**Algorithms Chosen:**

- **Code taken from Phanindra for Random Forest and Decision Tree algorithms.**
- **Code taken from Yoshita for Neural Network algorithms.**

We have decided to work with Neural Networks, Random Forest and Decision Trees. We have tried all the algorithms taught in class and choose the top three performing algorithms. We feel for this dataset these algorithms are best suited and hence we can see great accuracies and f-1 scores. For each dataset we have verified multiple factors, majorly these factors provided below.

- Knowledge of Data: The data's structure and complexity help dictate the right algorithm.
- Accuracy Requirements: Different questions demand different degrees of accuracy, which influences algorithm selection.
- Processing Speed: Algorithm choice may depend on the time constraints in place for a given analysis.
- Variable: The unique features considered while training the model for the optimal result and accuracy help to determine the right algorithm.
- Parameters: Factors such as the number of iterations directly relate to training time needed generating output.

These factors are applicable to all the datasets below.

**General Trends for different Algorithms.**

- **Neural Network:** In general if the model is too complex then we try to increase the lambda value to update the weights. Whereas for simpler architectures we might not need higher lambda values as they might further increase the cost and reduce the weights higher which might overfit but perform badly on test data. We need to use lower values of lambda for simpler models to penalize the model. This will help the cost function to depend only on the errors and modify the weights based on this. On the contrary, if we use higher lambda it would base the cost value on this lambda and update the network using that error functioned value.

- **Random Forest:** For varying values of n_tree, I train using a bootstrapping method where 9 folds were used for training the model and the remaining one fold was used for testing the mode. Repeated this procedure for 10 times with various combinations of folds and each time obtained all the metrics and the resulting metric is the average of all these metrics. After going through detailed and thorough analysis it made it clear that all the metrics had significant effect with increasing n_tree values. As we perform majority voting of predictions obtained from n_trees used in the Random Forest we get good performance metrics but usage of bigger values of n_trees could result in overfitting the model. So, we need to tune the value of n_tree such that it mitigates the effect of overfitting.

  This increase was only until it reached a threshold value and further increase resulted in decrease of performance. Specifically, for these datasets it has been 40 constantly. On the other hand, F1 score is a little harder metric to optimize and the increase in n_tree didn't affect much. As it is directly dependent on both precision and recall therefore whenever it decreases in one metric f-1 score also experiences the same. As we know that achieving both precision and recall at time can be challenging which is why

obtaining high f-1 scores can also be as challenging as this. It shows that predicting all the classes correctly is more important than predicting on class correctly. And this could be shown with an f-1 score which is a perfect blend of both precision and recall.

In this approach we used minimum instances stopping criteria and maximal depth and provided the analysis for both the hyper parameters. Here we stop splitting nodes whenever the data partition associated with the current node contains less than 5 instances. And for maximal depth we used the number of the columns as the max depth the tree can achieve.

**Dataset: Oxford Parkinson's Disease Detection Dataset:**

Parkinson's dataset has 195 instances with 22 attributes each corresponding to the measurement of one specific property of the patient's voice. As all the attributes contribute in predicting whether a person is diagnosed with disease or not. It would be ideal to use Neural Networks as it can learn well based on all the attributes. When choosing a model it is important to get a high recall metric for this dataset. As we don't want to miss out on predicting patients suffering from disease. So, we tried multiple algorithms and found high recall for Random forest. Training decision trees would be computationally faster for this dataset; all the attributes are numerical which will divide the whole dataset into two nodes at each level.

With Neural Networks for lambda = 0.1 we were able to achieve accuracy as high as 89.23% and f-1 score 93.26%. Whereas with Random Forest we achieved 93.71% accuracy and 91.4% with n_tree = 40. Base model decision tree here achieved accuracy of 89.83 and F-1 score of 86.84. Here Random forest achieved the higher metrics in comparison to other two algorithms there could be multiple reasons for this.

a. **Neural Network-** When analyzing this dataset with a neural network we used different lambda values, layers and neurons as hyper parameters. To fix on this architecture we computed accuracies on the training set first and if the accuracy is very high we have eliminated that architecture as it overfits and gives very lower accuracy on the test set. So, with [9, 4] neurons in the hidden layer performed well on training and test sets. Then fixing this architecture we tried to evaluate results for different lambda values such as [0,0.1,0.25,1]. As expected the model was performing better with lambda = 0.1 than 1 with accuracy as high as 89.23%. This could be because of all the attributes contributing to the model's learning process. Further, increase in lambda penalizes the model and gives lower accuracies. As the weights are not updated appropriately and the model might not have learnt patterns in the training model. Similarly, the f-1 score is highest when lambda = 0.1 i.e. 93.26%.
    - Step size: While fine tuning the model we verified the neural network model on multiple step sizes-0.5, 0.8, 1, 1.5. As we can see the model is best performing with 0.8 step size so we have provided all our results with the same step size. We observed that when we increased the step size too high the cost graphs oscillated.
    - Architecture: Similarly, we used multiple architectures where we used 2 hidden layers with [9,4] and [20,10] neurons and one hidden layer with [5], [10] neurons. Best performance was observed with [9,4] architecture as the model was able to learn the trends. Whereas with [20, 10] accuracy was really high for the training

set but observed very low accuracy on the test set. This explains that the model is overfitted but it hasn't learned to generalize well on the test set.

b. **Random Forest-** For maximal depth stopping criteria we were able to achieve higher accuracies and f-1 scores for n_tree = 40 i.e 93.71% and 91.4% respectively. Whereas for minimal size for split, highest accuracy was for n_tree = 10, 30 and highest f-1 score was for n_tree = 10. We could achieve 92.3% accuracy for both n_tree = 10, 30 and 90% f-1 score for n_tree = 10. For these datasets the accuracies increased from n_tree = 5 to n_tree = 40 and beyond this model overfits and were not able to generalize well for the test data. Whereas for F-1 score we couldn't identify any such trends here.
    - Minimal split size- We have implemented optimal split size i.e. 5 as if we increase the split size tree will stop early and performance will be very poor. Similarly, if we have a high split size model might overfit to training data but when tested on a test set it performs poorly.
    - Minimal depth size: Similarly, if the depth is high the model again overfits and if it is too low it might be able to split on all the best attributes and stop early. This will give low performance so we have chosen a 22 steps deep tree.

