MLP vs SVM: A Comparative Study for a Binary Classification Problem

Motivation of the problem

For the purpose of our study, we investigate a specific dataset from the German Socio-Economic Panel, a large panel study, extremely popular in Social Sciences Research for a classification problem (1). We choose to critique to two popular machine learning models namely Multi-Layer Perceptron and Support Vector Machines in a real-world scenario, with more complicated and noisy data (as opposed to using a popular machine learning dataset). The dataset contains sociodemographic information, as well as rarer psychometric variables, selected based on their importance in past research on behavioural prediction (2,3). Using the variable feature space, we will contrast the two models on the binary classification problem of predicting smoking patterns.

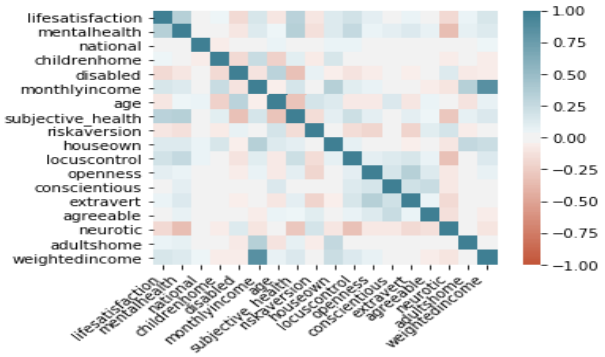
Description of the Dataset

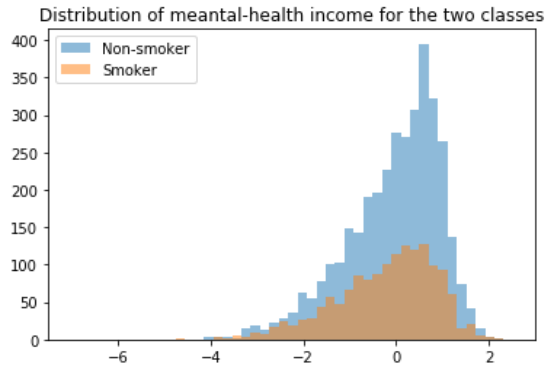
The original dataset was randomly sampled and a cross section of 5055 individuals was chosen, for the purpose of computational feasibility. 20 features were selected, including 2 categorical variables, that were transformed using one-hot-encoding. Continuous and discrete features were standardized and normalized before analysis. The problem is a binary classification one, with class imbalance consisting of 71% non-smokers. Features like log weighted income, risk-aversion and age, reveal a very different distribution for the two classes. Meanwhile, even categorical features like occupation and migration, also offered possibility for better discrimination.

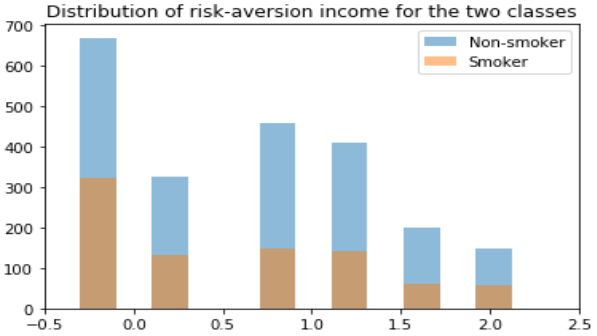
Dataset Summary Statistics

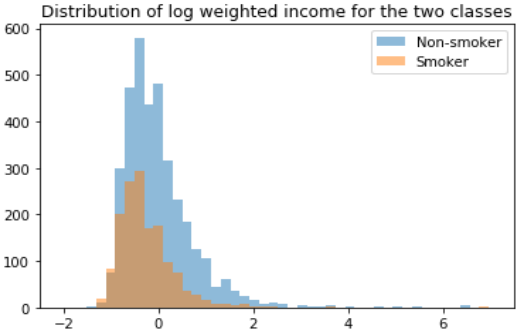
Statistic	Smoking	life satisfaction	mental health	children home	disabled	monthly income	age	subjective health	risk aversion	house own	locus control	openness	conscientious	extravert	agreeable	neurotic	adults home	Log weighted income
mean	NO	0.08	-0.05	0.48	-0.05	0.16	-0.21	0.07	0.02	0.05	0.01	0.05	-0.05	-0.01	0	0.01	0.08	0
	YES	-0.09	-0.22	0.48	-0.1	-0.19	-0.45	-0.12	-0.16	-0.4	-0.08	-0.02	-0.01	0.2	-0.13	0.11	-0.07	-0.27
std	NO	0.97	1.01	1.21	0.94	1.09	0.87	0.99	0.97	1	0.99	0.96	1	1.03	1	1.02	1.04	0.81
	YES	0.97	1.07	1.21	0.88	0.62	0.67	0.97	1.01	0.93	1.1	1	1	1.01	1.06	1.06	0.98	0.59
Skewness	NO	-1.53	-0.86	0.89	2.58	6.27	0.5	-0.49	0.1	-0.14	-0.22	-0.26	-0.75	-0.31	-0.5	0.16	1.25	4.37
	YES	-1.14	-0.7	1.22	2.87	1.39	0.19	-0.35	0.16	0.82	-0.19	-0.25	-0.59	-0.37	-0.37	0.14	1.2	2.72

