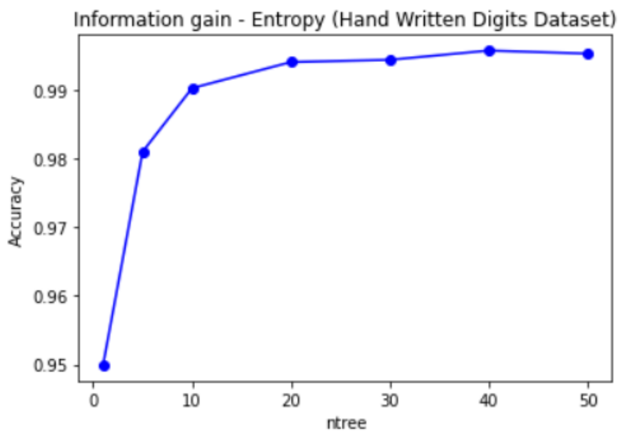
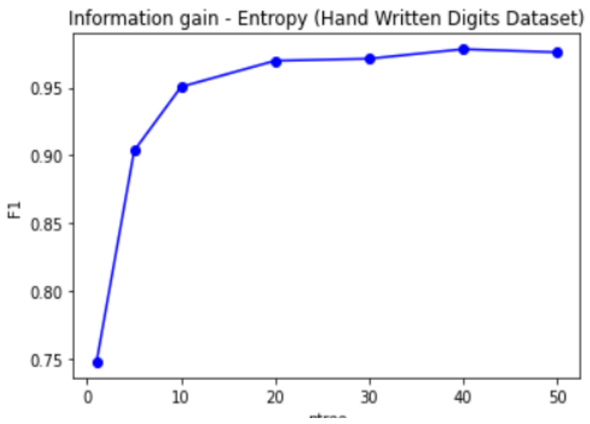
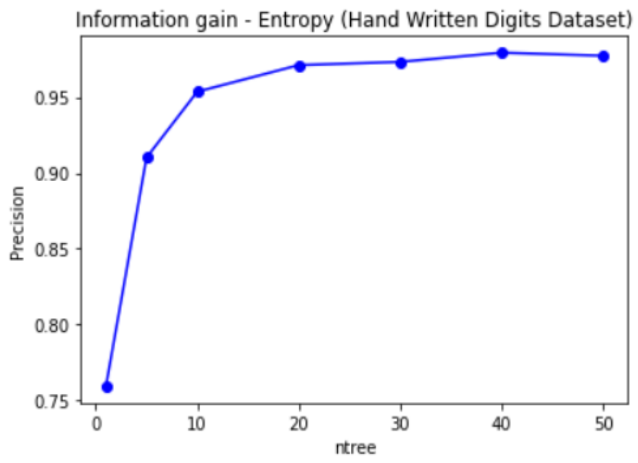
COMPSCI 589 PROJECT REPORT

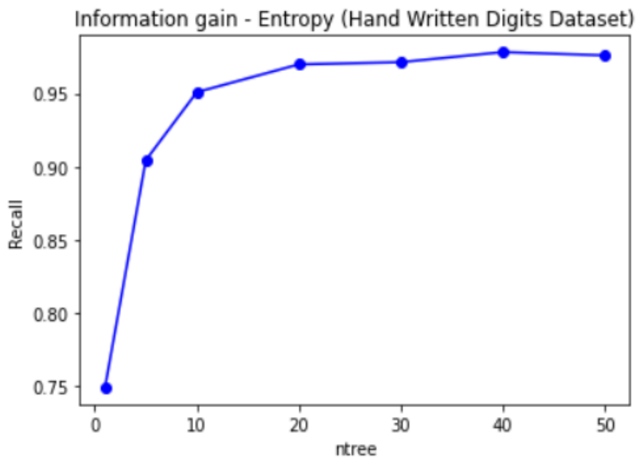
Hand-Written Digits Recognition Dataset :

RANDOM FORESTS -









Performance Results -

nTrees	Accuracy	Precision	Recall	F1 Score
1	94.99	75.94	74.89	74.75
5	98.09	91.01	90.43	90.34
10	99.01	95.35	95.08	95.06
20	99.39	97.11	96.99	96.98
30	99.43	97.32	97.14	97.13

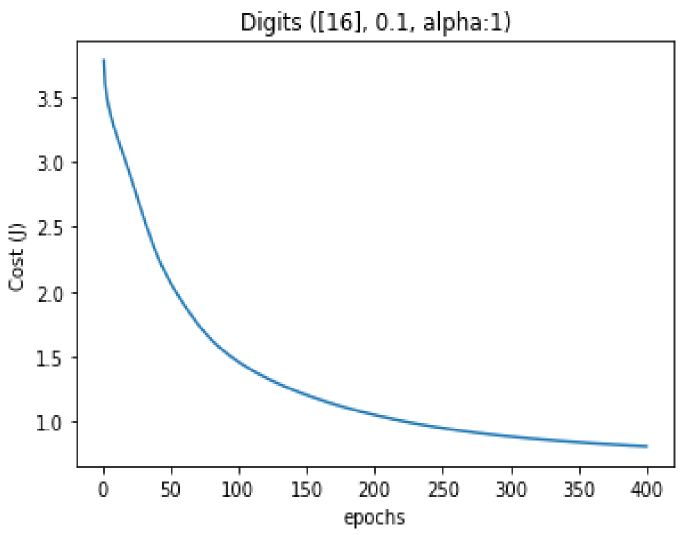
40	**99.56**	**97.93**	**97.83**	**97.84**
50	99.52	97.73	97.60	97.61

Best ntree = 40 (Acc = 99.56 , F1 = 97.84) Even though 40 gives best performance, for generalization probably 20 ntrees is better for novel data.

