Exploring Machine Learning Model Performance across Diverse Datasets: A Comparative Analysis

Yan Mazheika Polina Petrova

May 9, 2024

Introduction

This report provides an analysis of the application of various machine learning projects on several datasets. The goal is to identify the best-use scenarios of model variants given the nature of input data. In this report, the authors analyze the

- Decision Tree, *provided by Polina Petrova*
- k Nearest Neighbors, *provided by Polina Petrova*
- Neural Network, *provided by Yan Mazheika*
- Random Forrest, *provided by Yan Mazheika*

against the handwritten digits, titanic survival, loan eligibility, and Parkinson's classification datasets.

It is our aim to explain the nature of the data, our models, and the performance of these classifiers. Throughout our report, we justify our algorithm choice for the given dataset and our choice of hyper-parameters for the tuning of the algorithm. These insights are for the reader's benefit; we also aim to provide insight into how to solve novel machine problems, which algorithms may work best, and how to adjust their hyper-parameters for optimal performance.

Approach

The authors have chosen to coordinate on every dataset. For the first two algorithms of every dataset, one of them comes from *Yan Mazheika* while the other comes from *Polina Petrova*. These algorithms may have been modified to handle a new type of data but their fundamental logic stays consistent from previous use cases.

We use cross-validation of k=10 folds when testing the performance of our models (except for kNN). We also transform the data into the [0,1] range when using the neural network or kNN classifiers.

Performance Overview

Dataset:	Digits		Titanic		Loan		Parkinson's	
	Accuracy	F1-Score	Accuracy	F1-Score	Accuracy	F1-Score	Accuracy	F1-Score
k-NN	.8698	.9298	-	-	-	-	?	?
Decision Tree	-	-	.7852	.8793	.8083	.8930	-	-
Random Forrest	-	-	-	-	.7354	.6650	-	-
Neural Network	.9962	.9809	.9663	.9652	-	-	.9226	.8940
Additional Algorithm	-	-	-	-	-	-	-	-

Digits Dataset

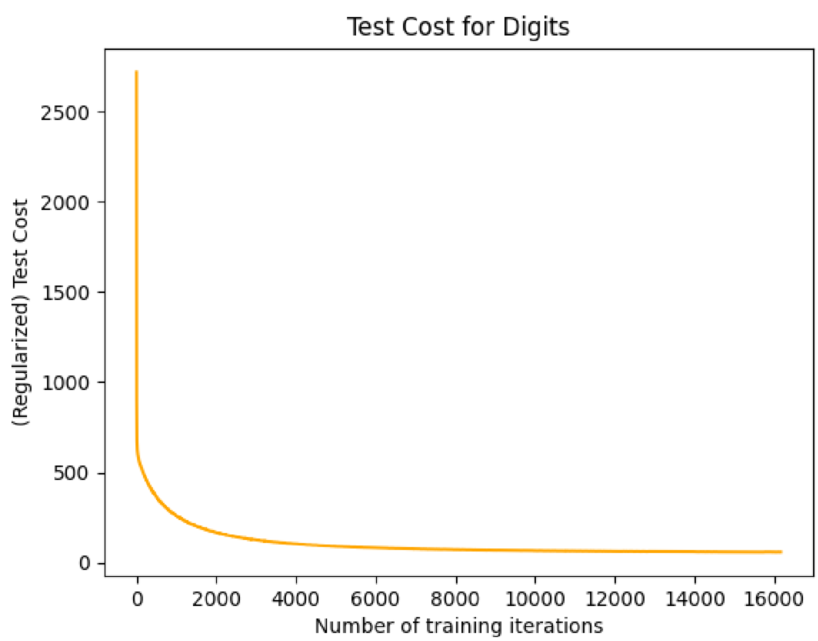
The digits dataset come's from the sci-kit learn library, available in Python. We have chosen to analyze the performance of a neural network classifier and the k-Nearest Neighbors classifier. Again, we mention that the neural network tuning was performed by *Yan Mazheika* while the kNN tuning was done by *Polina Petrova*.

We chose a neural network as one of our algorithms for this dataset because of the algorithm's ability to handle complex patterns and identify non-linear patterns in these data. Anecdotally, neural networks have shown impressive results in image classification tasks, making them a top contender for handwritten digit recognition, the focus of this dataset.

