ML Project Report

Vaishnavi Shah and Shreya Birthare

10 May 2024

1 Handwriting Dataset

K-Nearest Neighbors (KNN) was chosen primarily for its simplicity and efficacy in classification tasks that rely on feature similarity. KNN operates by identifying the closest training examples in the feature space, making it particularly suitable for image data where the resemblance in pixel values often correlates with class similarity. The code provided implements KNN from scratch, incorporating functions for calculating Euclidean distances between feature sets, a method widely used for its straightforwardness and effectiveness in spatial data comparisons. The decision to use KNN was also influenced by its non-parametric nature, requiring no explicit training phase, thus enabling quick adaptation to new data without retraining the entire model. This makes KNN exceptionally adaptable to scenarios where the model needs to be frequently updated with new examples. Neural Networks, known for their flexibility and power in modeling non-linear relationships, were chosen for their ability to learn hierarchical representations of data. The custom neural network implementation provided allows for extensive experimentation with different network architectures and regularization techniques to optimize performance. Neural Networks excel in image recognition tasks due to their deep learning capabilities, which can abstract features at various levels (edges at lower levels, more complex shapes at higher levels) that are crucial for effective image classification. The code reflects this through functions for forward propagation, backpropagation, and loss calculation, which are central to training deep learning models. Additionally, the inclusion of regularization methods in the implementation helps in avoiding overfitting, a common challenge with models as complex as neural networks. Random Forest was selected due to its robustness against overfitting and its capability to handle large datasets with numerous features both categorical and numerical. As an ensemble learning method that combines multiple decision trees to improve classification accuracy, Random Forest is adept at managing the complexity and high dimensionality typical of image data. The implementation of Random Forest in the code includes features like bootstrap sampling and the use of Gini index or information gain for optimal split finding, enhancing its ability to discern intricate patterns and interactions between pixels across different trees. This complexity management makes Random Forest an excellent choice for image recognition, where the relationships between features (pixels) can be nonlinear and require sophisticated modeling to capture effectively.

1.1 KNN

1. We trained and tested our code for k_values = [3, 5, 7] and calculated the metrics

2. The K-Nearest Neighbors (KNN) algorithm's performance on the Hand-Written Digits Recognition Dataset, as indicated by your results, shows a distinct improvement in both accuracy and F1 score as the value of K (the number of nearest neighbors considered) increases from 3 to 7

3. Increasing K Value: K = 3: The model achieves an accuracy of 0.9572 and an F1 score of 0.9569. This lower performance relative to higher K values might be due to the model being more sensitive to noise in the data. With fewer neighbors considered, the prediction is heavily influenced by the immediate few data points, which may not always represent the general pattern effectively, especially if those points are outliers or mislabeled. K = 5: There is a noticeable increase in both accuracy and F1 score (0.9730 and 0.9739, respectively). As more neighbors are considered, the model becomes more robust against noise and can generalize better, reducing the influence of anomalies in the data. K = 7: The model reaches its peak performance with an accuracy of 0.9776 and an F1 score of 0.9735. This increment indicates that the model benefits from considering even more neighbors, providing a more stable and reliable estimation of the true class by aggregating more information from the dataset.

4. Performance Analysis: The trend suggests that as K increases, the algorithm is better able to mitigate the effects of noisy data and outliers, leading to more accurate and consistent predictions. This is particularly beneficial for a dataset like the Hand-Written Digits Recognition, where variations in digit writing styles can introduce a significant amount of variability. The slight decrease in the F1 score from K = 5 to K = 7, despite an increase in accuracy, suggests that there might be a point where adding more neighbors does not necessarily contribute to better precision and recall balance but still aids in overall classification correctness. This could be due to the increase in true positive rates being offset by a relatively slower rate of decrease in false positives.

5. Optimal K Selection: Choosing the optimal K is crucial in KNN. Too small a K can make the model too sensitive to noise, while too large a K might lead to underfitting, where the model overly generalizes the training data. The results suggest that K = 7 offers the best accuracy without a significant trade-off in F1 score, making it a suitable choice for this particular dataset. The KNN model's increasing performance with a higher K indicates its effectiveness in handling datasets with substantial variability in class representation, such as the Hand-Written Digits Recognition Dataset. However, it also highlights the importance of tuning K to balance between noise sensitivity and the ability to generalize.

6. In summary, the improvement in performance metrics with increasing K values for KNN on the Hand-Written Digits Recognition Dataset underscores the algorithm's capacity to adapt to the intrinsic challenges of the dataset, providing robust, reliable predictions as more contextual information (neighbors) is considered. This analysis not only aids in understanding the behavior of KNN in different settings but also in optimizing its parameters for enhanced predictive performance.

7. Figure 1,2

1.2 Random Forest

1. We tried different values of ntree=10,30,50 and calculated the precision, recall, accuracy and f1 score for the same.

```
Net Average Accuracy for all Ks: 0.9705713318052641
Net F1 Score for all Ks: 0.9684013305100465
```

K	Accuracy	F1 Score
3	0.9572	0.9569
5	0.9730	0.9739
7	0.9776	0.9735

Figure 1: HW- Table

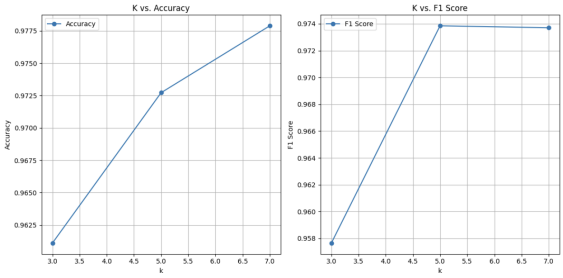


Figure 2: HW - Graph

2. The results from the Random Forest algorithm on the Hand-Written Digits Recognition Dataset, as shown in the table and graph, indicate that the model's performance generally improves with an increase in the number of trees (ntrees) used in the ensemble up to a certain point.

3. Performance with Different ntree Values: With 10 trees, the accuracy, F1 score, precision, and recall are relatively lower (around 0.933), suggesting that the model might not be sufficiently robust at this stage. The smaller number of trees could lead to higher variance and less generalization, causing the model to capture noise in the training data rather than the actual signal. Increasing to 30 trees, there is a noticeable improvement in all metrics, with accuracy and F1 score both around 0.966. This enhancement indicates that adding more trees helps in averaging out errors, reduces variance, and avoids overfitting, making the model more stable and accurate. However, when the number of trees is further increased to 50, the performance slightly decreases compared to 30 trees. This could be due to the model starting to converge towards a limit beyond which adding more trees doesn't significantly contribute to new information or better generalization; instead, it might introduce complexity that slightly reduces performance efficiency in terms of recall.

4. Interpreting the Metrics: Accuracy is very high for all configurations, suggesting that the Random Forest model is overall effective in classifying the digits correctly. Precision and recall are also high, indicating that the model has a good balance between identifying as many relevant results as possible (recall) and ensuring that the predictions it makes are correct as often as possible (precision). The F1 score, which balances precision and recall, is consistently high, reinforcing the model's robustness in handling this type of dataset.

5. Algorithm Suitability and Performance: Random Forest performs well on datasets like the

Hand-Written Digits Recognition because it can handle high-dimensional data efficiently. Each tree in the forest considers a subset of features, allowing the ensemble to explore a diverse set of decision paths, making good use of the dataset's multiple attributes. The slight performance dip when moving from 30 to 50 trees might indicate that there is an optimal number of trees beyond which the marginal gains in performance are minimal. This can be due to the law of diminishing returns, where each additional tree adds less to the overall accuracy and might even clutter the decision-making process.

6. In summary, the Random Forest algorithm demonstrates strong performance on the Hand-Written Digits Recognition Dataset, benefiting significantly from its ensemble approach, which enhances its prediction accuracy and general robustness against overfitting. The best performance is observed with 30 trees, suggesting a balanced point between complexity and learning capability.

7. 3,4

ntree	accuracy	f1score	precision	recall
10	0.9332339290395743	0.9335640581167135	0.9356166838279304	0.9315339239102531
30	0.9666305642411379	0.9654557586650263	0.9654214941280832	0.9655001977058385
50	0.9643207103 27691	0.9643500551609266	0.9652332222446942	0.9634852322338524

Figure 3: HW - table

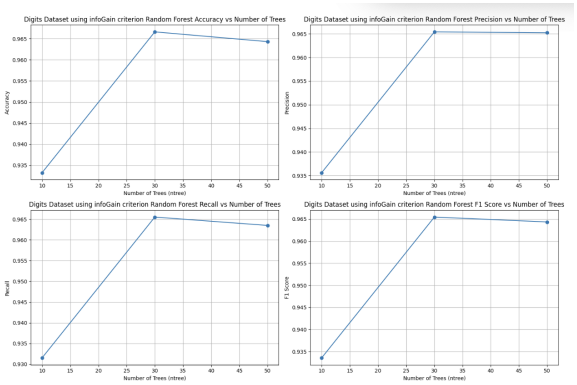


Figure 4: HW - graph

1.3 Neural Networks

1. We used the following architectures to train and test our neural network: k = 10 epochs = 1000 learning_rate = 0.8 lambdas = [0.01, 0.04, 0.07] dimensions = [[64, 32, 10], [64, 32, 64, 10]]

2. The neural network results for the Hand-Written Digits Recognition Dataset provide insightful data on the performance of different configurations across several training parameters. These results illustrate how factors such as the regularization parameter (lambda) and the network architecture significantly influence the model's ability to generalize and effectively learn from the dataset. Performance Across Different Configurations: The neural network was tested with a learning rate of 0.8, 1000 epochs, and 10-fold cross-validation under various lambda values and two different architectures ([64, 32, 10] and [64, 32, 64, 10]). The results show that accuracy and F1 scores generally hover above 0.97, indicating strong performance.

