Dataset 1 - Power Consumption of a house : After dropping lights and date variables, 26 independent variables such as pressure of house, temperature of house, wind speed outside, visibility outside etc.… are used in determining the power consumption of a house which is given in the variable Appliances. Scaling was done to ensure that the columns having higher numeric values not to dominate our model prediction. Minmax scalar brings the values in all columns in between 0 and 1.

Dataset 2 : Counter strike game

The 2-player online shooting game – Counter strike between cops and terrorists team has mission such as bomb defusal, hostage rescue and VIP assassination missions. The parameters such as Game Map, Day, Month, Year, Date, Wait Time(s), Match Time(s), Team A , Rounds, Team B, Rounds, Ping, Kills, Assists, Deaths, Mvp's, HS%, Points are the independent variables and the result of the game is given as output variable – win, lose or tie.

Tournaments involve 1,000,000$ prize pool and the players can view my model to improve their gaming and having a great likelihood of winning the game.

(i) Neural Networks: The neural network which resembles the human brain comprises of 3 layers – input layer, hidden, output layer; activation for each node and connections are termed as edges. I have used Multi-layer Perceptron classifier(MLP) algorithm in python to implement neural network.

Experiment 1: Hidden layer = 1 ; **Experiment 2:** Hidden layers = 2

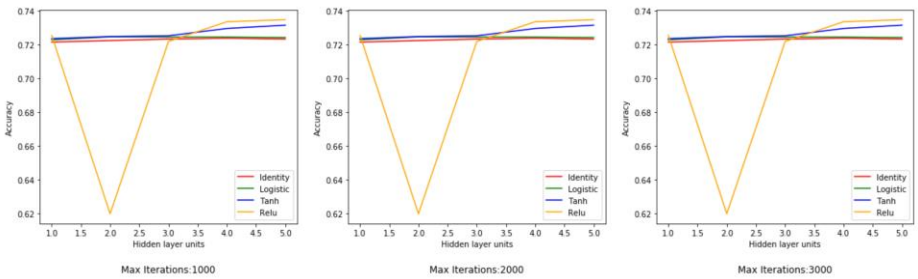
Common for both experiments & both datasets : Performing the experimentation, yields the following accuracy. The graph has been plotted for 3 types of iterations – 1000,2000,3000. The different color line represents different activations and the legends of the same are mentioned.

The parameters mentioned below are common for both the experiments.

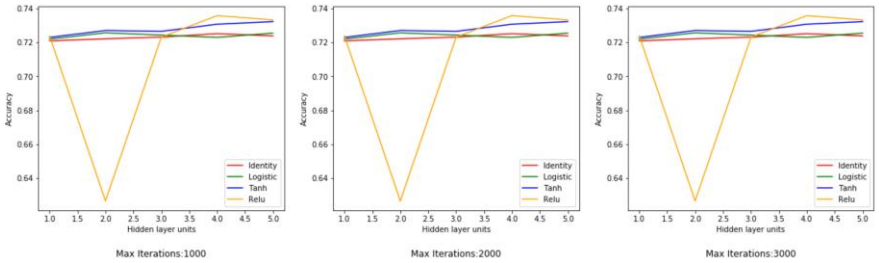
- Number of units in each hidden layer: 5 different types of units are used to compute the same - 1,2,3,4,5.
- Activation: There are 4 activation functions used - *'identity', 'logistic', 'tanh', 'relu'.*
- Max iterations : Iterations such as 1000,2000,3000 which are steps, are used and the accuracy are tested for the same.

Power Consumption of a house: Experiment 1: It is seen that the 3 types of iterations is almost same.

Training Data: relu activation function with 5 hidden units and 1 hidden layer has been performing better when compared to other activation functions. The order of the performance after relu are tanh, logistic and identity for 5 hidden units. Since there is no improvement with iterations, we can stop with 1000 iterations in this case.