Correlation Plot of continuous & Discrete Features









Hypothesis Statement

- First, we examine *Latkowski, T. & Osowski's (2017)* study that that SVM outperforms MLP for a binary classification algorithm(4).
- Second, we analyse, *Guang-Bin Huang 2015;* MLP with a single layer even with a large number of neurons is unable to perform well on classification algorithms due to its shallow architecture, which impacts its ability for feature learning (5).
- Third, we evaluate *Weissbart, Picek and Batina (2019*); that Re-LU activation function for hidden layer outperforms other activation functions in MLP models with smaller architecture (6).
- Next, we investigate *Suykens & Vandewalle's (1999)* study, that for classification algorithms, SVM with RBF kernel out-performs linear and polynomial kernels (7).

Support-Vector-Machines

The Support Vector machine algorithm finds a N-dimesional hyperplane in order to classify the dataset with the use of support vectors. The SVM maximises the margin of the hyperplane, i.e.) maximises the distance between hyperplane and the datapoints in the classes. Additionally, SVM allows us to convert not separable problem to separable problem by using the kernel function encompassing the kernel trick, which allows us to take a low dimensional input space and transforms it to a higher dimensional space.

The main advantages of SVM are that it is a quadratic programming problem that is computationally efficient (4). SVM is effective in cases where number of dimensions is greater than the number of samples and works even well with small and medium datasets. The main disadvantages are that SVM shows poor performance on noisy data with overlapping classes. Further, the classifying hyper plane offers no probabilistic explanation.

Multilayer Feedforward Neural Networks

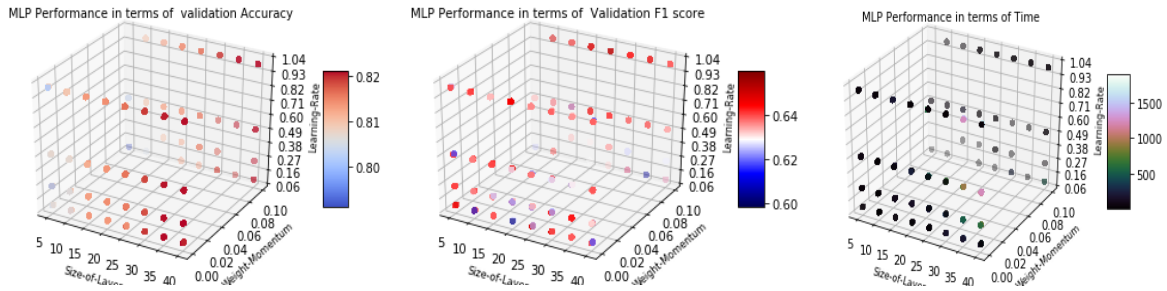
A multilayer feedforward neural network is a deep neural network that consists of an interconnection of perceptrons with an input layer, an output layer (that makes a decision or prediction about the input); and in between these are the hidden layers. The training on the input and output layers of MLP is executed by updating weights and biases of the nuerons composing the layers, each layer at a time, by the method of backpropagation, via a series of iterations each successively reducing the error. During backpropagation, for each epoch, the gradient of the error along which the parameters (synaptic weights and biases) are adjusted as they move the MLP one step closer to the error minimum. The parameters are updated until the error reaches the minimum, which is known as convergence.