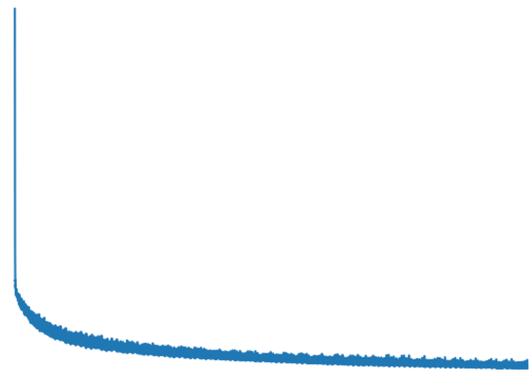
On the other hand, the k-NN algorithm is a simple yet effective classifier that makes predictions based on the closeness of past training instances in the feature space. We figured that similar pixel activations in the handwritten digits would translate into less distance in the 64-dimensional feature space. A concern we had when choosing this algorithm is precision loss; kNN performs poorly with a large number of features, which is 64 in our case. The distance measurement between two instances converges to zero as we add more features and normalize them. This (has/hasn't) proved to be a problem.

Performance Metrics

`Neural Network`

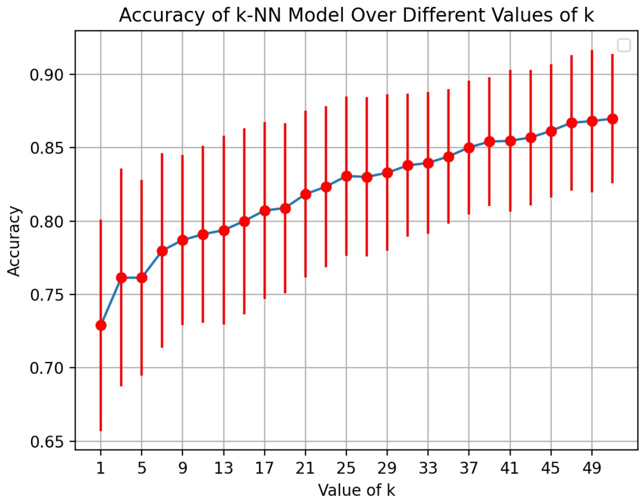


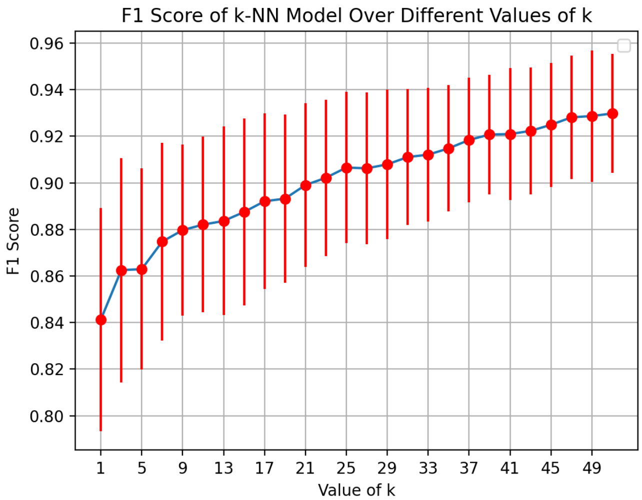
	Neural Network
Learning Rate α	0.0500
Regularization λ	0.0100
Architecture	[64, 32, 32, 10]
Mean Accuracy	0.9962
Mean F1-score	0.9809
Mean Test Cost	59.3100



Test cost over 179,700 instances

k-NN Algorithm





	kNN
Neighbors k	51
Mean Accuracy	0.8698
Mean F1-score	0.9298

Metrics for Preferred Model Variant

k	Test Accuracy	Test F-Score
1	0.7291	0.8413
3	0.7615	0.8625
5	0.7615	0.8629
7	0.7799	0.8748
9	0.7872	0.8797
11	0.7911	0.8821
13	0.7939	0.8836
15	0.8000	0.8875
17	0.8073	0.8921
19	0.8089	0.8932
21	0.8184	0.8991
23	0.8235	0.9022
25	0.8307	0.9066
27	0.8302	0.9062
29	0.8330	0.9079
31	0.8380	0.9111
33	0.8397	0.9121
35	0.8441	0.9148
37	0.8503	0.9184
39	0.8542	0.9208
41	0.8547	0.9209
43	0.8570	0.9223
45	0.8615	0.9249
47	0.8670	0.9281
49	0.8682	0.9287
51	0.8698	0.9298

Performance Metrics over various k

Tuning Hyper-parameters

For the neural network, we've chosen an α of 0.05 and a regularization constant of $\lambda = 0.01$. At first, with a $\lambda = 0$ and $\alpha = 0.50$, the model had good performance but a quick convergence in the training curve. We found increased performance and a slower convergence by decreasing α and increasing λ to a final accuracy of 99.62%. This indicates a misclassification of 6 total instances out of 1,797 total. The architecture was chosen because it had the lowest accuracy variance between each fold out of the models tested.

