Pima Indians Diabetes Classification: A Comparative Analysis Study of Multi-Layer Perceptron and Support Vector Machine Algorithms

Abin Abraham

1. Introduction

Diabetes is a medical condition that causes a person's blood sugar level to become too high. Diabetes prevalence in the UK is estimated to rise to 5 million by 2025 [1]. Quick and early diagnosis is beneficial to treating the disease. Deep learning and machine learning have been extensively applied in medical diagnosis. This paper aims to critically evaluate and perform a comparative analysis on two key supervised learning algorithms – multi layer perceptron(MLP) and support vector machines(SVM) , on a classification task on the Pima Indian diabetes dataset [2]. The support vector machines are evaluated in comparisons with different kernel functions. We perform hyperparameter tuning using grid search and Bayesian optimisation for both algorithms with stratified k-fold cross validation, oversampling of minority class and evaluate and compare model performance using confusion matrix and ROC curves. Section 2 describes initial data analysis. Section 3 describes the algorithms and their pros and cons. Section 4 & 5 describes the choice of training and evaluation methods. Section 6 & 7 describes in detail the choice of parameters and critically analyses the experimental results. Lastly Section 8 concludes the paper with lessons learnt and future work.

2. Description of the dataset

The dataset is originally sourced from the National Institute of Diabetes and Digestive and Kidney Diseases and originally published on UCI but is currently available in all public forums and is a popular dataset. It has data regarding 768 Pima Indian women with 8 numeric continuous features and the categorical diagnosis of diabetes. The binary target label has the following definition '1' means a positive sample for diabetes and '0' means negative sample for diabetes. Quick summary of the distribution and correlation plots follow

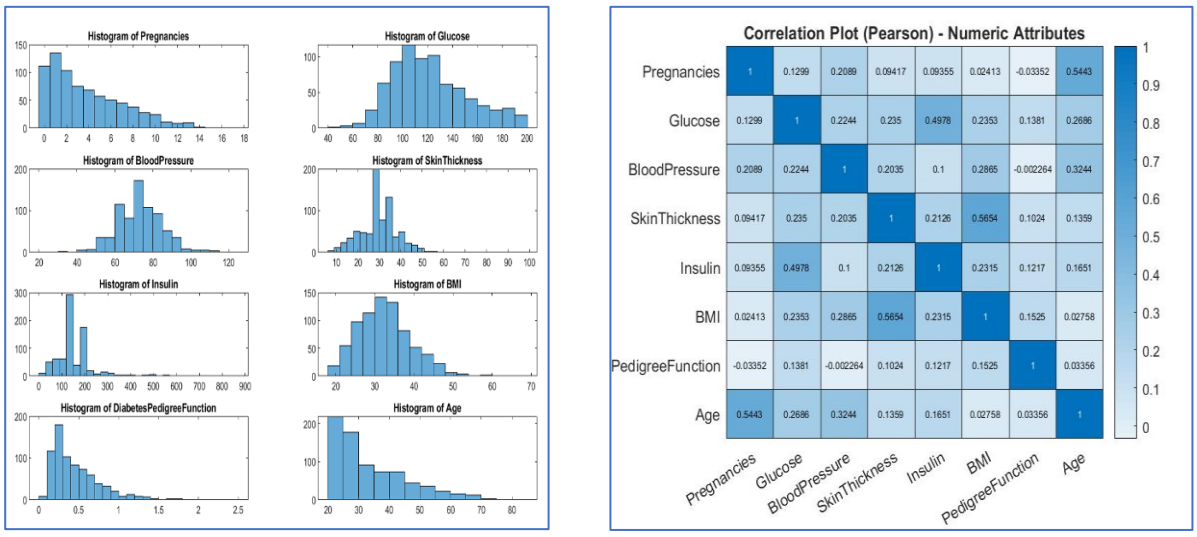

Figure 1 Histogram of features and Confusion Matrix

The dataset has missing values which were put as 0. As part of the pre-processing steps, the values are imputed based on the mean value for the attribute in the associated target class. There is a slight imbalance of the target class. We will oversample the minority class during the training using ADASYN [3]. The pre-processing was done using Python. All the other steps in this project were completed using Matlab.

