SEOUL BIKE DATA DEMAND

INTRODUCTION

Public bike-sharing systems have been gaining momentum only in the last decade. The main purpose other than convenience and easy-to-use service for customers is the mobility. More people are turning to healthier life styles and locations where bike riding can be easily available. There are many benefits in bike riding. Therefore, it is important to have rental bikes available to the customers (in our case, the public) to reduce their waiting time.

In this project, we implement a binary classification problem. It helps to give us the best model selected through experimentation and then evaluate the model for prediction, which is to predict the rental bike demand.

DATASET

This dataset has 8760 records comprising the details of every hour each day and also 14 columns. The data contains the hourly and daily count of rental bikes. It also contains the weather information (Rainfall, Snowfall, Temperature, Humidity, Visibility etc). Rented Bike Count is the current dependent variable when checked in the dataset.

We plan to implement Artificial Neural Network machine algorithm and see if our model generalises well.

DATA CLASSIFICATION PROBLEM

The dependent variable is Rented Bike Count which is a numerical variable. We will change our dependent variable to a binary classification problem by thresholding. Current threshold taken for our analysis is 75^{th} percentile. A good reason for creating binary predictors from numerical predictors is to overcome the problem of linearity. According to linear regression, we assume that X and Y to be in linear relation. If we can't find an appropriate equation to represent relation between X and Y variables , then creating binary predictors might be a way of obtaining some predictors

EDA has been conducted and we could see that positive correlation was seen for Temperature and Dew Point Temperature. Also, fields like Seasons showed Summer as the peak season for rented bike counts and also an increase in demand of bike count for higher temperature.

ALGORITHM

ARITIFICIAL NEURAL NETWORKS

Artificial Neural Networks or Neural Networks is an efficient computing system where the main concept is borrowed from the analogy of biological neural networks. Multiple input signals, referred as Input Layer Neurons, are transmitted to Hidden Layer Neurons, which on its turn are used to predict the output, Output Layer. In the sense, input signals are similar to the human senses, such as human sight, hearing, smelling, tasting and touching, only in the case of ANN, those input signals can be various type of features characterizing an observation

ANN acquires a large collection of units that are interconnected in some pattern to allow communication between the units. These units, also referred to as nodes or neurons, are simple processors which operate in parallel. Every neuron is connected with other neuron through a connection link. Each connection link is associated with a weight that has information about the input signal. This is the most useful information for neurons to solve a particular problem because the weight usually excites or inhibits the signal that is being communicated. Each neuron has an internal state, which is called an activation signal. Output signals, which are produced after combining the input signals and activation rule, may be sent to other units.

Data Normalization

Artificial Neural Networks are sensitive to the scale of the data. Therefore, we will use the Standard Scalar from the Scikit learn library to scale the data.Given that the dependent variable that can only take values 0 and 1, we only need to scale the features data, the set of all independent variables.

Data Splitting

In order to make sure we first train the model using only a part of the data and then use the trained model to predict the rented bike count whose data have not been used during the training.

Activation Function

We use a function to join signals from different input neurons into one value. Each synapse get assigned a weight, an importance value. These weights form the corner stone of how

Neural Networks learn. These weights determine whether the signals get passed along or not, or to what extent each signal gets passed along. If we define the input value of signal i by xi and its importance weight by wi then the sum of these signals, can be defined by

$$\phi\left(\sum_{i=1}^{m} w_i x_i\right)$$

where φ represents the activation function. This function is called Activation Function. We have used sigmoid, tanh and relu activation functions for our analysis.

For our classification problem, we created a model algorithm with the below layers and size and we got a good accuracy for both the train and test accuracy.