In our implementation of the k-NN algorithm, we tested different values of k in the range $[1, 52)$ for an ideal hyperparameter. We observed an initial jump in the evaluation metrics between $k = 1$ and $k = 2$ and a high variance. This points to the algorithm being sensitive to outliers at very small values of k. As the value of k increases, the values of the evaluation metrics increase, as well, and stabilise by $k = 51$ at 86.98% for accuracy and 92.98% for F1-score. Increasing this hyperparameter value also led to smaller variance, which is helpful when dealing with noisy datasets. In witnessing this convergence, we have chosen a final hyperparameter value for the k-NN algorithm to be $k = 51$. With a larger k value and smoother decision boundary, we mitigate the chance of overfitting our model to the data.

Analysis

Reporting on the final runs of the neural network and k-NN algorithms on the digits dataset, both models performed well on the testing instances. However, testing on the neural network resulted in higher accuracy and F1-score values as opposed to when testing on k-NN. Dealing with an 8x8 pixel image, k-NN is able to effectively measure distances in the feature space when making predictions. Although k-NN performs well on this dataset, it struggles in distinguishing between two or more digits that look alike when handwritten. For example, an analysis of its predictions made on an instance of class 0 may return a substantial number of predictions of class 9, which would be a factor in decreasing accuracy.

The neural network acts as a well-defined alternative to this issue, as it captures more complex patterns and data variability present in handwriting. Additionally, a higher neuron count fascilitates learning these handwriting variations of numbers of the same class, making the network more reliable in predicting instances that look alike and a higher accuracy.

Titanic Dataset

We pre-processed some of the titanic dataset. At first, there was a 'Sex' column which was encoded into a 'Male' column indicating a 1 if the 'Sex' of a given instance contained 'male' and a 0 if an instance contained 'female'. There was also a 'Name' column which was removed entirely. At first, we attempted to process this by encoding every English letter into a number. This resulted in a very large data-frame that led to poor results after processing.

Afterwards, we tried to one-hot encode the first word of the 'Name' column which indicated the title of a person. Because this dataset represents a time when a person's title had greater significance, we figured that there would be differences in the survival rates between prestigious and non-prestigious titles (such as 'Rev' or 'Master' versus 'Mr' or 'Miss'). We also hypothesized that persons with the title 'Rev,' which indicates religious affiliation, would be less likely to survive as we thought that their role as a religious figure would influence them to stay on the ship for longer than others. Indeed, every reverend in the dataset did not survive.

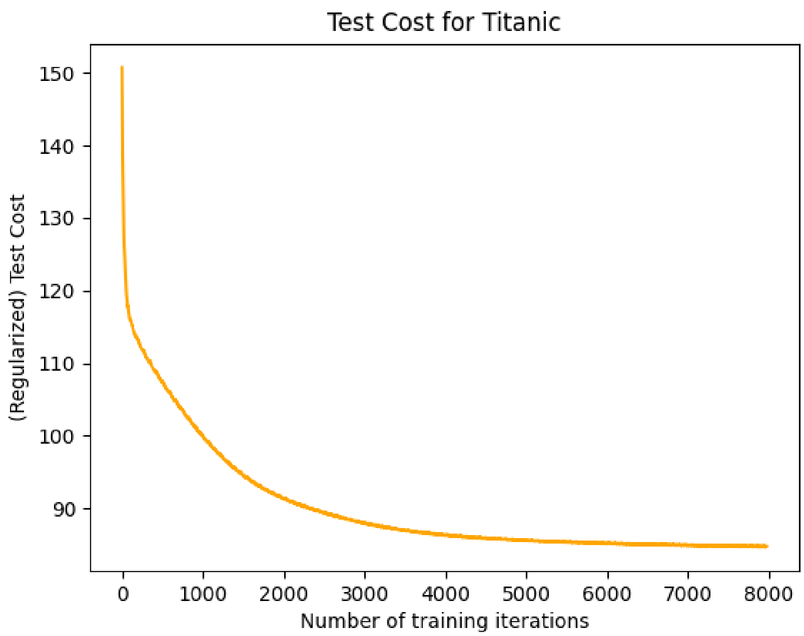
Empirically, we found that this didn't work well, which led to the removal of the 'Name' feature in the dataset. Perhaps this is because 'Pclass' contains the ticket class of a given passenger, which mitigates the role that the 'Name' column plays in indicating the socio-economic status of a passenger. Additionally, the added dimensionality of encoding around 8 titles could have mitigated the importance of the un-altered columns.

