A Comparative Study in Early Stage Diabetes Risk Prediction Using Multilayer Perceptron and Support Vector Machine

Vo Tri Duc Sang
tri.vo@city.ac.uk

Abstract

The goal of this research is to successfully apply Machine Learning to the medical field by building two models to predict early-stage diabetes. The two chosen models are Multilayer Perceptron and Support Vector Machine. The results from both models were evaluated and contrasted against each other to find the best model using two key measurements Confusion Matrix and Area under the ROC curve. Hyper-parameter search is applied to both models. The result shows that MLP produces better performance than SVM for this classification problem.

1. Description and Motivation

According to the latest report from World Health Organisation (WHO) in November 2021[1], there were 420 million people were diagnosed with diabetes in 2014. This is a significant increase when compared to 108 million people in 1980. The increasing rate is different between low, middle, and high-income nations where the low and middle-income nations increase more rapidly than the high-income nations. Diabetes is not the only disease by itself, which means that when people got diabetes they are more likely to have other diseases such as kidney failure, heart attacks, blindness, stroke, and many others. In another report from WHO in 2019, approximately 1.5 million people were caused by diabetes, which makes it into the top 10 reasons for all death. There are many reasons for diabetes such as obesity, living a sedentary lifestyle, increasing age, and a bad diet. It sounds horrifying but there are many ways or methods to prevent diabetes from developing such as regular exercise, a healthy diet, avoiding eating junk foods or tobacco, and maintaining normal body weight. There are also methods to treat it from an early-stage such as a healthy diet, physical activities, medication, and regular screening and treatments.

The rapid advancement of technology in the last few decades has had profound impacts on improving many aspects of human life. The medical field is one of those areas that is benefited the most from those innovations, and inventions. For example, the application of machine learning, computer vision, or neural network in disease early detection, prevention and treatment has changed the life of many people. From that standpoint, in this study, we will build two models with the purpose of predicting the early-stage diabetes risk based on the seventeen different attributes. Feed-forward Multilayer Perceptron and Support Vector Machine are the two chosen models for this task.

2. Dataset

The dataset used in this study was obtained from UCI Machine Learning Repository[2]. The data has been collected from the patients using a direct questionnaire from Sylhet Diabetes Hospital of Sylhet, Bangladesh. The dataset contains information on 17 attributes including dependent variables and 520 observations. The Table below gives the description of the data.

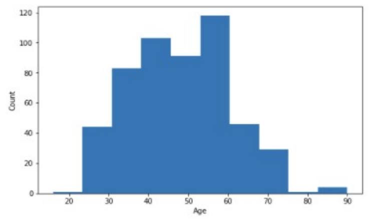
Feature Name	Data Type	Values	Yes Counts	No Counts
Age	Numerical (Discrete)	20 to 65	NA	NA
Sex	Categorical (Binary)	1.Male, 2.Female	NA	NA
Polyuria	Categorical (Binary)	1.Yes, 2.No.	258	242
Polydipsia	Categorical (Binary)	1.Yes, 2.No.	233	287
Sudden Weight Loss	Categorical (Binary)	1.Yes, 2.No.	217	303
Weakness	Categorical (Binary)	1.Yes, 2.No.	305	215
Polyphagia	Categorical (Binary)	1.Yes, 2.No.	237	283
Genital Thrush	Categorical (Binary)	1.Yes, 2.No.	116	404

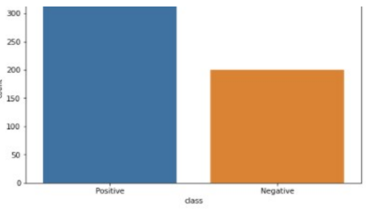
Visual Blurring	Categorical (Binary)	1.Yes, 2.No.	237	283
Itching	Categorical (Binary)	1.Yes, 2.No.	267	253
Irritability	Categorical (Binary)	1.Yes, 2.No.	123	394
Delayed Healing	Categorical (Binary)	1.Yes, 2.No.	239	281
Partial Paresis	Categorical (Binary)	1.Yes, 2.No.	224	296
Muscle Stiffness	Categorical (Binary)	1.Yes, 2.No.	195	325
Alopecia	Categorical (Binary)	1.Yes, 2.No.	179	341
Obesity	Categorical (Binary)	1.Yes, 2.No.	88	432
Class	Categorical (Binary)	1.Positive, 2.Negative.	320	200