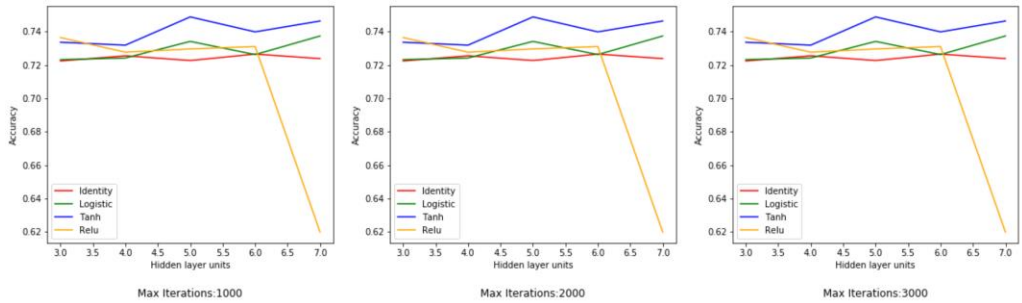


Validation Data: relu activation function with 4 hidden units and 1 hidden layer has been performing better when compared to other activations. The order of the performance after relu are tanh, identity and logistic for 5 hidden units. In training dataset 5 hidden units were best but here in validation it is 4 hidden units. We can perform with 1000 iterations, relu and 4 hidden units to build our model. With an increase in hidden units, accuracy keeps improving except for relu which has seen a downfall when the hidden unit is 2.

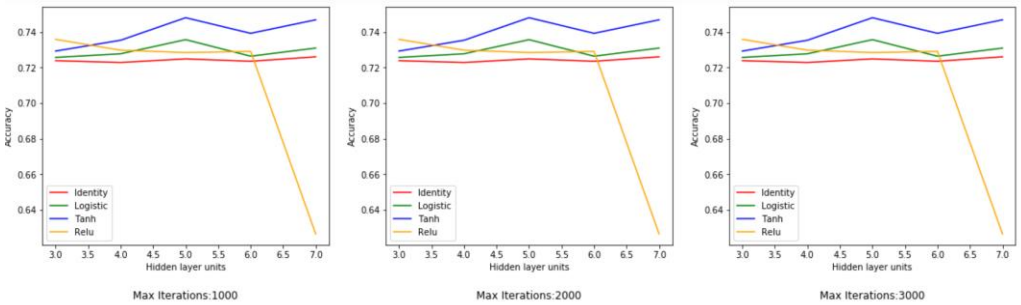


<u>Power Consumption of a house: Experiment 2:</u> It is seen that the 3 types of iterations is almost same.

Training Data: tanh activation function with 5,3 hidden units and 2 hidden layer has been performing better when compared to other activation functions. The order of the performance followed by tanh are logistic, relu and identity for 5 hidden units. Since there is no improvement with iterations, we can stop with 1000 iterations.



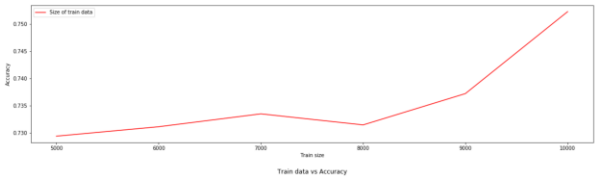
Validation Data: tanh activation function with 5,3 hidden units and 2 hidden layer has been performing better when compared to other activation functions. The order of the performance followed by tanh are logistic, relu and identity for 5 hidden units. Since there is no improvement with iterations, we can stop with 1000 iterations.



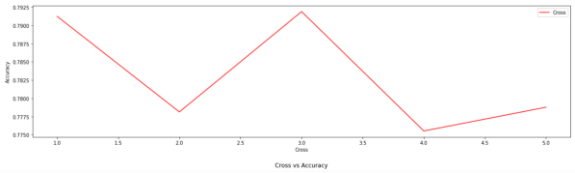
We can proceed with 2 hidden layers, 5 hidden units, 1000 iterations and tanh activation to obtain the best accuracy as it has been performing better than our previous experiment which had 1 hidden layer.

Experiment 3: Increase in training data size is causing an increase in accuracy except in 8000 rows which has a small decrease in accuracy when compared to its preceding 7000 rows.

Experiment 4: Cross validation: When performing cross validation with set on x-axis and accuracy on y-axis, the 1st and 3rd set are having the highest accuracy almost equal.