As per, the universal approximation theorem the MLP with a single hidden layer containing a finite number of neurons can approximate any continuous functions on compact subsets of Rn. However, the main limitation of the MLP algorithm is that the error function is highly sensitive to the choice of learning rate and is likely to converge to a local minimum. Further, it is a highly complex algorithm with many hyper parameters to consider, which makes computationally expensive and prone to overfitting.

Description of choice of training and evaluation methodology

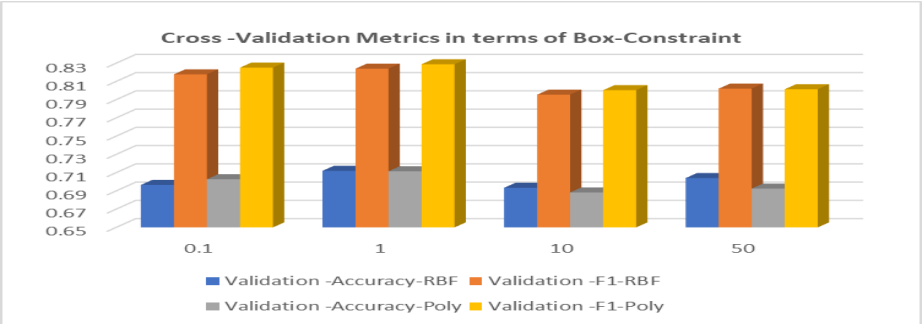
70% data was used as training data and 30% data was used as test data. On the training data, 10-fold stratified-cross validation was performed for the purpose of model selection. The class imbalance in favour of non-smokers, leads us to investigate F1 score as a performance metric for the model, as opposed to using accuracy.

Hyperparameter Selection-MLP: The SoftMax activation output function was used for the given classification problem which returns a probabilistic distribution of the classes. The standard cross entropy loss function was used to generate the loss function in order to iteratively adjust the errors in each epoch. For the purpose of our study, the maximum number of epochs was set to 400. Early stopping criterion after 10 epochs was implemented i.e. to stop training once the model performance stops improving on a hold-out validation dataset after 10 epochs; for the purpose of computational efficiency. The following hyper parameters using grid search were considered, number of hidden layers, number of nuerons in each layer, the learning rate, weight momentum and activation functions (namely Logistic-Sigmoind and Re-Lu). MLP is extremely sensitive to the choice of the initial weights, thus ten-fold cross validation is a potent tool for the purpose of hyper-parameter selection. Additionally, both the gradient descent back propagation and Bayesian Regularization for weight optimisation were evaluated, since gradient descent is expected to suffer from the drawbacks like local search nature and a tendency of overfitting the data.



Experimental results-MLP: The above plotted graphs are for a single hidden layer (since a model with a single hidden layer outperformed the model with two hidden layers) in terms of cross validation scores. As can be see, with increase in size of hidden layer, weight momentum and learning rate; increases the validation accuracy; however, increasing size of the hidden layer and learning rate, post a certain value leads to a fall in the validation-F1 score.

Hyperparameter Selection-SVM: For SVM, Radial basis function kernel and Polynomial kernel functions were evaluated. The hyper parameters are C; which is a regularization parameter that controls the trade-off between the achieving a low training error and a low testing error, that is the ability to generalize your classifier to unseen data (bias-variance trade off). In case of RBF kernel, gamma hyper-parameter which is the inverse of the standard deviation of the RBF kernel (Gaussian function) needs to be optimized. For, polynomial kernel, the degree of the polynomial is the hyper-parameter.



Experimental Results-SVM: As can be seen in figure above, for both RBF and Polynomial kernel, increase in the box constraint beyond the value of 1, led to a fall in cross-validation F1

score and accuracy. For SVM with polynomial kernel, while increase in polynomial degree after '2', increased training accuracy and F1-score, cross validation accuracy and F1score declined for further increase in polynomial degree. Further, for RBF kernel with gamma value 1, outperformed gamma values of 0.1, 10, 100 for cross validation F1 score.

Analysis and Critical evaluation of results

In line with expectations, the models showed that in case of class imbalance, the hyper parameters that minimize the misclassification error, are not ones giving optimality in terms of F1score.