Attributes	Missing
BloodPressure	35
BMI	11
Insulin	374
SkinThickness	227

Table 1 Missing Values corrected in pre-processing

3. Summary of Algorithms

3.1 Multi-Layer Perceptron

A multilayer perceptron(MLP) or artificial neural network is a supervised learning algorithm. It has an input layer, one or more hidden layers and an output layer. Each layer in the multilayer perceptron except the input layer, has weights, bias and a non-linear activation unit. The activation function maps the product of the weights and inputs and their sum with the biases, as the output of the layer, which is the subsequent input for the next layer. Depending on the activation layer at the output layer the MLP can be used for regression and classification[4]

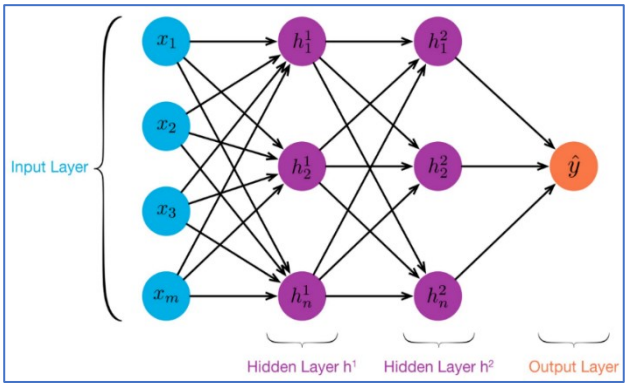

Figure 2: Multi-layer Perceptron (Source: Towards Data Science [12])

The weights and biases are learned by a process known as back propagation Each run is called an epoch and the training process is continued for multiple epochs till the error converges to a minimum. The error function varies according to the task being regression or classification.

Pros	Cons
• Can be used for both regression and classification • Automatic feature extraction • Can approximately model any function (universal approximator).	• Poor model explainability • Risk of overfitting • Can be computationally expensive and time consuming • No guarantee of global optimum

3.2 Support Vector Machine

A support vector machine (SVM) is a supervised learning algorithm primarily used for binary classification tasks that determines an optimum hyperplane to split the data. The kernel is a mathematical transformation of the data into a higher dimension space so that the optimum hyperplane can be determined. This step can be computationally very expensive if not for the kernel trick which achieves this by using the dot products between the features. A linear kernel is used for linear classification and a nonlinear- polynomial or gaussian is used for nonlinear classification. Here C1 and C2 are the 2 binary classes. SVM tries to build a hyperplane with maximum margin that separates the two classes. SVM can also be used to handle multi-class classification with the two methods: One vs One and One vs All.

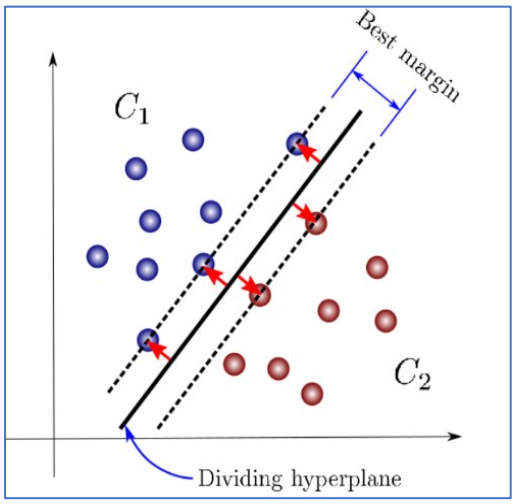

Figure 3: Multi-layer Perceptron (Source: Towards Data Science[13])

Pros	Cons
• Effective on high dimensional data and nonlinear data • Models as convex optimisation problem so no local minima • Less risk of overfitting	• Non probabilistic predictions • Training time complexity is O(n^2) • Extensive memory requirement for training. • Results are dependent on kernel

4. Hypothesis Statement

SVMs and MLP have comparable performance for classification tasks and performance is dependent on the dataset [5] with SVM occasionally better on small datasets as MLP can be prone to overfitting. MLP generally performs better in complex problems like image classification, NLP, etc. Here we have a relatively small dataset with a problem of lesser complexity and the expectation is for SVM to perform better. There have been multiple similar studies[5][6][7].

5. Methodology

We have split the dataset with stratification - 80 % is used for training and validation. 20% is used for testing and comparative evaluation of the performance between MLP and SVM algorithms. The choice of size of split is because we have a small dataset. As the size of data increases more can be used in training. The test set is never seen by the networks during training. The validation set error is an estimate of the generalisation error and this will be used for model selection.