The choice of applying a neural network for this task can be justified by it's performance (speed) and ability to learn complex patterns. It's arguably the easiest model to tune with respect to the datasets this paper presents.

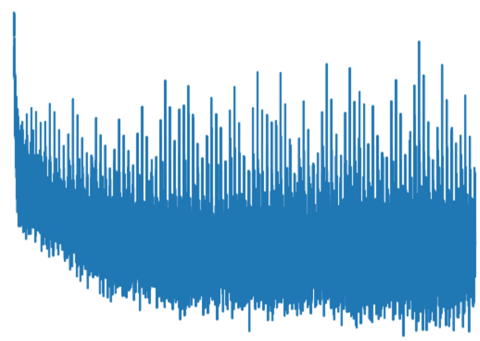
The choice of applying a decision tree for this task is natural as well. There are a number of more important features that a high-bias model can benefit from. These include passenger class, ticket fare, and gender. With three main features impacting survivability, we can benefit from a decision tree by quickly sub-setting the dataset on those features.

Performance Metrics

`Neural Network`

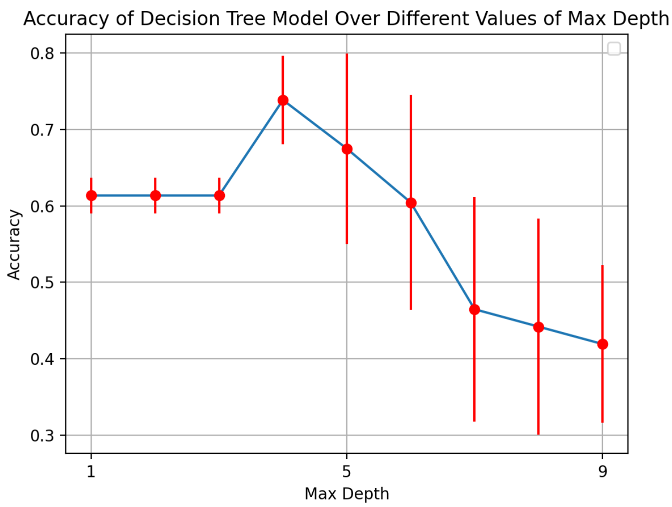


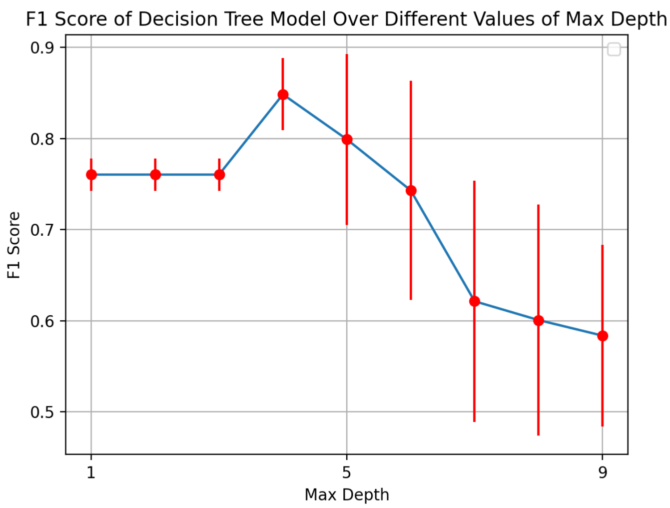
	Neural Network
Learning Rate α	0.0400
Regularization λ	0.0900
Architecture	[6, 4, 2]
Mean Accuracy	0.9663
Mean F1-score	0.9652
Mean Test Cost	84.8500



Test cost over 87,700 instances

Decision Tree





Maximum depth	Test Accuracy	Test F-score
1	0.7852	0.8793
2	0.6386	0.7774
3	0.5898	0.7372
4	0.7125	0.8291
5	0.6784	0.7920
6	0.6295	0.7540
7	0.5227	0.6619
8	0.4216	0.5802
9	0.4216	0.5802