Model Selection: MLP & SVM

Contrary to Guang-Bin Huang 2015, in case of MLP the cross validation F1score was maximised with a single hidden layer of 15 neurons, as opposed to two or more hidden layers. However, the cross-validation accuracy showed similar results with two hidden layers of ~15,25 neurons or a single layer with 40 neurons. **In line with F1 score as the metric of choice, we chose the single hidden layer with 15 neurons model**. For cross-validation F1 score, **learning rate of 0.5**, proved better (than learning rate of 0.25, 1 and 0.1), **and momentum of 0.001** proved optimal for both accuracy and F1 score. However, any further increase in weight momentum reduced the validation F1 score and accuracy. In line with expectations, **Bayesian regularized network** which assigns a probabilistic nature to the model allowing the network to automatically and optimally penalize excessively complex models and reduce overfitting (8), outperformed gradient descent, in terms of optimal cross validation F1 score. Contrary to Weissbart, Picek and Batina (2019), **Log-Sigmoid activation** function outperformed Re-Lu in terms of cross-validation F1- score. However, dispersion of F1-score was larger for Log-Sigmoid as compared to RE-Lu activation function.

For SVM with polynomial kernel, while increase in polynomial degree after '2' maximized the training F1 score, it reduced the validation F1 score. Contrary to Suykens & Vandewalle's (1999), SVM with polynomial kernel out-performed RBF kernel in terms of both validation F1 score and accuracy. **Thus, our model of choice was polynomial kernel with degree 2 and box-constraint value 1.**

Algorithm Comparison

Next, we compare the two optimal models, SVM and MLP models in order to evaluate Latkowski, T. & Osowski's (2017) claim that SVM is expected to outperform MLP for a binary classification problem. The two models are evaluated basis test-F1 score.

	Train Confusion Matrix			
	MLP		SVM	
Output-Class	41	1008	207	842
	10	2479	141	2348
	Test Confusion Matrix			
	MLP		SVM	
	7	443	79	371
	6	1061	61	1006
	Traget Class			

Model	F1-Validation	Precision Validation	Recall - Validation
SVM	0.829	0.71	0.99
MLP	0.87	0.92	0.83
Model	**F1-Test**	**Precision Test**	**Recall - Test**
SVM	0.825	0.7	0.99
MLP	0.82	0.73	0.94

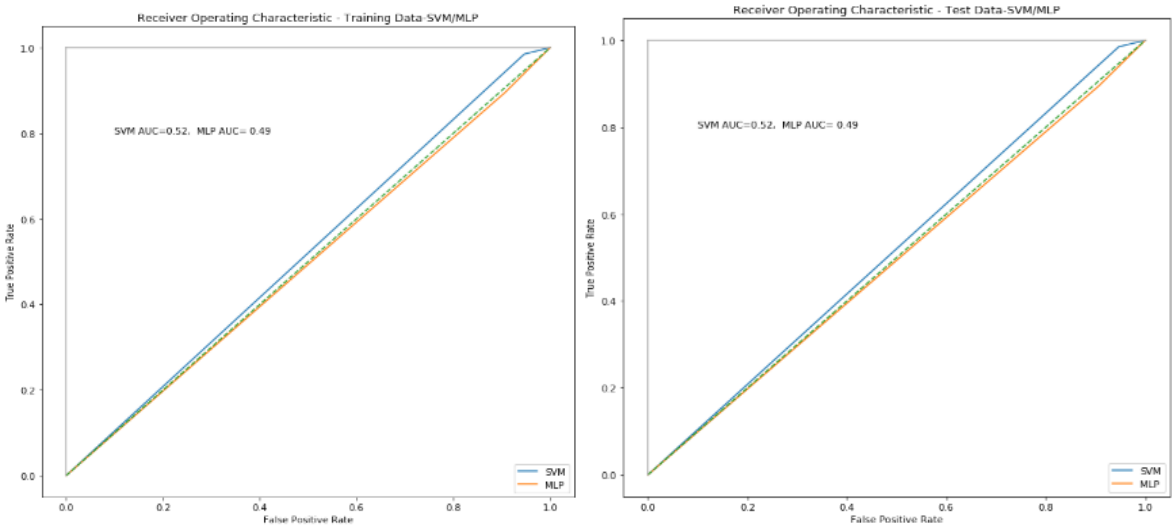
Contrary to expectations, looking at the validation F1 score, MLP with ~87%, out-performs SVM with ~82%. However, when the models are evaluated on 30% hold-out test data, SVM

and MLP both have F1 score of 82%. SVM with lesser hyper-parameters and fewer training iterations; shows test and validation F1 score in the narrow range of 77-83% for different hyper-parameter combinations. Meanwhile, for different iterations of the MLP models, great disparity in both cross-validation and test F1 score can be witnessed from 60-85%. Further, MLP is extremely sensitive to initialised random weight, as different iterations of MLP grid-search with same hypermeter values, gives rise to different results, and henceforth points towards sensitivity to overfitting. Further, in terms of training time complexity, the findings support our hypothesis that SVM is computationally more efficient.