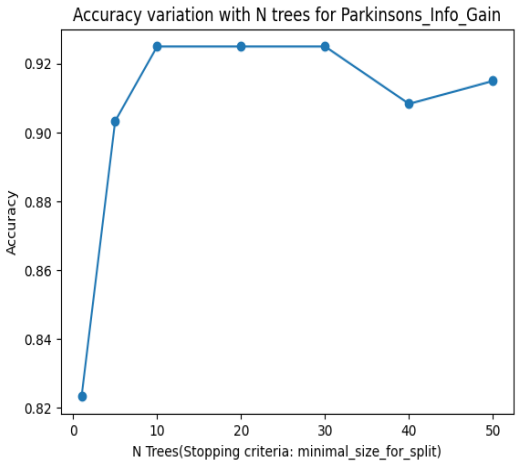
I would like to conclude that I will choose the Random Forest algorithm for this dataset as the best model since it has high accuracy and at the same time high f-1 score.

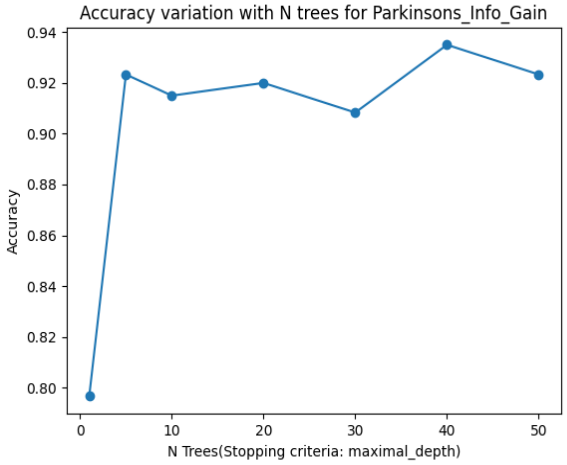
**Algorithm- 1 Neural Network for 1000 iterations and step size = 0.8.**

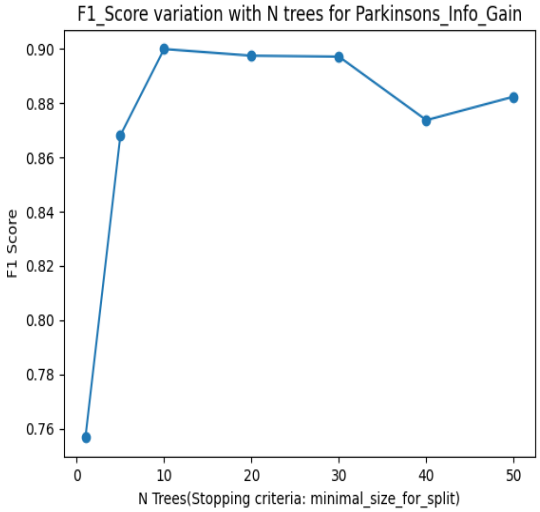
| **Metrics / λ** | **λ = 0** | **λ= 0.25** | **λ= 0.1** | **λ= 1** |
|---|---|---|---|---|
| **Accuracy** | 83.58 | 86.15 | 89.23 | 85.64 |
| **F-1 score** | 88.86 | 90.96 | 93.26 | 91.26 |

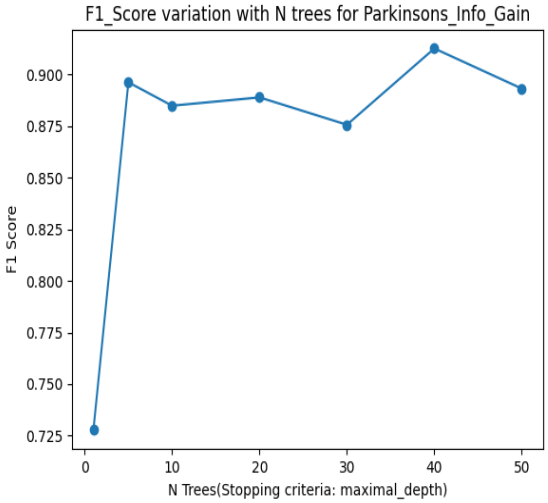
**Algorithm- 2 Random forest with different n_trees and different stopping criterion**

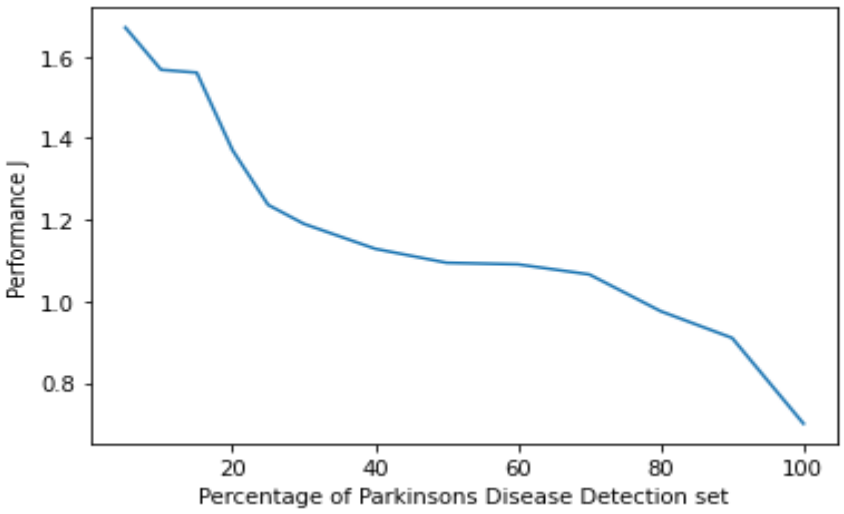
| **Minimal Size for split = 5** | | | | |
|---|---|---|---|---|
| **Metrics / n_trees=** | **n_trees= 10** | **n_trees= 30** | **n_trees= 40** | **n_trees= 50** |
| **Accuracy** | 92.3 | 92.3 | 90.6 | 91.94 |
| **F-1 score** | 90 | 88.82 | 87.8 | 88.4 |
| **Maximal Dept = number of columns** | | | | |
| **Metrics / n_trees** | **n_trees= 10** | **n_trees= 30** | **n_trees= 40** | **n_trees= 50** |
| **Accuracy** | 91.9 | 91 | 93.71 | 92.31 |
| **F-1 score** | 89 | 87.5 | 91.4 | 90.66 |


