Final Project

CS 589: Machine Learning
May 10, 2024

Aaron Tian
atian@umass.edu

Mihir Thanlanki
mthalanki@umass.edu

1 The Hand-Written Digits Recognition Dataset

1.1 Model Selection

We will evaluate the **k-NN**, **random forest** and **neural network** models on this dataset.

1. k-NN is a natural candidate for the problem as we expect same digits to be visually similar. If visual similarity of two images corresponds to "closeness" in their flattened vector representations and vice versa, then k-NN will perform very well (in general this is **not** the case, especially for large images, but we suspect that my statement is sound for the small images in this dataset).
2. Random forests are good choice for high dimensional data. The samples in this dataset are given as 8×8 images, i.e. 64 features in each instance. Random forests are capable of modeling non-linear relationships in these high-dimensional datasets to find intricate patterns that may arise.
3. Neural networks are another candidate for the problem as they are, in general, good black-box models for multi-class classification problems.

1.2 Model Tuning

For each of the selected models, we systematically evaluate the dataset on selected hyperparameters and identify the best hyperparameters to use on the dataset.

1.2.1 k-NN

k	`acc`	`f1`
1	0.988	0.988
5	0.989	0.989
10	0.983	0.984
20	0.976	0.976
30	0.967	0.967
40	0.966	0.966
50	0.956	0.957

Table 1: k-NN model performance on digits dataset for different number of neighbors.

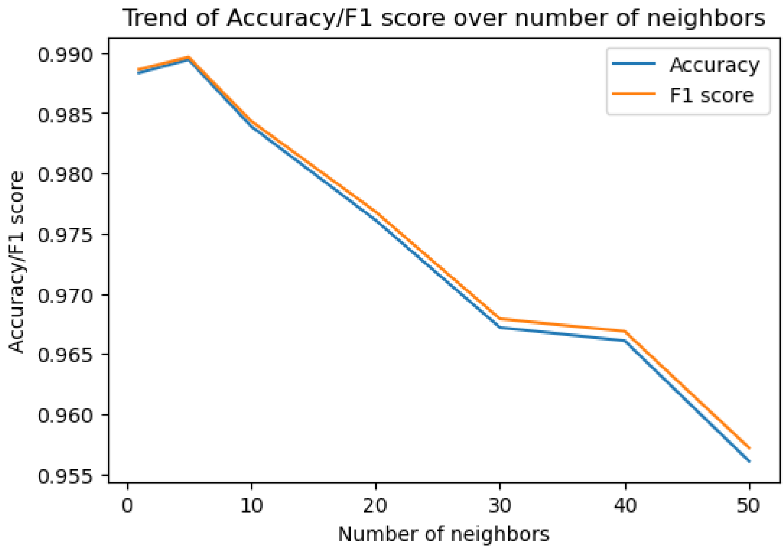

Figure 1: k-nn accuracy and f1 score vs. different values of k. It is observed that as the number of neighbors increases, the performance degrades

Based on the results above, we identify $k = 5$ as the best hyperparameter for the k-nn model.

1.2.2 Random Forest

ntrees	max-depth	acc	f1
1	10	0.376	0.376
5	10	0.527	0.524
10	10	0.669	0.665
20	10	0.772	0.771
30	10	0.811	0.811
40	10	0.836	0.838
50	10	0.852	0.856
50	20	0.855	0.858
50	30	0.847	0.850
50	40	0.845	0.846
50	50	0.847	0.850

Table 2: random forest model performance on digits dataset for different values of ntrees and max depth.

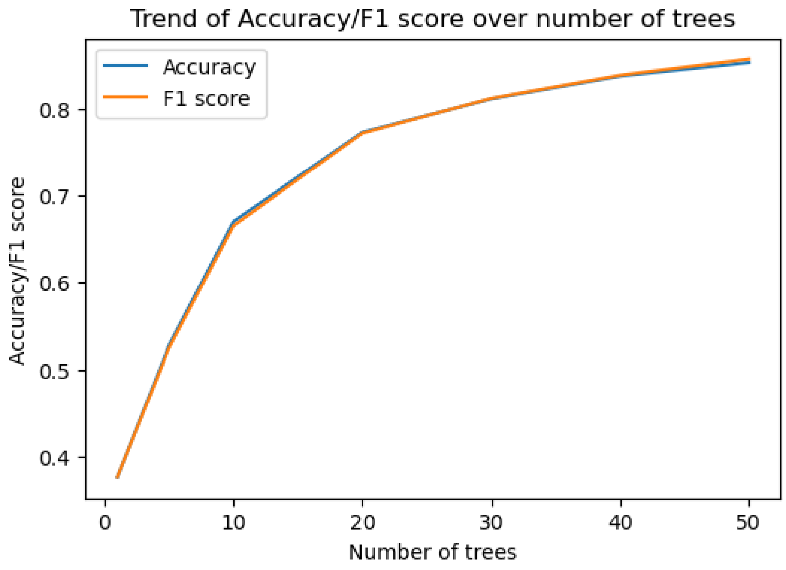

Figure 2: random forest accuracy and f1 score vs. different values of ntrees with $max\_depth = 10$. Performance increases as we increase the number of trees. However the rate of increase plateaus. This is expected behavior

