A Multi-Classification study comparing Multi-Layer Perceptrons (MLP) and Support Vector Machines (SVM)

Humza Khan

Abstract - This research paper aims to critically evaluate, compare, and study the performance of two neural networks on a multi-classification model. The two models we evaluate is a feedforward multi-layer perceptron (MLP) algorithm and support vector machines (SVM). The models are tweaked on varying hyperparameters using a random search function and validated through cross-validation. The best performing models and scores are compared using performance measures such as Accuracy, Precision, Recall, Receiver Operation Curves (ROC) and Confusion Matrices.

Keywords: *Machine Learning; Tensor; Python; Artificial Neural Networks; Multi-Layer Perceptrons; Support Vector Machines; Image Classification; MNIST datasets; Handwritten Digit Recognition; Character Recognition*

1. Brief Description and Motivation of Problem

Digital image processing is an area that enables the manipulation of digital images via computer algorithms. Its primary goal is the classification of objects into a number of classes or categories. Our goal is to implement a pattern classification method to recognise handwritten digits provided in the MNIST dataset. Handwritten digit recognition of characters has a history since the 1980s. This has significant value in areas such as online handwritten digit recognition on computers, processing cheques in banking, number plate recognition, postal mail sorting etc. However, there are challenges faced as handwritten digits are not always the same orientation, size, thickness and so on. In this paper we tackle the handwritten digit recognition problem. Our aim is to present classification methods based on statistical techniques with good performance for classifying handwritten digital images. We use two different statistical approaches and provide a reasonable understanding of models like SVM and MLP for handwritten digit recognition. It furthermore gives you the information about which algorithm is efficient in performing the task of digit recognition. [1]

2. Description of Dataset

MNIST dataset (Modified National Institute of Standards and Technology database) is the subset of the NIST dataset which consists of: Special Database 1 and Special Database 2. MNIST dataset can be accessed from the Yann Lecun website [2]. It has become a traditional dataset for testing theories of pattern recognition and machine learning algorithms. MNIST dataset contains 60,000 handwritten digital images (0-9) for classifier training and 10,000 handwritten digital images for classifier testing, which are derived from the same distribution. These are black and white digits which are normalised, fixed size, centered and the intensity lies at the centre of the image with 28 x 28 pixels. Each element is binary with dimensionality of each image sample vector is 28 * 28 = 784. We therefore can comment that the 28 x 28 pixels are flattened into a 1D vector which is 784 pixels in size. Our digits are labelled in a 0-9 vector that contains ten values, one for each digit. One of these digits is set to 1 to represent the digit in the index and the rest to 0. For example, the digit 4 is represented by the vector [0, 0, 0, 0, 1, 0, 0, 0, 0, 0]. Since we want to test neural networks as applied to real world practical problems, the MNIST dataset is ideal as it is already pre-processed, segmented, and normalised. Therefore, little effort is required in pre-processing and formatting. In this report we have used a 10% sample of the data which consists of 6000 training images and 1000 test images. The general problem we would expect are misclassifications between the digits as the same digit can be written in different ways as the uniqueness and variety of individuals differs when to comes to handwriting.

Fig 1. n.o of digit classes in Training Dataset

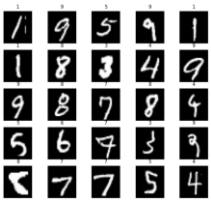

Fig 2. Plotting of random digits from MNIST

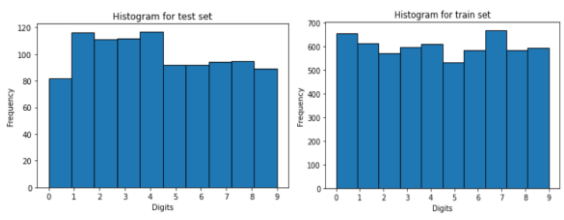

Fig 3. Histogram of test and train datasets

3. Summary of Neural Network Models

3.1. Multi-Layer Perceptron (MLP)

Multi-Layer Perceptron (MLP) algorithm is a feedforward Artificial Neural Network (ANN). It is a type of supervised learning classifier consisting of three types of layers: input layer, hidden layers, and output layer [3]. A perceptron is an artificial neuron also called a node that transfers information between nodes. Each layer consists of several nodes which are interconnected to every other node of the next layer. These are given adjustable weights by applying activation functions for each of the neurons in the hidden layer. Every hidden layer can have different activation functions for processing. MLP algorithm operates with values in a feed-forward way and travels through multiple layers to give an output result. Error term is used in this context to describe the difference between the actual and desired output. The supervised learning technique used here is called backpropagation. The error described above is backpropagated through the network by adjusting the weights to synchronize with each connection, which is repeated several times to obtain the target output [4].