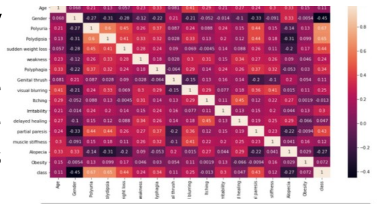
Exploratory Data Analysis (EDA)

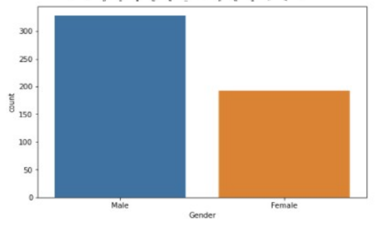
One of the most important parts of any Machine Learning task is exploratory data analysis. This process is extremely useful to identify common data problems such as missing values, duplicated observations, outliers, and class imbalance. After going through this process, here are some key findings:

- There are no missing values or duplicated values or error inputs.
- One of the most interesting about this dataset is that out of 17 attributes, 16 of them are categorical binary variables and one numerical variable is age. Hence, there are almost certain that there is no outliers issue that includes the age variable. When looking at the age distribution in Figure 1, we can confirm that there is indeed no outlier issue.
- From the correlation heat map, we can identify which variable has the strongest correlation with diabetes. Polydipsia and Polyphagia are the two variables that have the strongest positive effect on diabetes, while gender and Alopecia are the variables with the strongest negative effect on diabetes.
- Class imbalance is another key problem when dealing with binary classification. There are 320 positive results and 200 negative results, Figure 3. Difference between positive and negative results is not large enough to consider a class imbalance problem. Hence, we do not really need to use techniques such as SMOTE.
- In the final part of EDA, we will convert all the binary categorical variable values to numerical values (0 and 1) and apply data normalization to the age attribute using MinMaxScaler.









3. The Summary of Models with Pros and Cons

Multilayer Perceptron (MLP)

Multilayer Perceptron (MLP) is a fully connected class of feed-forward neural network with one or more hidden layers combined together with an input layer and an output layer to become a network[3]. Each of these layers performs its own task within the network. The input layer task is to receive the input data to be processed. The hidden layers, we can have any arbitrary number of hidden layers, are located in between the input layer and output layer. Its task is to perform a number of transformations on the data received from the input layer. This transformation will help to produce an output that is close enough to the expected output. The hidden layer is the key computational engine of the Multilayer Perceptron where all the complex computations happen. The output layer is required to perform the final task of making predictions and classification. Just like a feed-forward neural network, data are being fed in the forward direction from the input layer to hidden layers and finally to the output layer. Each of these layers is assigned weights and biases during the learning phase of the model.

The Back-Propagation Algorithm is used for updating the weights of an MLP. A loss function is defined usually on the defined regression or classification task. When data are being fed forward by the input layer to the hidden layers, the weights assigned to the hidden layers will modify those values from the input layer to produce the output. This output is then compared to the ground truth label by the loss function and a loss score is generated. According to this loss score, the weights of the MLP are updated in back-propagation from the output layer to the input layer. MLP is a go-to method for many tasks such as pattern classification, recognition, prediction, and approximation.

Pros

- Use for both regression and classification problems.
- Suitable to any size of input data, large or small.
- Produce fast predictions once trained.
- Work well to model with non-linear data with la arge number of inputs.
- Neural networks can be trained with any number of inputs and layers.

Cons

- MLP is black box. It is difficult to predict the effects each independent variable has on the dependent variable.
- MLP depends a lot on training data. This leads to the problem of over-fitting and generalization.
- Computation is difficult and time-consuming.
- A possible problem of local minimum. Different assigned weights may cause different results.
- Many hyper-parameters need to be tuned
- Sensitive to feature scaling,

Support Vector Machine (SVM)

Support Vector Machine is a supervised machine learning algorithm[4]. It is one of the simplest and arguably the most elegant method for classification but it can also use for regression tasks. Each object we want to classify is represented as a point in an n-dimensional space and the coordinates of this point are usually called features. SVM performs the classification test by drawing a hyperplane that is a line in 2D or a plane in 3D in such a way that all points of one category are on one side of the hyperplane and all points of the other category are on the other side. While there could be multiple hyperplanes, SVM tries to find the one that best separates the two categories. In the sense that it maximizes the distance to points in either category. This distance is called the margin and the points.