	Decision Tree
Maximum Depth	4
Mean Accuracy	0.7023
Mean F1-score	0.8234

Metrics for Preferred Model Variant

Tuning Hyper-parameters

...
Conducting multiple analyses of the decision tree algorithm on the titanic dataset with differing maximum tree depths, we have determined the preferred hyperparameter to be $max\_depth = 4$. At this value, accuracy and F1-score reach a peak at 70.23% and 82.34%, respectively, after which both metrics decrease substantially. Deeper trees lead to overfitting and decrease in generalisation ability, which is evidenced by the sharp increace in variance on both accuracy and F-1 score after tree depth 4.

Analysis

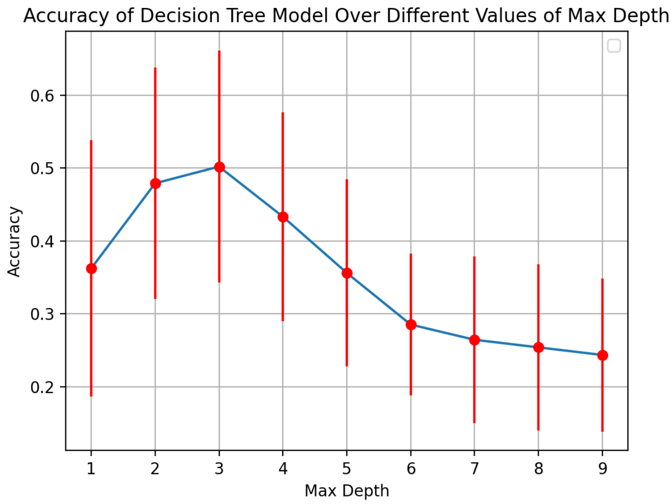
Our decision tree struggled in providing reliable predictions on the titanic dataset. This is possibly due to the fact that the dataset contained numerical attributes such as $Ticket\ Fare$ and Age that may be difficult to evaluate on this model. The decision nodes were split based on a numerical threshold that maximised information gain at each decision point. This approach is particularly challenging given the large range of numerical values present in the data. For example, there were passengers as old as 80 years-old on the ship, along with infants as young as 5 months-old. Similarly, the ticket fare values ranged from 0.0 to 512, showing a wide distribution of values. Although we employed feature normalisation, this variability in numerical attributes could have contributed to the difficulty in finding optimal splits during the tree construction process. Additionally, deeper trees tend to create more decision nodes based on numerical thresholds, which may struggle to effectively differentiate between instances with similar normalised values but different original scales.

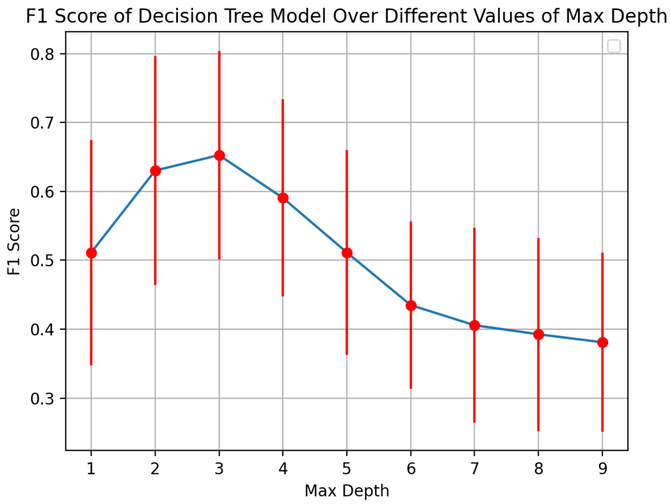
On the other hand, employing a neural network on this dataset proved to be substantially more accurate in predicting instance classes.