ANN Model	Train Accuracy	Test Accuracy
First Hidden Layer - 10 units, relu activation function Second Hidden Layer - 10 units, relu activation function Output Layer - Sigmoid Batch Size - 10 Epocks - 10	90.38%	90.41%

This model has also been fitted with batch size of 20 with epochs of 100 with a validation set split of 20%. The accuracy is on the training data, and val_accuracy is on the validation data. It's best to rely on val_accuracy for a fair representation of model performance because a good neural network will end up fitting the training data at 100%, but would perform poorly on unseen data.

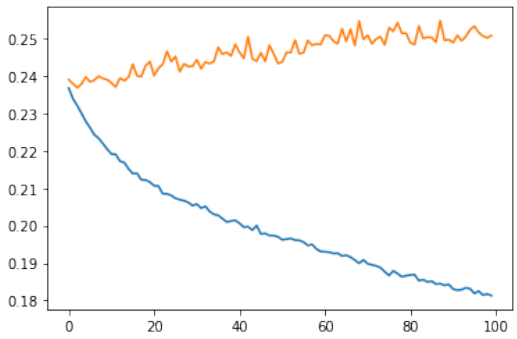


Fig 1.1

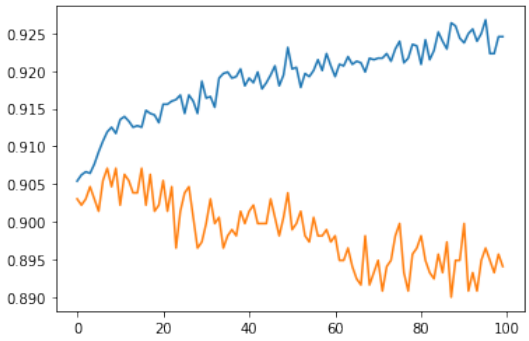
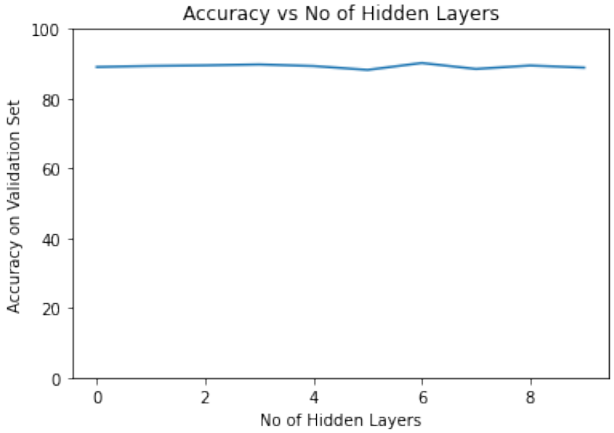


Fig 1.2

In Fig 1.1, the loss and the val_loss is plotted for the model algorithm. It's best to rely on the val_loss to prevent overfitting. Overfitting is when the model fits the training data too closely, and the loss keeps decreasing while the val_loss is stale, or increases. Fig 1.1 shows an overfitted view. In Fig 1.2, accuracy and the val_accuracy are plotted. It's best to rely on val_acc for a fair representation of model performance because a good neural network will end up fitting the training data at 100%, but would perform poorly on unseen data. The graph shows a good val_accuracy score.

EXPERIMENTATIONS

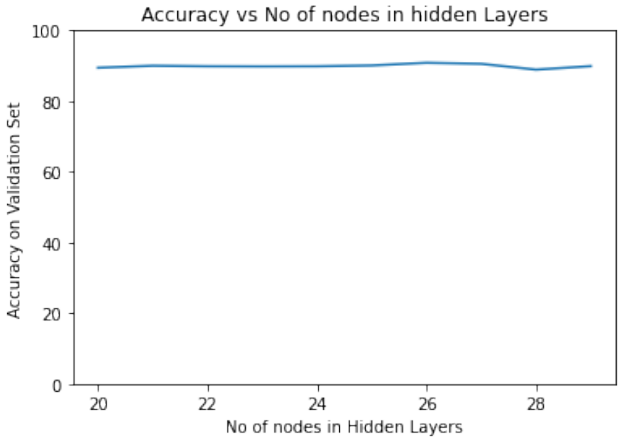
1.Accuracy Vs No Of Hidden Layers



We can see from Fig 2.1 that the no of hidden layers is not making any good progress in the accuracy of the model.So, we will keep 1 Hidden Layer in our Neural Network.