NEURAL NETWORKS -

→ alpha (learning rate) = 1.0

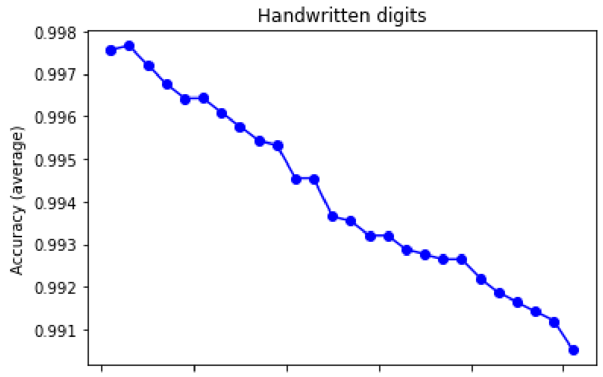
Lamda	Architecture	epochs	Accuracy	Precision	Recall	F1 Score
0.25	[64,8,10]	300	0.9678	0.8593	0.8390	0.8279
0.1	[64,8,10]	300	0.9713	0.8561	0.8552	0.8440
0.25	[64,16,10]	300	0.9844	0.9244	0.9218	0.9213
0.1	[64,16,10]	300	0.9872	0.9378	0.9356	0.9352
0.1	**[64,16,10]**	**400**	**0.989985**	**0.95235**	**0.9496**	**0.9495**
0.25	[64,8,8,10]	300	0.9683	0.8387	0.8404	0.8317
0.25	[64,16,8,10]	300	0.9743	0.8824	0.8706	0.8633
0.25	[64,16,16,10]	400	0.9872	0.9383	0.9359	0.9357
0.25	[64, 8, 8, 8,10]	400	0.9613	0.8085	0.8055	0.7891
0.25	[64, 16, 8, 8, 10]	400	0.9664	0.8434	0.8311	0.8234

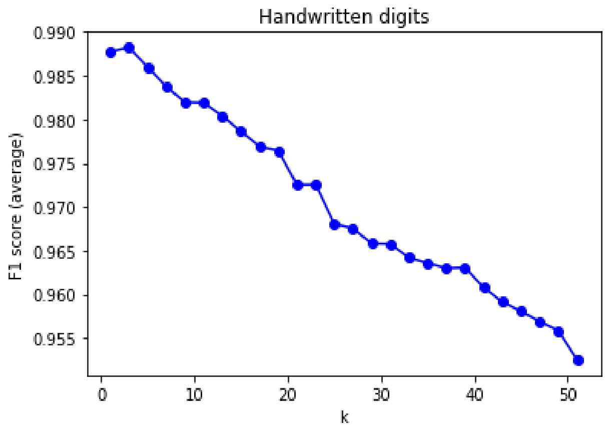


***** The best performance is given by the simplest architecture with a single hidden layer, and 16 hidden neurons which is comparable to sqrt(input size)=8. We have kept lambda 0.1 because we train the model on lesser epochs to reach good performance. [8] performed less compared to [16] because the number of neurons was not sufficient to encode the information of the input layer.***

Cost function(J) on test set. Each epoch here represents 1612 training samples. As the model trains through 400 epochs, cost (J) is decreasing.

K-NN :





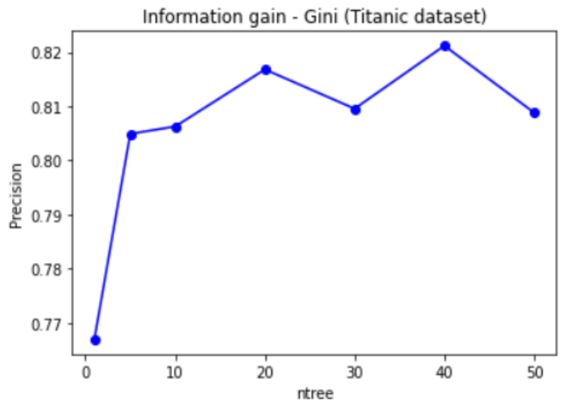
Handwritten digits dataset :

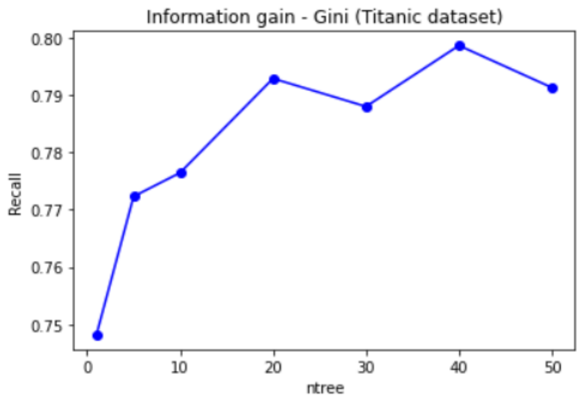
Highest Accuracy = 99.76576% (k = 3)
Highest F1 score = 98.82397% (k = 3)

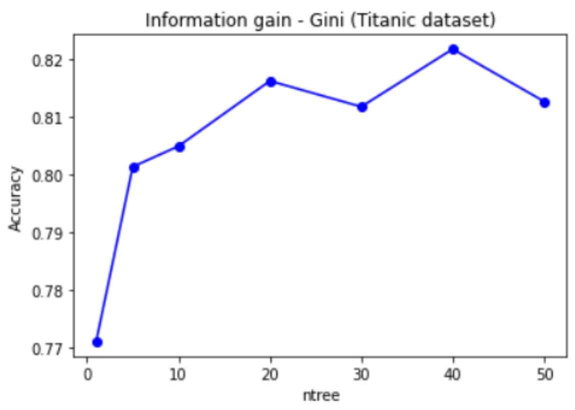
All digits can be written in various manners. So, intuity k > 1 (k = 3) makes sense to get better performance. As we increase k, we now consider numbers which are not correct also in the majority voting, so the performance decreases. kNN outperforms NNs and RFs probably due to the uniformity of the digits data. Else, the distance between two images may not properly signify "if both the digits are equal." - This depicts that as k increases, the decision boundary is smoother which means the variance is low but bias is increased.

