

Project Report

Executive Summary

This project aims at the experimentation of 2 important machine learning algorithms – Artificial Neural Networks (ANN) and K-Nearest Neighbours (KNN). Though both the algorithms can be used for Regression and Classification, various control parameters are experimented from a Classification perspective. Some of the control parameters that are studied are: *type of activation functions, No. of hidden layers and no. of nodes in each hidden layer, training size, type of algorithm for KNN, various distance metrics for KNN* etc. The results and plots are captured for plausible interpretations. The final summary aggregates the results and presents the overview of the project. The entire project is implemented using 'Python 3' (Jupyter Notebook).

Datasets

Dataset 1 – Energy Data

The Energy dataset, from Kaggle, is considered as the First dataset. The dataset has 29 columns and 19735 rows with 1 target variable – 'Appliances' and 28 dependent variables. Correlation matrix is plotted and visualized using a heatmap for all the independent variables. The features – 'date', 'rv1', and 'rv2' are dropped. The target variable is continuous in nature. It is converted to a binary classification problem using a median of 60 for implementing the above listed algorithms. The problem statement is to predict if the energy usage is above 60 units or not.

Dataset 2 – Audit data

The second dataset I have chosen is an Audit dataset from Kaggle. The Audit risk dataset consists of data of various firms and their risk factors. They belong to a multitude of sectors ranging from Irrigation, Public Health, Animal Husbandry to Fisheries, Tourism, Science and Technology. This dataset is developed by a 3rd party Audit company that wishes to calculate and assess 'Risk' by analysing the present and historical risk factors thereby facilitating the audit-planning process. The dataset has over 18 columns, with one 'target' variable – 'Risk' (binary).

The primary reason I found this dataset interesting is because I have a strong interest in fraud detection. Initial EDA is performed and columns 'TOTAL', 'Score' and 'LOCATION_ID' are dropped because of redundancy in information and 'Money_Value' feature is imputed with the mean value as it has an unusual number of zeroes. The problem statement is to predict if a firm is under 'Risk' or not.

Tasks – Task 1 – Artificial Neural Networks (ANN)

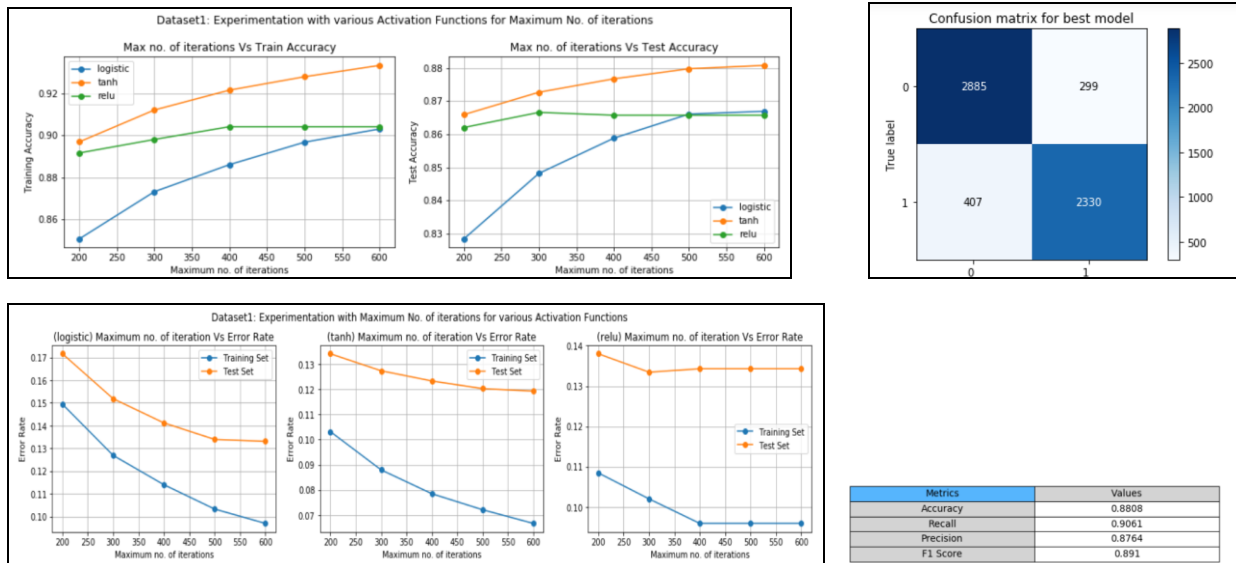
This algorithm is implemented using 'MLPClassifier' package from 'scikit-learn' library. I have performed 3 experiments on ANN. I have considered 3 activation functions – 'tanh', 'relu' and 'logistic'. Even, the most complicated data distribution can yield good results for the above functions. The best value of the hyperparameters such as 'No. of hidden layers and no. of nodes' can be obtained only by a 'GridSearch' and the below experiments are means to achieve the best values for the various control parameters.