5.1 Model Selection

A two stepped process is organized to build and validate the models that generalize the input space. Initially a grid search is performed on a wide range for each hyperparameter, which will later provide a filtered down range for hyperparameters, on which a Bayesian optimisation search is performed. K-fold stratified cross validation will be done, which is a reliable method to assess the model performance and reduce bias for small datasets. The F1 score error i.e. (1- F1 score) is used to assess the model and would be used as the minimization objective. By minimizing the F1 score error, we maximize the model's F1 score. To address the slight class imbalance (65% negative and 35% positive) during K fold cross validation we perform oversampling of minority class in each fold's training data to prevent data leakage. We will use ADASYN as the algorithm for oversampling [3].

While training neural networks, after a point it starts overfitting or stops generalizing and start learning the statistical noise in the training dataset [4]. To counter this we perform mitigation steps like early stopping (training stops when/if validation set error starts to increase or soon afterwards). Many studies have shown that adding noise during training can improve network generalization [10][11] and that adding noise during training is equivalent to data augmentation[4] and that noise injection can be much more powerful than simply shrinking the parameters [9]. Normalizing data makes the cost function symmetric and can help the classifier to model and generalize better. In Matlab, by default all the neural network training algorithms normalize inputs and targets to [-1 1]. In fitcsvm, the input data is normalized based on setting.

5.2 Algorithm comparison

For algorithm comparison, the best models from the model selection process were retrained using the full training data set with ADASYN oversampling of minority class and error curve is plotted for MLP to ensure there is no overfitting. Noise was added during training of the final models to ensure better generalization. Testing is done on the test set and results are compared. We use ROC and F1 score-based evaluation metrics because under the imbalanced learning condition, traditional overall classification accuracy may not be able to provide a comprehensive assessment of the observed learning algorithm.

5.3 Time complexity

We will also perform a simplified analysis on the time complexity – training + testing time for each of the algorithms across a small range of values of hyperparameters like an ablation study by keeping all values constant except one.

6. Choice of parameters and experimental results

6.1 Architecture and Parameters for MLP

We will tune the models using Number of hidden layers, Number of neurons per hidden layer, Learning rate, momentum, training function and transfer function. Learning rate is perhaps the most important hyperparameter for tuning neural networks [4] and decides the hyperparameter that decides the rate at which the model weights will be updated after each epoch evaluations of the error function. Scaled conjugate gradient backpropagation(trainscg in MATLAB) is used as the network training function which will update the neuron weights. Patternet automatically configures a network with random initial weights[0,1]. The primary reason behind initializing the weights randomly is to break symmetry. We want to make sure that different hidden units learn different patterns There are 2 output nodes and we will try different transfer function – tansig, logsig, softmax as this is a classification task. The number of epochs for training and validation have been set to 100 and 6 has been set as the minimum number of times when validation error falls as early stopping criteria to prevent overfitting.

6.2 Architecture and Parameters for SVM

We will explore different kernel types – Linear , Gaussian and Polynomial. The box constraint is the regularization parameter common to all kernel types, which defines how much to avoid misclassifying each training example. For large values of C, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly. Conversely, a very small value of C will cause the optimizer to look for a larger margin separating hyperplane, even if that hyperplane misclassifies more points. Gamma (kernel scale)in Gaussian kernel defines how far the influence of a single training example reaches, with low values meaning 'far' and high values meaning 'close'. Order defines the polynomial kernel affect the outcome of the mapping into a higher dimensional feature space

7. Analysis and critical evaluation of results

7.1 Model Selection

For MLP, based on grid search results we observe that the best value for objective error function (F1 score error) occurs even when the number of hidden neurons were 40. But studies have shown that MLP is prone to overfitting when there are many hidden layers. Hence, we filter the range down to 20-35 for the Bayesian optimization step. There is a wide range of results observed during grid search 75%-80% F1 score. Based on the patterns identified we did a 2nd level Bayesian optimisation which converges in less than 20 runs. Grid Search results have been included in Results folder in the submission. The results for the best 5 models are shown in the below table.

MLP					
Network Depth	No. of Neurons	Learning Rate	Momentum	Validation F1 Score Error	Validation F1 Score
2	34	0.78234	0.95682	0.17883	0.82117
2	28	0.084684	0.92862	0.18582	0.81418
2	35	0.54757	0.99985	0.18991	0.81009
2	35	0.18305	0.98174	0.18991	0.81009
2	35	0.17588	0.95065	0.20381	0.79619

Table 2 Top 5 models after Bayesian optimization

Based on the best hyperparameters for MLP we trained - one with noise and one without noise added to the full dataset along with ADASYN. Error curve below. Clearly the 1st model trained with data augmented with the gaussian noise has better generalized the input space and performs better in the validation data. The 2nd model without added noise was discarded.