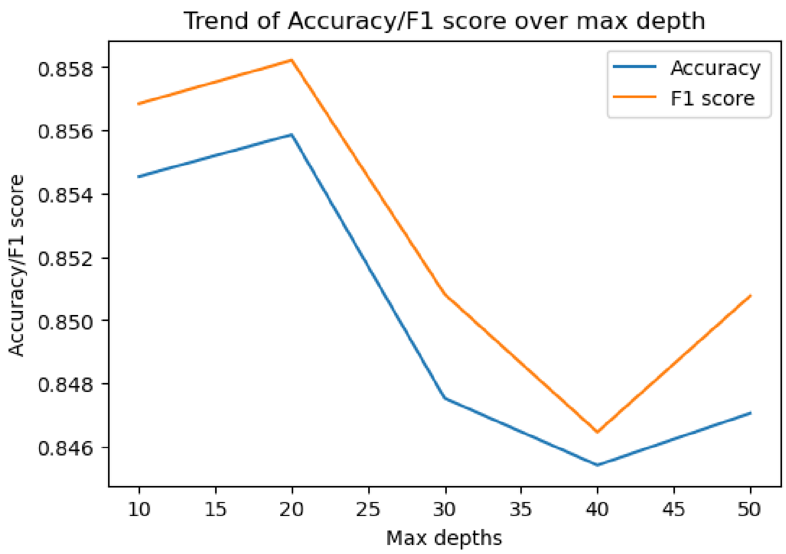

Figure 3: random forest accuracy and f1 score vs. different values of max_depth for ntrees=50. As we increase the depth, after $depth = 20$, the performance degrades. This is possibly because of overfitting.

The random forest model worked best with hyperparameters: `ntrees` = 50 and `max-depth` = 20.

1.2.3 Neural Network

The neural network model is tuned for different values of `num-layers`, `num-neurons`, and the regularization parameter λ.

num-layers	num-neurons	λ	acc	f1
1	64	0.001	0.979	0.979
1	128	0.001	0.984	0.984
1	128	0.01	0.982	0.982
2	128	0.01	0.984	0.984

Table 3: Neural network model performance on digits dataset. Mini-batch gradient descent with 32 batches and learning rate $\alpha = 0.5$ was used to train each model.

and the model with 1 hidden layer, 128 neurons per layer, and $\lambda = 0.001$ as the best neural network architecture.

1.3 Model Evaluation

1.3.1 Neural Network

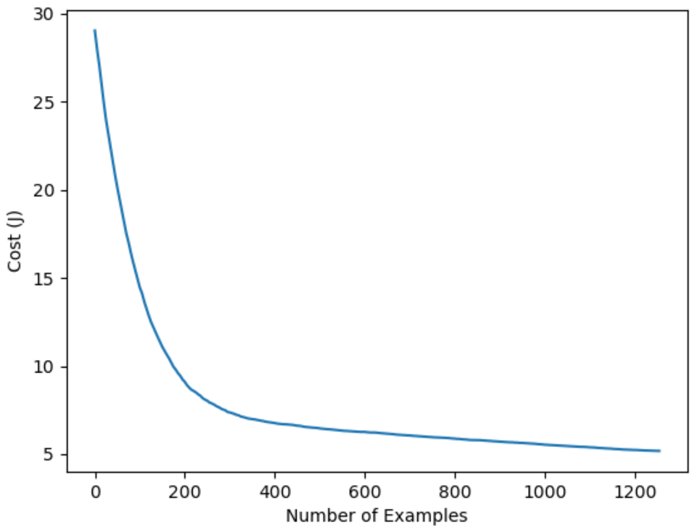

Figure 4: Neural network cost on test set vs. number of training examples seen. A learning rate of $\alpha = 0.004$ was used and weights were updated after every 5 examples.