3. Increasing the regularization strength (lambda) from 0.01 to 0.07 shows a slight decrease in performance metrics (both accuracy and F1 score), which can be attributed to the increased penalty on larger weights. This could lead to underfitting where the model becomes too simple to capture the underlying patterns in the data effectively.

4. Impact of Lambda and Architecture: The lambda parameter controls the extent of regularization. A lower lambda (0.01) provides enough regularization to prevent overfitting without overly constraining the model's capacity to learn complex patterns. As lambda increases, the network may become too restricted, leading to a drop in accuracy and F1 score as seen with lambda 0.07.

5. Comparing the two architectures, the configuration with one additional hidden layer ([64, 32, 64, 10]) does not consistently outperform the simpler one ([64, 32, 10]). This suggests that adding more complexity to the network (in terms of layers) does not necessarily translate to better performance and might actually be less efficient due to the potential increase in trainable parameters without significant gains in learning deeper representations for this particular dataset.

6. Learning Curve Analysis: The learning curve for the configuration with Alpha: 0.8, Lambda: 0.01 exhibits a sharp decrease initially, which levels off as more training instances are used. This pattern indicates rapid learning initially, with diminishing returns as the model begins to converge to its optimal state. The curve flattens, demonstrating that the model is not significantly improving with additional training instances past a certain point, which is typical in training neural networks where early epochs make the most substantial improvements to model weights.

7. In summary, the neural network's performance on the Hand-Written Digits Recognition Dataset is quite robust, with high accuracy and F1 scores across different configurations. The choice of regularization strength and network architecture significantly impacts performance, where too much regularization or unnecessarily complex architectures do not yield proportionate gains in learning efficiency. The learning curve further substantiates the model's rapid adaptation early in training, with a plateau indicating convergence. These insights suggest that while neural networks are powerful for this task, careful tuning of hyperparameters is essential to maximize their potential without incurring unnecessary computational costs or risking overfitting/underfitting.

8. Figure 5,6

ntree	accuracy	f1score	precision	recall
10	0.9332339290395743	0.9335640581167135	0.9356166838279304	0.9315339239102531
30	0.9666305642411379	0.9654557586650263	0.9654214941280832	0.9655001977058385
50	0.9643207103 27691	0.9643500551609266	0.9652332222446942	0.9634852322338524

Figure 5: HW Dataset - Table

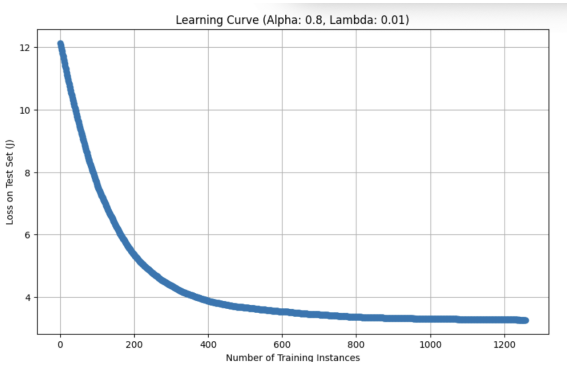


Figure 6: HW Dataset - Curve

2 Titanic Dataset

K-Nearest Neighbors (KNN) is particularly advantageous for the Titanic dataset because it can effectively model decision boundaries that are not necessarily linear, which is crucial given the mixed data types (categorical and continuous) present in the dataset. The attributes like age, sex, and passenger class likely interact in non-linear ways to affect survival chances. KNN's method of predicting the outcome based on the nearest data points makes it a suitable choice for capturing such complex pattern interactions without the need for a predefined model structure. Moreover, the simplicity of the algorithm allows for quick iterations and modifications, which is beneficial during the exploratory phase of analysis. Random Forest, an ensemble of decision trees, is known for its high accuracy and robustness against overfitting, which is crucial when dealing with datasets that, like the Titanic dataset, may have intricate interactions among features. This algorithm can handle both numerical and categorical data effectively, making it ideal for the Titanic dataset where predictors vary widely in type and scale. The Random Forest model works by building multiple decision trees and voting on the most popular outcome to improve prediction accuracy and stability. This is particularly useful for the Titanic dataset where variables like "passenger class" or "fare" might individually predict survival poorly, but when combined, they provide a strong basis for prediction. The Random Forest algorithm also provides valuable insights into feature importance, which helps in understanding which factors most significantly influenced survival, enhancing the interpretability of the model. Neural Networks offer the capability to learn complex, non-linear relationships through layers of neurons, making them highly effective for datasets with intricate patterns. For the Titanic dataset, where interactions between variables (like age, sex, and passenger class) may not be straightforward, Neural Networks can detect and model these subtleties in ways

that simpler models cannot. The flexibility in architecture—such as the number of layers and neurons—allows for detailed tuning and optimization of the model to fit the specific characteristics of the dataset. Furthermore, the ability of Neural Networks to integrate regularization techniques helps prevent overfitting, ensuring that the model generalizes well to new, unseen data. Neural Networks were chosen for their deep learning capabilities, necessary for modeling the complex and non-linear relationships expected in the dataset.

2.1 KNN

1. We trained and tested our code for k_values = [3, 5, 7] and calculated the metrics

2. The results from applying the K-Nearest Neighbors (KNN) algorithm to the Titanic dataset, which includes a combination of numerical and one-hot encoded categorical data, show a distinct trend where both accuracy and F1 score decrease as the number of neighbors (K) increases.

3. Decrease in Performance Metrics as K Increases: The results indicate a decline in both accuracy and F1 score as K increases from 3 to 7. This trend can be attributed to several factors inherent in the nature of KNN and the dataset's characteristics:

4. Sensitivity to K: KNN's performance is highly sensitive to the choice of K, especially in datasets with mixed data types. A smaller K means the classification is heavily influenced by the nearest few data points, which can be beneficial if these neighbors provide a strong signal about the target class. However, it also makes the algorithm more susceptible to noise and outliers in the data.

5. Curse of Dimensionality: With the increase in dimensions due to one-hot encoding of categorical variables, distances between points become less meaningful. This phenomenon can adversely affect KNN, which relies heavily on distance metrics to make predictions. As K increases, the algorithm might start considering neighbors that are not truly similar but appear close due to the increased dimensionality.

6. Impact of One-Hot Encoding on Distance Calculation: One-hot encoding transforms categorical variables into binary vectors, significantly increasing the dataset's dimensionality. In high-dimensional spaces, the distance between data points tends to be more uniform, reducing the effectiveness of distance-based methods like KNN. The more dimensions there are, the harder it becomes to discern truly meaningful neighbors from those that are near due to the sparsity of the space.

7. Challenges in Balancing Bias and Variance: K = 3: Provides the highest accuracy and F1 score, indicating that considering fewer neighbors helps in making more precise predictions, possibly because it reduces the variance of the prediction. This setting might be capturing the most relevant information while avoiding too much noise. K = 5 and K = 7: As K increases, the algorithm starts to average more neighbors, which can dilute the influence of truly relevant instances and increase the bias of the model. This could be leading to a decrease in performance metrics as the model becomes overly generalized and less sensitive to variations in the smaller, more informative patterns in the data.

8. The decreasing trend in performance with increasing K suggests that for the Titanic dataset, a smaller K is more effective, likely due to the dataset's mixed data types and the challenges posed by high dimensionality from one-hot encoding. To potentially improve KNN's performance, it might be beneficial to explore techniques like dimensionality reduction to reduce the impact of less informative features or weighting features differently based on their predictive power. Additionally, experimenting with different distance metrics that might be more robust to high-dimensional settings could also be advantageous.