Fig 2.1 Accuracy Vs No Of Layers

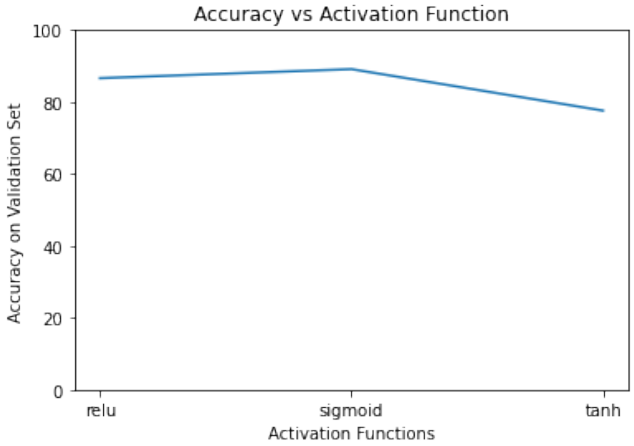
2. Accuracy Vs No Of Nodes in Hidden Layer



We can see from Fig 2.2, the accuracy with respect to increase in nodes for a hidden layer does not show much difference. So , 10 is the no of nodes taken in this experiment as an ideal case

Fig 2.2 Accuracy Vs No of Nodes in Hidden Layer

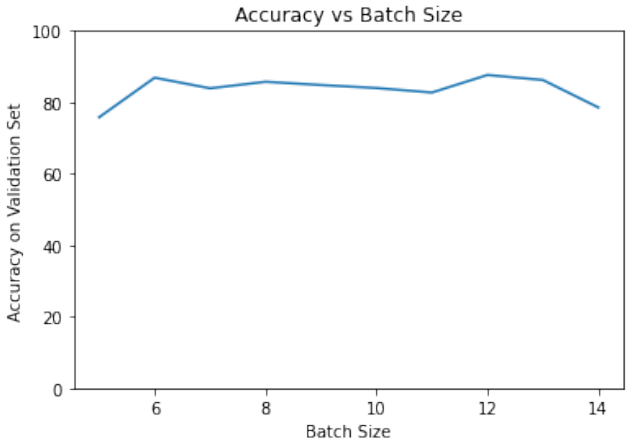
3. Accuracy Vs Activation Function



In Fig 2.3, the accuracy is shown highest for sigmoid function.

Fig 2.3 Accuracy Vs Activation Function

4. Accuracy Vs Batch Size



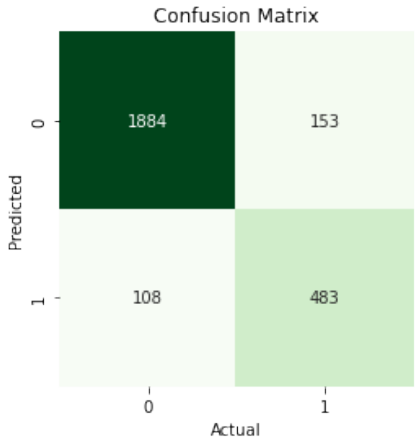
In Fig 2.4, optimum batch size chosen is 8

MODEL TUNING

Neural Network have a lot of parameters to find the optimum parameters with the best scores. We have used cost function optimizers like SGD (Stochastic Gradient Decent). It is an iterative approach for solving optimization problems with a different function. It aims to find the extreme or zero points of the stochastic model containing parameters that cannot be directly estimated. Post this, we will use Sigmoid activation function because our problem is a binary

classification rented bike count problem. Loss function used is cross entropy which gives the negative average of the log of corrected predicted probabilities. GridSearchCV from SkLearn package has been used for the analysis.