3.2. Support Vector Machines (SVM)

Support Vector Machines (SVM) is a supervised learning algorithm that aims to construct a hyperplane or set of hyperplanes in an n-dimensional space where n is the number of features [4]. The value of the feature is characterized by a specific coordinate. The classification is performed by finding the hyperplane that differentiates the two classes and the hyperplane that correctly separates the classes is therefore chosen. SVM chooses the extreme vectors in selecting which vectors are to be used in the creation of the hyperplane. These are defined as support vectors. An SVM seeks to find the optimal separating hyperplane between classes by concentrating on the training cases that lie at the edge of the class distributions and the support vectors. The other training cases are discarded. We can have linear and non-linear SVMS. In this research paper, we have used linear SVM as the basic model and linear/non-linear SVMs for our search for best parameters using random search cross validation.

Model	Pros	Cons
MLP	▪ Can be applied to complex non-linear problems ▪ Yields quality results with complex large data sets ▪ Same accuracy ratio can be achieved with smaller dataset	▪ Positive correlation between size of dataset and complexity. ▪ Computations can be time-consuming with traditional CPUs ▪ The accuracy of the model depends on the quality of the training
SVM	▪ Works well with a clear margin of separation between classes (no overfitting) ▪ SVMs are less complex as they emphasis on training samples at the edge of class distributions ▪ More effective in high dimensional spaces	▪ Not suitable for large datasets ▪ Not effective with overlapping target classes (more noise in data) ▪ Choosing an appropriate Kernel function to handle non-linear data can be difficult and complex

Fig.4 Advantages and Disadvantages of MLP and SVM

4. Hypothesis Statement

In this work, we are comparing MLP and SVM models for the MNIST handwritten image Dataset. Given the computational complexity, processing time and large dataset, the expectation is that SVM will perform better than MLP. The conclusion given by the author E.A. Zanaty [4] is in line with this. In work [5], the author concluded that random search performs better with larger datasets than a gridsearch. We perform a random search on the hyper-parameters for both models to tune and optimise in order to achieve better performance (accuracy, precision, recall, F1 and AUC).

5. Choice of Parameters, Training, Methodology and Architecture of Models

The data was downloaded directly from Yann LeCun website and transformed to Tensor. Data was large and each row represented one of the 60,000 training instances. Similarly, we have 10,000 testing instances. Data was already split into training and testing data. This dataset is already balanced and therefore no oversampling techniques such as SMOTE was required. 10% of this data was sampled in the train (6000 images) and test sets (1000). The input data was reshaped so all the images present in the dataset are converted to 2-dimensional images. Each column is represented by each of the 784 pixels in a 28 x 28 image. Each pixel value of the image remains between 0 and 255. We would usually normalise

these pixel values so that the input features range between 0 and 1. The data downloaded however is already size normalised and centered in a fixed size image. The MNIST dataset are uniformly distributed class labels 0-9 corresponding to the respective handwritten digit shown in each image. The shape of our training data is just (60,000,), which shows that is a 1-dimensional vector which contains numeric target labels (0-9) and therefore the output will have a one hot encoded vector with 10-dimensions. Both the models were trained on training data and predicted and tested on the testing dataset. RandomsearchCV algorithm was utilized to find the best parameters in both models. A 5-fold Cross-validation technique is also applied to both models. Although F1 score and precision are good scoring mechanisms, accuracy is vital in digit recognition hence we used this for ranking our performance for scoring purposes. Early stopping was used on the MLP model which is useful as it stops the neural network once it notices that the performance is degrading on the training dataset. This prevents the model from overfitting and better the neural network generalisation. The MLP and SVM models are compared on basis of time to execute the computation, confusion matrices, performance scores and ROC curve.