Experiment 1.1 – Experimentation with 'activation functions' for various values of 'Max iterations'

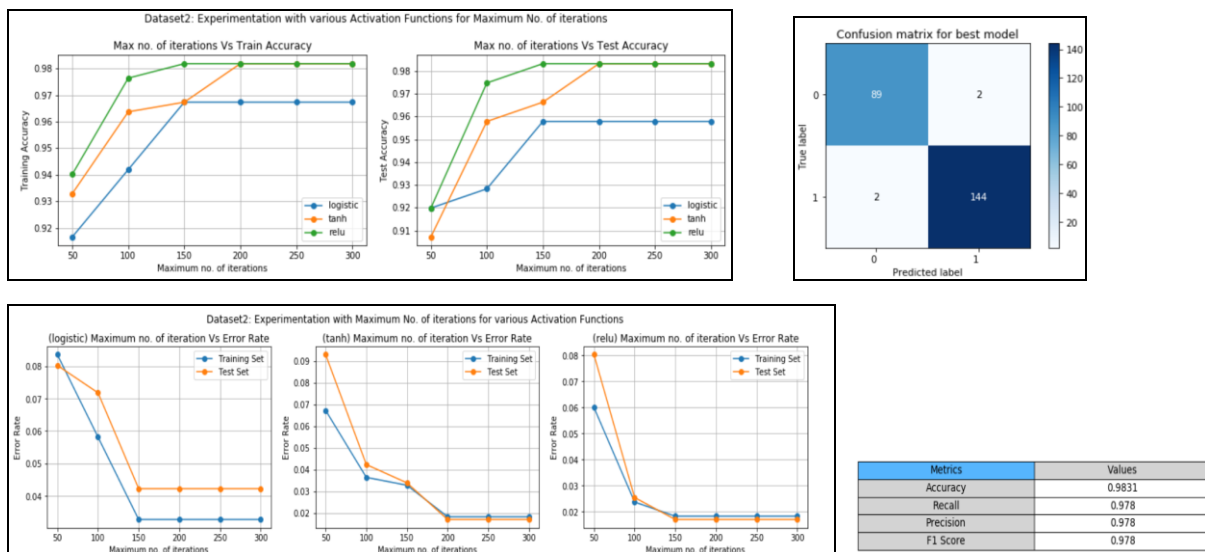
'Max iterations' parameter determines the maximum number of epochs an ANN model has to run in order to achieve convergence. The model that has achieved convergence will produce the best results on the test set. Default value of 'max iterations' is 200. I have experimented in the range of 200-600

with 3 activation functions as already mentioned. The best activation function and 'max iterations' value is reported for each dataset at the conclusion of the experiment.

Dataset 1 – From the below plots, it can be inferred that 'tanh' activation function produces the maximum test accuracy for 'maximum no. of iterations' = 600. We can expect the test accuracy to increase with increase in the control parameter, but the test accuracy almost saturates for values above 500. Though convergence isn't met we can expect the model to produce a test error rate in the range of 11.92%. The classification metrics are reported for the best model ('tanh' function, 600 max iterations)