Accuracy variation with N trees for Parkinsons_Info_Gain
Accuracy
N Trees(Stopping criteria: minimal_size_for_split)


Accuracy variation with N trees for Parkinsons_Info_Gain
Accuracy
N Trees(Stopping criteria: maximal_depth)


F1_Score variation with N trees for Parkinsons_Info_Gain
F1 Score
N Trees(Stopping criteria: minimal_size_for_split)


F1_Score variation with N trees for Parkinsons_Info_Gain
F1 Score
N Trees(Stopping criteria: maximal_depth)


Performance J
Percentage of Parkinsons Disease Detection set

**Dataset: Titanic Survival Dataset**

Titanic Survival dataset contains 887 instances with 8 attributes. From the list of these attributes we can see that few attributes don't contribute to training the model such as “Name” of the passenger. Similarly few attributes can have high priority for the model. So, we have preprocessed the dataset and included the features in model training. Neural Networks and Random Forest can work well for this featured dataset.

Random Forest were able to achieve accuracy 80.41% and f-1 score 79.3%. For this dataset Neural Network was able to perform better than Random Forest and obtain higher accuracy of 80.88% and f-1 score of 85.23%. This is slightly better than Random Forest but this shows that for the complex dataset Neural Network was able to learn better and achieve higher metrics for different hyperparameters. Also, with the base model decision tree we achieved only 77.85% accuracy and 76.43% f-1 score.

a. **Neural Network-** As mentioned above we have validated all the architectures on training data. We were able to get better performance with one hidden layer and [6] neurons.Then fixing this architecture we tried to evaluate results for different lambda values such as [0,0.1,0.25,1]. Model was performing reasonably well for all the lambda values ranging from 79% to 81% for accuracy and 84% to 85%. The highest accuracy of 80.88% and highest f-1 score of 85.23%, we could see this for lambda = 0.25.
    - Step size: While fine tuning the model we verified the neural network model on multiple step sizes-0.3, 0.9, 1.3, 1.7. As we can see the model is best performing with 0.9 step size so we have provided all our results with the same step size. We observed that when we increased the step size too high the cost graphs oscillated.
    - Architecture: Similarly, we used multiple architectures where we used 2 hidden layers with [9,4] and [6,3] neurons and one hidden layer with [6], [10] neurons. Best performance was observed with [6] architecture as the model was able to learn the trends. Whereas with [9, 4] accuracy was really high for the training set but observed very low accuracy on the test set. This explains that the model is overfitted but it hasn’t learned to generalize well on the test set.

b. **Random Forest-** For maximal depth stopping criteria we were able to achieve higher accuracies and f-1 scores for n_tree = 50 i.e 81% and 79.89% respectively. Whereas for minimal size for split, highest accuracy, f-1 score was for n_tree = 40. We could achieve 80.34% accuracy and 78.56% f-1 score for n_tree = 40. For these datasets the accuracies increased from n_tree = 5 to n_tree = 40 and beyond this model overfits and were not able to generalize well for the test data you can see the decreasing trends from n_tree = 50.

- Minimal split size- We have implemented optimal split size i.e. 5 as if we increase the split size tree will stop early and performance will be very poor. Similarly, if we have a high split size model might overfit to training data but when tested on a test set it performs poorly.
    - Minimal depth size: Similarly, if the depth is high the model again overfits and if it is too low it might be able to split on all the best attributes and stop early. This will give low performance so we have chosen a 7 steps deep tree.

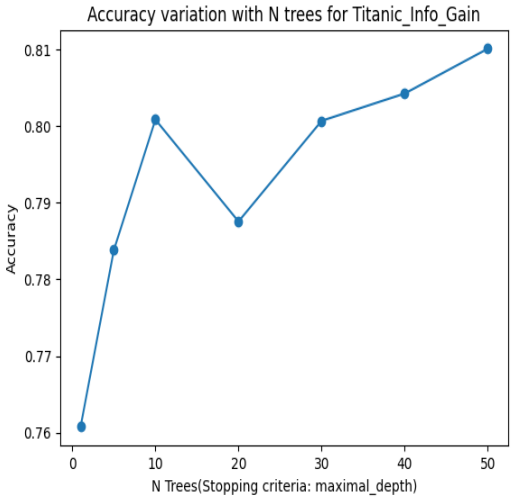
I would like to conclude that I will choose the Neural Network algorithm for this dataset as the best model since it has high accuracy and at the same time high f-1 score.