5.1 Architecture and Parameters - MLP

We utilised the Pytorch library to build the MLP neural network on the training data. Firstly, a basic MLP model was created without hyper-parameter tuning. A linear model is developed with 784 input features and 10 output features. Leaky Rectified Linear Unit (Leaky ReLU) is used as an activation function as this function overcomes the vanishing gradient problem [7], allowing the model to learn faster and perform better. It fixes the dying ReLU problem [8] as it doesn't have zero-slope parts. It also speeds up training and converge sooner as having the mean activation close to 0 makes training faster. Skorch is a scikit-learn compatible neural network library that wraps the PyTorch classifier. Cross-Entropy Loss is a measure of the difference in two probability distributions between the actual and predicted class. LogSoftmax and NLLLoss is combined into a single class by using Cross-Entropy. The calculated loss values drive the updating of the weights and hence the neural network model is trained. The ideal case would be when the actual and predicted values have the same distributions for a class label and in this case the cross-entropy loss will be 0. Early stopping is used in the MLP model to prevent overfitting of this model [3].

Secondly, utilising sklearn, a random search function is used to find the best parameters and hence the most optimised model for MLP. Hyper-parameter tuning is important as it is used to prevent over/under fitting of the model and hence give us high performance. The random search is conducted using the learning rate, optimiser, number of hidden layers, maximum number of epochs, drop rates with a 5-fold cross-validation and accuracy is used as the scoring parameter. Performance scores (accuracy, precision, recall, f1-score, AUC) is calculated as the test data is used to predict 10 output classes. We further evaluate using confusion matrices, loss curves and an ROC curve.

5.2 Architecture and Parameters - SVM

The SVM model is first trained on a basic model with default parameters where C=1 and Kernel=Linear. RandomsearchCV is then used to improve the performance of the SVM. The hyper-parameters used for tuning are C, Gamma, Kernel and Degree, with accuracy as a scoring factor. Similar to MLP, a 5-fold cross-validation technique is used on the training data and the target class values are predicted using the test data. Confusion matrices, ROC Curve and a classification report is used to appraise the model [6].

6. Analysis, experimental and critical evaluation of results

Table 1 and 2 shows us the results for the basic MLP and SVM model and also the best parameters achieved having carried out a random search on both models. We can clearly see that the models improved upon carrying out random search as MLP and SVM improved in all performance metrics (accuracy, precision, recall, f-score) when compared to the basic models. The MLP performance metrics was affected by the neuron weights every time the model was re-run. The SVM values were impacted less when re-running the data.

The MLP basic model (87% accuracy) was improved upon running a hyper-parameter random search (90%) to find the best parameters. Adam optimiser worked best in both models. The random search reduced the max epoch to 450. This suggests that the basic model had more epochs than necessary and therefore the model may lose generalisation capacity by overfitting to the training data. Drop out is reduced by our random search. There are several reasons why dropout can hurt performance and hence the optimal here is 0.3. The optimal number of hidden layers reduced from 120 to 100.

In the SVM model, default degree used initially was 3. Adding more degrees to the model helps it linearly separate the classes, but as we can see the best model reduced this value to 2. The type of kernel and the value of C used is important as it can have a great impact on the SVM models performance. In the basic model of the Linear SVM, we used the default parameters where C = 1. A smaller C hyper-parameter in SVM helps to control the error margin with low variance and this value also helps regularising the model. The C remained unchanged in the best model suggesting this was the most optimal value to use. As the training data has been balanced, a lower C value would be expected for optimum results. A larger C can potentially give rise to overfitting. The linear SVM model gave us a 91% value across all the performance measures. These measures improved to 95% when a non-linear kernel 'Poly was selected. This suggests that the performance is directly affected by the choice of kernels.

MLP Model	Learning Rate	Maximum Epoch	Drop-out	Optimiser	Hidden Layers	Accuracy	Precision	Recall	F-Score
Basic	0.1	1000	0.5	Adam	120	87%	88%	87%	87%
Best	0.1	450	0.3	Adam	100	90%	91%	90%	90%

Table 1 – MLP Model results of basic and random search model

SVM Model	C	Gamma	Degree	Kernel	Accuracy	Precision	Recall	F-Score
Basic	1	1	3	Linear	91%	91%	91%	91%
Best	1	0.0001	2	Poly	95%	95%	95%	95%

Table 2 – SVM Model results of basic and random search model

From the confusion matrices in Fig.5, we notice an improvement within the models between the basic model and best models. For example, most of the classes were classified correctly over the test data. MLP basic model had wrong predictions in classes 8 and 9 which was corrected in the MLP best model. The least correctly predicted class in the MLP best model was class 5 that predicted 73 cases as true from the total. SVM best model least correctly predicted was class 9 that predicted 99 as true out of the total.