Case 1: Batch Size and No Of Epochs



Batch Size: [10,20,50] , No Of Epochs :[10,50,100]

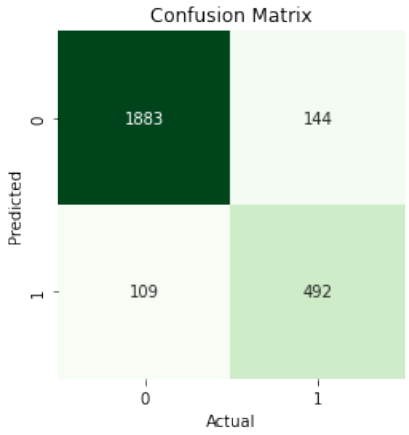
10 nodes have been used for the first hidden layer with sigmoid function as the activation function.

Best parameters: **Batch Size: 10, Epochs: 100**

Best Cross Validation Accuracy: **0.89**

```
Accuracy: 0.90
Sensitivity: 0.76
Specificity: 0.95
```

Case 2: Activation Function Of First Hidden Layer



Activation Function: Linear, relu, sigmoid, tanh

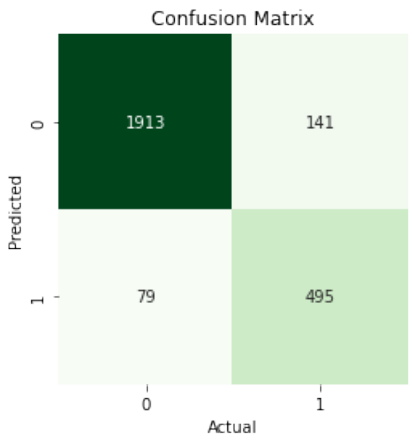
10 nodes have been used for the first hidden layer with sigmoid function as the activation function.

Best Parameters: **ReLU**

Best Cross Validation Accuracy: **0.90**

```
Accuracy: 0.90
Sensitivity: 0.77
Specificity: 0.95
```

Case 3: No Of Neurons in First Hidden Layer



	param_neurons	mean_train_score	mean_test_score
0	1	0.862076	0.859590
1	5	0.896119	0.885355
2	10	0.910225	0.898727
3	15	0.915688	0.901010
4	20	0.923394	0.905902
5	25	0.925024	0.903946
6	30	0.927266	0.909165

```
Accuracy: 0.92
Sensitivity: 0.78
Specificity: 0.96
```

No Of Neurons: 1, 5, 10, 15, 20, 25, 30

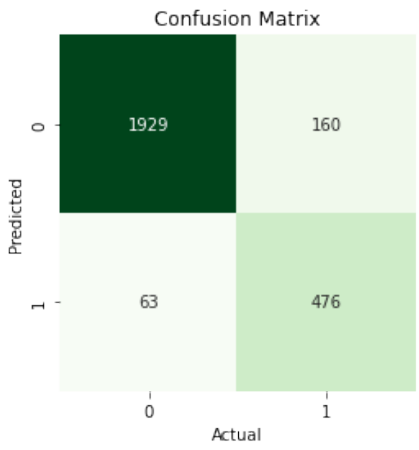
Learning Rate = 0.01

The first hidden layer has ReLU activation function and sigmoid as the output later.

Best Parameters: **25** neurons

Best Cross Validation Accuracy: **0.91**

Case 4: No Of Neurons in Second Hidden Layer



	param_neurons	mean_train_score	mean_test_score
0	1	0.909369	0.896771
1	5	0.911529	0.899054
2	10	0.912467	0.900685
3	15	0.913976	0.900032
4	20	0.912998	0.897422
5	25	0.917442	0.906717
6	30	0.916259	0.903945

No Of Neurons: 1, 5, 10, 15, 20, 25, 30

Learning Rate = 0.01

The first hidden layer has ReLU activation function with 15 nodes and sigmoid as the output.