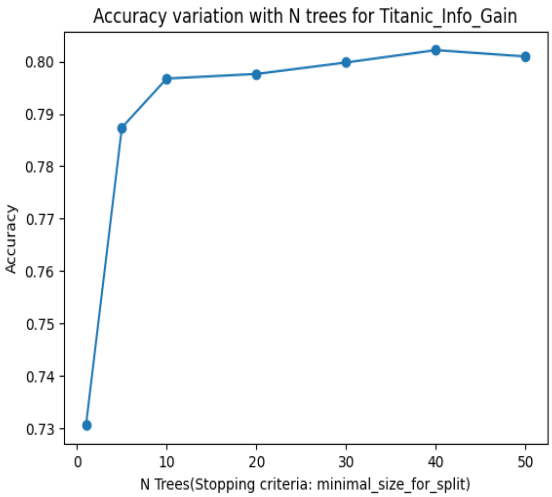
**Algorithm- 1 Neural Network for 1000 iterations and step size = 0.9.**

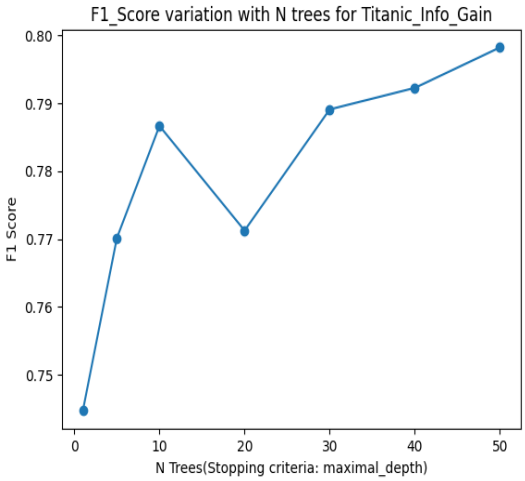
| **Metrics / λ** | **λ = 0** | **λ= 0.25** | **λ= 0.1** | **λ= 1** |
|---|---|---|---|---|
| **Accuracy** | 80.31 | 80.88 | 80.54 | 79.63 |
| **F-1 score** | 85.17 | 85.23 | 84.91 | 84.21 |

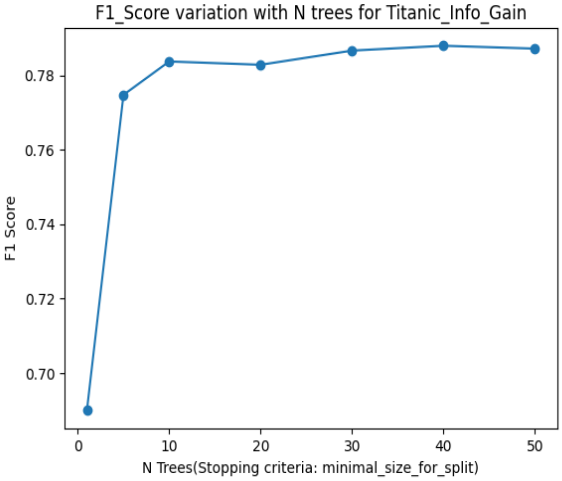
**Algorithm- 2 Random forest with different n_trees**

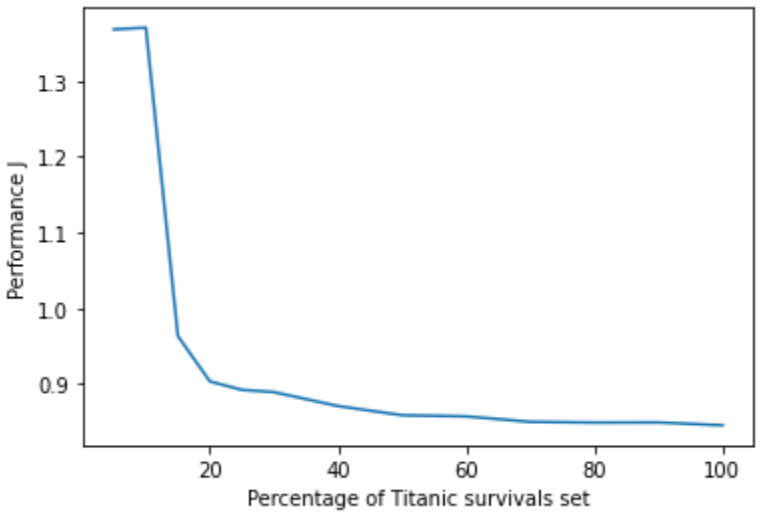
| **Minimal Size for split = 5** | | | | |
|---|---|---|---|---|
| **Metrics / n_trees=** | **n_trees= 10** | **n_trees= 30** | **n_trees= 40** | **n_trees= 50** |
| **Accuracy** | 79.5 | 79.8 | 80.34 | 80 |
| **F-1 score** | 78.3 | 78.45 | 78.56 | 78.38 |
| **Maximal Dept = number of columns** | | | | |
| **Metrics / n_trees** | **n_trees= 10** | **n_trees= 30** | **n_trees= 40** | **n_trees= 50** |
| **Accuracy** | 80.16 | 80.2 | 80.41 | 81 |
| **F-1 score** | 78.74 | 79 | 79.3 | 79.89 |


Accuracy variation with N trees for Titanic_Info_Gain
Accuracy
N Trees(Stopping criteria: maximal_depth)


Accuracy variation with N trees for Titanic_Info_Gain
Accuracy
N Trees(Stopping criteria: minimal_size_for_split)


F1_Score variation with N trees for Titanic_Info_Gain
F1 Score
N Trees(Stopping criteria: maximal_depth)


F1_Score variation with N trees for Titanic_Info_Gain
F1 Score
N Trees(Stopping criteria: minimal_size_for_split)


Performance J
Percentage of Titanic survivals set

**Dataset: Loan Eligibility Dataset**

For evaluating a loan application this dataset has multiple attributes where credit history, income information etc. features might have higher influence in training the model. Similarly, in preprocessing we have removed the attributes which don't have any contribution in the learning process like “Loan_id”. We need to reduce wrong predictions of ineligible candidates as eligible for loan. This might need us to focus on precision which helped us to focu on the top three algorithms for this dataset. So, we have chosen to use Neural Networks, Random Forest, and Decision trees.