We find that the average cost of the model against the test set trends downwards with the number of examples seen. This is the behavior we expect of the model; as we minimize cost with respect to the train set, we should see a corresponding decrease in cost on the test set. Based on the figure above, the cost of the model decreases very smoothly with respect to the number of examples seen. This indicates to us that the neural network was a good fit for the dataset, as it continually benefited from learning on the train set.

2 The Titanic Dataset

2.1 Model Selection

Again we will evaluate the **k-NN**, **random forest** and **neural network** models on this dataset.

1. k-NN will work well *under the assumption that people with similar demographics were similarly likely to survive.* This is a reasonable assumption to make for this dataset as an individual's demographic could have influenced whether they were able to make it onto one of the lifeboats.
2. Decision trees (and consequently random forests) will likely do well as there seem to be good ways of splitting the data. For example, partitioning the dataset by the "Sex" column already decreases entropy drastically; perhaps further splits will lead to even stronger models.
3. Again, we consider neural networks here as they are universally applicable to classification problems.

2.2 Model Tuning

For each of the selected models, we systematically evaluate the dataset on selected hyperparameters and identify the best hyperparameters to use on the dataset.

2.2.1 k-NN

k	`acc`	`f1`
1	0.757	0.746
5	0.807	0.794
10	0.802	0.791
20	0.818	0.807
30	0.819	0.809
40	0.802	0.790
50	0.800	0.788

Table 4: k-NN model performance on titanic dataset for different number of neighbors.

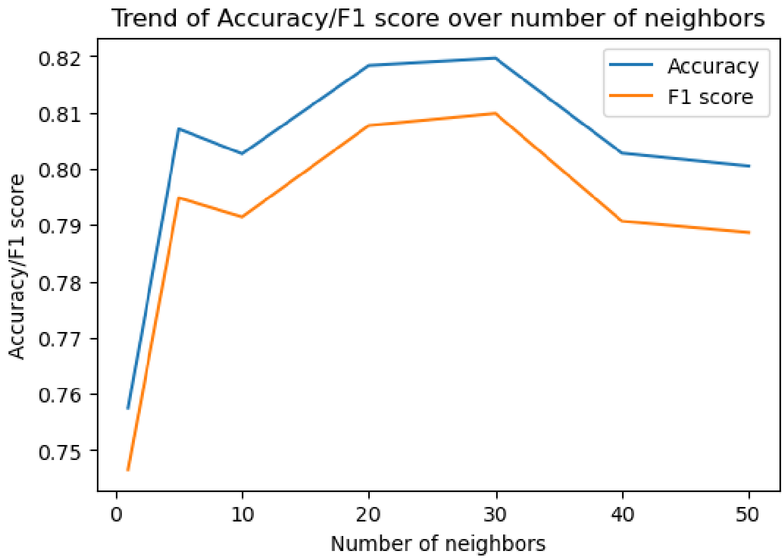

Figure 5: k-nn accuracy and f1 score vs. different values of k. Performance increases until $k = 30$. This means that by taking majority vote from samples upto 30, increased the performance of the model.

Based on the results above, we identify $k = 30$ as the best hyperparameter for the k-NN model.

2.2.2 Random Forest

ntrees	max-depth	acc	f1
1	10	0.670	0.645
5	10	0.727	0.704
10	10	0.759	0.742
20	10	0.771	0.753
30	10	0.774	0.758
40	10	0.769	0.753
50	10	0.782	0.768
50	20	0.780	0.765
50	30	0.776	0.761
50	40	0.780	0.763
50	50	0.785	0.770

Table 5: Random Forest model performance on titanic dataset for different number of trees.

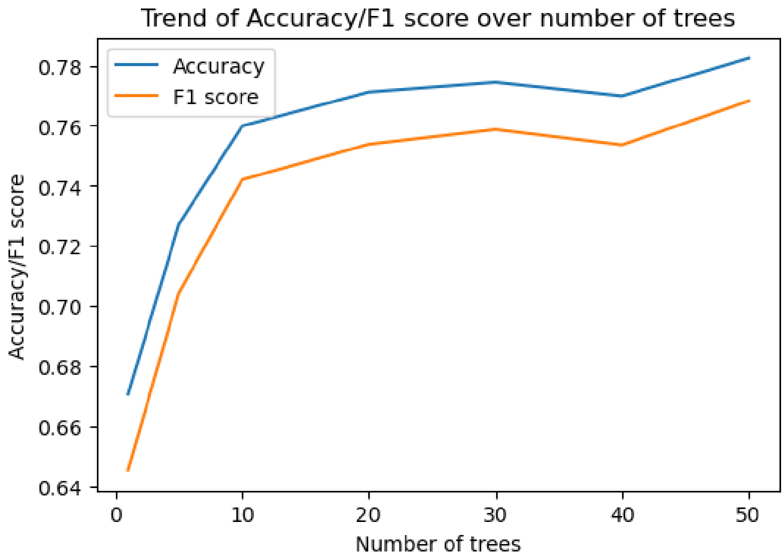

Figure 6: random forest accuracy and f1 score vs. different values of ntrees with $max\_depth = 10$. Performance increases as we increase number of trees.

