.90	96.8
polynomial	4	0.75	86.53	0.86	0.90	98.7
polynomial	4	0.8	87.35	0.88	0.93	91.9

Support Vector Machine Hyperparameters Bayesian Optimization						
Kernel	Box	Scale	Accuracy	F1	AUC	Time
RBF	998.58	119.2	86.28	0.86	0.92	0.84

Table 4. Left. SVM grid search. Right. SVM Bayes optimization.

Multilayer Perceptron Hyperparameters Grid Search								
Function	Hidden	learn	momentum	Reg.	Accuracy	F1	AUC	Time
trainrp	35	0.005	0.25	0.15	87.60	0.87	0.94	2.63
trainrp	**35**	**0.005**	**0.35**	**0**	**89.29**	**0.89**	**0.96**	**2.22**
trainrp	35	0.005	0.35	0.15	88.32	0.88	0.95	2.54
trainrp	35	0.015	0.25	0	88.09	0.88	0.95	2.92
trainrp	35	0.015	0.25	0.15	87.07	0.87	0.94	2.15
trainrp	35	0.015	0.35	0	88.96	0.88	0.95	1.96
trainrp	35	0.015	0.35	0.15	88.01	0.88	0.94	2.52
trainbfg	25	0.005	0.25	0	86.48	0.86	0.93	4.64
trainbfg	25	0.005	0.25	0.15	88.63	0.88	0.95	6.93

Multilayer Perceptron Hyperparameters Bayesian Optimization							
Function	Hidden	learn	momentum	Accuracy	F1	AUC	Time
trainrp	23	0.001	0.443	89.67	0.89	0.94	6.69

Table 5. Left. MLP grid search results. Right. MLP Bayes optimization

The best performing SVM model was a second-degree polynomial (quadratic polynomial) and a box constraint of 0.8 with an AUC of 0.93. Although another model with the same AUC with four-degree polynomial was found but we opted for the simpler one according to Occam's razor, and to have a better generalization on our unseen test set. Our experiments demonstrated that a large box constraint (C) leads to a drastic fall in accuracy despite the model flexibility.
In MLP, the highest AUC score obtained from Bayes optimization was 0.94, while our grid search found better hyper parameters with a 0.96 AUC on training set. The model has one hidden layer with 35 neurons, learning rate of 0.005, momentum of 0.35, trainrp function which is a resilient backpropagation training function that updates weight and bias values according to the resilient backpropagation algorithm (Rprop) and regularization set to zero. Initially our results were different with each grid search as random weights were assigned in the first iteration, to overcome this we set the control random number generation seed to default so our results will be reproducible. Smaller learning rates require more training epochs given the smaller changes made to the weights each update, whereas larger learning rates result in rapid changes and require fewer training epochs. Momentum can smooth the progression of the learning algorithm that, in turn, can accelerate the training process.

The best accuracy we obtained from the training stage is 89.29% on the MLP model which is almost the same results from the best model from Elsalamony et al. (2013) paper which they obtained 90.84 from their best MLP model, unfortunately, accuracy was the only metric they used [9].

5.2. Algorithm Comparison & Analysis

We trained both of our optimal models on the whole training set. Fig 4 displays the training and validation error while training. The maximum epochs were set on 150 to avoid overfitting and early stopping was applied. As we see the model stopped at epoch 144 with a cross-entropy of 0.258. At that point the validation curve started to increase which indicates the model started to overfit the data.

The main test was made on unseen data where unlike the training data where we applied SMOTE, here in the test set we kept the original balance.

A confusion matrix and ROC curves are used for evaluating the performance of our classification model. Fig 5 displays the confusion matrix for our two best models. SVM optimal model achieved an accuracy of 83.8% while MLP optimal model achieved an accuracy of 87.2% on the unseen test set. The predictions are compared to the original classes to identify the values of true positive, true negative, false positive and false negative and from those values the confusion matrix was built.