For this dataset Neural Networks performed better than other algorithms we could achieve 80.20% accuracy for lambda = 1. Whereas for Random Forest maximum accuracy that we could obtain is 80.15% which is closer to the Neural Network model. And for the Decision tree we could only obtain 71.87% of accuracy and an f-1 score of 65.53%. For this dataset we can see a different trend in f-1 score as Random forest was able to give higher f-1 scores than any other algorithm; the highest recorded f-1 score is 80.09%.

a. **Neural Network-** For this dataset after trying with multiple architectures we were able to achieve better performance with two hidden layers and [6, 4] neurons. Then fixing this architecture we tried to evaluate results for different lambda values such as [0,0.1,0.25,1]. The model was performing better with lambda = 1 with accuracy as high as 80.2%. Further, increase in lambda overfits the model and gives lower accuracies. Similarly, the f-1 score is highest when lambda = 0.1 i.e. 93.26%.
    - Step size: While fine tuning the model we verified the neural network model on multiple step sizes-0.5, 0.9, 1.5, 2. As we can see the model is best performing with 1.5 step size so we have provided all our results with the same step size. We observed that when we increased the step size too high the cost graphs oscillated.
    - Architecture: Similarly, we used multiple architectures where we used 2 hidden layers with [10,3] and [6,3] neurons and one hidden layer with [10], [6] neurons. Best performance was observed with [10,3] architecture as the model was able to learn the trends. We were able to obtain reasonably better results with [10] neurons as well. Here the model is performing better with higher neurons performance didn’t increase drastically with extra hidden layers.

b. **Random Forest-** For maximal depth stopping criteria we were able to achieve higher accuracies and f-1 scores for n_tree = 40 i.e 80.15% and 74.3% respectively. Whereas for minimal size for split, highest accuracy and f-1 score was for n_tree = 50. We could achieve 76.07% accuracy and 81% f-1 score. We obtained better results from n_tree = 30 to 40 in some cases it also performed better for n_tree = 50.

- Minimal split size- We have implemented optimal split size i.e. 5 as if we increase the split size tree will stop early and performance will be very poor. Similarly, if we have a high split size model might overfit to training data but when tested on a test set it performs poorly.
    - Minimal depth size: Similarly, if the depth is high the model again overfits and if it is too low it might be able to split on all the best attributes and stop early. This will give low performance so we have chosen a 11 steps deep tree.

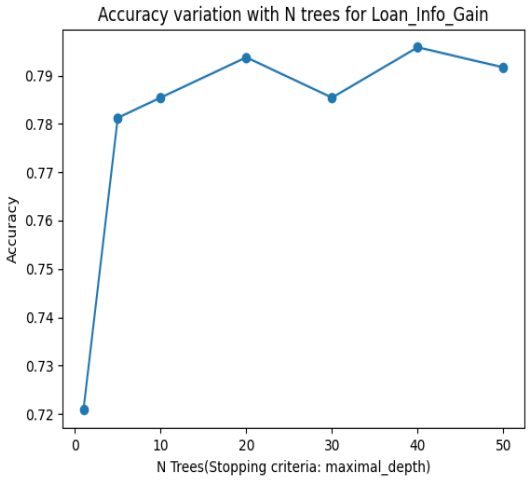
I would like to conclude that I will choose the Neural Network algorithm for this dataset as the best model since it has high accuracy, high precision and at the same time high f-1 score which is highly desirable for this dataset.

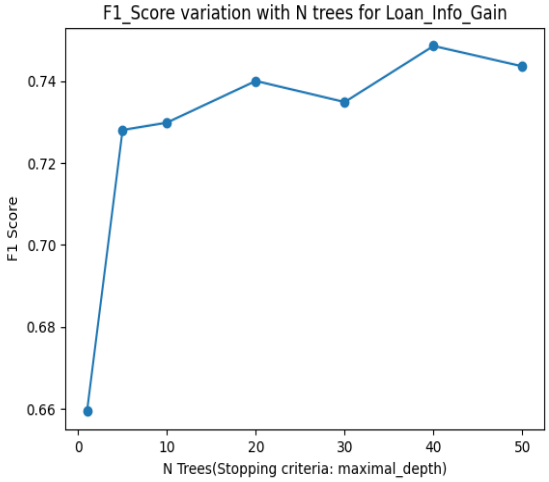
**Algorithm- 1 Neural Network for 1000 iterations and step size = 1.5.**

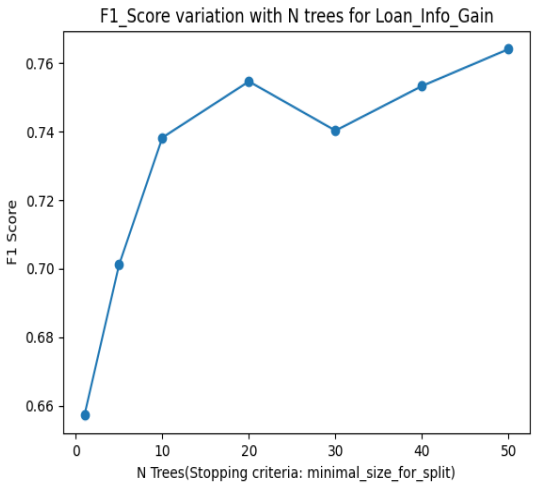
| **Metrics / λ** | **λ = 0** | **λ= 0.25** | **λ= 0.1** | **λ= 1** |
|---|---|---|---|---|
| **Accuracy** | 83.33 | 81.25 | 83.33 | 86.45 |
| **F-1 score** | 80.56 | 78.40 | 80.85 | 84.52 |

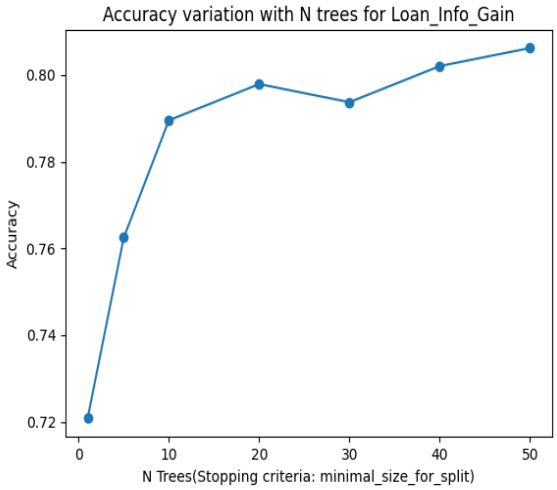
**Algorithm- 2 Random forest with different n_trees**