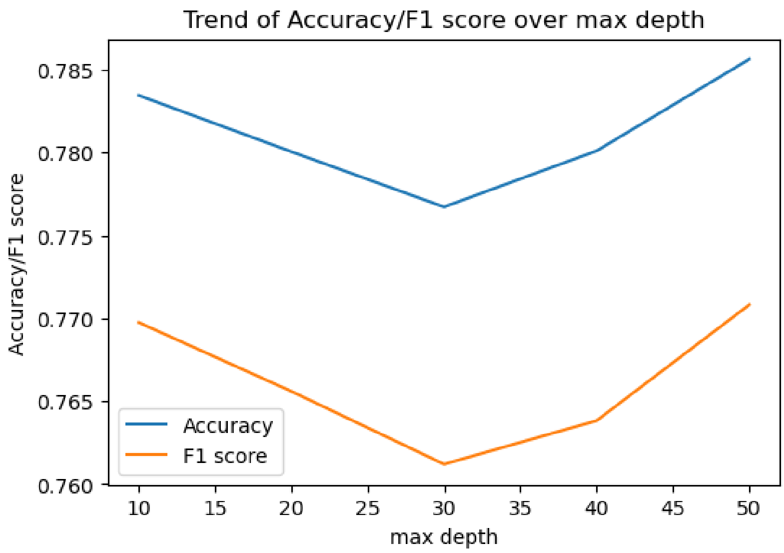

Figure 7: random forest accuracy and f1 score vs. different values of max_depth for ntrees=50. The performance degrades and then improves.

Based on the results above, we identify `ntrees` = 50 and `max-depth` = 50 as the best hyperparameters for the random forest.

2.2.3 Neural Network

num-layers	num-neurons	λ	acc	f1
1	64	0.001	0.812	0.730
1	64	0.01	0.820	0.742
1	128	0.001	0.819	0.740
2	128	0.001	0.804	0.719

Table 6: Neural network model performance on digits dataset. Mini-batch gradient descent with 32 batches and learning rate $\alpha = 0.5$ was used to train each model.

Based on the results above, we identify `num-layers` $= 1$, `num-neurons` $= 64$ and $\lambda = 0.01$ as the best set of hyperparameters for the dataset.

2.3 Model Evaluation

2.3.1 Neural Network

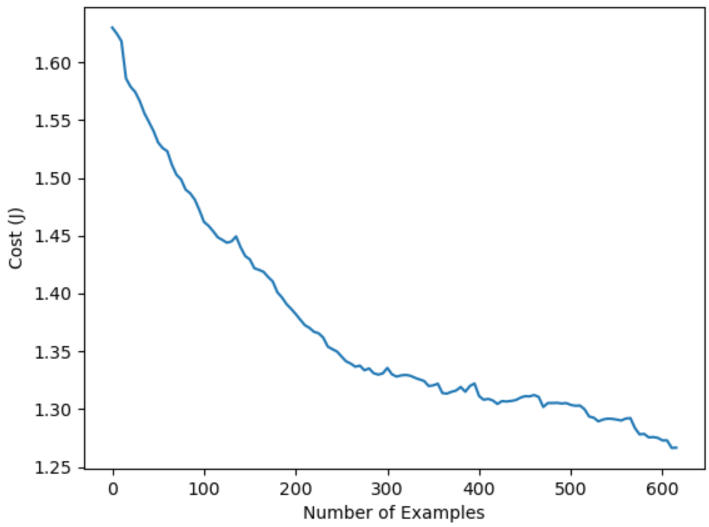

Figure 8: Neural network cost on test set vs. number of training examples seen. A learning rate of $\alpha = 0.004$ was used and weights were updated after every 5 examples.

We find that the average cost of the model against the test set trends downwards with the number of examples seen. This is the behavior we expect of the model; as we minimize cost with respect to the train set, we should see a corresponding decrease in cost on the test set. Based on the figure above, the cost decreases more gradually compared to other curves we have seen before; this could suggest that either the dataset is harder to learn in comparison to the other datasets we've seen before, or that the chosen hyperparameters are suboptimal for the dataset. A more exhaustive hyperparameter search (e.g. a grid search) could have helped to better understand this issue.