9. Figure 7,8

```
Net Average Accuracy for all Ks: 0.7670671749007599
Net F1 Score for all Ks: 0.7525726719074131
```

K	accuracy	f1score
3	0.7734520344689836	0.7606543653613816
5	0.7677745163021178	0.7522140383078119
7	0.7599749739311782	0.7448496120530458

Figure 7: Titanic Dataset - Table

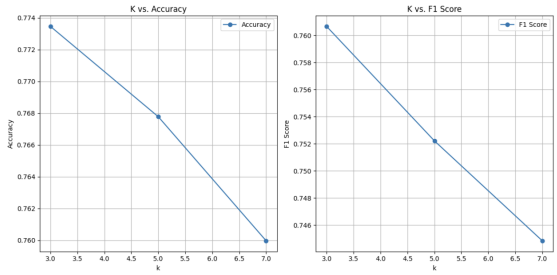


Figure 8: Titanic Dataset - Graph

2.2 Random Forest

1. Interpretation of the Results:

2. Performance Metrics Across Different Tree Counts: NTree 10: This configuration yields the highest overall performance metrics, with an average accuracy of 0.8286, precision of 0.8330, recall of 0.8052, and F1 score of 0.8186. These results suggest that even with a relatively small number of trees, Random Forest can effectively capture the essential patterns in the data, benefiting from both diversity (from random feature selection) and ensemble learning (aggregating multiple decision trees). NTree 30 and NTree 50: Surprisingly, increasing the number of trees does not correspond with an improvement in performance. Both configurations show a slight decrease in accuracy, precision, recall, and F1 score compared to NTree 10. This might indicate that adding more trees beyond a certain point does not significantly contribute to model accuracy but may affect the precision and recall balance.

3. Random Forest Stability and Overfitting: The results suggest a high level of stability in the Random Forest algorithm as performance metrics do not vary drastically with an increase in the number of trees. This stability is a key strength of Random Forest, providing reliable performance even when the model configuration changes slightly. The slight decrease in performance with more trees could be indicative of the model beginning to fit more to the noise in the data rather than the actual signal. While Random Forest is generally robust against overfitting, overly complex models with too many trees might still suffer from this issue, especially if the trees are highly correlated or if the diversity introduced by random features is insufficient.

4. Dimensionality and Feature Importance: The Titanic dataset contains a mix of categorical and numerical data, which are handled effectively by Random Forest through its inherent mechanisms of handling different types of data and feature importance evaluation. The use of infoGain helps in optimizing splits based on the most informative features, which is crucial for a dataset where some features might significantly influence survival predictions (e.g., age, sex, passenger class). The relatively stable performance across different numbers of trees also underscores Random Forest's capability to manage feature interactions and dependencies effectively, which is crucial for a dataset with complex relationships like the Titanic dataset.

5. Conclusion: The Random Forest algorithm demonstrates good performance on the Titanic dataset, with the highest metrics observed at NTree 10. This might suggest an optimal balance between model complexity and learning capability, where further increases in the number of trees do not necessarily equate to better performance and might slightly degrade the model's precision and recall balance. These findings highlight the importance of tuning the number of trees in a Random Forest model to align with the dataset's specific characteristics and the diminishing returns of adding more trees beyond a certain optimal point.

6. Figure 9, 10

ntree	accuracy	precision	recall	f1score
10	0.8285679832028148	0.8329691969 78125	0.8052169422476677	0.8186443024042867
30	0.8261792078084212	0.82771165308 37645	0.8017732764872546	0.8144194443091969
50	0.8265100442628531	0.8223348847 85407	0.8037916045289724	0.8128008397883646

Figure 9: Titanic Dataset - Table

2.3 Neural Networks

1. We used the following architecture to train and test our neural network: learning_rate = .8 epochs = 1000 k = 10 dimensions = [[dataset.shape[1]-1, 2, 5, 2], [dataset.shape[1]-1, 2, 3, 8, 2]] lamb = [0.01, 0.04, 0.07]

2. The results from applying a neural network to the Titanic dataset, which includes both numerical and categorical data processed through one-hot encoding, provide a nuanced view of how different configurations and regularization parameters influence model performance.

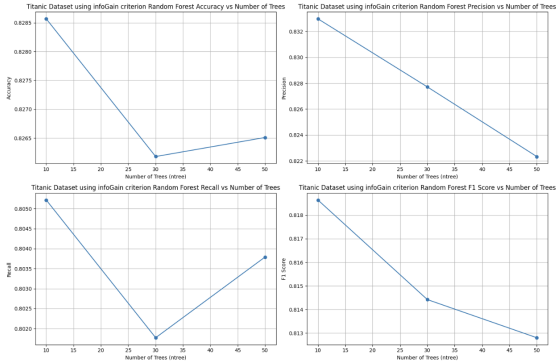


Figure 10: Titanic Dataset - Graph

3. Variability in Performance Metrics: The configurations tested show a varying degree of accuracy and F1 score, with both metrics peaking for certain lambda values and network architectures. The highest accuracy and F1 scores are observed for a lambda of 0.01 and architectures [102, 2, 5, 2] and [102, 3, 8, 2], suggesting that mild regularization combined with a modestly complex network structure tends to yield the best results in this scenario.

4. Impact of Lambda (Regularization Strength): The lambda parameter significantly affects the model's ability to generalize. With lambda = 0.01, the network maintains enough flexibility to learn from the data without heavily penalizing the weight sizes, which likely helps in capturing essential patterns without overfitting. As lambda increases to 0.04 and 0.07, there is a noticeable decline in both accuracy and F1 scores. This trend could indicate that the network starts to underfit the data, overly simplifying the model which prevents it from capturing the more subtle, complex relationships in the data.

5. Network Architecture and Dataset Complexity: The architecture variation results ([102, 2, 5, 2], [102, 3, 8, 2], [102, 3, 2, 8]) show that networks with an increased number of neurons in hidden layers don't consistently result in better performance. It suggests that while additional neurons provide a higher capacity to learn, they may also make the network prone to overfitting or increase the challenge of training effectively, particularly when combined with higher lambda values. The simpler architectures ([102, 2, 5, 2], [102, 3, 8, 2]) seem to provide a better balance between learning capability and complexity, resulting in higher performance.

6. Learning Curve Analysis: The learning curve for lambda = 0.04 shows a steady decrease in loss as the number of training instances increases, indicating that the network is learning and improving its predictions over time. The curve plateauing toward the end of training suggests that the model is approaching an optimal state given the current configuration and dataset. The gradual and consistent decline in loss without sharp drops or rises indicates that the learning rate and the batch processing are well-tuned for this training session, allowing stable convergence.

7. Specific Observations and Considerations: Convergence to Local Optima: Higher lambda values might be forcing the network into a too-restricted space where it can't escape local

optima, leading to poorer performance. Performance on Mixed Data Types: Neural networks can handle mixed data types effectively once they are correctly encoded. However, the encoding method (one-hot in this case) increases the dimensionality, which can dilute the impact of each individual feature unless the network architecture and training process are carefully managed. The neural network's ability to adapt to the encoded features of the Titanic dataset, combined with appropriate regularization (lambda) and network architecture adjustments, highlights its capacity to model complex relationships in mixed-type data. The key to optimizing performance lies in balancing model complexity with the ability to generalize from training data to unseen data, avoiding both overfitting and underfitting.

8. Figure 11,12

epochs	k_fold_count	learning_rate	lambda	dimensions	accuracy	f1_score
1000	10	0.8	0.01	[102, 2, 5, 2]	0.7924327545114062	0.7790488900550151
1000	10	0.8	0.01	[102, 2, 3, 8, 2]	0.7810563500170241	0.7458292061523294
1000	10	0.8	0.04	[102, 2, 5, 2]	0.7991493587561004	0.785343507120066
1000	10	0.8	0.04	[102, 2, 3, 8, 2]	0.766372148450 8	0.7286997930027208
1000	10	0.8	0.07	[102, 2, 5, 2]	0.7968896833503575	0.7834804263690422
1000	10	0.8	0.07	[102, 2, 3, 8, 2]	0.7924197026444217	0.7786898055311717

Figure 11: Titanic Dataset - Table

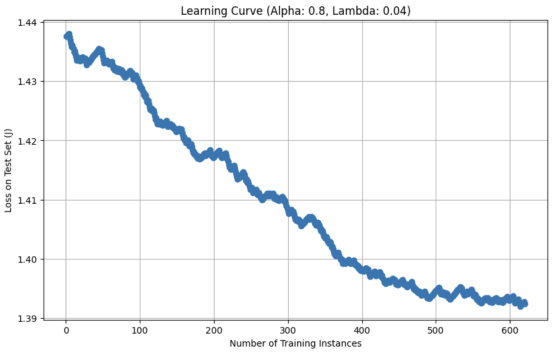


Figure 12: Titanic Dataset - Curve

3 Loan Dataset

The dataset encompasses both categorical and numerical attributes, rendering it suitable for algorithms such as KNN, neural networks, and random forests, which excel in handling diverse data types. Given that loan approval hinges on intricate interplays among factors like income, credit history, and marital status, algorithms like neural networks and random forests, renowned for capturing non-linear relationships and intricate patterns, prove fitting for this task. In real-world scenarios, datasets often harbor noise and outliers, yet KNN and random forests remain resilient, capable of delivering reasonable predictions despite such data imperfections. While neural networks

excel in unraveling complex relationships, their interpretability may be lacking, contrasting with the transparency afforded by KNN and random forests, a virtue particularly valued in finance-related domains. The selection of KNN, neural networks, and random forests strikes a balance between interpretability, robustness, and the ability to discern complex patterns within the loan dataset. In contrast, Naive Bayes, despite its effectiveness in certain scenarios, may not be as well-suited due to its assumption of feature independence, a presumption that may not hold in this context.