Best Parameters: **15** neurons

Best Cross Validation Accuracy: **0.91**

MODEL EVALUATION

We have also trained and evaluated models of all combinations of parameters specified in below Fig 3.1. The training accuracy and error (loss) for every model is noted. The same has been done for test data.

Hidden Layers	Epoch	Learning Rate	Activation Function	Training Accuracy	Training Error	Test Accuracy	Test Error
2	100	0.001	Sigmoid	**0.9137**	0.0684	**0.9091**	0.0684
			TanH	0.9114	0.0694	0.9068	0.0711
			ReLU	0.9103	0.0708	0.906	0.0719
3	100	0.001	Sigmoid	0.9132	0.0683	**0.9113**	0.0663
			TanH	0.9253	0.0566	0.9201	0.0616
			ReLU	**0.9163**	0.062	0.9113	0.0654
2	100	0.01	Sigmoid	**0.94**	0.0491	0.9102	0.0667
			TanH	0.9204	0.0742	**0.9136**	0.0788
			ReLU	0.7466	0.2534	0.758	0.242
3	100	0.01	Sigmoid	0.9481	0.0423	0.9216	0.0592
			TanH	**0.9534**	0.0406	0.9151	0.0644
			ReLU	0.9416	0.05	**0.933**	0.0562
2	200	0.001	Sigmoid	0.9224	0.062	0.9106	0.0673
			TanH	0.9145	0.0641	0.9136	0.0662
			ReLU	**0.9287**	0.056	**0.9224**	0.0591
3	200	0.001	Sigmoid	0.938	0.0498	**0.9254**	0.0542
			TanH	0.9359	0.0502	0.9254	0.0582
			ReLU	**0.9418**	0.0464	0.9239	0.0559
2	200	0.01	Sigmoid	**0.9488**	0.0443	0.9186	0.0645
			TanH	0.9274	0.0609	0.9203	0.0647
			ReLU	0.9305	0.0579	**0.9231**	0.0625
3	200	0.01	Sigmoid	**0.9631**	0.0313	0.9243	0.0603
			TanH	0.9543	0.0383	0.9273	0.0612
			ReLU	0.9444	0.0465	**0.9334**	0.0558