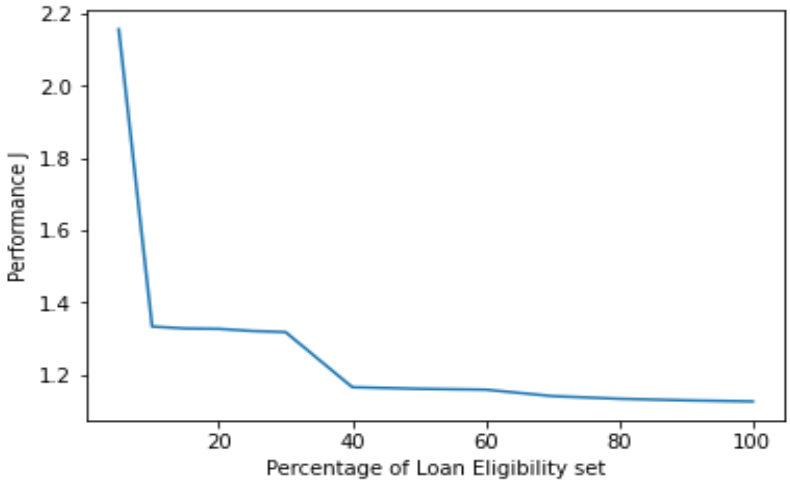
| **Minimal Size for split = 5** | | | | |
|---|---|---|---|---|
| **Metrics / n_trees=** | **n_trees= 10** | **n_trees= 30** | **n_trees= 40** | **n_trees= 50** |
| **Accuracy** | 74 | 74 | 75.4 | 76.07 |
| **F-1 score** | 79 | 79.35 | 80.09 | 81 |
| **Maximal Dept = number of columns** | | | | |
| **Metrics / n_trees** | **n_trees= 10** | **n_trees= 30** | **n_trees= 40** | **n_trees= 50** |
| **Accuracy** | 78.5 | 78.76 | 80.15 | 79.43 |
| **F-1 score** | 73 | 73.68 | 74.3 | 74.11 |


Accuracy variation with N trees for Loan_Info_Gain
Accuracy
N Trees(Stopping criteria: maximal_depth)


F1_Score variation with N trees for Loan_Info_Gain
F1 Score
N Trees(Stopping criteria: maximal_depth)


F1_Score variation with N trees for Loan_Info_Gain
F1 Score
N Trees(Stopping criteria: minimal_size_for_split)


Accuracy variation with N trees for Loan_Info_Gain
Accuracy
N Trees(Stopping criteria: minimal_size_for_split)


Performance J
Percentage of Loan Eligibility set

**Dataset: Handwritten Digits Recognition Dataset**

Handwritten Digits Recognition dataset has 64 attributes each explaining about one pixel and has 10 classes. We feel better performance can be achieved with neural networks as the model can be trained well for such a complex dataset. However, we observed significantly high performance with Random forest. The main reason for this could be as the image classification is based on 8*8 pixels and majority of the digits attributes are present in reasonably same pixels. So, the random forest is able to better split on these attributes and obtain high metrics.

It provides higher accuracy through cross validation. Random forest classifiers will handle the missing values and maintain the accuracy of a large proportion of data. If there are more trees, it won't allow over-fitting trees in the model. For Neural Networks it usually depends on the complexity of the network and various hyperparameters. Neural Networks, in my experience, have several hyper-parameters (number of layers, neurons per layer, activation functions, optimizers, regularizers, etc.) and are very hard in finding the best configuration for each task. Furthermore, NNs require caution as they are prone to overfitting.

a. **Neural Network-** When analyzing this dataset with a neural network we used different lambda values, layers and neurons as hyper parameters. In the second hidden layer we used half of the neurons from the first layer able to achieve higher performance. Then fixing this architecture we tried to evaluate results for different lambda values such as [0,0.1,0.25,1]. Model performed well with lower lambda values as it is giving maximum importance loss and increasing lambda penalized the model reducing the generality of the model. As expected the model was performing better with lambda = 0, 0.1 with accuracy as high as 96.92. Further, increase in lambda overfits the model and gives lower accuracies. Similarly, the f-1 score is highest when lambda = 0,0.1 i.e. 96.98% and 96.95% respectively.
    - Step size: While fine tuning the model we verified the neural network model on multiple step sizes-0.5, 0.9, 1.5, 2. As we can see the model is best performing with 0.9 step size so we have provided all our results with the same step size. We observed that when we increased the step size too high the cost graphs oscillated.
    - Architecture: Similarly, we used multiple architectures where we used 2 hidden layers with [25,20] and [30,20] neurons and one hidden layer with [40], [30] neurons. As the model has 64 attributes the model performed well and we have more neurons and with two hidden layers. With [30,20] the model gave little lower accuracies this shows that increasing neurons might overfit the model and increasing hidden layers also had significantly higher performance.

b. **Random Forest-** For maximal depth stopping criteria we were able to achieve higher accuracies and f-1 scores for n_tree = 40 i.e 93.71% and 91.4% respectively. Whereas for minimal size for split, highest accuracy was for n_tree = 10, 30 and highest f-1 score was for n_tree = 10. We could achieve 92.3% accuracy for both n_tree = 10, 30 and 90% f-1 score for n_tree = 10.

    -Minimal split size- We have implemented optimal split size i.e. 5 as if we increase the split size tree will stop early and performance will be very poor. Similarly, if we have a high split size model might overfit to training data but when tested on a test set it performs poorly.