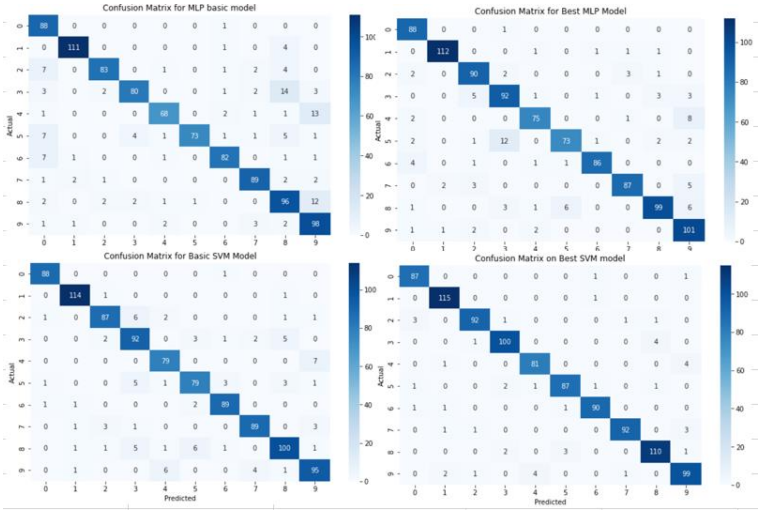

Fig. 5 Confusion matrices for MLP and SVM models

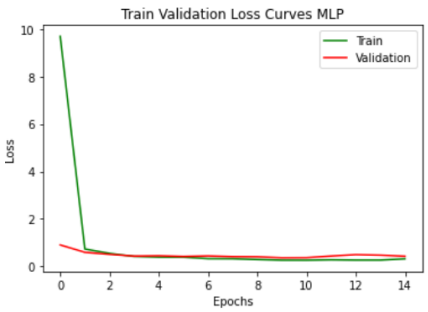

Fig. 6 Loss curves for MLP Best Model

Observing the results from Fig 6. We can conclude that there was no under/over fitting of the model as the loss curves for the train and test data overlap each other and are very similar. As the number of Epochs increase, the loss degrades over the test and train data.

The ROC curves are mapped using specificity (FPR) and sensitivity (TPR) scores. Area under curve (AUC) ranges between values 0 and 1, with 1 showing that 100% of the predictions are correct. A decent classifier ranges between the values 0.5 and 1. The higher the AUC, the better the model is in distinguishing the image. The AUC calculated for the best MLP model and best SVM model is 0.97 and 0.99 respectively. Hence the SVM model was able to correctly classify more of the target labels correctly when compared to the MLP model. The MLP model took longer to execute than the SVM model. MLP took approx. 288 seconds to run whereas SVM took approx. 186 seconds. In terms of complexity and time taken to execute, the SVM outperformed the MLP model.

During the initial experimental stage, we considered two parameter optimisation techniques random search and grid search. Random search performs better for larger datasets when compared with the grid search in terms of execution time. James & Yoshua [5] found that not all parameters are equally important to optimise, and random search generally performs better than a grid search in terms of computation time.