To find this hyperplane, SVM requires a set of points that are already labelled with the correct category. This is the reason why SVM is considered to be a supervised learning algorithm. In background structure, SVM solves a convex optimization problem that maximizes the margin and where the constraints say that points of each category should be on the correct side of the hyperplane.

Pros

- Works relatively well when there is a clear margin of separation between classes.
- More effective in high dimensional space.
- More effective when the number of dimensions is greater than the number of samples.
- Memory efficient.

Cons

- Low or poor performance on a large data set.
- Does not perform very well when classes are not distinct.
- When the number of features for each data point is greater than the number of training data samples, It will underperform.
- There are no probability estimates with SVM.

4. Hypothesis Statement

From some literature reviews, articles, and research papers, SVM model gives better results than MLP model in most cases[5]. For example in a very similar task of classification of Breast Cancer in Wisconsin hospital using these two algorithms, the SVM model performed better than MLP model. In this study, we also expect similar end results. This is our hypothesis question that we still have the answer by the end of this research.

5. Description of Choice of Training and Evaluation Methodology

In ML, validation set and "whether should we use it?" are among the most confusing term and commonly debated questions. After doing lots of research, there is no 100% clear answer about this. One side says the validation set is particularly useful and necessary to access the generalisation of the models and if we have a large data sample, it would be perfect to use a validation set. Another side said a validation set is not necessary when we already have the testing set. If we have a small sample size, a validation set will reduce the performance of the model since we have less data for training and testing. In this research paper, even though we have a small sample size, we still decide to split the data into a training (80%) and a testing (20%). And we also split the training data into a training set (80&) and a validation set (20%) for evaluating and comparing both models. One of the reasons is also because we want to test the performance of both models when having less training data to confirm what we find out about the pros and cons of both models above which is a decrease in performance with less data or smaller data. First, we train both models using training sets. In both models, hyper-parameter tuning is required to find the models with their best parameters which results in the most accurate prediction using the validation set. The remaining 20% of the data, the testing set, is used for testing the two chosen models with their best parameters and contrasting the results with each other. The two figures that we will use to graphically illustrate quantitative aspects of our results are the confusion matrix and the Area under the ROC curve.

6. Choice of Parameters and Experimental Results

Choice of Parameters - MLP

MLP was implemented using Pytorch library with binary cross-entropy loss to update the weights of the architectures. The initial MLP model was designed with an input layer consisting of 16 different nodes, which was also the count of input features. This was followed by a hidden layer with 32 nodes. The final output layer consisted of 1 node which was activated by the sigmoid function. The sigmoid function resulted in the output of the last node as the prediction probability of the given input. The hidden layers were activated by a non-linear activation function called ReLu. The ReLu function helps in solving the vanishing gradient problem that most neural networks suffer from. A dropout layer was also attached to the hidden layers for decreasing the variance of the MLP model. Dropout layers help in better generalisation by switching off random nodes in the hidden layer during training. Adam optimizer function was used to optimize the model. In the initial step of the neural network process, a set of random weights was chosen to avoid the networks being trapped into the local minima each time the training happens and because of this, the final performance of accuracy and results may be different each time. Most machine learning and deep learning algorithms have some parameters that can be adjusted which are called hyper-parameters. The chosen hyper-parameters for MLP are the number of layers (input layer, 1 or 2 or 3 hidden layers, and 1 output layers), learning rate, number of epochs, momentum, dropout (regularization),

Choice of Parameters - SVM

SVM was implemented using Scikit-learn library. SVM does not have the problem of local minima since the design of the model does not assign any random weights. Instead, SVM creates a decision boundary that makes the distinction between two or more classes. How to draw or determine the decision boundary is the most critical part of SVM algorithms and this is where hyper-parameters tuning will help to find that optimal decision boundary. In SVM, Kernel Function, C (Box Constraint), G (Gamma) are the three hyper-parameters.