3.1 KNN

1. Figure 13,14,15,16

	k_val_1	k_val_7	k_val_13	k_val_19	k_val_25	k_val_31	k_val_37	k_val_43	k_val_49
0	70.212766	80.851064	74.468085	72.340426	72.340426	74.468085	72.340426	70.212766	70.212766
1	69.387755	83.673469	77.551020	77.551020	79.591837	77.551020	77.551020	75.510204	73.469388
2	70.833333	83.333333	79.166667	77.083333	72.916667	72.916667	68.750000	70.833333	68.750000
3	73.469388	75.510204	71.428571	73.469388	73.469388	75.510204	75.510204	75.510204	75.510204
4	64.583333	72.916667	72.916667	72.916667	70.833333	70.833333	70.833333	70.833333	70.833333
5	58.333333	79.166667	72.916667	79.166667	79.166667	79.166667	79.166667	77.083333	77.083333
6	68.085106	78.723404	74.468085	74.468085	72.340426	74.468085	74.468085	74.468085	74.468085
7	68.750000	81.250000	81.250000	81.250000	75.000000	75.000000	72.916667	70.833333	70.833333
8	77.083333	77.083333	70.833333	75.000000	70.833333	66.666667	72.916667	75.000000	70.833333
9	70.833333	79.166667	75.000000	77.083333	77.083333	77.083333	77.083333	72.916667	72.916667
Mean	69.157168	79.167481	74.999910	76.032892	74.357541	74.366406	74.153640	73.320126	72.491044

Figure 13: Loan Dataset - Accuracy

	k_val_1	k_val_7	k_val_13	k_val_19	k_val_25	k_val_31	k_val_37	k_val_43	k_val_49
0	0.668347	0.754783	0.618919	0.484388	0.484388	0.548077	0.484388	0.412500	0.412500
1	0.616588	0.765550	0.687536	0.640906	0.685897	0.640906	0.640906	0.591667	0.537400
2	0.647059	0.764128	0.684211	0.639098	0.535369	0.535369	0.407407	0.475000	0.407407
3	0.693308	0.591667	0.477134	0.537400	0.537400	0.591667	0.591667	0.591667	0.591667
4	0.595037	0.573479	0.573479	0.573479	0.521368	0.475000	0.475000	0.475000	0.475000
5	0.515152	0.684211	0.628350	0.684211	0.684211	0.684211	0.684211	0.639098	0.639098
6	0.625995	0.682432	0.587719	0.587719	0.533231	0.548077	0.548077	0.548077	0.548077
7	0.629439	0.742704	0.742704	0.742704	0.621053	0.589744	0.535369	0.475000	0.475000
8	0.737965	0.702535	0.587224	0.646192	0.557895	0.494737	0.573479	0.621053	0.521368
9	0.647059	0.684211	0.621053	0.639098	0.639098	0.639098	0.639098	0.535369	0.535369
Mean	0.637595	0.694570	0.620833	0.617519	0.579991	0.574688	0.557960	0.536443	0.514288

Figure 14: Loan Dataset - F1 Score

2. The training accuracy does not reach 100% for k = 1 because of the rounding of floating point value, additionally because of one hot encoding, most of the instances have similar attributes. As k increases, more neighbors are considered, initially improving accuracy since the model gains more diverse insights. However, as k continues to increase, accuracy tends to decline. This is because a higher k means the target value is influenced by more neighbors, potentially introducing noise or irrelevant data points.

   The stability of the model across different subsets of data is indicated by the histogram for training accuracy, which shows low variance. When testing the model on the training data, a similar pattern emerges: accuracy rises initially with increasing k, then declines. However, the training accuracy consistently surpasses the testing accuracy because the model is trained on the training data, highlighting its ability to generalize.

   The high standard deviation suggests that the model's performance may vary significantly across different runs or subsets of the data.

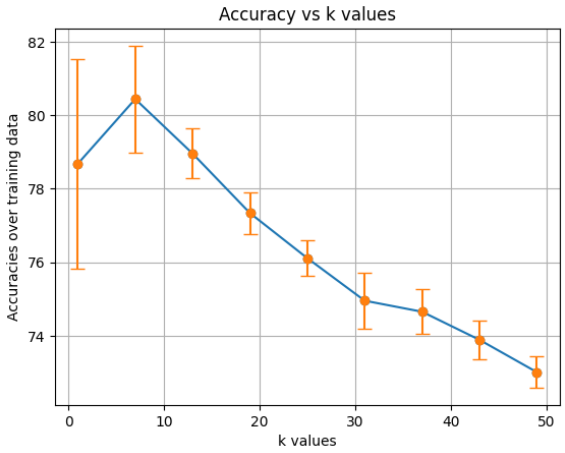


Figure 15: Loan Dataset - Training Accuracy

When $k \geq 45$, both the training and testing accuracy are relatively low. This suggests that the model is too simple and unable to capture the underlying patterns in the data. Therefore, it is reasonable to characterize this range as underfitting.

At the k value of 31, the model demonstrates a commendable testing accuracy, suggesting its ability to generalize effectively to unseen data. Similarly, the training accuracy remains reasonably high, indicating that the model adequately captures the underlying patterns present in the dataset. This k value strikes a balance between bias and variance, making it a fitting choice for the model.

3.2 Random Forest

For this algorithm, I have tried three different hyperparameters:

- 75% threshold 50 min instances information gain
- 85% threshold 20 min instances information gain
- 95% threshold 5 min instances information gain

The three figures summarise the performance of the random forest on this dataset, with the third hyperparameter setting working the best. Thus, I have displayed the accuracy and F1 score graphs for the third hyperparameter setting.

1. Figure 17,18,19,20,21
2. We can see that as the number of trees (ntree) increases, there is a general trend of improvement in all performance metrics. However, beyond a certain point, the improvements become marginal. Accuracy: The accuracy increases steadily with the increase in the number of trees, indicating that the classifier's overall correctness improves as more trees are added to the ensemble. Recall: Similarly, the recall improves with more trees, suggesting that the classifier

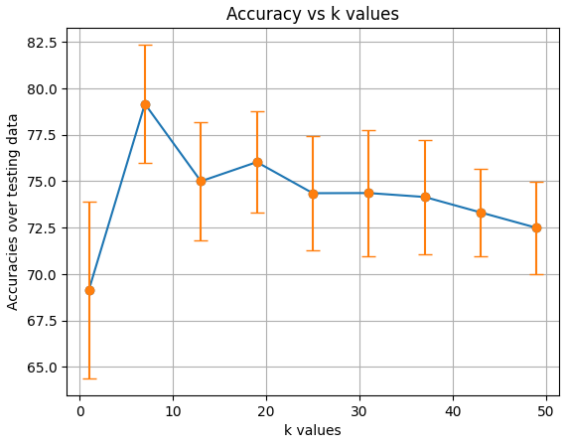


Figure 16: Loan Dataset - Testing Accuracy

	Trees	Accuracy	Recall	Precision	F1 Score
0	1	73.756151	0.620871	0.717237	0.628084
1	5	76.447659	0.640241	0.763914	0.646588
2	10	78.557045	0.665153	0.832062	0.677962
3	20	79.798723	0.683855	0.847794	0.703444
4	30	80.423723	0.693855	0.852654	0.715749
5	40	80.632056	0.699007	0.847884	0.721352
6	50	80.215389	0.692340	0.844044	0.713804

Figure 17: Loan Dataset - Hyperparameter 1

becomes better at identifying positive instances from the dataset. Precision: Precision also improves with more trees, indicating that the classifier makes fewer false positive predictions. F1 Score: The F1 score, which balances precision and recall, also shows improvement with increasing ntree, although the improvement becomes less significant as ntree grows larger.

Considering these trends, if I were to deploy this classifier in real life, I would select the value of ntree where the performance metrics start to plateau or show diminishing returns. In this case, it seems that around ntree = 30, the performance improvements become less substantial, and the computational cost of adding more trees may not justify the marginal gains in performance. Therefore, ntree = 30 would be reasonable choices for deployment.

Overall, increasing the number of trees in the random forest generally improves performance across all metrics. However, there is a point of diminishing returns where adding more trees does not significantly enhance performance but may increase computational complexity.