Fig 3.1 Training and Testing Accuracy For Different Models

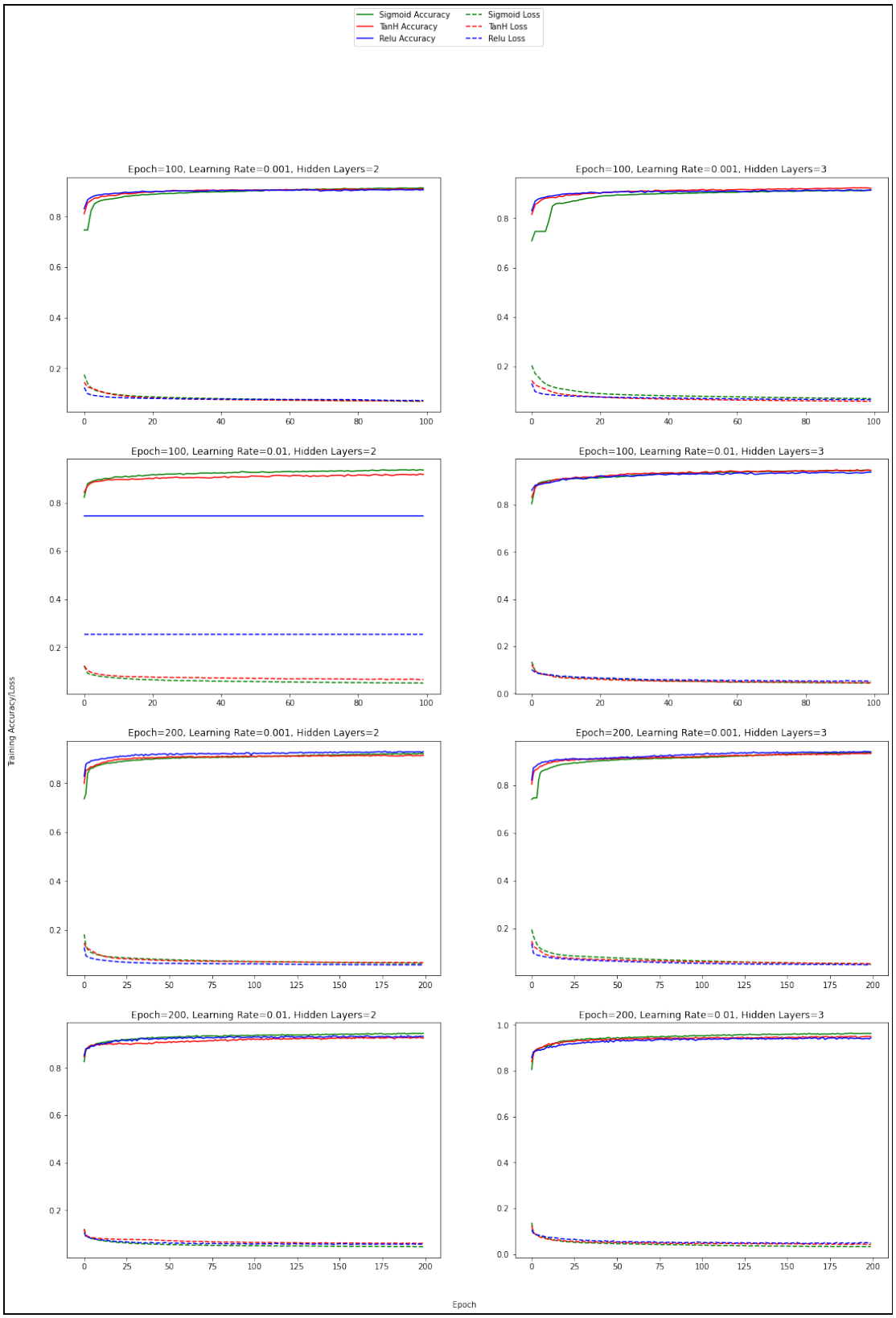


Fig 3.2 Plot of Accuracy Vs Training Steps

COMPARISON OF CLASSIFIERS

From below Fig 4.1, we can see that the some models have given a good predictability which shows the efficiency of the model. For those with several results, we would also need to check computationally which model is the fastest one.

ANN has performed well for this dataset in comparison to other models.

Model	Best Parameters	Train Accuracy	Test Accuracy
Linear Regression	Selected 8 suited features for the analysis	0.9108	0.9201
SVM (Linear)	C = 1	0.8657	0.8668
SVM(Polynomial)	C = 100	0.9022	0.9212
SVM (rbf)	C = 100, gamma = 0.1	0.9075	0.9243
SVM (sigmoid)	C = 1	0.7508	0.7671
Decision Tree - GINI	max_depth = 6, max_leaf_nodes:10, min_samples_split:2	0.9223	0.914
Decision Tree - Entropy	max_depth = 7, max_leaf_nodes:10, min_samples_split:2	0.878	0.8889
Decision Tree - Post Pruning	max_depth = 7,max_leaf_nodes:10,min_samples_split:2, ccp_alpha = 0.002	0.9274	0.9334
ANN	2 hidden layers with 10 nodes ,ReLU function Output Layer: Sigmoid Batch Size- 9, Epoch - 20	0.9185	0.9178
ANN	2 hidden layers, ReLU function 200 epochs Learning Rate - 0.001	0.9287	0.9224
ANN	3 hidden layers, Sigmoid function 200 epochs Learning Rate - 0.001	0.938	0.9254
ANN	3 hidden layers, ReLU function 200 epochs Learning Rate - 0.01	0.9444	**0.9334**

Fig 4.1 Comparison of Different Models

CONCLUSION

For the binary classification problem, model has been generalised well based on the algorithms that has been introduced, ANN.We initialized the ANN by calling the *Sequential* class from the *Keras* library in Python. Keras comes in combination with Tensorflow library. Then. We have added the fully connected Input Layer to the Sequential ANN by calling Dense class from Keras. We also fully connected Output Layer in the same way.For the first layer, Rectifier Activation function is used followed by Sigmoid Function in the output layer to find the probability.