- Minimal depth size: Similarly, if the depth is high the model again overfits and if it is too low it might be able to split on all the best attributes and stop early. This will give low performance so we have chosen a 64 steps deep tree.

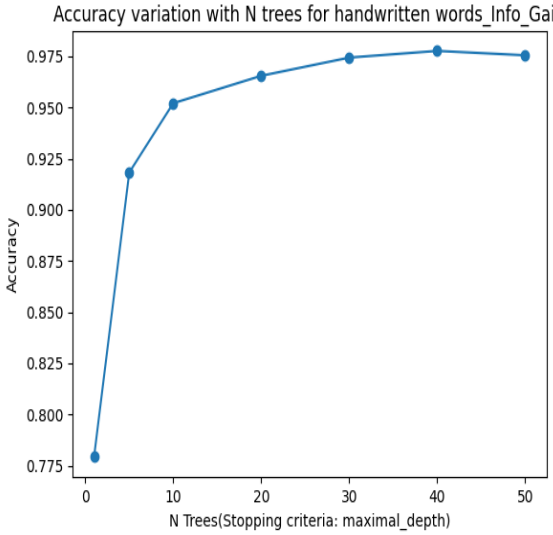
I would like to conclude that I will choose both Neural Network and Random Forest algorithms for this dataset as the best model since it has high accuracy and at the same time high f-1 score which is highly desirable for this dataset.

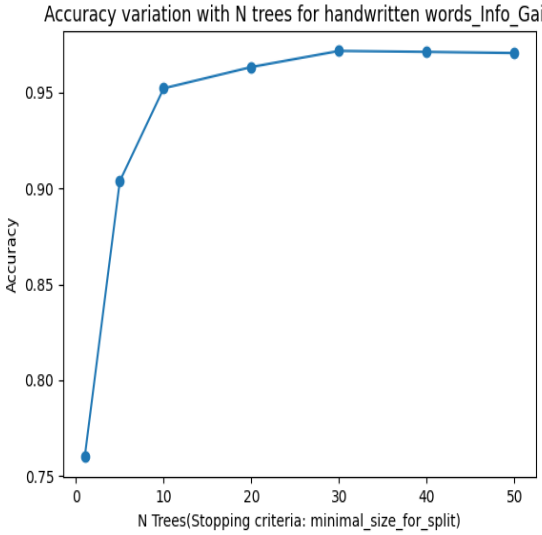
**Algorithm- 1 Neural Network for 1000 iterations and step size = 0.9.**

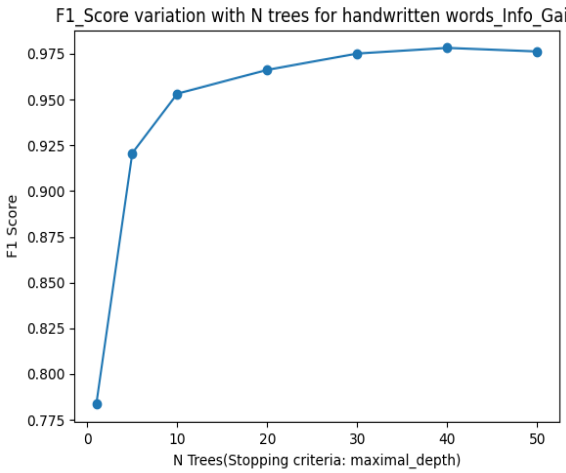
| **Metrics / λ** | **λ = 0** | **λ= 0.25** | **λ= 0.1** | **λ= 1** |
|---|---|---|---|---|
| **Accuracy** | 96.92 | 95.81 | 96.92 | 95.25 |
| **F-1 score** | 96.98 | 95.90 | 96.95 | 95.32 |

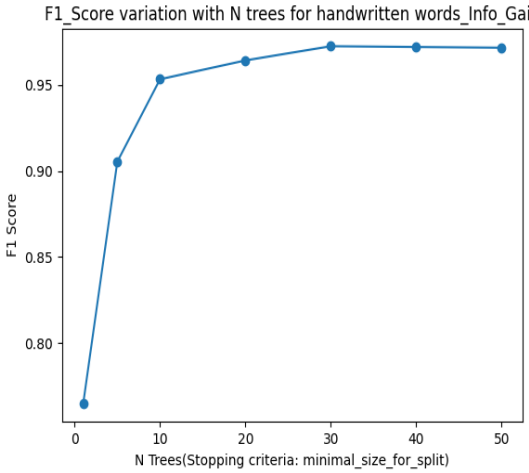
**Algorithm- 2 Random forest with different n_trees**

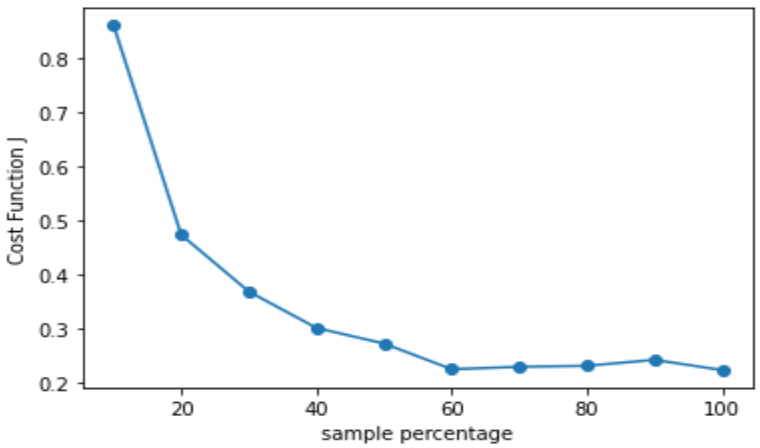
| **Minimal Size for split = 5** | | | | |
|---|---|---|---|---|
| **Metrics / n_trees=** | **n_trees= 10** | **n_trees= 30** | **n_trees= 40** | **n_trees= 50** |
| **Accuracy** | 95 | 96.1 | 96 | 96 |
| **F-1 score** | 90.25 | 96.78 | 96.6 | 96.6 |
| **Maximal Dept = number of columns** | | | | |
| **Metrics / n_trees** | **n_trees= 10** | **n_trees= 30** | **n_trees= 40** | **n_trees= 50** |
| **Accuracy** | 95.6 | 97.2 | 97.5 | 96.89 |
| **F-1 score** | 95.3 | 97.5 | 97.54 | 97 |