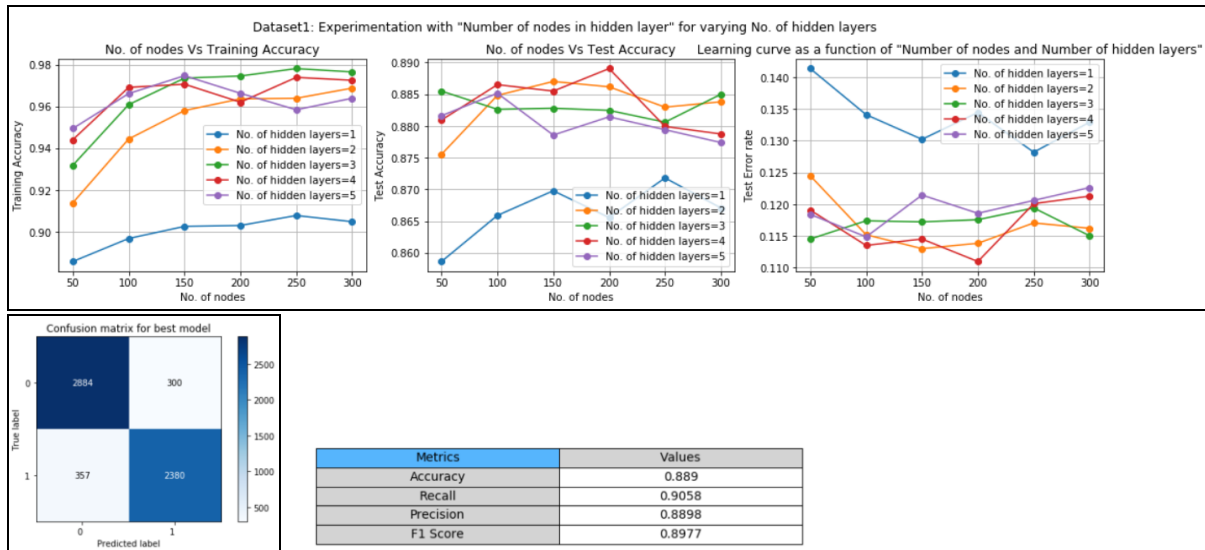
Dataset 2 – The 'relu' function records the best test accuracy of 98.31% for 'max no. of iterations' = 150. 'tanh' function also performs similar to 'relu' at the 150 iterations but it is inferred that 'relu' is better because it records a higher test accuracy for almos all values of the control parameter when compared with 'tanh'. It can be seen that the model achieves convergence at 150 iterations as both training & test accuracy values are flat for values>150. Hence, we fix 150 as the optimal value for 'max no. of iterations' control patameter. The classification metrics are reported for the best model ('relu', 150 iterations).



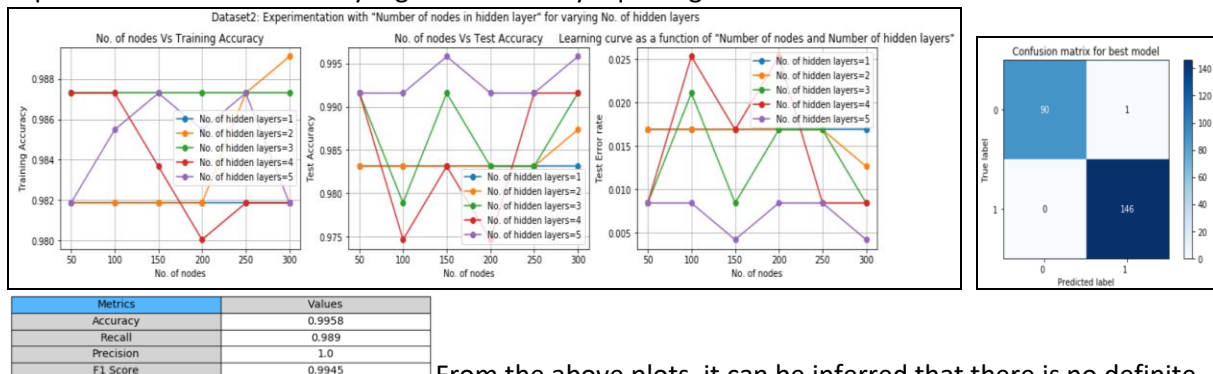
Experiment 1.2 – Experimentation with ‘Hidden layer size’ and ‘Number of nodes’ control parameters

The primary challenge of implementing an ANN model is to find the best values for ‘No. of Hidden layers’ and ‘No. of nodes in each layer’ that reports the best test accuracy. This can be done by using Grid search hyper-parameter optimization, which is not implemented for the entire parameter space, but implemented in a specific range of 1-5 hidden layers and 200 - 600 nodes for each hidden layer considering computational expenses and complexity. The models obtained in Experiment 1.1 are used for further experimentation. (Dataset 1 -‘tanh’ activation function, Dataset 2 -‘relu’ activation function)

Dataset 1 – The below plots record the test error rates for various ‘hidden layer size’ and ‘no. of nodes’ in each hidden layer. The lowest test error rate of 11.1% is recorded for 4 hidden layers and 200 nodes for each hidden layer. This corresponds to a neural net model with 1 input layer, 4 hidden layers with 200 nodes each and 1 output layer. This model is slightly better than the one obtained in Expt 1.1.



Dataset 2 – The best test accuracy of 99.58% is recorded for a model with 5 hidden layers and 300 nodes. This corresponds to a neural net model with 1 input layer, 5 hidden layers with 300 nodes each and 1 output layer. The ‘precision’ metric is 100% which tells us that all the values that were predicted by the model as positive are indeed positive actually. This model is better than the one obtained in experiment 1.1 and has a very high test accuracy equalling almost 100%.