3 The Loan Eligibility Prediction Dataset

3.1 Model Selection

Again we will evaluate the **k-NN**, **random forest** and **neural network** models on this dataset.

1. k-NN will work well *under the assumption that people with similar demographics are similarly likely to receive a loan.* This is a highly reasonable assumption to make as many algorithms for deciding loan eligibility make this assumption in practice.
2. Decision trees (and similarly, random forests) are a natural candidate for the dataset as they emulate a very straightforward approach to determining loan eligibility. They also make sense given the nature of the dataset as we are working with a mix of categorical and numerical features.
3. Again, we consider neural networks here as they are universally applicable to classification problems.

3.2 Model Tuning

For each of the selected models, we systematically evaluate the dataset on selected hyperparameters and identify the best hyperparameters to use on the dataset.

3.2.1 k-NN

k	`acc`	`f1`
1	0.685	0.621
5	0.781	0.727
10	0.808	0.764
20	0.797	0.755
30	0.781	0.734
40	0.766	0.714
50	0.747	0.693

Table 7: k-NN model performance on loan eligibility dataset for different number of neighbors.

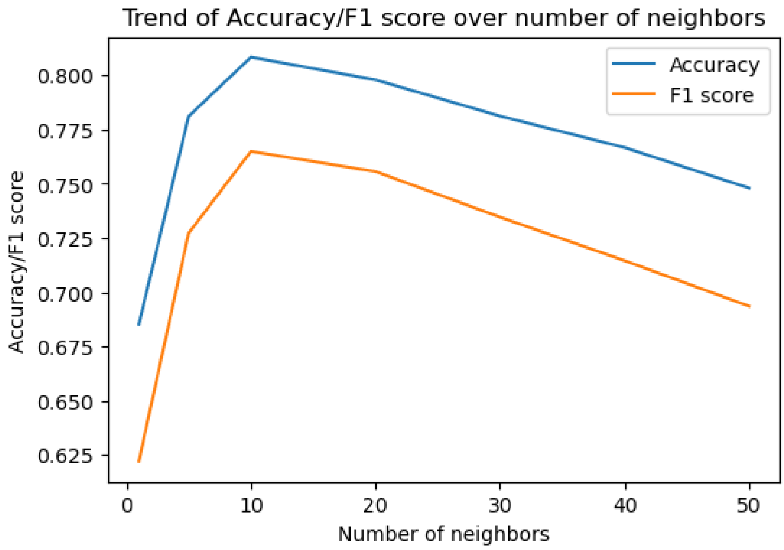

Figure 9: k-nn accuracy and f1 score vs. different values of k. Performance first increases sharply and then decreases.

Based on the results above, we identify $k = 10$ as the best hyperparameter for the k-NN model.

3.2.2 Random Forest

ntrees	max-depth	acc	f1
1	10	0.633	0.540
5	10	0.720	0.559
10	10	0.693	0.499
20	10	0.718	0.608
30	10	0.716	0.625
40	10	0.695	0.466
50	10	0.714	0.571
30	20	0.733	0.653
30	30	0.716	0.582
30	40	0.714	0.571
30	50	0.706	0.535

Table 8: Random Forest model performance on loan eligbility dataset for different number of trees

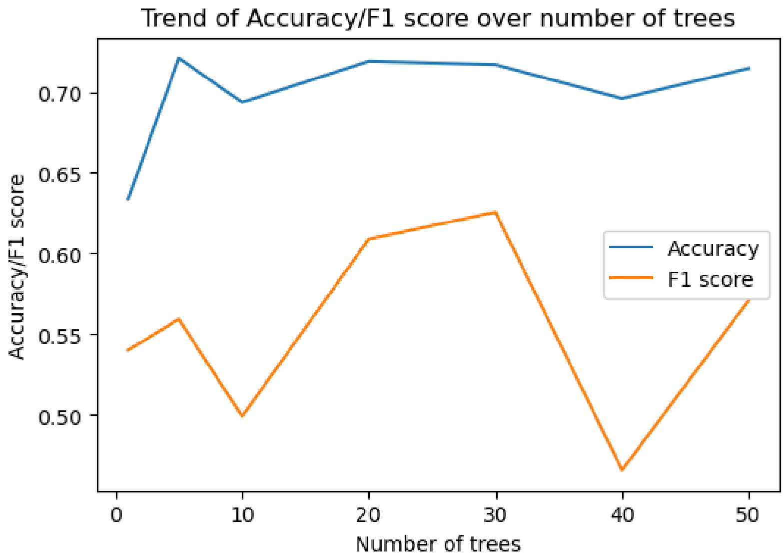

Figure 10: random forest accuracy and f1 score vs. different values of ntrees with *max_depth* = 10. Performance does not increase as we increased the number of trees which is unexpected behavior.