Experimental Results – MLP

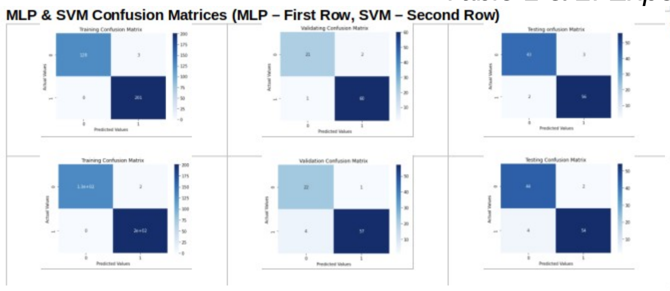
Number of Layer	Dropout	Learning Rate	Epochs	Momentum	Validation Accuracy	Hidden Layer	Dropout	Learning Rate	Epochs	Momentum	Validation Accuracy
3	0.2	0.005	40	0.5	90.48	3	0.4	0.01	50	0.5	90.48
4	0.2	0.005	40	0.5	86.90	3	0.4	0.01	100	0.5	90.48
5	0.2	0.005	40	0.5	89.29	3	0.4	0.01	200	0.5	92.86
3	0.2	0.005	40	0.5	89.29	3	0.4	0.01	300	0.5	94.05
3	0.3	0.005	40	0.5	86.90	3	0.4	0.01	400	0.5	90.48
3	0.4	0.005	40	0.5	90.48	3	0.4	0.01	500	0.5	91.67

3	0.5	0.005	40	0.5	90.48	3	0.4	0.01	300	0.01	88.10
3	0.4	0.05	40	0.5	89.29	3	0.4	0.01	300	0.05	96.43
3	0.4	0.01	40	0.5	91.67	3	0.4	0.01	300	0.1	94.48
3	0.4	0.005	40	0.5	90.48	3	0.4	0.01	300	0.5	86.90
3	0.4	0.0001	40	0.5	72.67	3	0.4	0.01	300	0.9	90.48
3	0.4	0.0005	40	0.5	82.14	3	0.4	0.01	300	0.99	88.10

Experimental Results – SVM

Kernel Function	C	Gamma	Polynomial Order	Validation Accuracy	Kernel Function	C	Gamma	Polynomial Order	Validation Accuracy
Linear	1	NA	NA	89.29	Poly	0.7	1	3	94.05
Poly	1	NA	NA	94.05	Poly	0.8	1	3	94.05
Poly	1	1	1	89.29	Poly	0.9	1	3	94.05
Poly	1	1	2	92.86	Poly	1	1	3	94.05
Poly	1	1	3	94.05	Poly	0.7	0.1	3	80.95
Poly	1	1	4	94.05	Poly	0.7	0.2	3	88.10
Poly	1	1	5	94.05	Poly	0.7	0.3	3	92.86
Poly	1	1	6	92.86	Poly	0.7	0.4	3	92.86
Poly	0.1	1	3	91.67	Poly	0.7	0.5	3	92.86
Poly	0.2	1	3	91.67	Poly	0.7	0.6	3	91.67
Poly	0.3	1	3	90.48	Poly	0.7	0.7	3	90.48
Poly	0.4	1	3	92.86	Poly	0.7	0.8	3	92.86
Poly	0.5	1	3	92.86	Poly	0.7	0.9	3	92.86
Poly	0.6	1	3	92.86	Poly	0.7	1	3	94.05