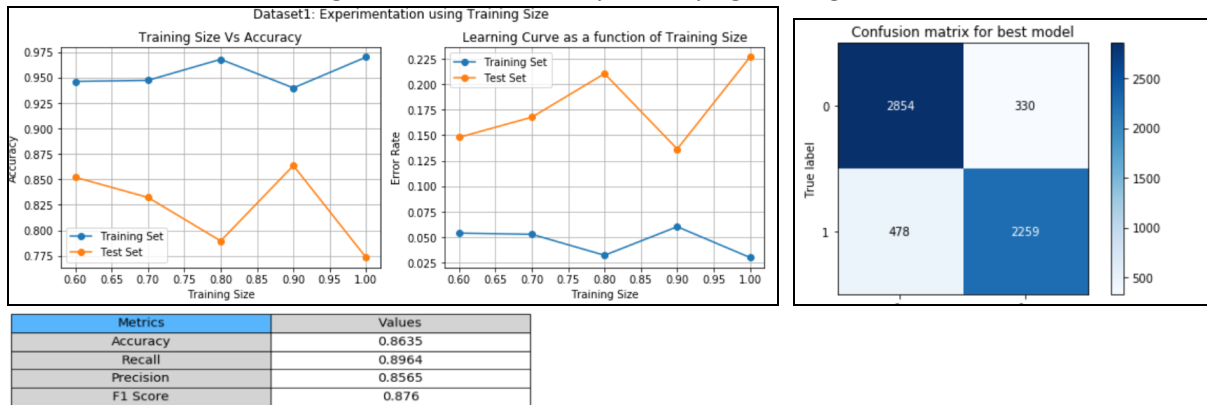


From the above plots, it can be inferred that there is no definite relationship between ‘test accuracy’ and ‘hidden layer size’. As the No. of nodes increase, we don’t see any discernible trend in test accuracy curve. The best values for ‘hidden layer size’ can’t be determined by a rule of thumb and have to be arrived at by thorough experimentation only.

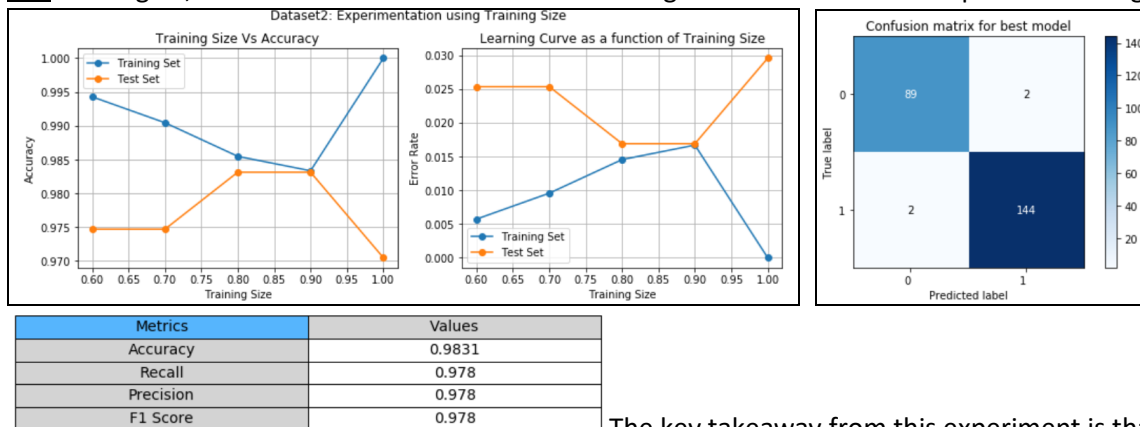
Experiment 1.3 – Experimentation using ‘Training Size’ parameter

In this experiment, the control parameter is the training size. This experiment is performed to understand the relationship between Training set size and train/test error rates. The best model that is obtained until now is used for further experimentation here. (Dataset 1 – ‘tanh’, 600 max iterations, 4 hidden layers, 200 nodes; Dataset 2 – ‘relu’, 150 max iterations, 5 hidden layers, 300 nodes)

Dataset 1 – From the below plots, it can be inferred that for a training size of 90% of total training set, the best test accuracy of 86.35% is reported. The classification metrics for the best model is reported below. There is a fluctuating trend in test accuracy for varying training set sizes.