Loan Eligibility Dataset

Performance Metrics

Decision Tree



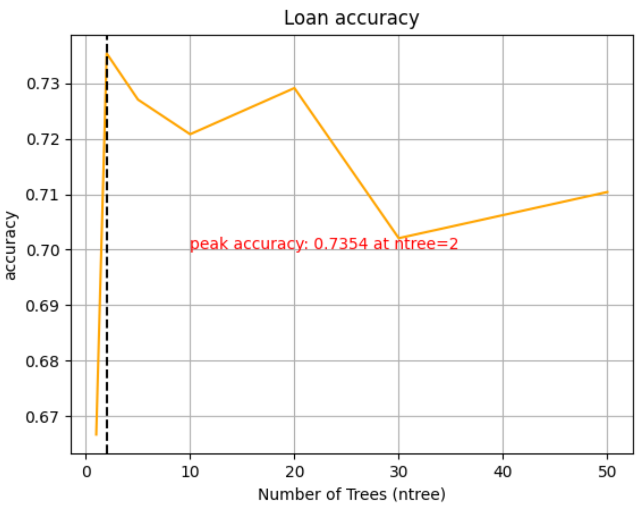


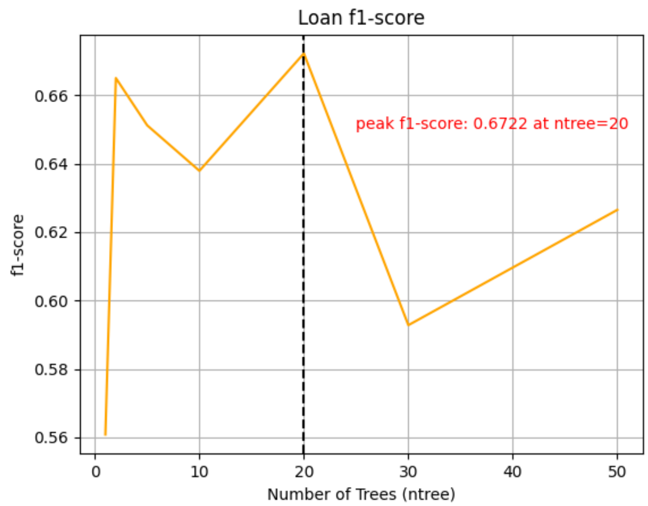
	Decision Tree
Maximum Depth	k
Mean Accuracy	?
Mean F1-score	?

Metrics for Preferred Model Variant

Maximum depth	Test Accuracy	Test F-score
1	0.8083	0.8930
2	0.6458	0.7657
3	0.4708	0.6357
4	0.3583	0.5139
5	0.3104	0.4606
6	0.3042	0.4609
11	0.2437	0.3894
16	0.2437	0.3894
21	0.2437	0.3894
26	0.2437	0.3894
31	0.2437	0.3894
36	0.2437	0.3894
41	0.2437	0.3894
46	0.2437	0.3894
51	0.2437	0.3894
56	0.2437	0.3894
61	0.2437	0.3894
66	0.2437	0.3894
71	0.2437	0.3894
76	0.2437	0.3894
81	0.2437	0.3894
86	0.2437	0.3894
91	0.2437	0.3894
96	0.2437	0.3894

Random Forrest





accuracy	precision	recall	f1-score	ntree
0.6667	0.5729	0.5493	0.5609	1
0.7354	0.6875	0.6440	0.6650	2
0.7271	0.6755	0.6286	0.6512	5
0.7208	0.7436	0.5585	0.6379	10
0.7292	0.8348	0.5627	0.6722	20
0.7021	0.6756	0.5281	0.5928	30
0.7104	0.7538	0.5360	0.6265	50