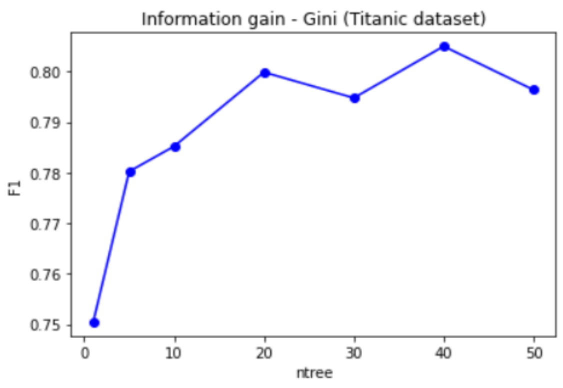
The Titanic Dataset :

RANDOM FORESTS -









Performance Results -

nTrees	Accuracy	Precision	Recall	F1 Score
1	77.12	76.69	74.81	75.04
5	80.14	80.49	77.23	78.02
10	80.49	80.62	77.64	78.52
20	81.63	81.67	79.29	79.98
30	81.18	80.95	78.80	79.47
40	**82.18**	**82.11**	**79.86**	**80.49**
50	81.27	80.88	79.13	79.63

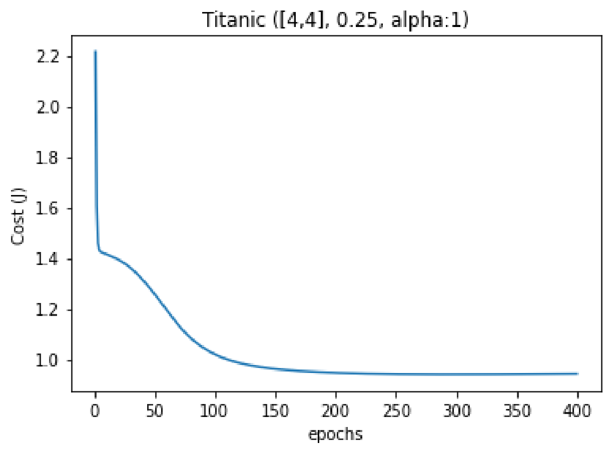
Best ntree = 40 (Acc = 82.18 , F1 = 80.49) **– accuracy, F1 score increased almost steadily as the number of trees are increasing, but with slight differences.**

NEURAL NETWORKS -

Alpha (learning rate) = 1.0

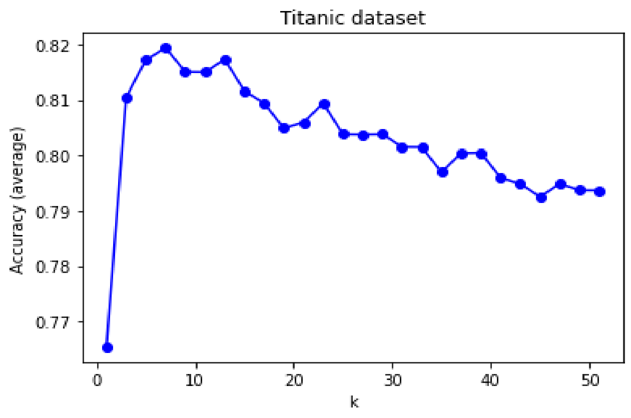
Lamda	Architecture	epochs	Accuracy	Precision	Recall	F1 Score
0.5	[6, 3, 2]	300	79.61	79.42	77.34	77.9
0.5	[6, 4, 2]	300	80.05	80.004	77.71	78.34
0.5	[6, 3, 3, 2]	300	79.93	80.22	76.895	77.647
0.5	**[6, 4, 4, 2]**	**300**	**80.737**	**80.971**	**78.147**	**78.888**
0.25	[6, 4, 2]	400	80.506	80.591	77.9579	78.671
0.25	**[6, 4, 4, 2]**	**400**	**80.95**	**81.2**	**78.32**	**79.11**
0.1	[6, 8, 4, 2]	300	80.504	80.821	77.794	78.572

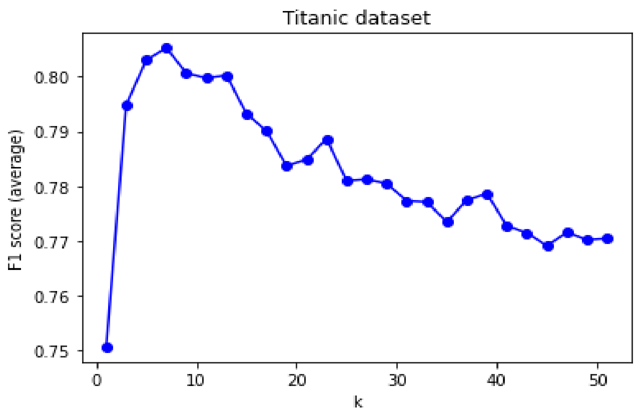
Best working model is [4,4]. Given that there are 6 input features, and not much difference between [4] & [4,4], we can consider even [4] architecture because it is even simpler. The smallest model [3] doesn't perform well compared to [4] probably because of fewer weight connections to capture data patterns.