3.3 Neural Networks

For this algorithm, I have tried six different architectures:

	Trees	Accuracy	Recall	Precision	F1 Score
0	1	70.214847	0.630774	0.675241	0.645801
1	5	79.386127	0.701762	0.795089	0.719872
2	10	79.364687	0.697189	0.798213	0.716002
3	20	80.002623	0.701823	0.812079	0.721528
4	30	80.206705	0.701520	0.823426	0.721939
5	40	80.623191	0.710437	0.826172	0.731276
6	50	80.410606	0.703184	0.830619	0.724561

Figure 18: Loan Dataset - Hyperparameter 2

	Trees	Accuracy	Recall	Precision	F1 Score
0	1	68.972807	0.626855	0.683903	0.648375
1	5	73.951549	0.665900	0.709190	0.678856
2	10	74.993215	0.678930	0.715447	0.686462
3	20	78.956343	0.712995	0.775428	0.729660
4	30	79.572659	0.708316	0.790746	0.727968
5	40	78.535425	0.697148	0.777625	0.715184
6	50	79.151740	0.703467	0.784380	0.722298

Figure 19: Loan Dataset - Hyperparameter 3

- Architecture 1:- Layers: [22, 13, 2], Regularization Param: 0
- Architecture 2:- Layers: [22, 13, 2], Regularization Param: 0.1
- Architecture 3:- Layers: [22, 8, 9, 2], Regularization Param: 0
- Architecture 4:- Layers: [22, 8, 9, 5, 2], Regularization Param: 0
- Architecture 5:- Layers: [22, 8, 9, 5, 2], Regularization Param: 0.1
- Architecture 6:- Layers: [22, 12, 18, 13, 15, 2], Regularization Param: 0

For this dataset, regularization has no major impact on the accuracy or the F1 Scores. Regularization techniques are employed to prevent overfitting by penalizing large weights in the model and improve generalization but since this particular dataset is not complex enough, it ends up decreasing the accuracy a little. Additionally, it may also be performing badly because of the irrelevant features in the training data. Increasing the number of layers in this dataset has resulted in a decline in performance. When comparing architectures 2, 3, and 4, it's evident that as we increase the number of layers, there's a consistent decrease in performance. This suggests that for this particular dataset, architectures with fewer layers and more neurons tend to outperform deeper architectures with fewer neurons. A comparison between architectures 2 and 4 highlights a significant drop in both accuracy and F1 score with architecture 4, indicating that there's a point where constructing and training more complex networks doesn't enhance performance and may even degrade it. This phenomenon is likely attributed to overfitting, where the model becomes too specialized to the training data, hindering its ability to generalize to unseen data.

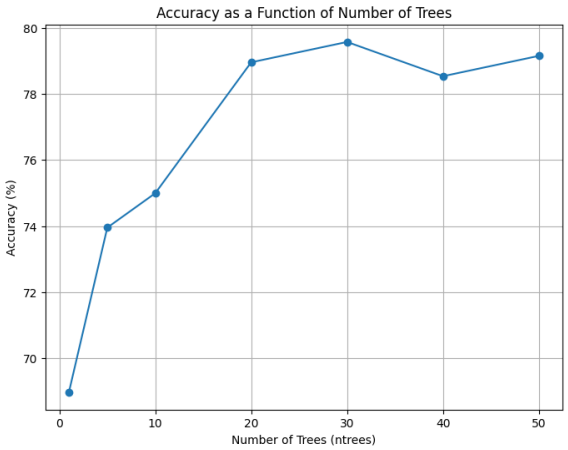


Figure 20: Loan Dataset - Hyperparameter 3 Accuracy Graph

In light of these observations, architecture 1 emerges as the preferred choice for deployment in real-life scenarios. Architecture 1 consistently demonstrates the highest average accuracy and F1 score among the provided architectures, striking a balance between depth and width to achieve improved performance without unnecessary complexity. Despite not employing regularization, architecture 1 outperforms architectures with similar or even higher complexity that utilize regularization techniques. This indicates that the inherent design of architecture 1 may be robust enough to mitigate overfitting without relying on additional regularization methods. Furthermore, architecture 1 exhibits consistent performance across both accuracy and F1 score metrics, suggesting its stability and reliability across various evaluation criteria.

I have displayed the learning curve using architecture 1.

1. Figure 22,23,24
2. From the figure we can see that as we train more and more instances the cost reduces and eventually flattens. The batch size used here is 120 and learning rate is 0.02

4 Parkinson Dataset

Several factors influenced the decision to evaluate the Parkinson's disease dataset using KNN, neural networks, and random forests rather than Naive Bayes or other techniques. To begin, the dataset is entirely composed of numerical attributes, making it ideal for algorithms such as KNN, neural networks, and random forests, all of which can handle such data types. Furthermore, the diagnosis of Parkinson's disease frequently requires complicated interactions between numerous voice measurements, which neural networks and random forests excel at capturing because to their capacity to represent non-linear connections. Furthermore, KNN and random forests are known for their ability to withstand noise and outliers, which are common in biomedical datasets. This robustness ensures that forecasts are accurate even in the presence of faulty data. Furthermore, while neural networks may be less interpretable than KNN and random forests, their capacity to

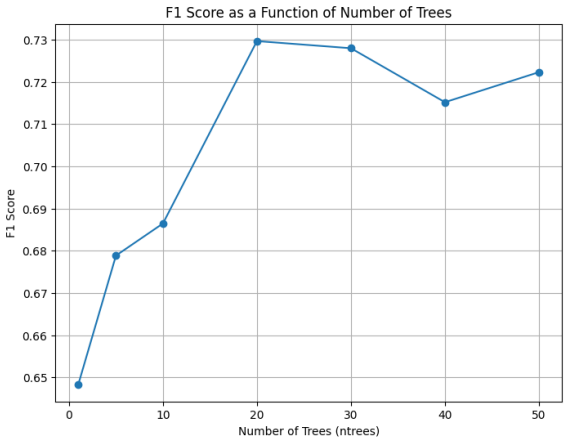


Figure 21: Loan Dataset - Hyperparameter 3 F1 Score Graph

	Architecture 1	Architecture 2	Architecture 3	Architecture 4	Architecture 5	Architecture 6
0	100.0	100.000000	100.000000	70.212766	70.212766	100.000000
1	100.0	100.000000	100.000000	68.750000	68.750000	93.750000
2	100.0	97.916667	91.666667	68.750000	68.750000	89.583333
3	100.0	100.000000	100.000000	69.387755	69.387755	69.387755
4	100.0	100.000000	100.000000	70.212766	70.212766	70.212766
5	100.0	100.000000	91.666667	89.583333	68.750000	68.750000
6	100.0	100.000000	93.750000	68.750000	68.750000	68.750000
7	100.0	100.000000	95.833333	68.750000	68.750000	68.750000
8	100.0	100.000000	100.000000	68.750000	68.750000	68.750000
9	100.0	100.000000	100.000000	69.387755	69.387755	95.918367
Mean	100.0	99.791667	97.291667	71.253438	69.170104	79.385222

Figure 22: Loan Dataset - Accuracy

detect nuanced patterns in data is especially useful in identifying diseases with diverse presentations such as Parkinson's. In contrast, Naive Bayes is based on the premise of feature independence, which may not be valid in this scenario, thereby restricting its usefulness. Thus, the use of KNN, neural networks, and random forests provides a holistic strategy to studying the Parkinson's disease dataset, utilizing their individual strengths in numerical data processing, complicated relationship capture, and robust diagnosis prediction.

4.1 KNN

1. Figure 25,26,27,28

2. The training accuracy reaches 100% for k = 1 because the model essentially compares each instance with its own data record, inevitably resulting in perfect matches. As k increases, more neighbors are considered, initially improving accuracy since the model gains more diverse insights. However, as k continues to increase, accuracy tends to decline. This is because a higher k means the target value is influenced by more neighbors, potentially introducing noise or irrelevant data points.