In our task which is predicting a customer purchasing a term deposit, it's important to understand what is more valuable to the client, in the case of reducing costs such as man power then precision is more important. The confusion matrix shows a precision of 46.1% and 37.9% for MLP and SVM, respectively. In the case, that the client target was higher sale even if associated with higher expenses and effort contacting non-buyers then looking at the best TP and FP trade-off will be best. MLP achieved a true positive (TP) of 718 and false positive (FP) of 33, while SVM has a true positive (TP) of 691 and false positive (FP) of 37, thus the better classifier is MLP.

In classification problems AUC - ROC curve is a performance measurement at various thresholds settings. ROC is a probability curve and AUC represent the measure of separability. Higher the AUC, the better the model. We can examine the ROC curve plot in fig 6, the MLP had the highest AUC between the two algorithms at 0.89 while SVM achieved 0.87. While both classifiers performed very well on the test data, the better classifier is MLP for our dataset. Overall, these results support our hypothesis that MLP have better prediction accuracy than SVM in classification tasks.

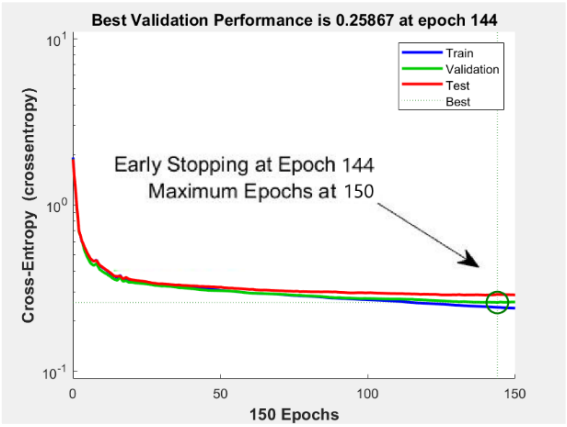

Fig 4. MLP learning curve with early stopping

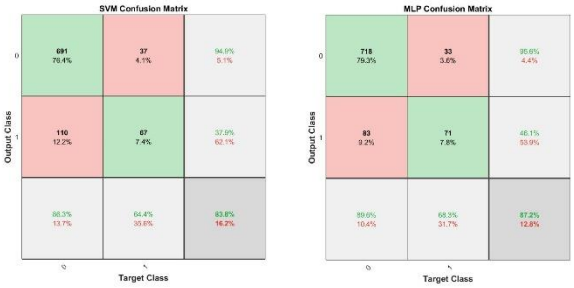

Fig 5. SVM Vs MLP test set confusion matrix

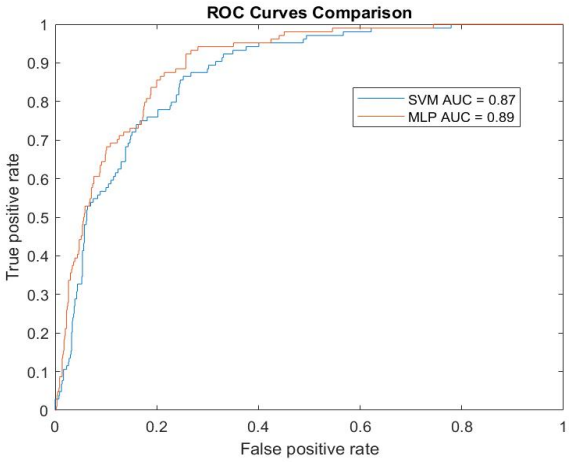

Fig 6. SVM Vs MLP test set ROC curves

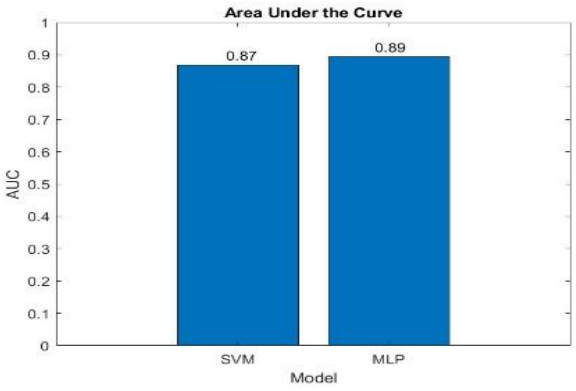

Fig 7. SVM Vs MLP test set AUC bar chart

SVM Vs. MLP Final Test Metrics Comparison				
	Accuracy	F1	AUC	Time
Support Vector Machines Scores (SVM)	83.8	0.90	0.87	131.3
Multilayer Perceptron Scores (MLP)	87.2	0.92	0.89	4.1