Accuracy variation with N trees for handwritten words_Info_Gain
Accuracy
N Trees(Stopping criteria: maximal_depth)


Accuracy variation with N trees for handwritten words_Info_Gain
Accuracy
N Trees(Stopping criteria: minimal_size_for_split)


F1_Score variation with N trees for handwritten words_Info_Gain
F1 Score
N Trees(Stopping criteria: maximal_depth)


F1_Score variation with N trees for handwritten words_Info_Gain
F1 Score
N Trees(Stopping criteria: minimal_size_for_split)


Cost Function J
sample percentage

**SUMMARY:**

| | Parkinson's Disease Detection | | Loan Eligibility Dataset | | Titanic Survival Dataset | | Hand-written Dataset | |
|---|---|---|---|---|---|---|---|---|
| | **Accuracy** | **F-1 Score** | **Accuracy** | **F-1 Score** | **Accuracy** | **F-1 Score** | **Accuracy** | **F-1 Score** |
| **Neural Network** | 89.23 | 93.26 | 80.2 | 67.58 | 80.88 | 85.23 | 96.92 | 96.98 |
| **Random Forest** | 93.71 | 90.66 | 80.15 | 80.09 | 81 | 79.89 | 97.5 | 97.54 |
| **Decision** | 89.83 | 86.84 | 71.87 | 65.55 | 77.85 | 76.43 | 78.5 | 79 |

An important finding in machine learning is that no single algorithm works best across all possible scenarios. Thus, no algorithm strictly dominates in all applications; the performance of machine learning algorithms varies wildly depending, for example, on the application and the dimensionality of the dataset. Accordingly, a good practice is to compare the performance of different learning. For most tasks where you deal with structured data, I've found tree-based algorithms (especially boosted ones) to outperform NNs.

According to the results, Random Forest is clearly the best classifier as it achieves the best classification results in over 90% of the cases. The results of Neural Networks are on average worse, but close to those of Random Forests. One additional finding is that Neural Networks are getting better when the complexity of the data set increases. Another study is concerned with the comparison of Neural Networks and Random Forests in predicting building energy consumption, which is a numerical prediction and not a classification case. According to the findings, Neural Networks performed marginally better than Random Forests. But the Random Forest models were able to effectively handle any missing values in the application. Thus, the Random Forests were able to accurately predict even when some of the input values were missing.

## EXTRA CREDIT-1: ALGORITHM-3

The main idea of a decision tree algorithm is to identify the features that contain the most information regarding the target feature and then split the dataset along the values of these features. The target feature values at the resulting nodes are as pure as possible. A feature that best separates the uncertainty from information about the target feature is said to be the most informative feature. The search process for the most informative feature goes on until we end up with pure leaf nodes.

**Accuracies and F-1 scores predicted using decision tree:**

| **Parkinson's Disease Detection Dataset** | |
|---|---|
| **Accuracy** | 89.83 |
| **F-1 score** | 86.84 |
| **Titanic Survival Dataset** | |
| **Accuracy** | 77.85 |
| **F-1 score** | 76.43 |
| **Loan Eligibility Dataset** | |
| **Accuracy** | 71.87 |
| **F-1 score** | 65.55 |
| **Handwritten Digits Recognition Dataset** | |
| **Accuracy** | 94.21 |
| **F-1 score** | 92.74 |

## EXTRA CREDIT-2: ALGORITHM-3

**About Dataset:** We have selected a dataset from kaggle which has numerical and categorical attributes with 6 classes. Imagine that you are a medical researcher compiling data for a study. You have collected data about a set of patients, all of whom suffered from the same illness. During their course of treatment, each patient responded to one of 5 medications, Drug A, Drug B, Drug C, Drug X and Y.

Part of this project is to build a model to find out which drug might be appropriate for a future patient with the same illness. The features of this dataset are Age, Sex, Blood Pressure, and the Cholesterol of the patients, and the target is the drug that each patient responded to. It is a sample of a multiclass classifier, and you can use the training part of the dataset to build a decision tree, and then use it to predict the class of an unknown patient, or to prescribe a drug to a new patient.

**Algorithm- 1 Neural Network for 1000 iterations and step size = 0.9.**
**Architecture = [7,3]**

| **Metrics / λ** | **λ = 0** | **λ= 0.25** | **λ= 0.1** | **λ= 1** |
|---|---|---|---|---|
| **Accuracy** | 95.37 | 92.58 | 93.37 | 90.62 |
| **F-1 score** | 85.47 | 77.09 | 83.99 | 78.8 |

**Algorithm- 2 Random forest with different n_trees**

| **Minimal Size for split = 5** | | | | |
|---|---|---|---|---|
| **Metrics / n_trees=** | **n_trees= 10** | **n_trees= 30** | **n_trees= 40** | **n_trees= 50** |
| **Accuracy** | 91 | 95 | 96.47 | 95 |
| **F-1 score** | 83.23 | 85.78 | 88.92 | 86 |
| **Maximal Dept = number of columns** | | | | |
| **Metrics / n_trees** | **n_trees= 10** | **n_trees= 30** | **n_trees= 40** | **n_trees= 50** |
| **Accuracy** | 94.3 | 96 | 95.38 | 95 |
| **F-1 score** | 86 | 88.79 | 88 | 87.63 |

**Algorithm- 3 Decision Tree**

| | |
|---|---|
| **Accuracy** | 91.29 |
| **F-1 score** | 84.79 |