Table 1 & 2: Experimental Results



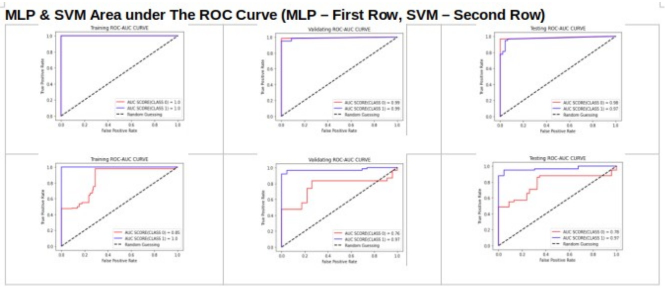


Table 3: Confusion Matrices & Area under ROC Curves

7. Analysis and Critical Evaluation of Results

The above two tables display the hyper-parameters searches for both MLP and SVM. Each red block of rows represents a search for one hyper-parameter and the best result of that search was highlighted in green and the cell with red colour is the best value of the search. The yellow rows are the final results with the best hyper-parameters of both models. One important thing which needs to be mentioned is that the values in both tables are not the results of ONLY one run from both models. We have repeatedly run the experiment many times in order to select the most influence value for each hyper-parameter. The reason we ran it many times is because the results for the MLP model are easily impacted by the initial weights which lead to different results for every run. SVM does not have this problem which means the result is very consistent every time we ran. The results from the hyper-parameters search show that both models are strongly influenced by the different combinations of the hyper-parameters. Based solely on the results from the tables, MLP seems to be more impacted by different combinations when the accuracies range from 96.43% to 72.67%. However, this may be caused by the range of hyper-parameter values in SVM like C(0.1-1) and gamma(0.1-1).
In the training process, SVM achieved 99.39% accuracy and MLP achieved 99.10%. SVM misclassification is 2 compared to 3 of MLP. In the validation process, both SVM achieved 94.05%

with 5 misclassifications and MLP achieved 96.43% with 3 misclassifications. In the testing process, SVM achieved 94.23% and MLP achieved 95.19%. SVM misclassification is 6 compared to 5 of MLP. We can say that the MLP model is more generalisation than SVM even though SVM performed better in the training phase but MLP performed better in validation and test phases. Again, we performed this experiment many times to account for the random weight initialization but the performance of MLP is better than SVM or at least equal most of the time.

Achieving high prediction accuracies using machine learning in the medical field is very important but that is not a complete picture. Anything related to medical such as diseases, life, and death is on the line. We need our model to identify the people having diseases with the lowest misclassification possible, in our case class 1. We don't want to fall into a situation where a person with a disease to be predicted without a disease. And because of that he/she does not get treatment as early as possible and the disease gets worst without his or her knowing and eventually die. This is where we can use Area Under ROC curve to visualize and measure the classification accuracy for each class. From the graphs, we can visually see that MLP curves have better performance than SVM curves again. MLP was able to predict people having disease with much lower misclassification than SVM. The total misclassification of class 1 of MLP for train, validate, and test is 3, while this number for SVM is 8. This can also visually see using Area under the ROC curves.

8. Conclusions, Lessons Learned, References and Future Work

In this paper, we try to apply two machine learning algorithms, MLP and SVM, to predict early-stage diabetes risk and evaluate and contrast the results from both models to see how effective they are and which one is a better model. Based on the above analysis of the results using confusion matrices and the Area under ROC curves, we can finally answer our hypothesis question. And it is not what we proposed because MLP is performed better than SVM in this particular dataset or problem. We have performed the experiment many times and most of the time MLP gives better results even though MLP results is strong influence by the initial weights assigned which causes the results to be different each run. This can be explained by the pros and cons of both models. SVM is not performed well in a small data set, which is our case. In an additional effort to prove this point, we performed another experiment by splitting the dataset into training and testing without a validation set. By doing this, we increase the amount of data for training. The result is significantly increased even better than MLP with more than 98%. This can be a lesson learned to keep in mind when performing an SVM model, we need more data or use a large dataset.

For future work, we did not apply k-fold cross-validation in both models. K-fold cross-validation is a very useful and effective method to improve ML performance, especially for small data sets. For SVM, we can use more kernels such as rbf or sigmoid, and for MLP we can use different activation functions such as Swish or Leaky-ReLu. We can also use different techniques such as boosting and bagging.

References

[1] Diabetes: https://www.who.int/news-room/fact-sheets/detail/diabetes.

[2] Data set: https://archive.ics.uci.edu/ml/datasets/Early+stage+diabetes+risk+prediction+dataset.

[3] "Neural Networks for Pattern Recognition" by Christopher Bishop 1995.

[4] "An Introduction to Support Vector Machines and Other Kernel-based Learning Methods" by Nello Cristianini, University of London, John Shawe-Taylor, Royal Holloway, University of London.

[5] E.A.Zanaty: Support Vector Machines (SVMs) versus Multilayer Perception (MLP) in data classification. https://doi.org/10.1016/j.eij.2012.08.002.