Experiment 3



Experiment 4

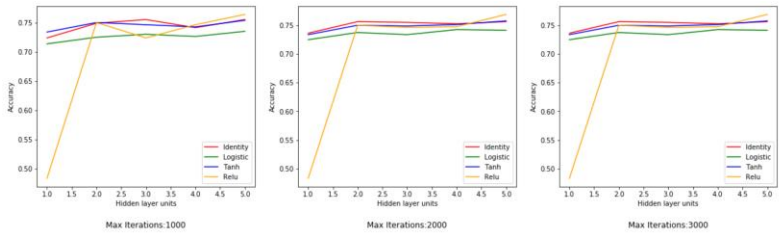
Confusion matrix for the best set of parameters:

	Predicted 0	Predicted 1
Actual 0	1376 (TN)	835 (FP)
Actual 1	657(FN)	3053 (TP)

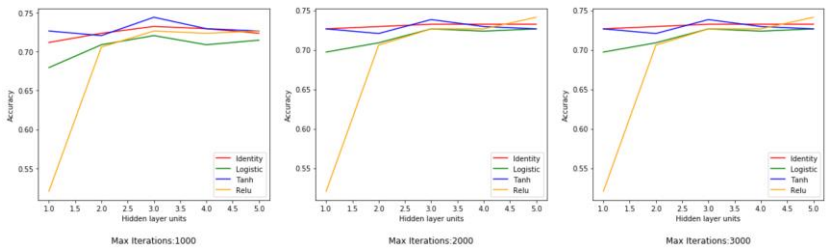
Parameters	Formula	Computation
Accuracy	(TP + TN) / (TP+TN+FP+FN)	0.7480
Sensitivity	TP/(TP+FN)	0.8229
Specificity	TN/(TN + FP)	0.6223
Precision	TP / (TP+FP)	0.7852

Counter Strike: Experiment 1:

Training Data: relu activation function with 5 hidden units and 1 hidden layer has been performing better when compared to other activation functions. The order of the performance after relu are tanh, identity and logistic for 5 hidden units. Since there is no improvement with iterations, we can do 1000.

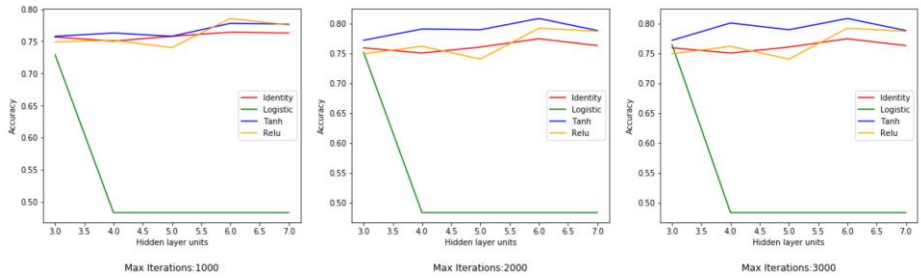


Validation Data: tanh activation function with 3 hidden units and 1 hidden layer has been performing better when compared to other activation functions. The order of the performance after identity, relu and logistic for 5 hidden units. We can go with 1000 iterations.

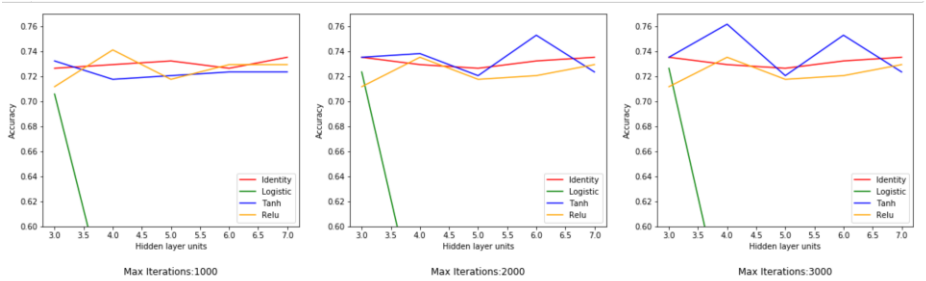


Counter Strike: Experiment 2:

Training Data: relu activation function with 6,3 hidden units and 2 hidden layer has been performing better when compared to other activation functions. We can stop with 2000 iterations.



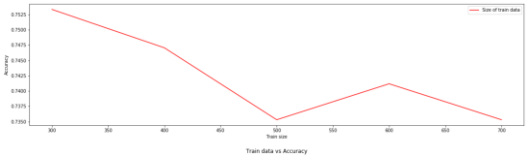
Validation Data: tanh activation function with 4,3 hidden units and 2 hidden layers has been performing better when compared to other activation functions. We can stop with 3000 iterations.



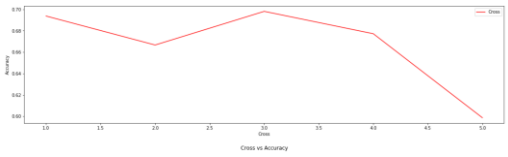
We get better results when there are 2 hidden layers with (4,3) units, 3000 iterations and tanh activation.