In our case, where there are 70% non-smokers, it is more costly to misclassify a smoker as a non-smoker, during public policy formulation, as opposed to misclassifying a non-smoker as a smoker. Misclassifying a smoker as a non-smoker even for a single individual, can have hazardous health repercussions. Thus, for the purpose of our study, the value of recall outweighs precision during algorithm comparison.

On further examining the test metrics, while SVM outperforms in terms of better recall, MPL offers better precision. While SVM test recall is 99%, MLP test recall is 94%. In the test confusion-matrix it can be seen, MLP misclassified 93% (10 over 144) more non-smokers 90% (6 over 61) more in training data.

Further, the AUC for SVM is larger both for training and test data; i.e., given a true positive (non-smoker); there is greater probability of a false positive (classifying a non-smoker as a smoker) in the MLP model, vis-à-vis the SVM model.



Conclusion and Way Forward

Thus, despite higher validation F1 score, we select SVM for the binary classification problem in consonance with Latkowski, T. & Osowski's (2017). Generally better results for SVM are consistent with the initial hypothesis; the reduction in the number of parameters contributes to the smaller variance of the model and in general helps achieve regularly higher out-of-sample scores, further MLP is prone to overfitting.

Going forward, the dataset as the present one with class imbalance can be handled using SMOTE and ADASYN. These techniques help understand better what features are being extracted from the data which sampling can't achieve

Additionally, more efficient SVM with a new kernel function that is called Gaussian Radial Basis Polynomials Function (GRPF) that combines both Gaussian Radial Basis Function (RBF) and Polynomial (POLY) kernels can be investigated (9).

In order to improve the overfitting problem and high dispersion in errors due to large hyper-parameters, adjustable training signals and multi-layer perceptron neural network (ASMLP) can be used. The ASMLP uses two multi-layer perceptron neural networks (MLPs) to infer respiration position alternately and the training sample will be updated with time (10).

Further, to tackle the convergence issue of MLP, a Hybridization of Gradient-based and Meta-heuristic training algorithms can be investigated. In addition, Random Search is expected to give better results for some datasets than grid search, and can be evaluated (11).

References

1) Pham, B.T. and Prakash, I., (2019). Evaluation and comparison of LogitBoost Ensemble, Fisher's Linear Discriminant Analysis, logistic regression and support vector machines methods for landslide susceptibility mapping. Geocarto International, 34(3), pp.316-333.
2) Goebel, J., Grabka, M., Liebig, S., Kroh, M., Richter, D., Schröder, C. and Schupp, J. (2019). The German Socio-Economic Panel (SOEP). Jahrbücher für Nationalökonomie und Statistik, 239(2), pp.345-360. 5.
3) Dohmen, T., Falk, A., Huffman, D., Sunde, U., Schupp, J. and Wagner, G. (2011). INDIVIDUAL RISK ATTITUDES: MEASUREMENT, DETERMINANTS, AND BEHAVIORAL CONSEQUENCES. Journal of the European Economic Association, 9(3), pp.522-550.
4) Latkowski, T. and Osowski, S., 2017. Gene selection in autism – Comparative study. Neurocomputing, 250, pp.37-44.
5) J. Tang, C. Deng and G. Huang, "Extreme Learning Machine for Multilayer Perceptron," in *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 4, pp. 809-821, April 2016.
6) Suykens, J., Vandewalle, J. Least Squares Support Vector Machine Classifiers. Neural Processing Letters 9, 293–300 (1999). https://doi.org/10.1023/A:1018628609742
7) <https://www.researchgate.net/publication/220356380_Support_Vector_Machines_SVMs_with_Universal_Kernels> [Accessed 14 April 2020].
8) Bishop, 1999, Bayesian Neural Networks<https://www.researchgate.net/publication/220356380_Support_Vector_Machines_SVMs_with_Universal_Kernels> [Accessed 14 April 2020].
9) Sun, W, Jiang, M, and Yin, F. SU-F-E-09: Respiratory Signal Prediction Based On Multi-Layer Perceptron Neural Network Using Adjustable Training Samples. United States: N. p., 2016. Web. doi:10.1118/1.4955695.
10) Kulala, Hemalatha & Rani, K.. (2017). Advancements in Multi-Layer Perceptron Training to Improve Classification Accuracy. International Journal on Recent and Innovation Trends in Computing and Communication. 5. 353-357.
11) Bacanin, N., Bezdan, T., Tuba, E., Strumberger, I. and Tuba, M., 2020. Optimizing Convolutional Neural Network Hyperparameters by Enhanced Swarm Intelligence Metaheuristics. Algorithms, 13(3), p.67.