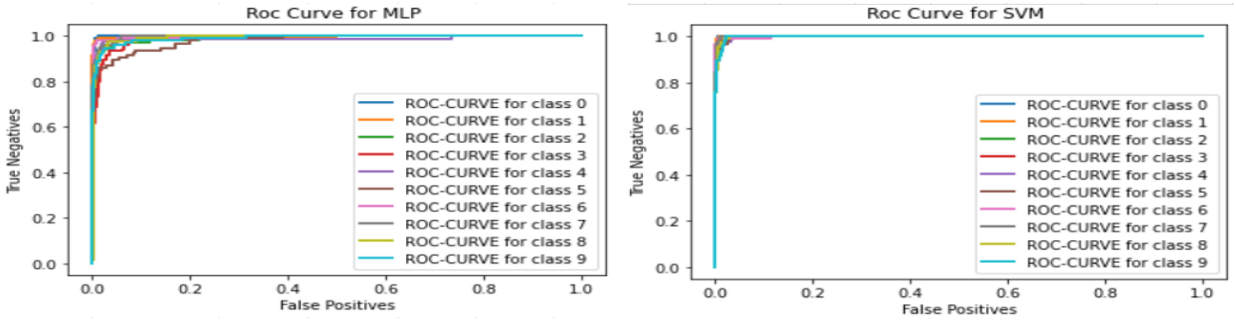

Fig. 7 RoC curves for Best MLP and SVM Models

7. Conclusion, Lessons Learned, Future Work & References

In this paper, we implemented two models for handwritten digit recognition using MNIST datasets based on MLP and SVM Models. We compared the models based on their characteristics to appraise which model provides the higher accuracy. The size of the dataset and complexity level of the MLP model is positively correlated. MLP also takes longer to train than SVM. SVMs are one of the basic classifiers which is why it is faster than most other models. Random search provided both models with high performance metrics such as accuracy etc hence random search is a better option in contrast to a grid search in finding the optimal parameters. MLP results also varied during training due to it being affected by the weights of the neurons. Confusion matrices is a key tool in determining whether class predictions are correct. We found that SVM gave the most accurate results for handwritten digit recognition. We can therefore conclude that SVM is most suitable for prediction problem on image data.

Future work can include creating a hybrid algorithm using more complex data. We can investigate how the complexity of the data and the size of the data can affect the MLP and SVM models. We can extend these applications and algorithms to tackle many handwritten digit recognition problems from banks to daily administration. We took a sample of the dataset available. We could take a larger sample of the data in the future and see how much more improvement we can obtain. We could also look to see if other models perform better such as KNN.

References

[1] Shamim, S. M., Miah, Md Badrul, Sarker, Angona, Rana, Masud, Jobair, Abdullah. (2018). Handwritten Digit Recognition Using Machine Learning Algorithms. Indonesian Journal of Science and Technology. 18. 10.17509/ijost.v3i1.10795.
[2] MNIST Database of Handwritten digits: http://yann.lecun.com/exdb/mnist/
[3] Ramchoun, Hassan & Amine, Mohammed & Janati Idrissi, Mohammed Amine & Ghanou, Youssef & Ettaouil, Mohamed. (2016). Multilayer Perceptron: Architecture Optimization and Training. International Journal of Interactive Multimedia and Artificial Inteligence. 4. 26-30. 10.9781/ijimai.2016.415.
[4] Support Vector Machines (SVMs) versus Multilayer Perception (MLP) in data classification. E.A. Zanaty
[5] Random Search for Hyper-Parameter Optimization James Bergstra, Yoshua Bengio Departement d'Informatique et de recherche op ´ erationnelle ´ Universite de Montr ´ eal ´ Montreal, QC, H3C 3J7, Canada
[6] Anguita, Davide & Boni, A. & Ridella, Sandro. (2003). A Digital Architecture for Support Vector Machines: Theory, Algorithm, and FPGA Implementation. IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council. 14. 993-1009. 10.1109/TNN.2003.816033.
[7] https://towardsdatascience.com/the-exploding-and-vanishing-gradients-problem-in-time-series-6b87d558d22
[8] https://medium.com/@danqing/a-practical-guide-to-relu-b83ca804f1f7
[9] https://towardsdatascience.com/pros-and-cons-of-various-classification-ml-algorithms-3b5bfb3c87d6
[10] Norhidayu binti Abdul Hamid, Nilam Nur Binti Amir Sharif, (2017). Handwritten recognition using SVM, KNN, and Neural networks, https://arxiv.org/pdf/1702.00723.pdf

[11] Dutt, Anuj, Dutt, Aashi, (2017). Handwritten Digit Recognition Using Deep Learning, International Journal of Advanced Research in Computer Engineering & Technology (IJARCET).
[12] Dixit, Ritik & Kushwah, Rishika & Pashine, Samay. (2020). Handwritten Digit Recognition using Machine and Deep Learning Algorithms. International Journal of Computer Applications.