Appendix

Glossary

Confusion Matrix: A confusion matrix is a technique for summarizing the performance of a classification algorithm. Classification accuracy alone can be misleading if you have an unequal number of observations in each class or if you have more than two classes in your dataset.

Area under ROC: As the area under an ROC curve is a measure of the usefulness of a test in general, where a greater area means a more useful test, the areas under ROC curves are used to compare the usefulness of tests. The term ROC stands for Receiver Operating Characteristic.

Black Box: is shorthand for models that are sufficiently complex that they are not straightforwardly interpretable to humans. Lack of interpretability in predictive models can undermine trust in those models.

Class imbalance: In a binary classification problem with data samples from two groups, class imbalance occurs when one class, the minority group, contains significantly fewer samples than the other class, the majority group. In many problems [3,4,5,6,7], the minority group is the class of interest, i.e., the positive class.

Hyper-parameters: is choosing a set of optimal hyper-parameters for a learning algorithm. A hyper-parameter is a model argument whose value is set before the learning process begins. The key to machine learning algorithms is hyper-parameter tuning.

Feature Scaling: Feature scaling is a method used to normalize the range of independent variables or features of data. In data processing, it is also known as data normalization and is generally performed during the data preprocessing step.

Generalizability: Very simply, generalizability is a measure of how useful the results of a study are for a broader group of people or situations. If the results of a study are broadly applicable to many different types of people or situations, the study is said to have good generalizability.

Over-fitting: is a modelling error in statistics that occurs when a function is too closely aligned to a limited set of data points. As a result, the model is useful in reference only to its initial data set, and not to any other data sets.

Hidden Layer: A hidden layer in an artificial neural network is a layer in between input layers and output layers, where artificial neurons take in a set of weighted inputs and produce an output through an activation function.

Learning Rate: in machine learning and statistics, the learning rate is a tuning parameter in an optimization algorithm that determines the step size at each iteration while moving toward a minimum of a loss function.

Momentum: is a variant of the stochastic gradient descent. It replaces the gradient with a momentum which is an aggregate of gradients.

Dropout: Dropout is a technique where randomly selected neurons are ignored during training. They are “dropped-out” randomly. This means that their contribution to the activation of downstream neurons is temporally removed on the forward pass and any weight updates are not applied to the neuron on the backward pass.

Kernel Function: maps the observations into some feature space. Ideally the observations are more easily (linearly) separable after this transformation. There are mainly four different types of kernels: Linear, Polynomial, RBF, and Sigmoid that are popular in SVM classifier.

C (Box Constraint): is a regularization parameter for SVMs model. The C parameter tells SVM optimization how much we want 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 misclassified more points. For a very tiny values of C, we should get misclassified examples, often even if our training data is linearly separable.

G (Gamma): is a hyper-parameter which we have to set before training model, Gamma decides that how much curvature we want in a decision boundary. A high Gamma means more curvature. Conversely, a low Gamma means less curvature.

Implementation Details

After performing exploratory data analysis, we split the data into train, validate and test sets. We create two separate python files, one for each model. In the first step, we train both models using the training data set and produce the training accuracy. In this step, we use the default setting for both models. There is no hyper-parameter search or tuning in this step. In the second step, we did hyper-parameter search or tuning for both models using the validation data set. The search or tuning was run separately for each hyper-parameter in each model. The hyper-parameter with the value which produced the highest accuracy was selected. After finishing tuning all the hyper-parameters for each model, the best values of all hyper-parameters were saved into a model for both SVM and MLP. In the final step, we load the save model for SVM and MLP and evaluate the models using test data set. In this final step, we also evaluate the final model using training and validating data sets for model comparison. With the result from the evaluation process, confusion matrices and the Area under ROC curves were created for model analysis.

For example, in the SVM model, we first tuned the kernel function hyper-parameter. After getting the best value of the kernel function, we used that value for the next tuning of the polynomial order hyper-parameter and collected the best value. The best values of kernel function and polynomial order were used in the next hyper-parameter tuning of C and collected its best results. The best result of C, kernel function, and polynomial order were used to tune gamma and collected the best result. The final model was reached after we tuned all the hyper-parameters and collected all the best values for each one of them. The same implementation process was done for MLP.