Appendix 1: Glossary

Term	Meaning
Training Data	The training data is an initial set of data used to help a program understand how to apply ML technologies to learn and produce sophisticated results.
Test Data	Training data's output is available to model whereas testing data is the unseen data for which predictions have to be made in oredr to evaluate model performance
Validation Data	The fitted model on training data is used to predict the responses for the observations in a second dataset called the validation dataset.The validation dataset provides an unbiased evaluation of a model fit on the training dataset while tuning the model's hyperparameters
Perceptron	A Perceptron is an algorithm for supervised learning of binary classifiers. This algorithm enables neurons to learn and processes elements in the training set one at a time.
One hot encoding	One hot encoding is a process by which categorical variables are converted into a form that could be provided to ML algorithms to do a better job in prediction.
SVM Kernel	SVM algorithms use a set of mathematical functions that are defined as the kernel. The function of kernel is to take data as input and transform it into the required form. Different SVM algorithms use different types of kernel functions.
Softmax function	Softmax function is an activation function for MLP that turns numbers aka logits into probabilities that sum to one. Softmax function outputs a vector that represents the probability distributions of a list of potential outcomes.
Cross-entropy loss	Cross-entropy loss, or log loss, measures the performance of a classification model whose output is a probability value between 0 and 1.
Hyperparameter	Hyperparameter is a parameter whose value is set before the learning process begins. By contrast, the values of other parameters are derived via training.
Gradient	The gradient can be interpreted as the "direction and rate of fastest increase". If at a point p, the gradient of a function of several variables is not the zero vector, the direction of the gradient is the direction of fastest increase of the function at p, and its magnitude is the rate of increase in that direction.
SMOTE	SMOTE is an oversampling technique that generates synthetic samples from the minority class. It is used to obtain a synthetically class-balanced or nearly class-balanced training set, which is then used to train the classifier.
k-Fold Cross-Validation	k-Fold Cross-Validation is a resampling procedure used to evaluate machine learning models on a limited data sample. . The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into
Hyperplane	In geometry, a hyperplane is a subspace whose dimension is one less than that of its ambient space. If a space is 3-dimensional then its hyperplanes are the 2-dimensional planes, while if the space is 2-dimensional, its hyperplanes are the 1-dimensional lines.
ADASYN	Its a improved version of Smote. What it does is same as SMOTE just with a minor improvement. In other words instead of all the sample being linearly correlated to the parent they have a little more variance in them i.e they are bit scattered.
Confusion Matrix	A confusion matrix is a table that is often used to describe the performance of a classification model (or “classifier”) on a set of test data for which the true values are known. It allows the visualization of the performance of an algorithm.
Precison	Precision means the percentage of your results which are relevant.
Recall	Recall refers to the percentage of total relevant results correctly classified by your algorithm.
Learning Rate	Specifically, the learning rate is a configurable hyperparameter used in the training of neural networks that has a small positive value, often in the range between 0.0 and 1.0. The learning rate controls how quickly the model is adapted to the problem
Weight Momentum	When training neural networks, it is common to use "weight decay," where after each update, the weights are multiplied by a factor slightly less than 1. This prevents the weights from growing too large, and can be seen as gradient descent on a quadratic regularization term.
Overfitting	Overfitting is a modeling error that occurs when a function is too closely fit to a limited set of data points. Overfitting the model generally takes the form of making an overly complex model to explain idiosyncrasies in the data under study.
Grid-search	Grid-search is used to find the optimal hyperparameters of a model which results in the most 'accurate' hypermarameters
Re-Lu Function	The rectified linear activation function is a piecewise linear function that will output the input directly if is positive, otherwise, it will output zero.
Logistic-Sigmoind Function	Another function that is often used as the output activation function for binary classification problems (i.e. outputs values that range between [0,1]), is the logistic sigmoid.
Bayesian regularization	Bayesian regularization is a mathematical process that converts a nonlinear regression into a "well-posed" statistical problem in the manner of a ridge regression for the purpose of hyperparamater tuning during ML task.
Epoch	An epoch is a term used in machine learning and indicates the number of passes through the entire training dataset the machine learning algorithm has completed. If the batch size is the whole training dataset (batch mode) then batch size and epoch are equivalent.