Table 6. SVM & MLP final test metrics

In Table 6 the final test metrics are displayed. Our models generalized very well on the unseen test set. The AUC dropped for both models on the test set. MLP model dropped from 0.96 to 0.89, while SVM dropped from 0.93 to 0.87, these drops are within a reasonable threshold. It's worth noting that the time it took SVM to fit the model was more than 33 times higher than the MLP fitting time and this result did not support our hypothesis.

6. Conclusion

In our study, we trained and tuned a Multilayer Perceptron (MLP) & Support Vector Machine (SVM) optimal models, we then predicted with high accuracy and AUC on the unseen test set. Our models generalized very well and there were no signs for overfitting. We also critically analysed and compared the SVM & MLP models.

SVM & MLP both performed very well and had similar behaviour, both models had very high AUC on training set, then dropped on validation sets and finally dropped further on the test set, all within the recommended thresholds which indicated no overfitting. In our bank marketing problem, MLP was a better choice as all metrics were high including accuracy, F1-score, AUC and finally the precision which is very important in imbalanced datasets. SMOTE was applied successfully on the training set and the algorithms had better generalization. Confusion matrices and ROC curves proves to be excellent tools in evaluating classifiers and measuring accuracies.

Our main results supported our hypothesis for MLP being a better algorithm. But the following results did not support our hypothesis, firstly, training a polynomial SVM training was more time consuming and computational expensive because of the complexity of the model, while single layer MLP computation time was very fast and this was not expected. Secondly, the optimization methods, where we used to optimize our hyper-parameters, random search followed by a grid search produced better combinations for our optimal models and better results than the Bayes optimization.

Future work would be exploring and comparing decisions boundaries for the two classifiers. Also adding noise can help in generalization although we did not need it in our case, as the results were within reasonable thresholds. Another exploration might be exploring deeper neural networks to test or experimenting with ensemble methods.

In practice, it is less useful to use SVM polynomial for performance reasons. So, the rule of thumb for future work is linear kernel for linear problems, and RBF kernel for non-linear problems.

7. Reference

[1] Apampa, O. (2016). Evaluation of Classification and Ensemble Algorithms for Bank Customer Marketing Response Prediction. Journal of International Technology and Information Management, 25(4), 6.
[2] Srivastava, Durgesh & Bhambhu, L.. (2010). Data classification using support vector machine. Journal of Theoretical and Applied Information Technology. 12. 1-7.
[3] V. Vapnik. The Nature of Statistical Learning Theory. NY: Springer-Verlag. 1995.
[4] C.-C. Chang, C.-J. Lin, "Libsvm: a library for support vector machines", *ACM Transactions on Intelligent Systems and Technology* (TIST), vol. 2, no. 3, pp. 27, 2011.
[5] Cristianini, N. & Shawe-Taylor, J. 2000, An Introduction to Support Vector Machines and Other Kernel-based Learning Methods, Cambridge University Press, Cambridge.
[6] Makridakis S, Spiliotis E, Assimakopoulos V. Statistical and Machine Learning forecasting methods: Concerns and ways forward. PLoS One. 2018;13(3):e0194889. Published 2018 Mar 27
[7] N. V. Chawla, K. W. Bowyer, L. O.Hall, W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," Journal of artificial intelligence research, 321-357, 2002.
[8] Moro, Sérgio et al. "A data-driven approach to predict the success of bank telemarketing." *Decis. Support Syst.* 62 (2014): 22-31.
[9] Elsalamony, Hany & Elsayad, Alaa. (2013). Bank Direct Marketing Based on Neural Network. International Journal of Engineering and Advanced Technology. 2. 392-400.

Appendix 1 – glossary

Support Vector Machine (SVM): Supervised machine learning is used in pattern recognition where there is an assumption of separability.

Multilayer Perceptron (MLP): A multilayer perceptron (MLP) is a class of feedforward artificial neural network (ANN).