Experiment 3: Increase in training data size is causing a decrease in accuracy in this dataset.

Experiment 4: Cross validation: When performing cross validation with set on x-axis and accuracy on y-axis, the 1st and 3rd set are having the highest accuracy almost equal.



Experiment 3



Experiment 4

Confusion matrix for the best set of parameters:

	Predicted lose (0)	Predicted win (1)	Predicted Tie (2)
Actual Lose (0)	129 (TN)	36 (FP)	12
Actual Win (1)	20 (FN)	113 (TP)	6
Actual Tie (2)	7	0	17

Considering only win and lose, ignoring tie:-

Parameters	Formula	Computation
Accuracy	(TP + TN) / (TP+TN+FP+FN)	0.7118
Sensitivity	TP/(TP+FN)	0.8496
Specificity	TN/(TN + FP)	0.7818
Precision	TP / (TP+FP)	0.7584

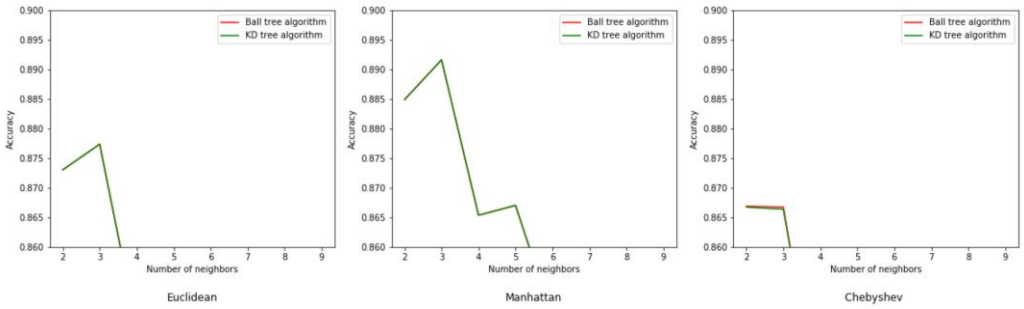
(ii) K-Nearest Neighbor: KNN, a supervised learning algorithm for solving classification problem in sorting the output variables in both the datasets.

Experiment set

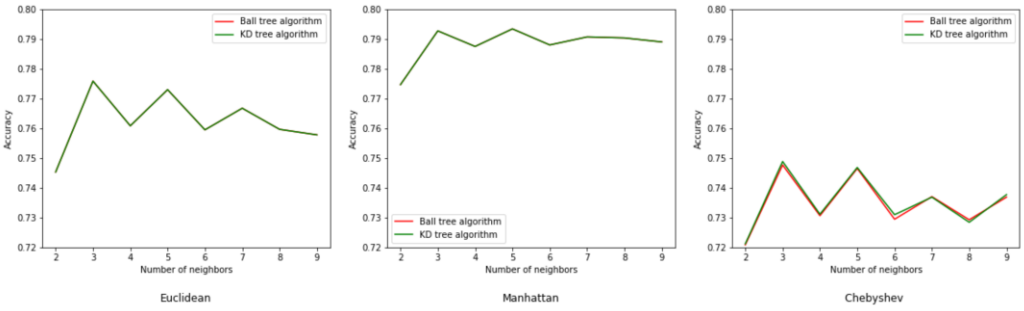
- Metrics: Distance metric used in the algorithm - 'Euclidean', 'Manhattan' and 'Chebyshev'
- Algorithms used : 'ball_tree' and 'kd_tree'
- Neighbors: [2,3,4,5,6,7,8,9]

Power Consumption of a house: Experiment 1

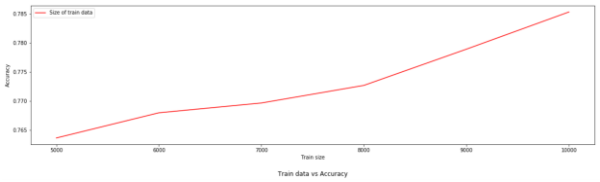
Training Data: Performing the experimentation, yields the following accuracy. The maximum accuracy is achieved when there are 3 nearest neighbors, Manhattan distance. Ball tree and KD tree algorithms are not making any difference and so we can use any of them to achieve our classification goal.