Dataset 2 – The lowest test error rate of 1.69% is recorded for a training set size of 90% of total training set. Here again, the trend of test error rate is fluctuating and non-linear with respect to training set size.



The key takeaway from this experiment is that training set size and test accuracy have no definite relationship. It is wrong to infer that, with more training data, we can produce a better model which would generalize better in the presence of new data. It can be inferred that the distribution of data is of primary importance when a model is trying to learn using a specific machine learning algorithm. Though we don't really have a control over the nature of the data we collect, we can address this aspect by performing K-fold cross validation which will give the best subset of data that produces the best test accuracy. 5-fold cross validation is implemented as a separate experiment for both the algorithms and both the datasets after the experimentation for KNN.

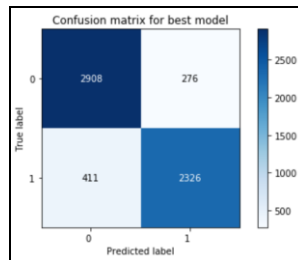
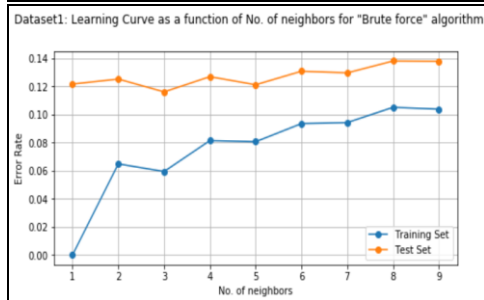
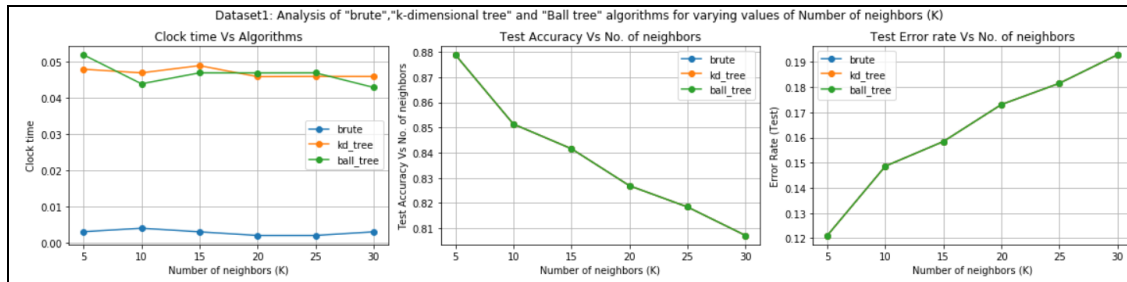
Tasks – Task 2 – K Nearest Neighbours (KNN)

This supervised learning algorithm is implemented using ‘KNeighborsClassifier’ package from scikit-learn library. I have performed 3 experiments involving control parameters - *algorithm* used to compute nearest neighbours, *distance metric*, *training set size* and *No. of nearest neighbours (k)*.

Experiment 2.1 – Experimentation with type of ‘algorithm’ used as a function of No. of neighbors (k)

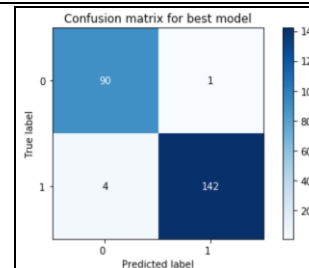
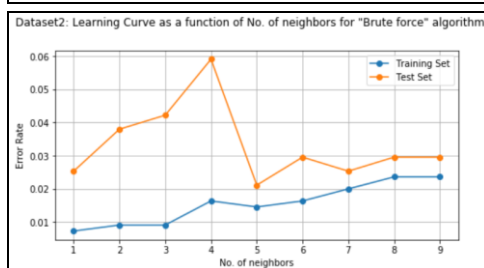
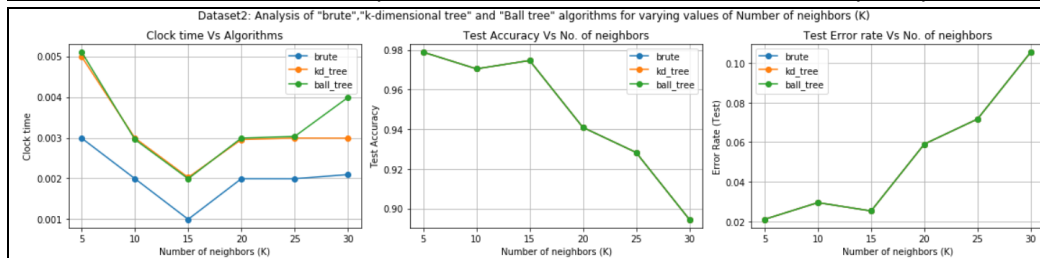
The ‘No. of neighbors’ control parameter denotes the number of neighbors that are to be clustered together as similar. The ‘algorithm’ that does this function can be ‘brute-force’, ‘kd_tree’ and ‘ball_tree’. Brute force involves calculation of distances between 2 points for all the data space. K-d tree involves indexing the data by creating trees which are easier to traverse for finding similarity. Ball-tree is also an indexing algorithm wherein data is visualized as hyperspheres in a binary tree. All the tree algorithms perform similarly with a difference only in time taken. The tree algorithms are faster for higher dimensions. Clock time is also projected to understand the fastest working algorithm for each dataset.