Appendix 2: Implementation Details

MATLAB Script:

- Load "SVMFinal.mat"
- Read csv file: NN5000.csv containing dataset
- Partition the data into stratified train and stratified test set (class imbalance), with 30% test hold-out dataset.
- On the 70% training data, perform stratified k-Fold cross validation for model selection.
- ****%% Hyper-Parameter Tuning for RBF Kernel**********
- G=Gamma Value
- C=Box constraint for RBF Kernel
- start measuring computation time
- K-fold Cross validation for different values of C and Gamma
- Model fit on training data and evaluated on validation data for every fold
- Confusion Matrix, F1 -score, misclassification erorr; Precision and Recall for every out of sample validation fold
- Calculating the mean F1 -score, misclassification erorr;Precision and Recall for all out-of sample validation scores to get cross-validation metrics
- Time calculation post K-fold cross validation for the given hyperparameter combination
- Evaluate every-hyper parameter-combination on hold-out test data to study dispersion of test scores\
- Save the grid-search in a csv file
- **********%%Hyper-Parameter Tuning for Polynomial Kernel%%**********
- G=Polynomial Degree Value
- C=Box constraint for RBF Kernel
- start measuring computation time
- K-fold Cross validation for different values of C and Gamma
- Model fit on training data and evaluated on validation data for every fold
- Confusion Matrix, F1 -score, misclassification erorr; Precision and Recall for every out of sample validation fold
- Calculating the mean F1 -score, misclassification on erorr; Precision and Recall for all out-of sample validation scores to get cross-validation metrics
- Time calculation post K-fold cross validation for the given hyperparameter combination
- Evaluate every-hyper parameter-combination on hold-out test data to study dispersion of test scores
- Save the grid-search in a csv file
- Load "MLPFinal.mat"
- Read csv file: NN5000.csv containing dataset
- Partition the data into stratified train and stratified test set (class imbalance), with 30% test hold-out dataset.
- On the 70% training data, perform stratified k-Fold cross validation for model selection.
- ****%%Hyper-parameter tuning for MLP with single Hidden Layer%%*****
- In order to evaluate the dispersion of Test F1 score, every hyper-parameter combination is evaluated on the 30% Hold-Out test set
- N=No. of neurons in hidden layer
- learn-rate combinations
- two optimisation algorithms
- "trainbr"=Bayesian Regularization
- "traingd"=Gradient Descent
- Two activation Functions: Log-sig and Poslin
- weightreg corresponds to weight momentum
- Create a Fitting Network & set number of neurons
- set cross-entrophy loss function to minimise back propagation cross entropy error in each iteration
- Set maximum no. of epochs to 400
- activation f(n) for input layers
- setting output layer as soft-max for classification
- Confusion Matrix, F1 -score, misclassification erorr; Precision and Recall for every out of sample validation fold
- Calculating the mean F1 -score, misclassification erorr; Precision and Recall for all out-of sample validation scores to get cross-validation metrics
- Time calculation post K-fold cross validation for the given hyperparameter combination
- Evaluate every-hyper parameter-combination on hold-out test data to study dispersion of test scores\
- Save the grid-search in a csv file
- ****%%Hyper-parameter tuning for MLP with two Hidden Layer%%*****
- In order to evaluate the dispersion of Test F1 score, every hyper-parameter combination is evaluated on the 30% Hold-Out test set
- N=No. of neurons in 1st hiddenlayer
- N1=No. of neurons in 2nd hiddenlayer
- Steps from last section follow
- save the grid-search results for two-hidden layers