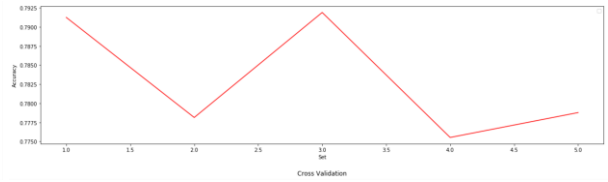


Validation Data: Performing the same for the validation data, we have the maximum accuracy for Manhattan distance and neighbor 5/neighbor 3. Ball tree and KD tree algorithms are not making any difference and so we can use any of them to achieve our classification goal.



Power Consumption of a house: Experiment 2: It is seen that the accuracy is increasing with the increase in training data size.





Experiment 2 Experiment 3

Power Consumption of a house: Experiment 3: The cross validation is giving us best results for the 1st and 3rd set as seen in the diagram above.

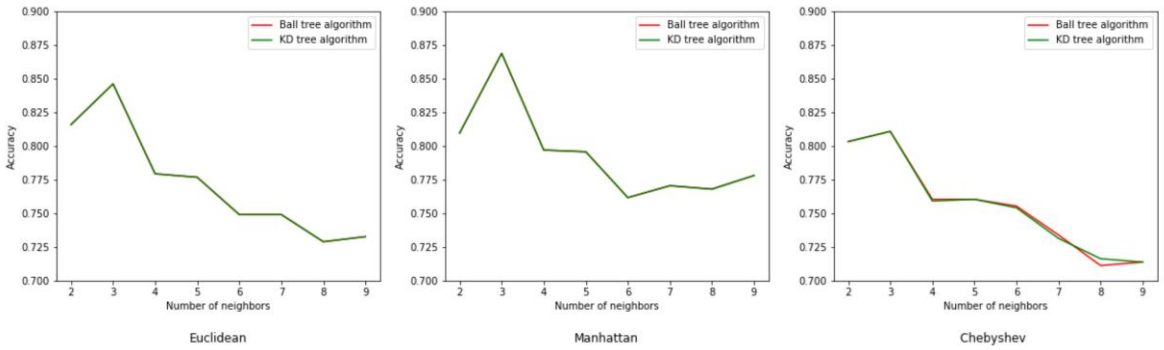
Confusion matrix for the best set of parameters:

	Predicted 0	Predicted 1
Actual 0	1661 (TN)	550 (FP)
Actual 1	677(FN)	3033 (TP)

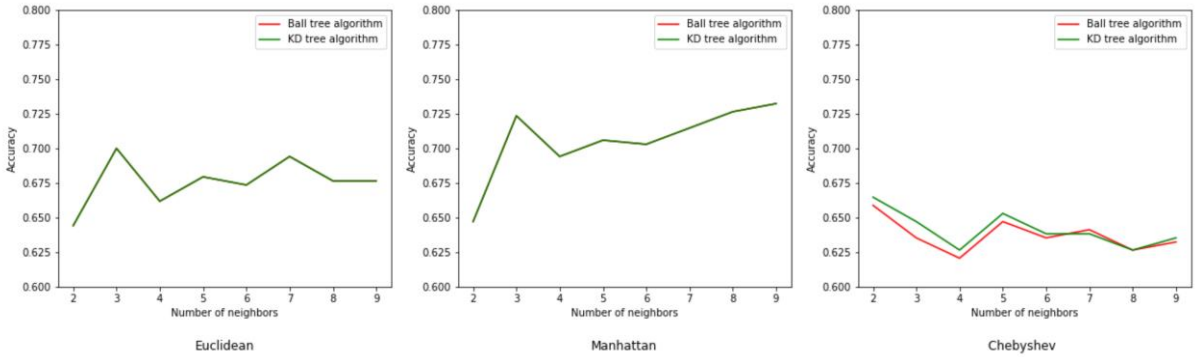
Parameters	Formula	Computation
Accuracy	(TP + TN) / (TP+TN+FP+FN)	0.7928
Sensitivity	TP/(TP+FN)	0.8175
Specificity	TN/(TN + FP)	0.7512
Precision	TP / (TP+FP)	0.8465

Counter Strike : Experiment 1

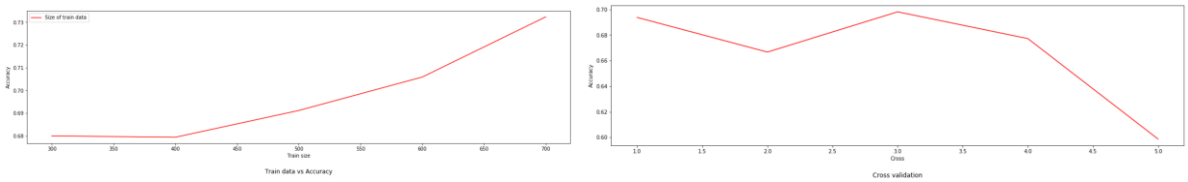
Training Data: Performing the experimentation, yields the following accuracy. The maximum accuracy is achieved when there are 3 nearest neighbors, Manhattan distance. Ball tree and KD tree algorithms are not making any difference and so we can use any of them to achieve our classification goal.