Glossary

Area under curve (AUC) The measure of the ability of a classifier to distinguish between classes.
Accuracy Ratio of correctly predicted observation to the total observations.
Activation Function A artificial neuron that delivers an output based on inputs.
Artificial Neural Network (ANN) Computational model that receive inputs and deliver outputs based on their predefined activation functions.
Backpropagation Used widely on feedforward neural networks, computes the gradient of the loss function concerning the weights of the network for a single input-output.
Cross Validation Procedure used to evaluate machine learning models on a limited data sample.
Confusion Matrix is a specific table layout that allows visualization of the performance of an algorithm.
Cross-Entropy Loss functions that measures the performance of a classification model whose output is a probability value between 0 and 1.
Classification Report Used to measure the quality of predictions from a classification algorithm.
Dropout Regularisation technique to reduce overfitting in neural networks by removing a percentage of trained data, preventing complex co-adaptations on the trained dataset.
Epochs The number of iterations to complete a process. Large datasets are usually grouped into batches.
Early Stopping Regularisation technique to avoid overfitting when training with an iterative method, such as gradient descent.
F Score(F1) The weighted average of Precision and Recall.
Feedforward Neural Network An artificial neural network (ANN) algorithm widely used for training feedforward neural networks, where information between the nodes travel forward from an input layer to hidden layers to output layers.
Grid Search The process of scanning the data to configure optimal parameters for a given model.
Hyperparameters Parameters where values are set to observe best performance models. In any machine learning algorithm, the parameters would be initialised before training a model.
Hyperplane Supervised learning tool that separates the dataspace into one less dimension for easier linear classification.
Kernel A SVM parameter, applied on an instance mapping the original non-linear observations to a higher-dimensional space.
K-fold The k-fold cross-validation procedure divides a limited dataset into k non-overlapping folds.
Loss Function Measures the average of the squares of the errors, also referred to mean squared error.
Learning Rate A tuning parameter to an optimisation algorithm that determines the step size at each iteration while moving toward a minimum of a loss function.
Normalisation A scaling technique in which numerical values are shifted and rescaled ranging between 0 and 1 also known as Min-Max scaling.
Overfitting Occurs when the neural network has so much information processing capacity but limited information contained in the training set, not enough to train all the neurons in the hidden layers.
Optimiser Algorithm used to change different parameters like weights or learning rate of neural network to reduce the loss.
Precision Ratio of correctly predicted positive observations to the total predicted positive observations.
Recall Ratio of correctly predicted positive observations to all observations in actual class.
Rectified Linear Activation Function (ReLU) A non-linear activation function used in multi-layer neural networks.
Random Search Set of optimal hyperparameter configurations by trying random combinations of hyperparameters.
ROC Curve ROC curve is a performance measurement for the classification problems at various threshold settings.
SoftMax An activation function in the output layer that predicts for multi-class classification problems i.e. more than two classification labels.
Specificity (FPR) Defines how many incorrect positive results occur among all negative samples available during the test.

Sensitivity (TPR) TPR defines how many correct positive results occur among all positive samples available during the test
Stochastic Gradient Descent (SGD) A gradient descent optimiser that uses an iterative method for optimizing an activation function for random selections of data samples.
Underfitting When model performs poor on training data and well on evaluation data.
Weight Decay Regularisation technique applied to the weights of a neural network by introducing a penalty to (and thus reducing) the loss function.

Implementation Details

The steps taken to process the data are:

- Torch Vision was used to download the dataset directly from the API.
- Images of the output labels were printed to do the initial investigation.
- Input/output labels were extracted from both train and test sets.
- Randint Numpy package was used to generate a sequence of random numbers, which was used to take a 10% sample of the original data to generate train_samp, trainlab_samp, test_samp, testlab_samp).
- X_train, Y_train, X_test, Y_test is represented by train_samp, trainlab_samp, test_samp, testlab_samp respectively.
- We then formatted these samples above to reshape from 3D to 2D images.
- Histogram to show the frequency/distribution across the 10 classes was created. This showed that the data is quite balanced.
- MLP and SVM were built on train dataset and tested on the test dataset (used to evaluate the final two models).
- The experiments were relatively efficient, therefore we selected CPU as a processor for training on both models. The data being numerical helped in this situation but we did allow it to use GPU if it needed.

MLP Model Implementation

- Pytorch was utilised to create the basic MLP model and Leaky ReLU as the activation function. Predictors were processed as an input to the model using default parameters for the basic MLP model training and the label class was taken as an output.
- Early stopping was applied to stop the training when no significant loss is further recognised.
- Dropout function was used to improve the training models and further observed.
- Random search function was used to obtain the parameters (epoch, learning rate, drop rate, hidden layers, optimiser type) that generate the best model and an improvement in the performance was observed. The performance was measured using performance analysis using metrics accuracy (used as scoring parameter), precision, recall, F1 Score and AUC/ROC. Classification report was used to display the results.

SVM Model Implementation

- Sklearn was utilised to create a basic SVM model with default parameters. The performance was generated.
- Random search function was used to obtain the parameters that generate the best model and an improvement in the performance was observed. Kernel, C, Gamma and degree parameters were used to train the best SVM model and accuracy was again used as a scoring parameter.