Figure 4 Selected model - trained with noise and Discarded model – not trained with noise

Similar to the process followed for MLP, grid search was performed on a wide range of hyperparameters with F1 score 60%-80% and Bayesian optimisation results for top 5 models are displayed below. Grid Search results are included in Results folder in the submission. The results for the best 5 models are shown in the below table. We select the best model based on highest validation accuracy – linear kernel and discard the rest of the models.

SVM					
Kerneltype	Box Constraint	KernelScale	Order	Validation F1 Score Error	Validation F1 Score
Linear	0.060645	-	-	0.21197	0.78803
Linear	0.063211	-	-	0.21197	0.78803
Linear	0.065474	-	-	0.21197	0.78803
Linear	0.066186	-	-	0.21197	0.78803
Linear	0.06584	-	-	0.21197	0.78803
Gaussian	0.37328	2.9344	-	0.21627	0.78373
Gaussian	0.39458	2.9687	-	0.21782	0.78218
Gaussian	0.34285	2.1902	-	0.21826	0.78174
Gaussian	0.32033	2.2443	-	0.2184	0.7816
Gaussian	0.33558	2.9712	-	0.21899	0.78101
Polynomial	0.30748		2	0.26731	0.73269
Polynomial	0.35036		2	0.26731	0.73269
Polynomial	0.34356		2	0.26731	0.73269
Polynomial	0.35291		2	0.26731	0.73269
Polynomial	0.36096		2	0.26731	0.73269

Table 3 Top 5 models after Bayesian optimization for each SVM kernel

7.2 Time complexity

For MLP, we observe that there is no clear pattern for increasing number of hidden neurons. For SVM, increasing the box constraint assigns fewer support vectors and can lead to longer training times. We have shown this in the comparative study between different kernels. It is observed that the increase is sharper for linear SVM and polynomial kernel SVM with degree 2. All figures are provided in 'Figures' folder in the code.

7.3 Algorithm comparison

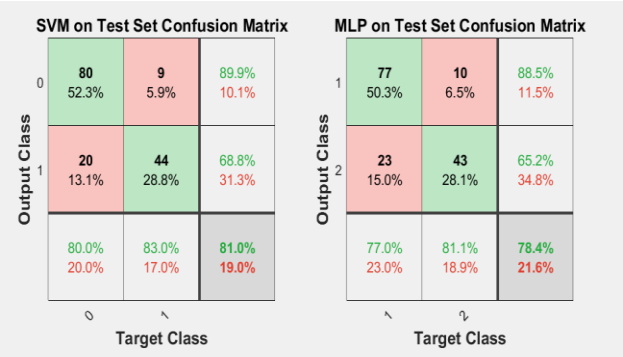

Figure 5 Confusion Matrix SVM v/s MLP on Test Data

We compare the best models from MLP and SVM on the test dataset and have plotted confusion matrix. Figure 1 displays the results of the best model from MLP and SVM on the test dataset. Here we can see that both have similar performance, with the SVM having a slight edge in classifying the accurate labels and lesser misclassification. In the model selection process the best MLP model achieved 82% and SVM model achieved 78% as F1 score on the validation dataset. During algorithm comparison stage on the test set, we achieve F1 score of 72% for MLP and 75% for SVM. Thus we can see that SVM has better generalized the input space and performs better.

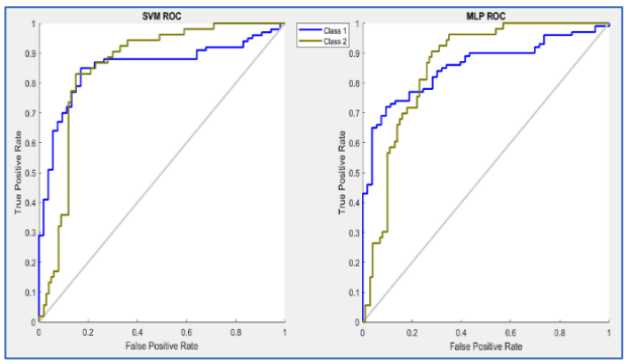

Figure 6 ROC Curve SVM v/s MLP on Test Data

We also plot the Receiver operating characteristic curve(ROC) on test data. We can observe that the curves for both classes in SVM are more oriented to the left side and thus offer better performance in classification. The more oriented to the left, the better the true positive rate for the class with minimal false positive rate. For class 1, it is important that true positive rate is higher so that most diabetes patients are identified. A slight false positive rate can be tolerated as the objective is to identify the diabetes patients. Similarly, it is important that the for class 0, the number of false positives are minimal to not miss any actual diabetes patients. Thus SVM is better suited as a classifier on this dataset.