AUC (Area under the ROC Curve): An evaluation metric that considers all possible classification thresholds. The Area Under the ROC curve is the probability that a classifier will be more confident that a randomly chosen positive example is actually positive than that a randomly chosen negative example is positive

Early Stopping: A method for regularization that involves ending model training before training loss finishes decreasing. In early stopping, you end model training when the loss on a validation dataset starts to increase

Bayesian optimization: Bayesian optimization is a sequential design strategy for global optimization of black-box functions that doesn't require derivatives

Grid search: The traditional way of performing hyperparameter optimization has been grid search, or a parameter sweep, which is simply an exhaustive searching through a manually specified subset of the hyperparameter space of a learning algorithm.

SMOTE: Synthetic Minority Oversampling Technique

Cross-entropy: Calculates a network performance given targets and outputs, with optional performance weights and other parameters. The function returns a result that heavily penalizes outputs that are extremely inaccurate (`y` near `1-t`), with very little penalty for fairly correct classifications (`y` near `t`). Minimizing cross-entropy leads to good classifiers.

SVM Kernel: The function of kernel is to take data as input and transform it into the required form. These functions can be different types. For example, linear, polynomial & radial basis function (RBF).

Box Constraint (C): A parameter that controls the maximum penalty imposed on margin-violating observations, which helps to prevent overfitting.

Activation Function: A function that takes in the weighted sum of all of the inputs from the previous layer and then generates and passes an output value (typically nonlinear) to the next layer.

Momentum: Momentum can smooth the progression of the learning algorithm that, in turn, can accelerate the training process

Learning Rate: A high learning rate, the system contains too much kinetic energy and the parameter vector bounces around chaotically, unable to settle down into deeper, but narrower parts of the loss function. A very low learning rate be wasting computation bouncing around chaotically with little improvement for a long time.

Trainrp function: A resilient backpropagation training function that updates weight and bias values according to the resilient backpropagation algorithm (Rprop). Multilayer networks typically use sigmoid transfer functions in the hidden layers. These functions are often called “squashing” functions, because they compress an infinite input range into a finite output range. The purpose of the resilient backpropagation (Rprop) training algorithm is to eliminate these harmful effects of the magnitudes of the partial derivatives. **Trainrp** can train any network as long as its weight, net input, and transfer functions have derivative functions.

Tansig: Hyperbolic tangent sigmoid transfer function. This neural transfer function can be found in the hidden layer. Transfer functions calculate a layer’s output from its net input.

Logsig: log-sigmoid transfer function. This neural transfer function can be found in the output layer.

Appendix 2 – Implementation details

Structure of the submitted files are shown in **README** file which includes also the data set used for training SMOTE ready and the test file used in the prediction stage. Both files are in CSV format.

The **SMOTE** technique used is imported from python library imblearn.over_sampling, this object is an implementation of SMOTE - Synthetic Minority Over-sampling Technique as presented in Chawla et al.[7]

MLPModelFinal.mat is our MLP final model, saved automatically when Final_Test_SVM_MLP.m file is run.

SVMModelFinal.mat is our SVM final model, saved automatically when Final_Test_SVM_MLP.m file is run.

In order to produce the same results, run the **Final_Test_SVM_MLP.m** file, the code will automatically save the models in mat format then the code will clear all workspace and finally loads the saved models and make predictions on test set.

PCA was applied on a separate data set, but the experiments showed no improvement, thus is was not included in the report.

All pre-processing stage including univariate and bivariate analysis and initial visualization was made in **python**, these files are not included in the submission folder as instructed.

The **grid search** result tables in the report shows a subset of the grid searches made, originally the searches contain several hundred of parameters combinations and their corresponding metrics.

The **maximum objective evaluation** configuration for Bayesian optimization for both models was set to 100 in order to get a more accurate evaluation of our models. The best model with its hyper parameters then is automatically plugged into our code to generate the metrics needed for the comparison with the grid search optimization method, thus the reason of displaying only the best model parameters in our table. The comparisons are based on AUC scores.

Training functions for the MLP differ in training and some of them needed different parameters combinations that depends on them.

All training and evaluation of the models was created with **MATLAB 2019b (Code included).**