For our classification problem, initially based on our analysis and random experiments, test accuracy of **90.5%** was obtained. The very set was modelled agained with validation_split as 20% with epochs as 100 and batch_size 20. We could see, from the graphs, that *val_accuracy* showed a good score but *val_loss* showed signs of overfitting. Different experiments for the above dataset was then conducted.

Accuracy with respect to different units like hidden layers, nodes in hidden layers, activation function and batch size has been noted. According to the analysis, 10 nodes, sigmoid function and batch size of 9 topped the results. We also conducted an analysis with respect to 10 nodes for 2 hidden layers with ReLU activation function with a batch size of 9 and epochs of 20. *Adam Optimizer* is the optimizer used for the compilation. It is an extended version of SGD with momentum method.This algorithm helps define different learning rates for different parameters.It calculates the individual learning rates for each parameter based on the estimates of the first two moments of the gradient.So each unique learning rate is being updated. This algorithm also is a way to minimise the loss function and update the weights. The test accuracy for this model gave **91.78%** which is a better experiment than the previous random experiment conducted.

Cross Validation accuracy has been validated in different algorithms and can be used as a performance metric to compare the efficiency of different models. The CV used here is *k-Fold* which generally produces less biased models as every point from the original dataset will appead in both training and testing set. This method is optimal if limited data is used for the analysis.

We could see that on cross validation, we will able to get the optimum hyperparameters for ANN. They are usually fixed before the actual training process begins. These parameters express important properties of the model such as its complexity or how fast it should learn. Therefore, hyperparameter tuning helps to improve the performance of the models. In our analysis, we could see that batch size of 10 and epoch of 100 gave a cross validation accuracy of 0.89.The best activation function for the first hidden layer is ReLU activation function with a cross validation accuracy of 0.90.No of neurons was suggested best at 25 and no of neurons in second hidden layer showed an accuracy as 0.91 which showed that adding another layer might not increase efficiency. Then, the history curve was plotted for all models performed and we could see good results in few models.

On comparing all models, ***ANN*** performance is better than all the other algorithms. Several factors made the ANN algorithm better than the rest. One of them is how ANN works in parallel which reduces the processing time with respect to other algorthms that provide the same results. This algorithm also has the ability to handle fuzzy data unlike other algorithms Moreover, the confusion metrics along with the ROC curve is a great advantage on validating ANN as the best model.

Additionally, best parameter for ***learning rate*** hyperparameter and ***momentum*** for the hiddel layers can be validated for.Also, best parameter for ***drop-out rate*** and ***weight constraint*** can be validated. However, computationally, this is very expensive and should have a good processor. Based on this analysis, we can select learning rate such that it does not diverge from the global minimum in SGD.Therefore, it is always good to select low learning rates to provide better classification metrics. Accuracy can be improved by using techniques like ***bagging*** and ***boosting*** technique. Different boosting techniques like ***XGBoost*** can be implemented to see if the performance can be increased.Another aspect that can be done is conduct the evaluation of the model through methods like ***Log Loss*** and ***Gain and Lift Charts***. Log Loss is very effective to see how likely the model thinks the observed set of outcomes was.The Gain and Lift chart evaluate the model on portion sof the whole population. Also, performance of the model can also be monitored in addition to other criterias. If there is scope to increase the sample size in training the model, it can increase the model's prediction efficiency.