Cost function(J) on test set. Each epoch here represents 797 training samples. As the model trains through 400 epochs, cost (J) is decreasing because the weights are learnt in the right direction

K-NN





Titanic Dataset

Highest accuracy & F1 score for k = 7 :

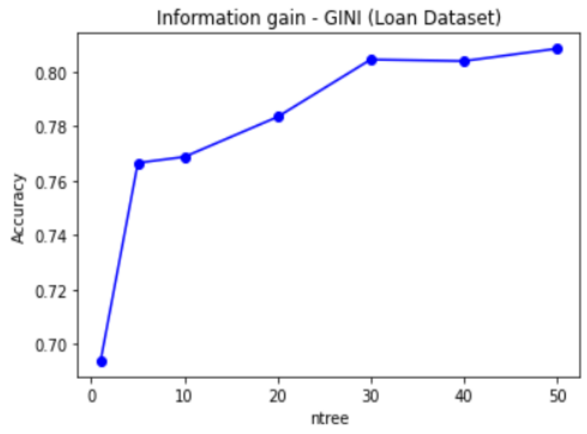
Accuracy (avg) = 81.94%
F1 score (avg) = 80.53%

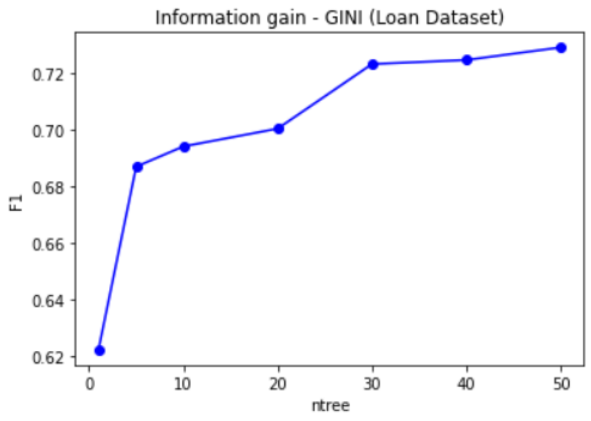
Titanic data has fewer features. But the classes in the dataset seem to be less separable, but not too complex that moderate k = 7 is giving best results.

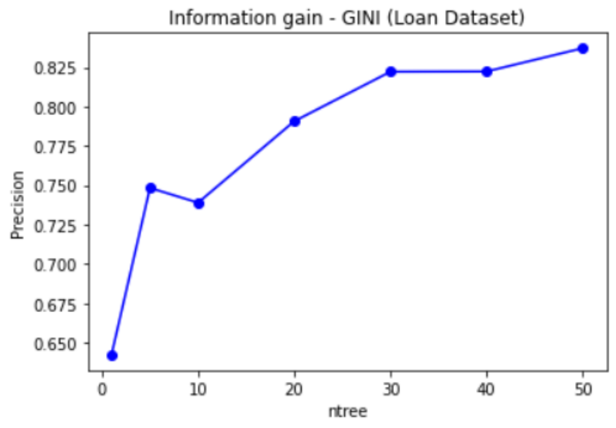
“We have calculated accuray, F1- score using k-NN algorithm by varying k from 1 to 51. We observed that they are at peak when k=7. As k value is increased further, both accuracy and F1 score has steadily decreased. This depicts that as k increases, the decision boundary is smoother which means the variance is low but bias is increased.”

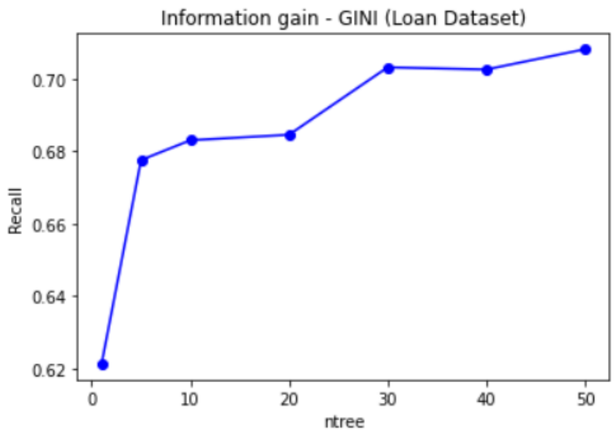
The Loan Eligibility Prediction Dataset :

RANDOM FORESTS -









Performance Results -

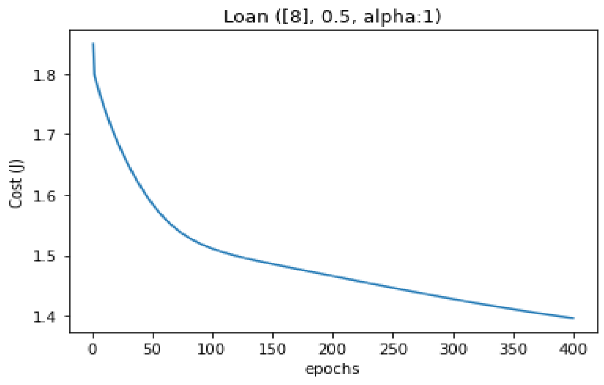
nTrees	Accuracy	Precision	Recall	F1 Score
1	69.39	64.22	62.12	62.24
5	76.65	74.82	67.76	68.71
10	76.87	73.88	68.31	69.43
20	78.34	79.06	68.46	70.06
30	80.45	82.21	70.32	72.34
40	80.39	82.28	70.26	72.48
50	**80.85**	**83.71**	**70.82**	**72.93**

Best ntree = 50 { Acc = 80.85 , F1 = 72.93 } But not much difference from ntree 30. Probably 30 ntree should suffice to generalize for this dataset, and novel data.