   The stability of the model across different subsets of data is indicated by the histogram for training accuracy, which shows low variance. When testing the model on the training data,

	Architecture 1	Architecture 2	Architecture 3	Architecture 4	Architecture 5	Architecture 6
0	1.0	1.000000	1.000000	0.412500	0.412500	1.000000
1	1.0	1.000000	1.000000	0.407407	0.407407	0.922705
2	1.0	0.976179	0.894505	0.407407	0.407407	0.864789
3	1.0	1.000000	1.000000	0.409639	0.409639	0.409639
4	1.0	1.000000	1.000000	0.412500	0.412500	0.412500
5	1.0	1.000000	0.894505	0.864789	0.407407	0.407407
6	1.0	1.000000	0.922705	0.407407	0.407407	0.407407
7	1.0	1.000000	0.949580	0.407407	0.407407	0.407407
8	1.0	1.000000	1.000000	0.407407	0.407407	0.407407
9	1.0	1.000000	1.000000	0.409639	0.409639	0.950000
Mean	1.0	0.997618	0.966130	0.454610	0.408872	0.618926

Figure 23: Loan Dataset - F1 Score

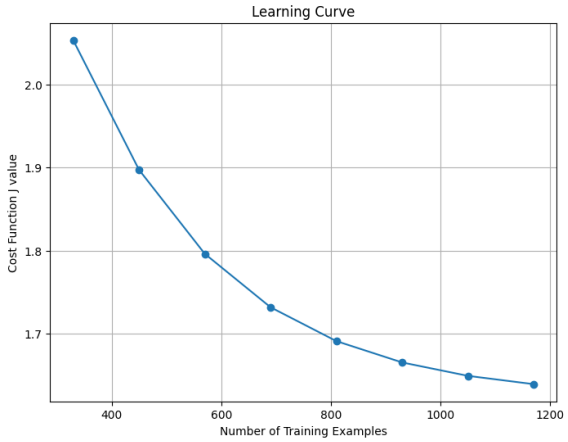


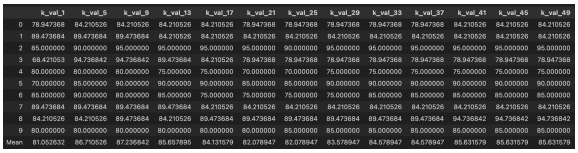
Figure 24: Loan Dataset - Learning Curve

a similar pattern emerges: accuracy rises initially with increasing k, then declines. However, the training accuracy consistently surpasses the testing accuracy because the model is trained on the training data, highlighting its ability to generalize.

The high standard deviation suggests that the model's performance may vary significantly across different runs or subsets of the data.

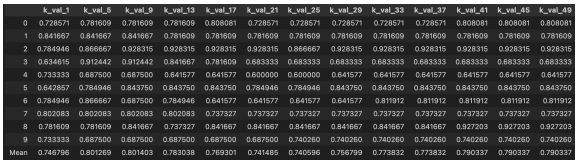
When $k \geq 25$, both the training and testing accuracy are relatively low. This suggests that the model is too simple and unable to capture the underlying patterns in the data. Therefore, it is reasonable to characterize this range as underfitting. When k= 1 the training accuracy is significantly higher than the testing accuracy. This suggests that the model is memorizing the training data too closely and is not generalizing well to new, unseen data. Therefore, it is reasonable to characterize these ranges as overfitting.

At the k value of 41, the model demonstrates a commendable testing accuracy, suggesting its ability to generalize effectively to unseen data. Similarly, the training accuracy remains reasonably high, indicating that the model adequately captures the underlying patterns present in the dataset. This k value strikes a balance between bias and variance, making it a fitting choice for the model.



	k_val_1	k_val_5	k_val_9	k_val_13	k_val_17	k_val_21	k_val_25	k_val_29	k_val_33	k_val_37	k_val_41	k_val_45	k_val_49
0	78.947368	84.210526	84.210526	84.210526	84.210526	78.947368	78.947368	78.947368	78.947368	78.947368	84.210526	84.210526	84.210526
1	89.473684	89.473684	89.473684	84.210526	84.210526	84.210526	84.210526	84.210526	84.210526	84.210526	84.210526	84.210526	84.210526
2	85.000000	90.000000	95.000000	95.000000	95.000000	95.000000	90.000000	95.000000	95.000000	95.000000	95.000000	95.000000	95.000000
3	68.421053	94.736842	94.736842	89.473684	84.210526	76.947368	76.947368	76.947368	76.947368	76.947368	76.947368	76.947368	76.947368
4	80.000000	80.000000	80.000000	75.000000	75.000000	70.000000	70.000000	75.000000	75.000000	75.000000	75.000000	75.000000	75.000000
5	70.000000	85.000000	90.000000	90.000000	90.000000	85.000000	85.000000	90.000000	90.000000	90.000000	90.000000	90.000000	90.000000
6	85.000000	90.000000	80.000000	85.000000	75.000000	75.000000	75.000000	75.000000	85.000000	85.000000	85.000000	85.000000	85.000000
7	89.473684	89.473684	89.473684	89.473684	84.210526	84.210526	84.210526	84.210526	84.210526	84.210526	84.210526	84.210526	84.210526
8	84.210526	84.210526	89.473684	84.210526	89.473684	89.473684	89.473684	89.473684	89.473684	89.473684	94.736842	94.736842	94.736842
9	90.000000	90.000000	90.000000	90.000000	90.000000	90.000000	85.000000	85.000000	85.000000	85.000000	85.000000	85.000000	85.000000
Mean	81.052632	86.710526	87.236842	85.657895	84.131579	82.078947	82.078947	83.578947	84.578947	84.578947	85.631579	85.631579	85.631579

Figure 25: Parkinson Dataset - Accuracy



	k_val_1	k_val_5	k_val_9	k_val_13	k_val_17	k_val_21	k_val_25	k_val_29	k_val_33	k_val_37	k_val_41	k_val_45	k_val_49
0	0.728571	0.781609	0.781609	0.781609	0.808081	0.728571	0.728571	0.728571	0.728571	0.728571	0.808081	0.808081	0.808081
1	0.841667	0.841667	0.841667	0.781609	0.781609	0.781609	0.781609	0.781609	0.781609	0.781609	0.781609	0.781609	0.781609
2	0.784946	0.866667	0.928315	0.928315	0.928315	0.928315	0.866667	0.928315	0.928315	0.928315	0.928315	0.928315	0.928315
3	0.634615	0.912442	0.912442	0.841667	0.781609	0.683333	0.683333	0.683333	0.683333	0.683333	0.683333	0.683333	0.683333
4	0.733333	0.687500	0.687500	0.641577	0.641577	0.600000	0.600000	0.641577	0.641577	0.641577	0.641577	0.641577	0.641577
5	0.642857	0.784946	0.843750	0.843750	0.843750	0.784946	0.784946	0.843750	0.843750	0.843750	0.843750	0.843750	0.843750
6	0.784946	0.866667	0.687500	0.784946	0.641577	0.641577	0.641577	0.641577	0.811912	0.811912	0.811912	0.811912	0.811912
7	0.803083	0.803083	0.803083	0.803083	0.737327	0.737327	0.737327	0.737327	0.737327	0.737327	0.737327	0.737327	0.737327
8	0.781609	0.781609	0.841667	0.737327	0.841667	0.841667	0.841667	0.841667	0.841667	0.841667	0.937203	0.937203	0.937203
9	0.733333	0.687500	0.687500	0.687500	0.687500	0.687500	0.740260	0.740260	0.740260	0.740260	0.740260	0.740260	0.740260
Mean	0.746796	0.801269	0.801403	0.783038	0.769301	0.741485	0.740596	0.756799	0.773832	0.773832	0.790337	0.790337	0.790337

Figure 26: Parkinson Dataset - F1 Score

4.2 Random Forest

For this algorithm, I have tried three different hyperparameters:

- 85% threshold 25 min instances information gain
- 95% threshold 15 min instances gini
- 95% threshold 15 min instances information gain

The three figures summarise the performance of the random forest on this dataset, with the third hyperparameter setting working the best. Thus, I have displayed the accuracy and F1 score graphs for the third hyperparameter setting.

1. Figure 29,30,31,32,33
2. We observe a trend of performance improvement across all metrics as the number of trees (ntree) increases. However, beyond a certain point, the improvements become marginal.

   Accuracy: The accuracy steadily increases with the increase in the number of trees, indicating overall improvement in the classifier's correctness. Recall: Recall also improves with more trees, indicating better identification of positive instances (healthy individuals) from the dataset. Precision: Precision generally improves with more trees, suggesting fewer false positive predictions. F1 Score: The F1 score, which balances precision and recall, shows improvement with increasing ntree, although the rate of improvement diminishes as ntree grows larger. To select the optimal value of ntree for deployment in real life, we would again look for the point where the performance metrics start to plateau or show diminishing returns. In this case, it appears that around ntree = 30 or ntree = 40, the performance improvements become less substantial, similar to the previous dataset. Therefore, either ntree = 30 or ntree = 40 would be reasonable choices for deployment.

   Overall, increasing the number of trees in the random forest generally improves performance across all metrics, but there comes a point of diminishing returns where adding more trees does not significantly enhance performance but may increase computational complexity.