8. Conclusion and lessons learnt

We have critically evaluated and compared models from Multi-Layer Perceptron and Support Vector Machine on the pima Indian diabetes dataset in terms of F1 score and time complexity. There was a clear difference in predictive performance with SVM better than MLP but not statistically significant. During training and validation MLP had performed marginally better while on the test set SVM has performed marginally better on F1 score. Both models have similar performance in training and validation with an expected dip in performance on test set. On the test set, SVM claims an advantage with better true positive and false positives. SVM's improvement may be attributed to additional data during final training and degradation in MLP as it may have slightly overfit. Our hypothesis thus stands proven via experimentation. Future work can include exploring dropout to reducing overfitting. Based on studies, it has been shown to provide the best accuracy on the pima Indian diabetes dataset [14].

Reference

1. https://www.diabetes.co.uk/diabetes-prevalence.html
2. Pima Indian Diabetes UC Irvine data set (originally contributed by Blake, Keogh, & Merz 1998) sourced from https://www.kaggle.com/uciml/pima-indians-diabetes-database
3. Haibo He, Yang Bai, E. A. Garcia and Shutao Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), Hong Kong, 2008, pp. 1322-1328.
4. Goodfellow Ian, Bengio Yoshua, Courville Aaron," Deep Learning" , in MIT Press 2016
5. Zanaty, E. (2012). Support Vector Machines (SVMs) versus Multilayer Perception (MLP) in data classification. Egyptian Informatics Journal. 13. 177–183. 10.1016/j.eij.2012.08.002.
6. Kayaer, Kamer & Yildirim, Tülay. (2003). Medical diagnosis on Pima Indian diabetes using general regression neural networks. Proceedings of the International Conference on Artificial Neural Networks and Neural Information Processing.
7. S. Osowski, K. Siwek and T. Markiewicz, "MLP and SVM networks - a comparative study," Proceedings of the 6th Nordic Signal Processing Symposium, 2004. NORSIG 2004., Espoo, Finland, 2004, pp. 37-40.
8. Zou Q, Qu K, Luo Y, Yin D, Ju Y, Tang H. Predicting Diabetes Mellitus with Machine Learning Techniques. Front Genet. 2018;9:515. Published 2018 Nov 6. doi:10.3389/fgene.2018.00515
9. Bishop, C.M., 1995. Neural networks for pattern recognition. Oxford university press.
10. https://machinelearningmastery.com/train-neural-networks-with-noise-to-reduce-overfitting/
11. L. Holmstrom and P. Koistinen, "Using additive noise in back-propagation training," in IEEE Transactions on Neural Networks, vol. 3, no. 1, pp. 24-38, Jan. 1992.
12. https://towardsdatascience.com/multi-layer-neural-networks-with-sigmoid-function-deep-learning-for-rookies-2-bf464f09eb7f (MLP image source)
13. https://towardsdatascience.com/support-vector-machines-for-classification-fc7c1565e3 (SVM image)
14. Ashiquzzaman A. et al. (2018) Reduction of Overfitting in Diabetes Prediction Using Deep Learning Neural Network. In: Kim K., Kim H., Baek N. (eds) IT Convergence and Security 2017. Lecture Notes in Electrical Engineering, vol 449. Springer, Singapore
15. Course slides for INM427 Neural Computing

Glossary

Perceptron : a perceptron is an algorithm for supervised learning of binary classifiers. They consist of four main parts including input values, weights and bias, net sum, and an activation function. They are of two type- single or multi-layer.

Weights : Coefficients used to multiply with inputs to a neural network layer. It is like regression coefficients. Values are generally initialized to small values during start of training. There are more complex methods like Adam, RMSProp used to update weights adaptively during backpropagation.

Bias: Intercept added to shift the weighted sum of inputs before activation.

Activation : An activation function is a simple mapping of summed weighted input to the output of the neuron. It is called an activation function because it governs the threshold at which the neuron is activated and strength of the output signal

Backpropagation : An algorithm for supervised learning of artificial neural networks using gradient descent based on the chain rule in differentiation, which consists of multiple runs of a two-step process -a forward pass and backward pass. During the forward pass the signal flows from the input layers, through the hidden layers to the output layers. The output is calculated based on the intermediate layer's setup. The error with the target is determined. During the backward pass, the gradient of the error with respect to each learned parameter is propagated backwards through the network. This modifies the weights and bias for each hidden layer.