NEURAL NETWORKS -

Alpha (learning rate) = 1.0 or 1.5

Lamda	Architecture	epochs	Accuracy	Precision	Recall	F1 Score
0.25	[11, 5, 2]	300	80.639	85.066	70.149	72.231
0.25	[11, 8, 2]	300	80.426	84.757	69.9979	72.019
0.1	**[11, 8, 2]**	**400**	**80.852**	**85.427**	**70.301**	**72.45**
0.1	[11, 4, 4, 2]	400, 1.5	80.01	82.49	70.288	71.972
0.5	**[11, 8, 2]**	**400**	**80.852**	**85.427**	**70.301**	**72.45**
0.1	[11, 8, 8, 2]	400	80.431	84.205	69.9979	72.036
0.25	[11, 8, 4, 2]	300	80.222	84.134	69.851	71.835
0.25	[11, 8, 8, 4, 2]	300	80.435	84.089	69.997	72.077

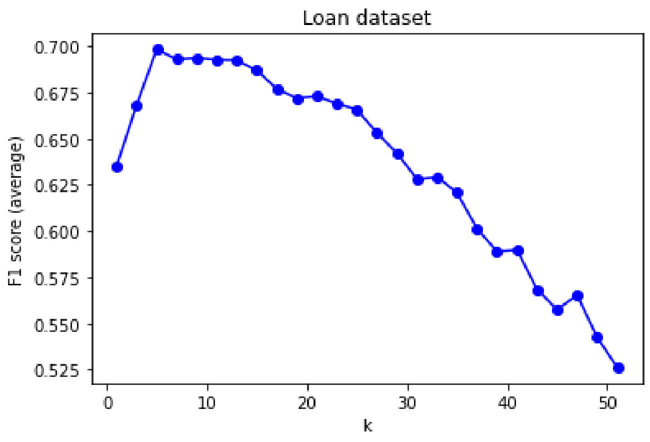


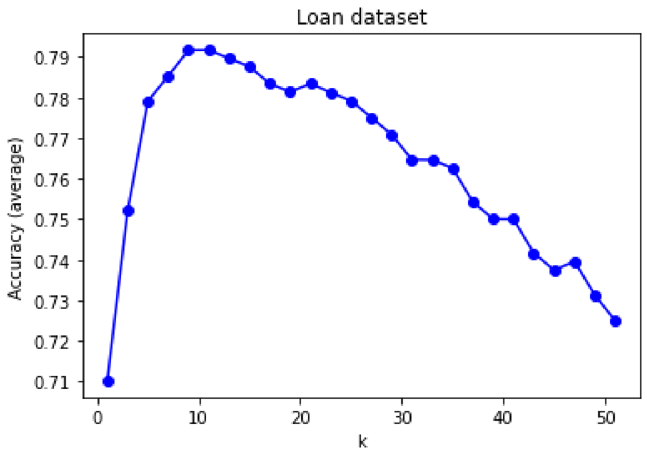
Best model is [8]. In real life, if the training data is lesser, probably [5] should also work well to capture data patterns.

Cost function(J) on test set. Each epoch here represents 431 training samples. As the model trains through 400 epochs, cost (J) is decreasing because the weights are learnt in the right direction.

K-NN

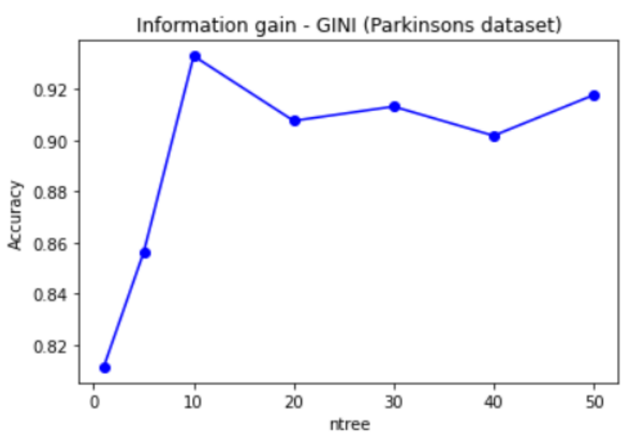
Highest accuracy : 79.17% (k = 11)
Highest F1 score : 69.83% (k = 5)
**** K = 9, 11 almost same performance**