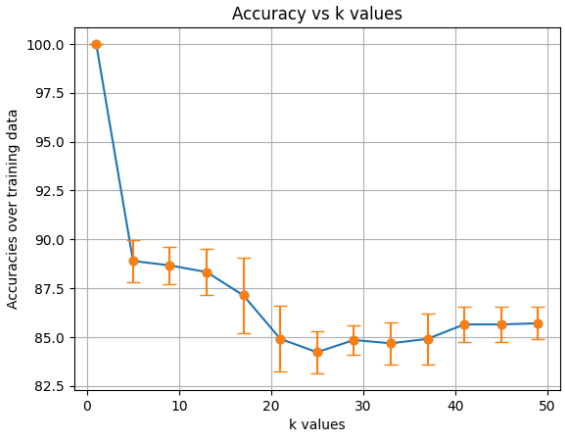


Figure 27: Parkinson Dataset - Training Accuracy

4.3 Neural Networks

For this algorithm, I have tried five different architectures:

- Architecture 1:- Layers: [22, 15, 2], Regularization Param: 0
- Architecture 2:- Layers: [22, 15, 2], Regularization Param: 0.1
- Architecture 3:- Layers: [22, 8, 9, 2], Regularization Param: 0
- Architecture 4:- Layers: [22, 8, 9, 5, 2], Regularization Param: 0
- Architecture 5:- Layers: [22, 12, 18, 13, 15, 2], Regularization Param: 0

Regularization helps prevent overfitting by penalizing large weights in the network. In this case, architectures with a regularization parameter of 0.1 outperformed those with lower regularization values, indicating that stronger regularization can lead to better generalization. When comparing, architecture 1 and architecture 2 - we can see a clear increase in accuracies and f1 scores. Increasing the number of layers in this dataset has resulted in a decline in performance. When comparing architectures 1, 3, and 4, it's evident that as we increase the number of layers, there's a consistent decrease in performance. This suggests that for this particular dataset, architectures with fewer layers and more neurons tend to outperform deeper architectures with fewer neurons. A comparison between architectures 2 and 5 highlights a significant drop in both accuracy and F1 score with architecture 5, indicating that there's a point where constructing and training more complex networks doesn't enhance performance and may even degrade it. This phenomenon is likely attributed to overfitting, where the model becomes too specialized to the training data, hindering its ability to generalize to unseen data.

In light of these observations, architecture 2 emerges as the preferred choice for deployment in real-life scenarios. Architecture 2 consistently demonstrates the highest average accuracy and F1 score among the provided architectures, striking a balance between depth and width to achieve

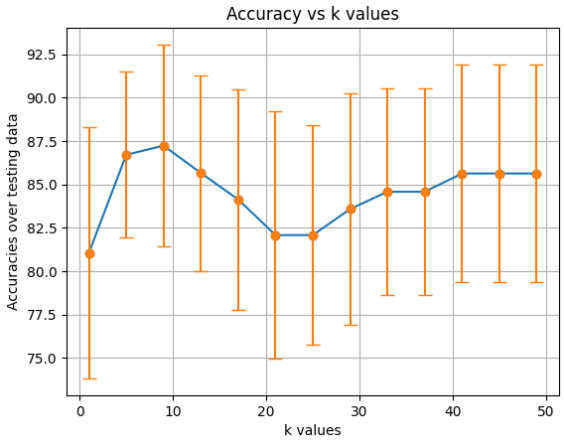


Figure 28: Parkinson Dataset - Testing Accuracy

	Trees	Accuracy	Recall	Precision	F1 Score
0	1	81.763158	0.742976	0.779254	0.740104
1	5	84.289474	0.753929	0.817870	0.767140
2	10	89.312865	0.800833	0.916814	0.832463
3	20	86.260234	0.758333	0.864472	0.783766
4	30	88.368421	0.803333	0.887904	0.823374
5	40	86.286550	0.758333	0.879302	0.783808
6	50	86.815789	0.757500	0.895502	0.786703

Figure 29: Parkinson Dataset - Hyperparameter 1

improved performance without unnecessary complexity. Despite not employing regularization, architecture 2 outperforms architectures with similar or even higher complexity that utilize regularization techniques. This indicates that the inherent design of architecture 2 may be robust enough to mitigate overfitting without relying on additional regularization methods. Furthermore, architecture 2 exhibits consistent performance across both accuracy and F1 score metrics, suggesting its stability and reliability across various evaluation criteria.

I have displayed the learning curve using architecture 2.

1. Figure 34,35,36
2. From the figure we can see that as we train more and more instances the cost reduces and eventually flattens. The batch size used here is 20 and learning rate is 0.1

5 EXTRA CREDIT DATASET: UCI Heart Disease Data

This dataset is a multivariate type, meaning it involves various separate mathematical or statistical variables and pertains to the analysis of multivariate numerical data. It comprises 14 attributes,

	Trees	Accuracy	Recall	Precision	F1 Score
0	1	83.681287	0.790119	0.817602	0.782767
1	5	83.654971	0.767143	0.808567	0.774889
2	10	86.786550	0.783690	0.857868	0.799363
3	20	88.315789	0.794167	0.894014	0.818997
4	30	91.394737	0.853929	0.914534	0.871543
5	40	89.342105	0.820833	0.893255	0.841764
6	50	88.842105	0.810833	0.887274	0.828865

Figure 30: Parkinson Dataset - Hyperparameter 2

	Trees	Accuracy	Recall	Precision	F1 Score
0	1	83.988304	0.787976	0.798950	0.780966
1	5	89.339181	0.849524	0.878671	0.849596
2	10	86.230994	0.795119	0.834238	0.800957
3	20	89.286550	0.811429	0.904223	0.833903
4	30	90.368421	0.840833	0.904296	0.858220
5	40	90.894737	0.844167	0.904407	0.860902
6	50	89.839181	0.828333	0.896171	0.848731

Figure 31: Parkinson Dataset - Hyperparameter 3

including age, sex, chest pain type, resting blood pressure, serum cholesterol, fasting blood sugar, resting electrocardiographic results, maximum heart rate achieved, exercise-induced angina, ST depression induced by exercise relative to rest (oldpeak), the slope of the peak exercise ST segment, number of major vessels, and Thalassemia.

Although the dataset contains 76 attributes, research studies have focused solely on a subset of 14 attributes. Among these attributes, 5 are categorical, while the remainder are numerical.

The objective with this dataset is to predict whether a given individual has heart disease based on the provided attributes.

Several factors influenced the choice of machine learning algorithms—K-Nearest Neighbors (KNN), Neural Networks, and Random Forests—over Naive Bayes for heart disease prediction. These algorithms offer versatility in handling both numerical and categorical data, aligning well with the multivariate nature of the dataset. Given the potentially complex and nonlinear relationships between input features and heart disease presence, Neural Networks and Random Forests excel in capturing such complexities, leading to more accurate predictions.

Furthermore, KNN, Neural Networks, and Random Forests demonstrate robustness to noise and outliers commonly found in medical datasets, enhancing the reliability of predictions compared to Naive Bayes, which assumes feature independence. Their flexibility enables them to capture intricate patterns within datasets with multiple attributes and potential interactions. Additionally, the ensemble nature of Random Forests mitigates overfitting and enhances predictive performance, particularly beneficial for datasets with numerous attributes and complex relationships.

While Naive Bayes offers simplicity and interpretability, it may not effectively capture the dataset's complex relationships, unlike KNN, Neural Networks, and Random Forests, which prior-

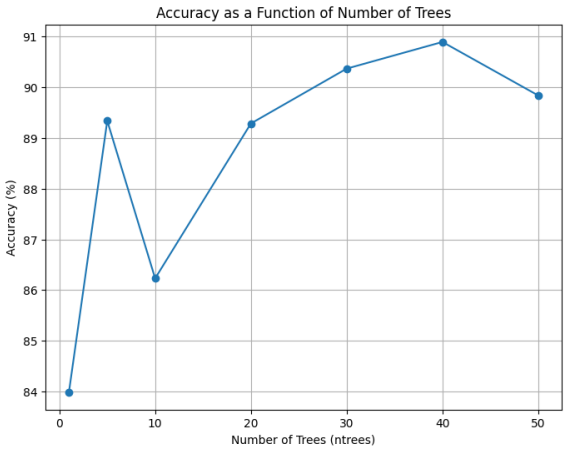


Figure 32: Parkinson Dataset - Hyperparameter 3 Accuracy Graph

itize predictive accuracy. Therefore, despite sacrificing some interpretability, these algorithms offer superior performance in capturing the intricate patterns present in the heart disease dataset.

5.1 KNN

1. Figure 37,38,39,40

2. The training accuracy does not reache 100% for k = 1 because of the rounding of floating point value, additionally because of one hot encoding, most of the instances have similar attributes. As k increases, more neighbors are considered, initially improving accuracy since the model gains more diverse insights. However, as k continues to increase, accuracy tends to decline. This is because a higher k means the target value is influenced by more neighbors, potentially introducing noise or irrelevant data points.

   The stability of the model across different subsets of data is indicated by the histogram for training accuracy, which shows low variance. When testing the model on the training data, a similar pattern emerges: accuracy rises initially with increasing k, then declines. However, the training accuracy consistently surpasses the testing accuracy because the model is trained on the training data, highlighting its ability to generalize.

   The high standard deviation suggests that the model's performance may vary significantly across different runs or subsets of the data.

   When k= 1 the training accuracy is significantly higher than the testing accuracy. This suggests that the model is memorizing the training data too closely and is not generalizing well to new, unseen data. Therefore, it is reasonable to characterize these ranges as overfitting.