Validation Data: Performing the same for the validation data, we have the maximum accuracy for Manhattan distance and neighbor 3. Ball tree and KD tree algorithms are not making any difference and so we can use any of them to achieve our classification goal.



Counter strike: Experiment 2: It is seen that the accuracy is increasing with the increase in training data size.



Experiment 2 Experiment 3

Counter strike: Experiment 3: The cross validation is giving us best results for the 1st and 3rd set as seen in the diagram above.

Confusion matrix for the best set of parameters:

	Predicted lose (0)	Predicted win (1)	Predicted Tie (2)
Actual Lose (0)	134 (TN)	37 (FP)	6
Actual Win (1)	40 (FN)	97 (TP)	2
Actual Tie (2)	6	3	15

Considering only win and lose, ignoring tie:-

Parameters	Formula	Computation
Accuracy	(TP + TN) / (TP+TN+FP+FN)	0.6794
Sensitivity	TP/(TP+FN)	0.7080
Specificity	TN/(TN + FP)	0.7836
Precision	TP / (TP+FP)	0.7239

Discussion:

Comparison between 3 algorithms : Power Consumption dataset

Parameters	SVM	Decision Tree	Boosting	Neural net	KNN

Accuracy	0.7226	0.6985	0.7162	0.7480	0.7928
Sensitivity / True positive rate	0.8416	0.9023	0.9669	0.8229	0.8175
Specificity	0.4339	0.2042	0.1078	0.6223	0.7512
Precision	0.7829	0.7334	0.7245	0.7852	0.8465

Comparison between 3 algorithms : Counter Strike dataset

Parameters	SVM	Decision Tree	Boosting	Neural net	KNN
Accuracy	0.7692	0.7865	0.7586	0.7118	0.6794
Sensitivity / True positive rate	0.7914	0.6503	0.5889	0.8496	0.7080
Specificity	0.8342	0.9016	0.8756	0.7818	0.7836
Precision	0.8012	0.848	0.8	0.7584	0.7239

Accuracy, Sensitivity, Specificity and Precision is tabulated above for all 5 classification algorithms. When observed which of the parameters are performing better for each of these models it is seen that the KNN classification performing better than all other 4 models for classifying power consumption dataset.

But when we compare the same for the counter strike dataset it has been seen that the decision tree performs better when compared to all other models and so neural networks and KNN has not made significant process in computing accuracy for this dataset.

If ranked by accuracy, for Power consumption the ranking from 1 to 5 goes as follows – KNN, Neural net, SVM, Boosting and decision tree. For counter strike data it goes as – Decision tree, SVM, Boosting, Neural nets and KNN.

More the data better will be the accuracy of the data and so as we collect more of the data and train in these different models we are likely to find a good increase in our results accuracy. Also, more experiments could be performed to improve the accuracy of the model, reduce errors and avoid overfitting.

The K-fold cross validation helps in finding out how our model performs in training dataset in terms of accuracy and it has been helpful in all 5 algorithms to figure out the same.

For neural networks more the hidden layers, better is the performance of the model as we saw the improvement of the model with 2 layers over 1 layer and I picked that. Activation function is another sensitive thing in improving the model and so it was chosen. If the model has no improvement over iterations, we can stop the process early and so iterations were chosen and tested.

In case of KNN, distance measurement determines to which class, our dataset belongs and so choosing the correct distance measurement is the important thing and so I chose 4 common distance measures to build the model. Want to choose the best algorithm to work on, but ball tree and kd tree algorithms didn't make much difference. Choosing the correct neighbors are important as the maximum number of points in a neighbor circle determines the class of the dataset.

Overall I understood that just because an algorithm works better on a dataset it doesn't mean that it works best in all the datasets and repeated experiments are required to figure out the best model to solve our prediction problem.