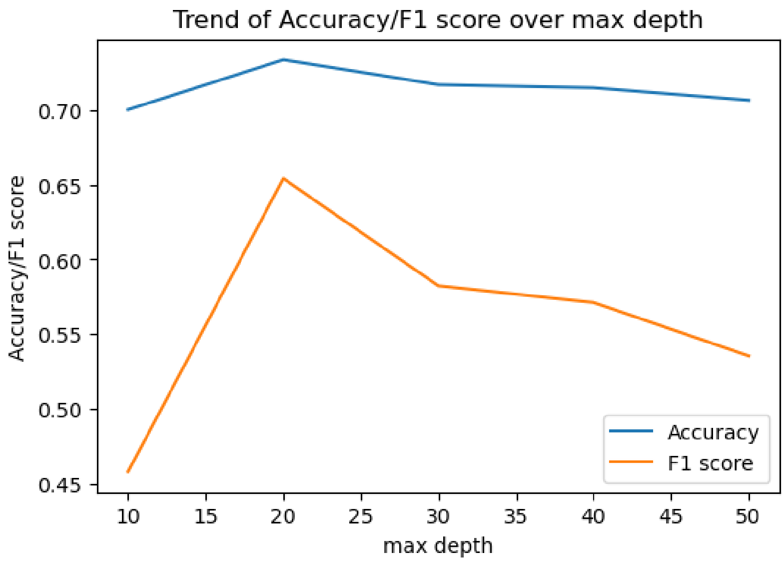

Figure 11: random forest accuracy and f1 score vs. different values of *max_depth* for ntrees=30. Performance does not increase as we increase the depth which is unexpected behavior.

Based on the results above, we identify `ntrees` = 30 and `max-depth` = 20 as the best hyperparameters for the random forest.

3.2.3 Neural Network

num-layers	num-neurons	λ	acc	f1
1	64	0.001	0.736	0.814
1	64	0.01	0.771	0.847
1	128	0.001	0.737	821
2	128	0.001	0.727	0.807

Table 9: Neural network model performance on digits dataset. Mini-batch gradient descent with 32 batches and learning rate $\alpha = 0.5$ was used to train each model.

Based on the results above, we identify `num-layers` $= 1$, `num-neurons` $= 64$ and $\lambda = 0.01$ as the best set of hyperparameters for the dataset.

3.3 Model Evaluation

3.3.1 Neural Network

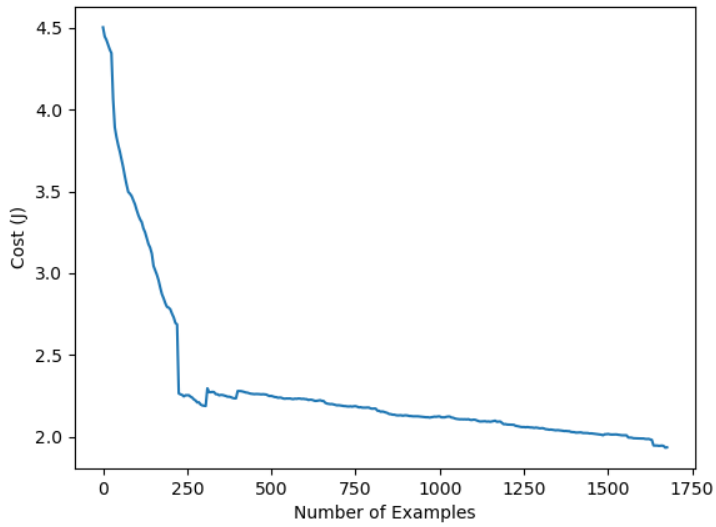

Figure 12: Neural network cost on test set vs. number of training examples seen. A learning rate of $\alpha = 0.004$ was used and weights were updated after every 5 examples.

We find that the average cost of the model against the test set trends downwards with the number of examples seen. This is the behavior we expect of the model; as we minimize cost with respect to the train set, we should see a corresponding decrease in cost on the test set. Based on the figure above, the cost of the model drops sharply around 250 examples and decreases more gradually after that; this suggests that a smaller subset of the training data was enough for the neural network to "learn" the dataset.

4 The Oxford Parkinson's Disease Detection Dataset

4.1 Model Selection

Again we will evaluate the **k-NN**, **random forest** and **neural network** models on this dataset.