Dataset 1 – It can be inferred from below plots that ‘Brute-force’ is the fastest algorithm outperforming others by taking very little time. As expected, all the algorithms record the same test accuracy for varying ‘No. of neighbors’ (k). The best model is the one with k=3 recording a test accuracy of 88.4%.



Metrics	Values
Accuracy	0.884
Recall	0.9133
Precision	0.8762
F1 Score	0.8944

Dataset 2 – It is interesting to note that brute is faster compared to other tree algorithms in this dataset as well. This is primarily dependent on the ‘No. of features’. The best model is in the range of k=5. The lowest error rate of 2.11% is reported for a value of k=5 as evidenced by the plots below.

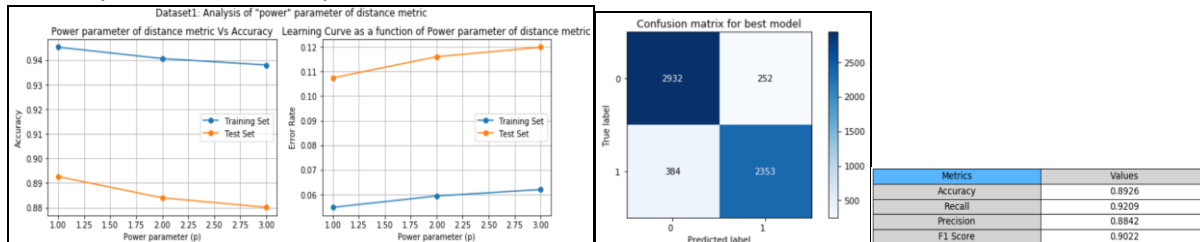


Metrics	Values
Accuracy	0.9789
Recall	0.989
Precision	0.9574
F1 Score	0.9729

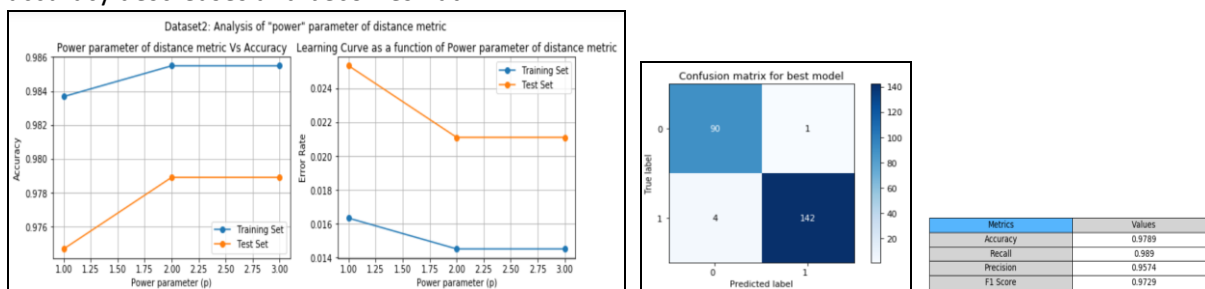
Experiment 2.2 – Experimentation using distance metrics (Euclidean, Manhattan)

In this experiment, the control parameter is the distance metric. The power of the minkowski metric is varied in the range of 1 to 3 to correspond to various distance metrics such as Manhattan (power=1) and Euclidean (power=2). We try to find the best distance metric through this experiment for each dataset.

Dataset 1 – It can be inferred that though there isn't a significant difference in test accuracy for various metrics, the Manhattan distance(p=1) records the best test accuracy of 89.26%. It can be seen that accuracy decreases as the power of the distance metric (minkowski metric) increases.