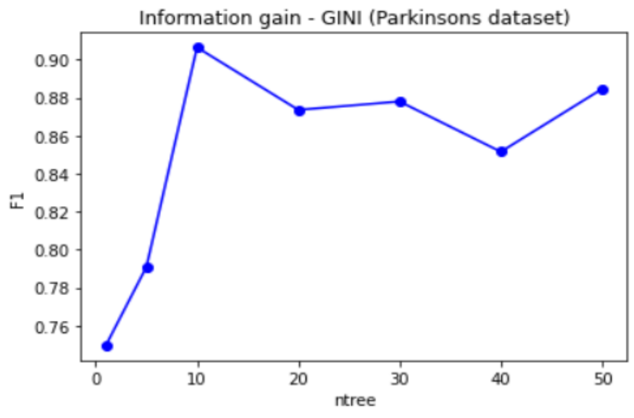


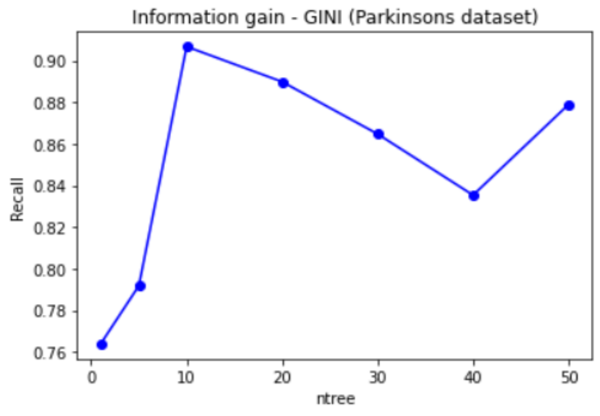


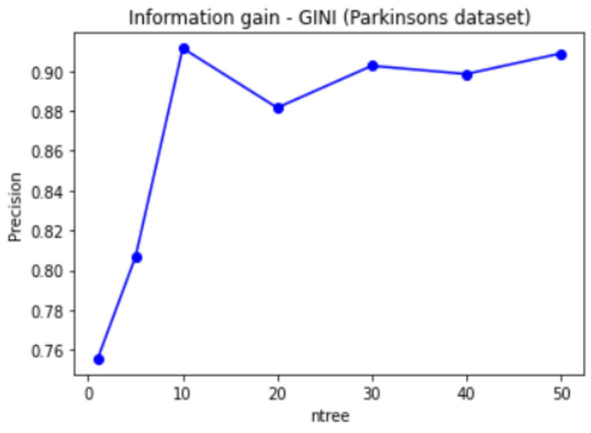
The Oxford Parkinson's Disease Detection Dataset :

1. *RANDOM FORESTS -*









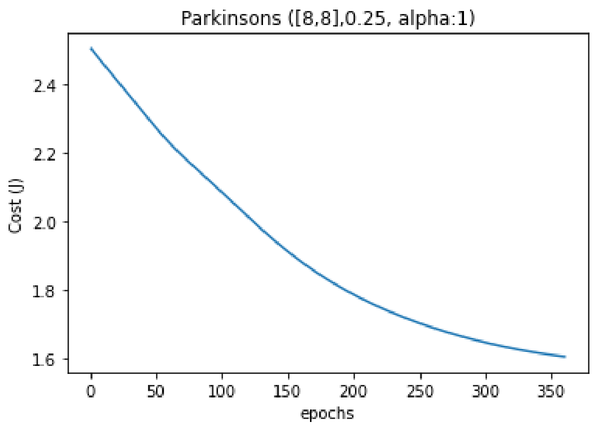
Performance Results -

nTrees	Accuracy	Precision	Recall	F1 Score
1	81.14	75.56	76.38	74.95
5	85.61	80.67	79.17	79.08
10	**93.28**	**91.13**	**90.67**	**90.65**
20	90.75	88.15	88.98	87.35
30	91.30	90.25	86.47	87.80
40	90.16	89.84	83.53	85.15
50	91.75	90.88	87.89	88.46

Best ntree = 10 : { Accuracy = 98.28, F1 score = 90.65 }

The value of ntree with best performance is optimal in real life when training data is moderate.

We have implemented the random forests algorithm and tested the dataset. We have varied the hyperparameter-nTree in random forests and we have captured the accuracies and F1 scores for the same. We have observed that the accuracy, F1 score has increased steadily and slightly dropped and increased again as the number of trees are increasing. As the number of trees are increasing, the model is training well but the learning rate will slow down.



The graph represents the Cost function(J) on the test set. Each epoch here represents 175 training samples. As the model trains through 400 epochs, cost(J) is decreasing.

NEURAL NETWORKS -

Alpha (learning rate) = 1.0

Lamda	Architecture	epochs	Accuracy	Precision	Recall	F1 Score
0.25	[22, 8, 2]	360	86.169	84.637	76.869	79.236
0.25	[22, 16, 2]	360	86.169	84.637	76.869	79.236
0.25	**[22, 8, 8, 2]**	**360**	**86.169**	**85.495**	**76.869**	**79.286**
0.25	[22, 16, 8, 2]	360	82.003	79.547	73.464	74.409
0.25	[22, 16, 8, 4, 2]	360	86.225	88.6958	76.226	78.306
0.25	[22, 16, 8, 4, 2]	400	84.725	86.104	74.333	76.421

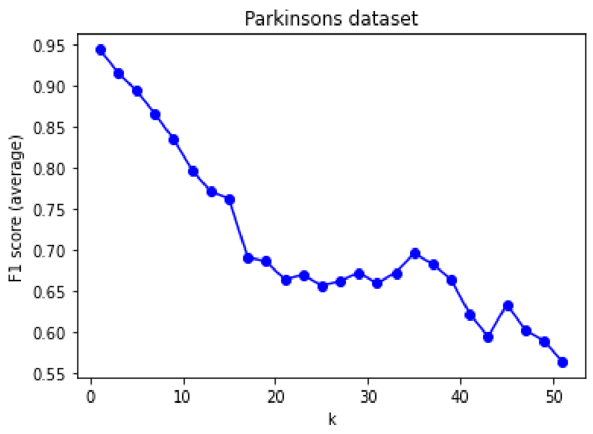
Best architecture :
Alpha = 1.0, Lambda = 0.25, Architecture = [22,8,8,2] =>
Accuracy = 86.169, F1 Score = 79.286