1. k-NN makes the assumption that the voice measurements of Parkinson's patients are in some sense "close." In general this is not a safe assumption to make, as there are countless external factors that could impact a person's voice (biological sex, genetics, lifestyle choices, etc.) but there may still be a correlation between the particular properties measured in the dataset and the healthiness of the patients.
2. Decision trees (and hence random forests) may work well on this dataset if there are particular thresholds in the feature space at which Parkinson's is likely to occur.
3. Again, we consider neural networks here as they are universally applicable to classification problems.

4.2 Model Tuning

For each of the selected models, we systematically evaluate the dataset on selected hyperparameters and identify the best hyperparameters to use on the dataset.

4.2.1 k-NN

k	`acc`	`f1`
1	0.948	0.939
5	0.907	0.881
10	0.907	0.873
20	0.851	0.746
30	0.855	0.753
40	0.846	0.741
50	0.790	0.614

Table 10: k-NN model performance on parkinsons dataset for different number of neighbors.

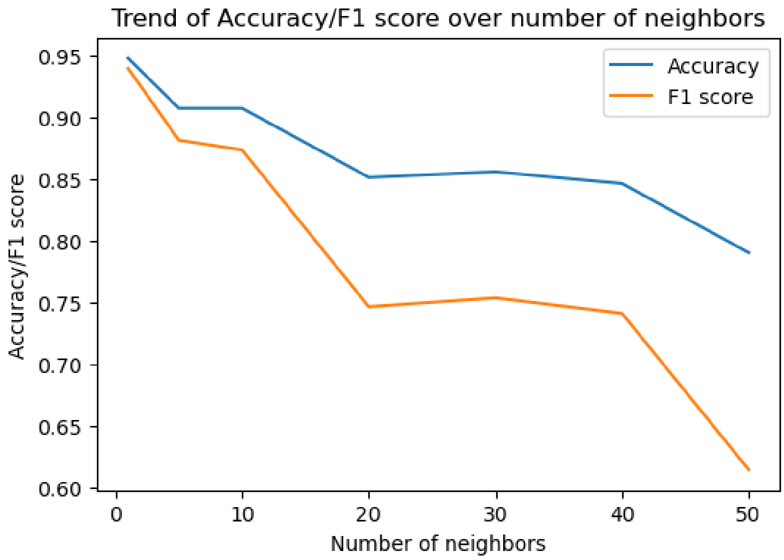

Figure 13: k-nn accuracy and f1 score vs. different values of k. Performance decreases sharply as we increase k

Based on the results above, we identify $k = 1$ as the best hyperparameter for the k-NN model.

4.2.2 Random Forest

ntrees	max-depth	acc	f1
1	10	0.861	0.816
5	10	0.856	0.809
10	10	0.872	0.824
20	10	0.892	0.851
30	10	0.903	0.866
40	10	0.897	0.858
50	10	0.913	0.879
50	20	0.913	0.880
50	30	0.902	0.865
50	40	0.914	0.880
50	50	0.892	0.850

Table 11: Random Forest model performance on Parkinson's dataset for different number of trees. Performance increases as we increase the number of trees. It is possible that more than 50 trees will also lead tp better performances.

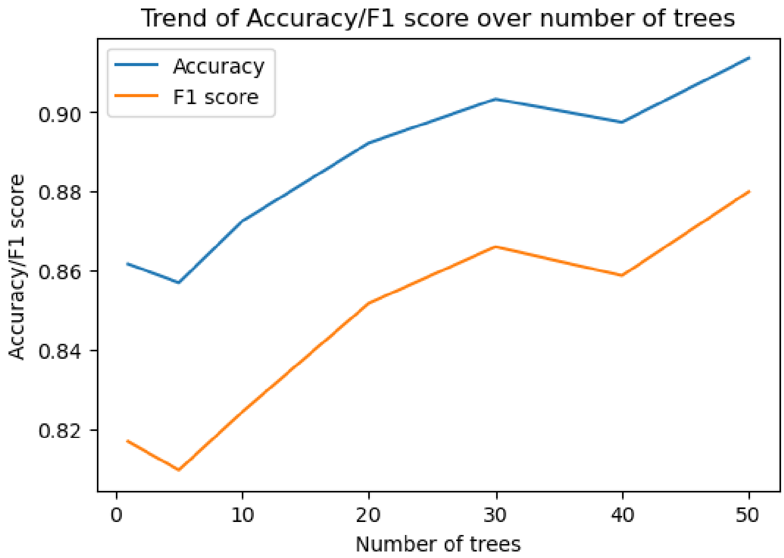

Figure 14: random forest accuracy and f1 score vs. different values of ntrees with $max\_depth = 10$.