Matlab command: Fitcsvm : Train support vector machine (SVM) classifier for one-class and binary classification in Matlab. Patternet : Pattern recognition networks are feedforward networks that can be trained to classify inputs according to target classes in Matlab.

Bayesian optimization : The Bayesian optimization algorithm attempts to minimize a scalar objective function $f(x)$ for x in a bounded domain

Stochastic gradient descent : Stochastic gradient descent refers to calculating the derivative from each training data instance and calculating the update immediately.

Scaled Conjugate gradient descent: SCG is a variation of a conjugate gradient method which avoids the line-search per learning iteration by using a Levenberg-Marquardt approach in order to scale the step size.

Generalizability : Ability of a model to learn from a given dataset and adapt to predict accurately on a previously unseen data dataset. An assumption is that the new dataset is drawn from the same data distribution.

Kernel Trick : In SVM, to transform data into a higher dimensional data, to classify better using a hyperplane, it is not necessary to compute the exact transformation but just the inner product of the data in that higher dimensional space. The inner product between two vectors is the sum of the multiplication of each pair of input values.

Box constraint : A parameter that controls the maximum penalty imposed on margin-violating observations, and aids in preventing overfitting (regularization). If you increase the box constraint, then the SVM classifier assigns fewer support vectors. However, increasing the box constraint can lead to longer training times.

F1 score : The F1 score can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst at 0

Implementation details

All data is available in the 'Data' folder. Code is available in 'Code' folder. Best trained models are available in 'Models' folders. Grid Search and Bayesian optimization results are available in 'Results' folder.

To run the model evaluation on the test data set, please run Evaluation.m script. There are multiple MATLAB scripts which are run to arrive at the final models. They can be categorized as below:

1. Data analysis – InitialDataAnalysis.m
2. Data Load and Split – DataLoadAndSplit.m

3. 1st level Hyper parameter tuning using Grid Search and Time complexity study:
    a. MLP_GridSearch.m
    b. SVM_Linear_GridSearch.m
    c. SVM_Gaussian_GridSearch.m
    d. SVM_Poly_GridSearch

4. 2nd level Hyper parameter tuning using Bayesian optimization and training best models on full training data
    a. MLP_Hyper_BayesOpt.m
    b. SVM_Linear_Hyper_BayesOpt.m
    c. SVM_Gaussian_Hyper_BayesOpt.m

5. Evaluation of best models from MLP and SVM - Evaluation.m

6. Helper functions
    a. ADASYN.m (Source: Author: Dominic Siedhoff , Matlab)
    b. PerformanceMetrics.m
    c. [ModelName]KFold _F1_Score Loss.m

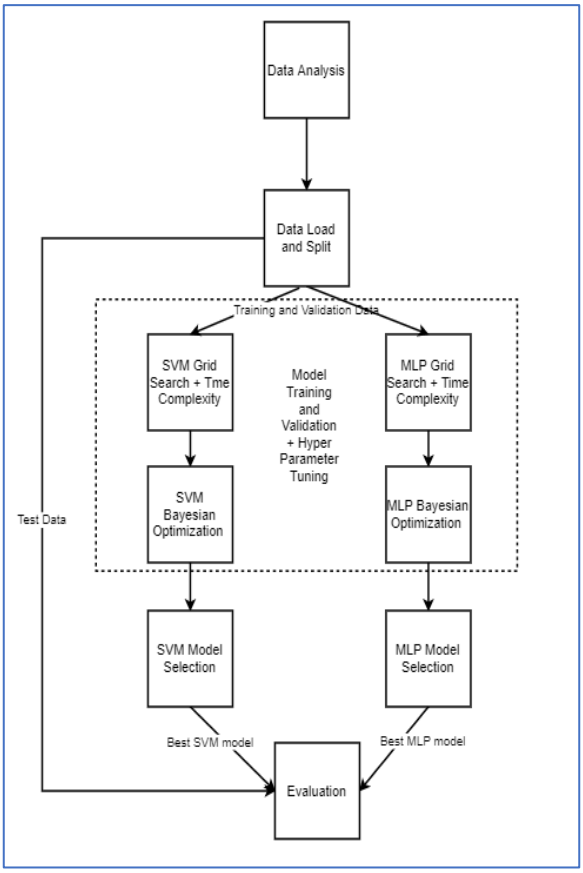

Figure 7 Overall algorithm comparison process