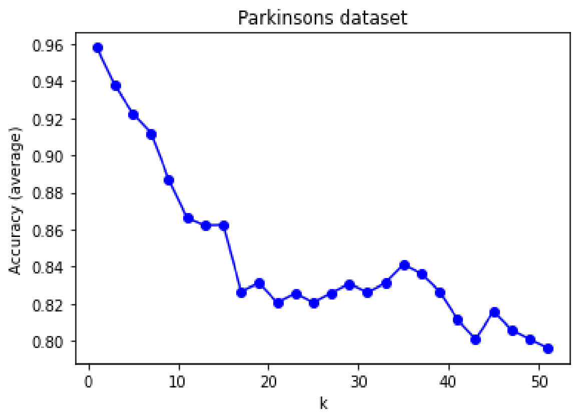
We have implemented the Neural networks algorithm and tested it by varying the hyperparameters, regularization parameter, learning rate, depth and width of the layers(number of neurons and layers respectively), number of epochs. We have observed the best performance for a simple network with 2 hidden layers(highlighted in the table above). Based on our experiments with various hyperparameters, we have observed that simpler networks with an appropriate selection of these parameters are performing well. And as the complexity increased, the performance of the model started to decrease.

K-NN

Parkinsons dataset
Highest accuracy & F1 score for k = 1 :

Accuracy = 95.8%
F1 score = 94.42%





We have calculated accuracies, F1- score using k-NN algorithm by varying k from 1 to 51. We have observed that the accuracies and F1-score are at peak when k=1 and it almost steadily decreases as k is increased. As the k values are increased, the decision boundary is smoother which means the variance is low but bias is increased. For generalization k = 1 may not work well, so moderate k should be chosen with better F1 score than just accuracy for this dataset.

Q1.

For all the datasets, we have studied RFs, NNs, kNN.

- **We have deleted “LoanID”, “Name” columns in loan & titanic datasets. This is because they are mostly unique, and do not contribute to any class patterns.**
- We have decided to test k-NN because there is no training involved, and for smaller values of k, kNN learns complex decision boundaries.
- We have decided to test Neural Networks because of its ability to learn complex data patterns ranging from low-level features to high-level features as we increase the number of hidden layers from 1 to 2 to 3 etc. This is very appropriate for image recognition – “handwritten digits”
- We have decided to study Random Forests because of their low variance property, and robustness associated with the aggregated decision coming from individual DTs.

- **We have taken minimum_gain = 0.1 in RFs for entropy. And for gini, we compare the best feature's gini value >= (1 - (1/classes)) – to identify if the split will result in the most non-homogenous distribution and avoid it to generate deeper trees.**
- **We have used "Spurthi Tallam"s implementation for all the algorithms. Code modifications have been done by both of us (Susmita & Spurthi) for data preprocessing, calculating additional performance parameters for this project.**

Overall Results

	Digits		Titanic		Loan		Parkinson	
	Accuracy	**F1**	**Accuracy**	**F1**	**Accuracy**	**F1**	**Accuracy**	**F1**
Neural Network	98.99	94.95	80.95	79.11	80.852	72.45	86.17	79.29
Random Forests	99.56	97.84	82.18	80.49	80.85	72.93	91.75	88.46
k-NN	99.76	98.82	81.94	80.53	79.17	69.83	95.8	94.42

Observations :

1. **KNN for small values of k learns very complex decision boundaries**. And this ability is evident from the above results for all the datasets. It outperformed NNs and RFs. This can be due to the uniformity of the datasets. That is, for example, in the digits dataset, if the pixel values are very noisy or shifted, then direct pixel-pixel distance comparison may not generalize well for novel dataset. Additionally, with the number of features increasing, computational time increases due to the curse of dimensionality for k-NN.

2. **Best k for each dataset :**

a. Digits	=	3
b. Titanic	=	7
c. Loan	=	9, 11
d. Parkinsons	=	1 (drops to good extent for k > 1)

3. In the entire process, **we noticed that when the same number of hidden layer models are compared, having more neurons in the initial layers is helpful**. ([4] & [8]), ([3,3] & [8,4]). And having less number of hidden layers trains well compared to complex models with deep architectures in comparable epochs.

4. Higher ntrees performed well almost for all datasets in the project. However, to generalize well, since the difference in their performances is very less, a moderate number of trees (5 to 20) can be chosen in real life if we have to. If training data is huge, and the number of classes are huge, probably higher ntrees will capture better. Further training of many trees will be time consuming, and inference time is also higher.

5. In the project, for all the datasets, the single layer neural networks performed well. This is because of their extreme ability to learn from data. However, if there are not sufficient numbers of neurons in the single layer, performance will be worse because of its inability to capture all patterns.

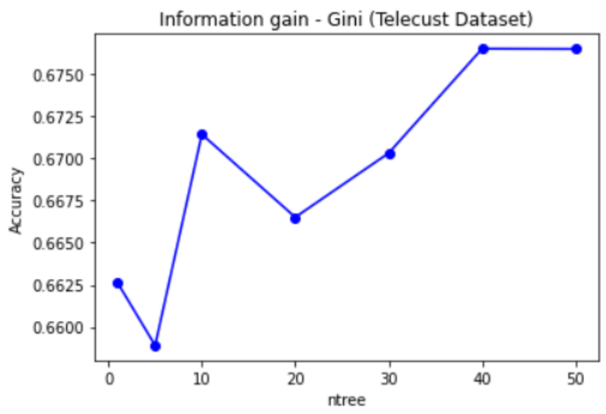
6. Although k = 1 performed well for parkinsons, it will not generalize well for novel data. This is because only 1 nearest neighbor is considered for classification. So, a moderate value of k should be chosen in real life to generalize well.