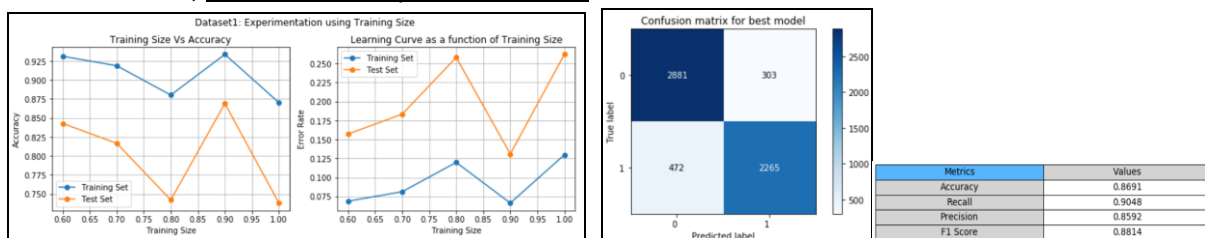
Dataset 2 – For this dataset, it can be interpreted from the below plots that monkowski metric with power=2 and power=3 perform better than the Manhattan distance. We choose the Euclidean metric as the best distance metric with the lowest error rate of 2.11%. In this scenario, as power increases the accuracy decreases and becomes flat.



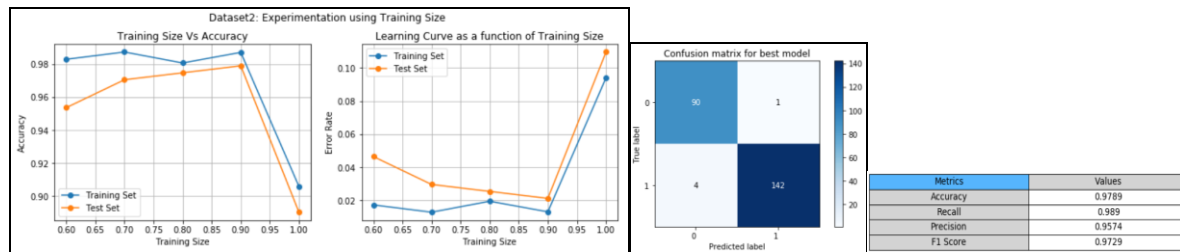
Experiment 2.3 – Experimenting with Training Size

In this experiment, the control parameter is the training size. This experiment is performed to understand the relationship between Training set size and train/test error rates. The best model that is obtained until now is used for further experimentation here. (Dataset 1 – 'brute', k=3, Manhattan distance; Dataset 2 – 'brute', k=5, Euclidean distance)

Dataset 1 - The best model is the one with a training size of 90% of the entire train set. Looking at the confusin matrix, the test accuracy is 86.91% with a Recall rate of 90.48%.



Dataset 2 – Here also, the best model is the one with traning size of 90% of original training set, the Test accuracy is peaking at 97.89%. As already interpreted, the distribution of data is the key point to consider. Based on the distribution of the data, the learnings can be different, and the model can perform differently on the test set.

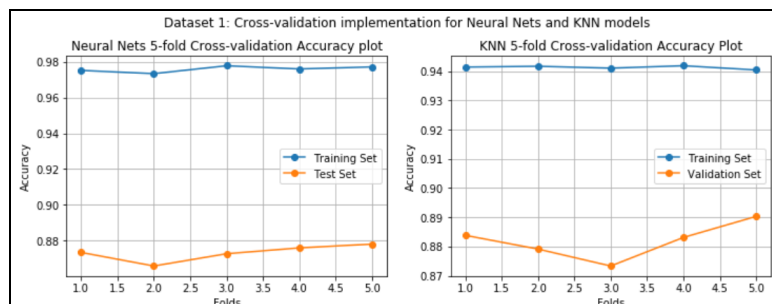


Cross Validation Implementation

Cross validation is performed to mainly understand if we can obtain a subset of the training data that can generalize better than the entire training set taken into account. Such a subset can be used for future experimentation to yield better results compared to the conventional models built using the entire training set. In this experiment, 5-fold cross-validation is implemented for both datasets for both the algorithms – ANN and KNN to understand if cross-validation can help boost the test accuracy by giving us the best subset of data that generalizes the best on the test set.