Performance metrics across various n-tree

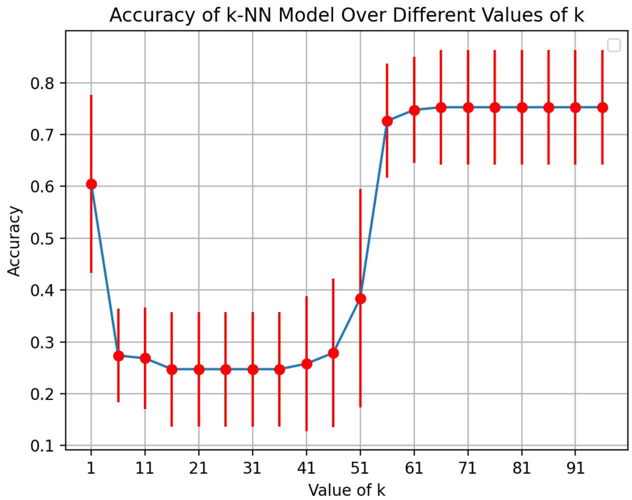
Tuning Hyper-parameters

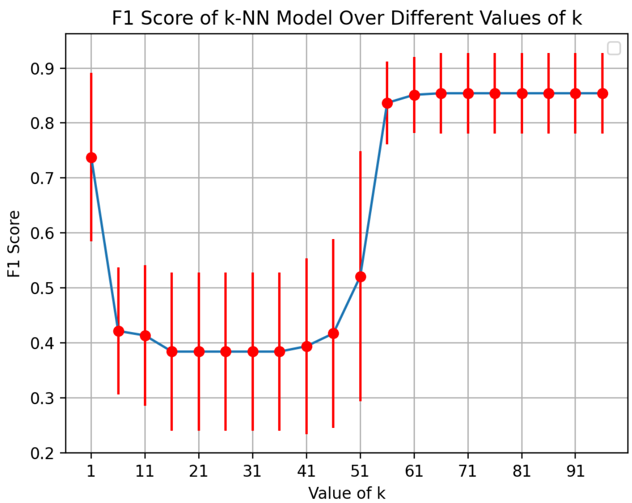
Analysis

Oxford Parkinson's Disease Dataset

Performance Metrics

k-NN Algorithm



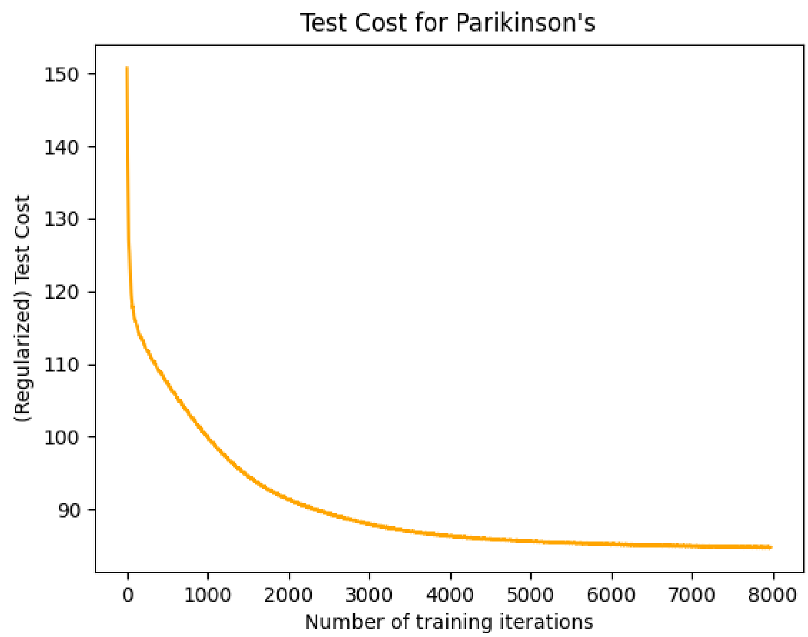


k	Test Accuracy	Test F-Score
1	0.5947	0.7367
6	0.2789	0.4319
11	0.2789	0.4319
16	0.2474	0.3921
21	0.2474	0.3893
26	0.2474	0.3893
31	0.2474	0.3893
36	0.2474	0.3893
41	0.2579	0.4030
46	0.4211	0.5625
51	0.6000	0.7320
56	0.7211	0.8350
61	0.7526	0.8561
66	0.7526	0.8561
71	0.7526	0.8561
76	0.7526	0.8561
81	0.7526	0.8561
86	0.7526	0.8561
91	0.7526	0.8561
96	0.7526	0.8561
101	0.7526	0.8561

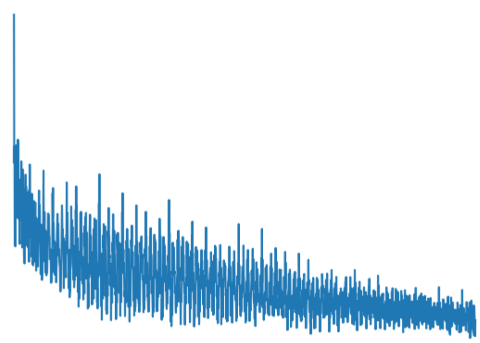
	kNN
Neighbors k	k
Mean Accuracy	?
Mean F1-score	?

Metrics for Preferred Model Variant

Neural Network



	Neural Network
Learning Rate α	0.1000
Regularization λ	0.0001
Architecture	[22, 12, 2]
Mean Accuracy	0.9226
Mean F1-score	0.8940
Mean Test Cost	13.4800



Training Cost Over 19,500 Instances

Tuning Hyper-parameters

.7538 accuracy of random classifier.

Analysis