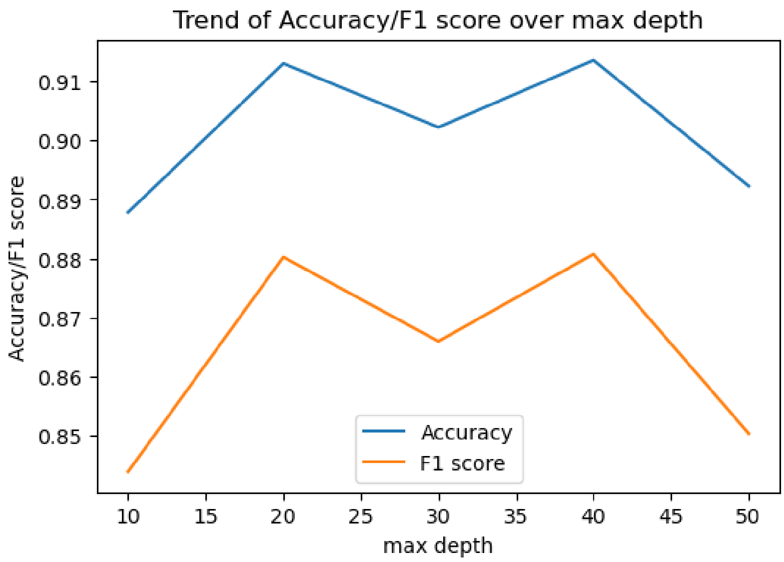

Figure 15: random forest accuracy and f1 score vs. different values of max_depth for ntrees=50. No visible pattern is seen when depth increases.

Based on the results above, we identify `ntrees` = 50 and `max-depth` = 40 as the best hyper-parameters for the random forest.

4.2.3 Neural Network

num-layers	num-neurons	λ	acc	f1
1	64	0.001	0.928	0.949
1	64	0.01	0.861	0.913
1	128	0.001	0.938	0.960
2	128	0.001	0.917	0.943

Table 12: Neural network model performance on digits dataset. Mini-batch gradient descent with 32 batches and learning rate $\alpha = 0.5$ was used to train each model.

Based on the results above, we identify `num-layers` = 1, `num-neurons` = 128 and λ = 0.001 as the best set of hyperparameters for the dataset.

4.3 Model Evaluation

4.3.1 Neural Network

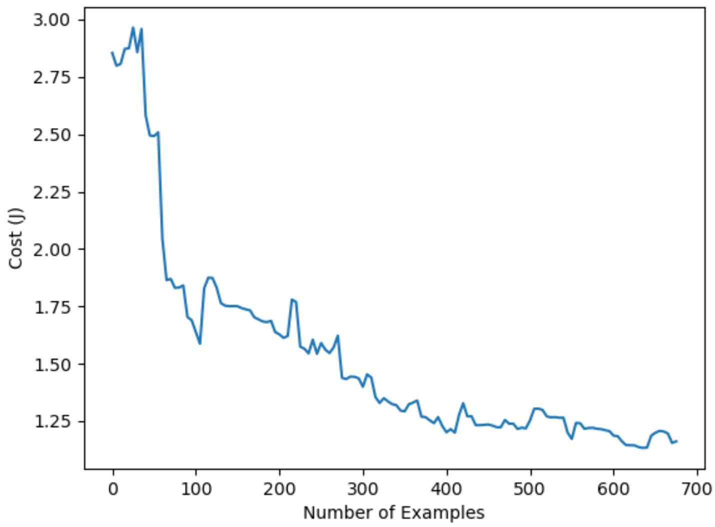

Figure 16: Neural network cost on test set vs. number of training examples seen. A learning rate of $\alpha = 0.004$ was used and weights were updated after every 5 examples.

We find that the average cost of the model against the test set trends downwards with the number of examples seen. This is the behavior we expect of the model; as we minimize cost with respect to the train set, we should see a corresponding decrease in cost on the test set. Based on the figure above, the cost of the model drops sharply around 100 examples and decreases more gradually after that; this suggests that a smaller subset of the training data was enough for the neural network to "learn" the dataset.

5 Summary

	Digits Dataset		Titanic Dataset		Loan Dataset		Parkinsons Dataset	
	Accuracy	F1-Score	Accuracy	F1-Score	Accuracy	F1-Score	Accuracy	F1-Score
KNN	0.989	0.989	0.819	0.809	0.808	0.764	0.948	0.939
RF	0.855	0.858	0.785	0.770	0.733	0.653	0.914	0.880
NN	0.984	0.984	0.820	0.719	0.771	0.847	0.938	0.943