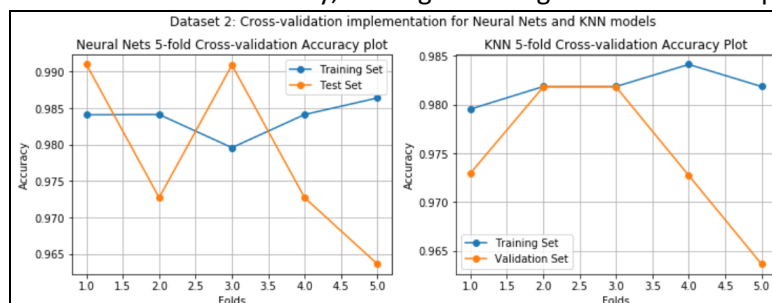
Dataset 1

- The model with entire training set records a test accuracy of 88% for both the algorithms.
- In Cross-validation, for ANN, there isn't a significant improvisation in the test accuracy (flat at ~88%).
- For KNN, it is to be noted that the 5th fold records an accuracy of 89% which is more than the base accuracy. This 5th fold subset of data can be considered for future experimentation for yielding better results.



Dataset 2

- The model with entire training set records a test accuracy of 98.31% for ANN & 97.89% for KNN.
- In Cross-validation, for ANN, the 1st fold records a test accuracy of 99% which is considerably more than the base model's accuracy. The 3rd fold also records similar accuracy but yields a poorer training accuracy. The 1st fold subset of data is a good candidate for future experimentation.
- For KNN, it is to be noted that the 2nd and 3rd folds record an accuracy of 98% which is a little more than the base accuracy, making them eligible for further experimentation for better models(results).



Conclusion / Discussion

Dataset 1 – Energy Data			Dataset 2 – Audit Data		
Rank	Algorithms	Test Accuracy (%)	Rank	Algorithms	Test Accuracy (%)
1	K-Nearest Neighbours	89.26	1	Artificial Neural Networks	99.58
2	Artificial Neural Networks	88.9	2	SVM	99.16
3	Boosted model of Decision Tree	88.81	3	Decision Tree	98.31
4	SVM	88.16	4	Boosted model of Decision Tree	97.89
5	Decision Tree	85.29	4	K-Nearest Neighbours	97.89

The table consisting of the ranking of various algorithms for both the datasets is listed above. The key takeaways are summarised below:

- Dataset 1 – Best model – K Nearest neighbours – 89.26% test accuracy
The Boosted model of the decision tree produced the best accuracy of 88.41%, but with the introduction of ANN and KNN, both the algorithms outperformed the previous algorithms. ANN model records a test accuracy of 88.9% which is also higher than the boosted model.
- Based on the analysis of various algorithms for dataset 1, it can be concluded that K-Nearest neighbours is the best algorithm within the scope of performed experimentation. A reasonable inference based on the above conclusion is that the data distribution is roughly homogeneous, meaning similar records are present near each other. SVM is also a good algorithm in this case and this is evident by the reasonably comparable accuracies (differ by 1%). KNN, however has outperformed owing to its simplicity, although being very slow when the data is huge.
- Dataset 2 – Best model – Artificial Neural Networks – 99.58% test accuracy
The ANN model has outperformed the previously best SVM model which had a test accuracy of 99.16%. K-Nearest neighbours has performed poorly and has the lowest test accuracy of 97.89%.
- For dataset 2, the experimentation with various algorithms has yielded that ANN is the best model with a stunning accuracy of almost 99.58%. We can also see that SVM and Decision Tree have comparable test accuracies supporting an intuition that the data is simplistic in nature with respect to features. The data space can be easily classified using simple rules on the feature variables.
- Although, we have inferred that KNN is the “best” algorithm for Dataset 1 and ANN is the “best” algorithm for Dataset 2, these inferences are valid only under the scope of the performed experimentation, because, to find the “best” model we would need to perform hyper-parameter optimization for each algorithm in the entire parameter space to find the parameters that yield the model with the best test accuracy.
- Also, here is no “best” algorithm on a universal basis. It solely depends on the dataset, its size, distribution, variance etc. From my perspective, the “best” algorithm would have optimal training accuracy and the best testing accuracy. The “best” model never occurs by chance, it must be worked upon incessantly. Extensive pre-processing, visualization and proper tuning using hyperparameters of the built model is mandatory for to get the “best model”.
- Recalling the results of Cross-validation, I would conclude that it is one operation which can be performed to get better results. It was inferred that training size of 90% of total training set yielded a better test accuracy than the entire training set. This nature of the dataset can be leveraged to produce better models.
- Overall, KNN performed better than ANN for Dataset 1 recording an accuracy of 89.26%. ANN outperformed KNN for Dataset 2 reporting a spiking accuracy of 99.58%. This experiment strongly asserts my previous conclusion that the best algorithm depends on the data, its characteristics and our understanding of it.