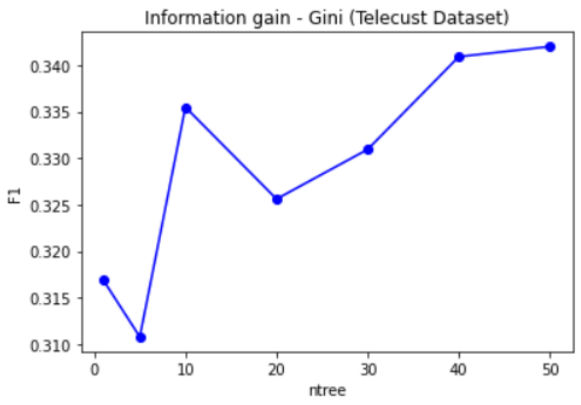
7. Although kNN worked well for digits, it may not work for image data with shifted objects etc since it is L2 distance. So, data preprocessing should be heavily performed before computing distances.
8. Multiple models can work well for a given dataset. But training process, generalization, hyperparameters, inference time should be kept in mind before finalizing the model.
9. **All the accuracies, f1 scores in this report are the average of all the classes values**.

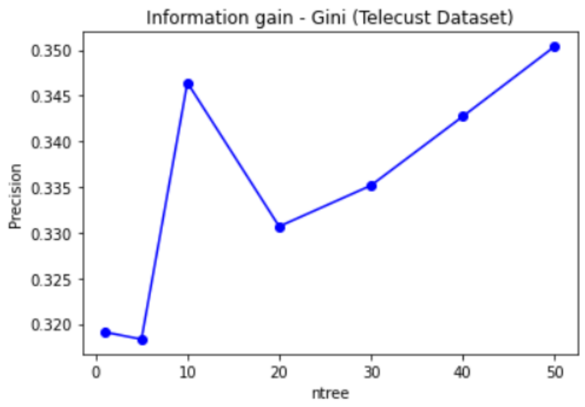
EXTRA CREDIT :

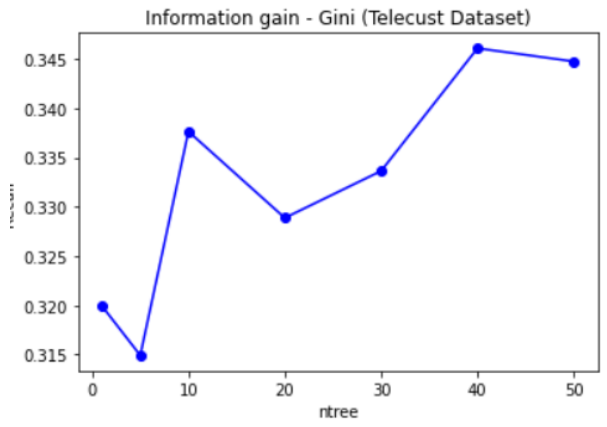
- Selected a new challenging dataset and evaluated the performance of different algorithms on it. The dataset has both categorical and numerical attributes and has more than two classes.

RANDOM FORESTS -









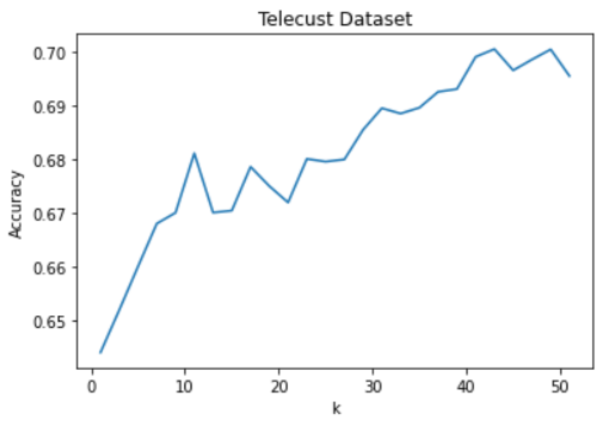
Performance Results for Gini -

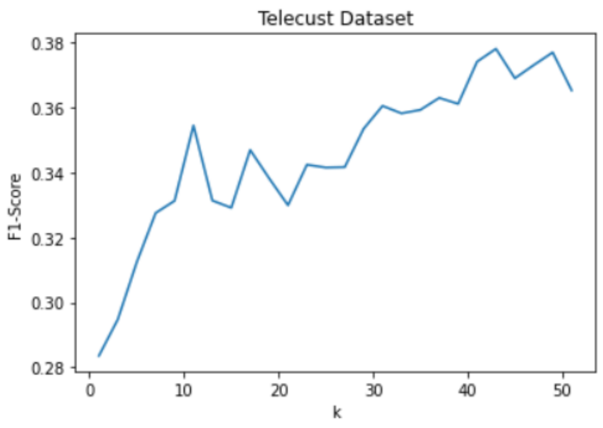
nTrees	Accuracy	Precision	Recall	F1 Score
1	66.26	31.93	31.99	31.68
5	65.88	31.83	31.49	31.07
10	67.14	34.64	33.76	33.54
20	66.65	33.06	32.88	32.56
30	67.03	33.51	33.36	33.09
40	67.65	34.27	34.60	34.09
50	67.65	35.03	34.47	34.20

Best Hyperparameter = 50 : (Accuracy = 67.65, F1 score = 34.20)

We have implemented the random forests algorithm and tested the Telecust dataset. When we have varied the hyperparameter-nTree in random forests, we have observed that the accuracy, F1 score has increased slightly as the number of trees are increasing. But the accuracies have not increased after a certain point.

K-NN -





Best performance is seen for k = 45.