   At the k value of 43, the model demonstrates a commendable testing accuracy, suggesting its ability to generalize effectively to unseen data. Similarly, the training accuracy remains reasonably high, indicating that the model adequately captures the underlying patterns present in the dataset. This k value strikes a balance between bias and variance, making it a fitting choice for the model.

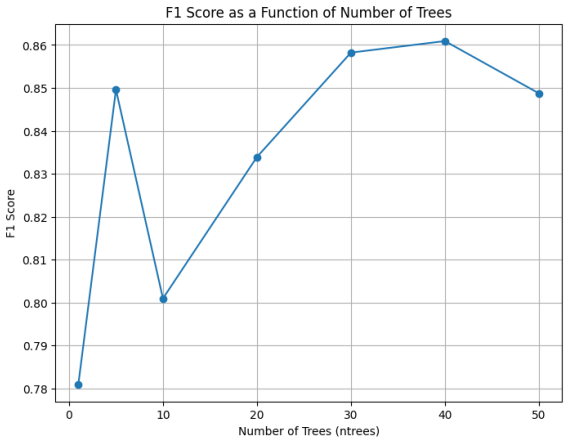


Figure 33: Parkinson Dataset - Hyperparameter 3 F1 Score Graph

	Architecture 1	Architecture 2	Architecture 3	Architecture 4	Architecture 5
0	90.000000	90.000000	75.000000	75.000000	75.000000
1	68.421053	73.684211	73.684211	73.684211	73.684211
2	75.000000	75.000000	75.000000	75.000000	75.000000
3	10.000000	10.000000	0.000000	0.000000	0.000000
4	95.000000	95.000000	75.000000	75.000000	75.000000
5	95.000000	95.000000	75.000000	75.000000	75.000000
6	75.000000	85.000000	75.000000	75.000000	75.000000
7	88.888889	88.888889	77.777778	77.777778	77.777778
8	94.736842	94.736842	73.684211	73.684211	73.684211
9	73.684211	73.684211	84.210526	78.947368	78.947368
Mean	76.573099	78.099415	68.435673	67.909357	67.909357

Figure 34: Parkinson Dataset - Accuracy

5.2 Random Forest

For this algorithm, I have tried three different hyperparameters:

- 75% threshold 50 min instances information gain
- 95% threshold 5 min instances information gain
- 95% threshold 10 min instances gini

The three figures summarise the performance of the random forest on this dataset, with the third hyperparameter setting working the best. Thus, I have displayed the accuracy and F1 score graphs for the third hyperparameter setting.

1. Figure 41,42,43,44,45
2. We can see that as the number of trees (ntree) increases, there is a general trend of improvement in all performance metrics. However, beyond a certain point, the improvements become marginal. Accuracy: The accuracy increases steadily with the increase in the number of trees, indicating that the classifier's overall correctness improves as more trees are added to

	Architecture 1	Architecture 2	Architecture 3	Architecture 4	Architecture 5
0	0.843750	0.843750	0.428571	0.428571	0.428571
1	0.406250	0.562212	0.424242	0.424242	0.424242
2	0.428571	0.428571	0.428571	0.428571	0.428571
3	0.181818	0.181818	0.000000	0.000000	0.000000
4	0.928315	0.928315	0.428571	0.428571	0.428571
5	0.928315	0.928315	0.428571	0.428571	0.428571
6	0.641577	0.784946	0.428571	0.428571	0.428571
7	0.800000	0.800000	0.437500	0.437500	0.437500
8	0.927203	0.927203	0.424242	0.424242	0.424242
9	0.562212	0.562212	0.654545	0.441176	0.441176
Mean	0.664801	0.694734	0.408339	0.387002	0.387002

Figure 35: Parkinson Dataset - F1 Score

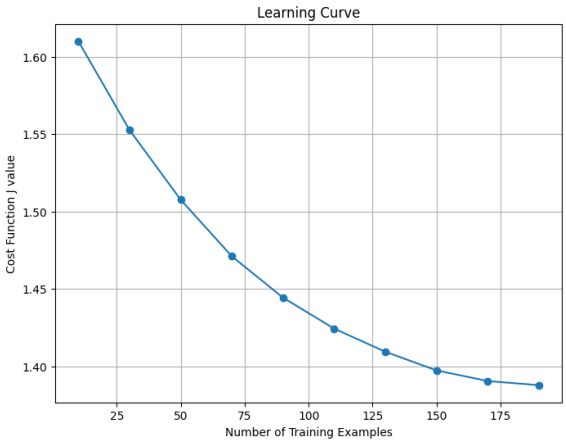


Figure 36: Parkinson Dataset - Learning Curve

the ensemble. The other metrics for this model are very poor, suggesting the model is likely making predictions that are not much better than random guessing. A very low precision indicates that the model is making a large number of false positive predictions. A very low recall indicates that the model is failing to correctly identify a large number of positive instances. A very low F1 score suggests that both precision and recall are low, indicating poor performance overall.

Considering these trends, if I were to deploy this classifier in real life, I would select the value of ntree where the accuracy metrics start to plateau or show diminishing returns. In this case, it seems that around ntree = 40, the accuracy improvements become less substantial, and the computational cost of adding more trees may not justify the marginal gains in performance. Therefore, ntree = 40 would be reasonable choices for deployment.

Overall, increasing the number of trees in the random forest generally improves performance across all metrics. However, there is a point of diminishing returns where adding more trees does not significantly enhance performance but may increase computational complexity.

	k_val_1	k_val_7	k_val_13	k_val_19	k_val_25	k_val_31	k_val_37	k_val_43	k_val_49
0	58.064516	64.516129	61.290323	70.967742	64.516129	58.064516	61.290323	61.290323	61.290323
1	58.064516	58.064516	51.612903	45.161290	48.387097	48.387097	51.612903	51.612903	54.838710
2	48.275862	68.965517	65.517241	55.172414	58.620690	55.172414	55.172414	62.068966	55.172414
3	56.666667	56.666667	56.666667	53.333333	53.333333	60.000000	53.333333	56.666667	60.000000
4	51.612903	61.290323	61.290323	54.838710	54.838710	58.064516	54.838710	61.290323	64.516129
5	53.571429	60.714286	60.714286	64.285714	60.714286	60.714286	60.714286	60.714286	57.142857
6	41.935484	54.838710	58.064516	54.838710	54.838710	54.838710	61.290323	61.290323	61.290323
7	53.333333	66.666667	60.000000	56.666667	60.000000	56.666667	60.000000	60.000000	66.666667
8	56.666667	56.666667	66.666667	56.666667	56.666667	56.666667	56.666667	56.666667	53.333333
9	67.857143	67.857143	64.285714	57.142857	60.714286	60.714286	64.285714	64.285714	60.714286
Mean	54.604852	61.624662	60.610864	56.907410	57.262991	56.928916	57.920467	59.588617	59.496504

Figure 37: Heart Dataset - Accuracy

	k_val_1	k_val_7	k_val_13	k_val_19	k_val_25	k_val_31	k_val_37	k_val_43	k_val_49
0	0.334872	0.370909	0.349198	0.481925	0.412026	0.339341	0.314286	0.316117	0.316667
1	0.407566	0.389305	0.246364	0.171111	0.184127	0.182121	0.191228	0.188664	0.197436
2	0.248889	0.434248	0.380280	0.220915	0.261818	0.215873	0.211429	0.351429	0.211429
3	0.448214	0.290000	0.304100	0.266840	0.253506	0.306061	0.212162	0.256607	0.303926
4	0.460153	0.420469	0.358058	0.230116	0.211111	0.239316	0.223701	0.265281	0.279640
5	0.358299	0.303810	0.285972	0.302973	0.235088	0.239640	0.239640	0.235088	0.168421
6	0.185621	0.199190	0.270769	0.244103	0.230769	0.230769	0.337436	0.347209	0.314828
7	0.306006	0.408095	0.307895	0.208547	0.290000	0.204444	0.271111	0.271111	0.367143
8	0.298586	0.315729	0.415873	0.315817	0.248526	0.252638	0.252638	0.252638	0.215038
9	0.388235	0.519640	0.257310	0.164103	0.225564	0.218421	0.287912	0.268421	0.210000
Mean	0.343644	0.365139	0.317582	0.260645	0.255254	0.242862	0.254154	0.275256	0.258453

Figure 38: Heart Dataset - F1 Score

5.3 Neural Networks

For this algorithm, I have tried six different architectures:

- Architecture 1:- Layers: [35, 13, 5], Regularization Param: 0
- Architecture 2:- Layers: [35, 13, 5], Regularization Param: 0.1
- Architecture 3:- Layers: [35, 8, 9, 5], Regularization Param: 0
- Architecture 4:- Layers: [35, 8, 9, 5, 5], Regularization Param: 0.1
- Architecture 5:- Layers: [35, 8, 9, 5, 5], Regularization Param: 0
- Architecture 6:- Layers: [35, 12, 18, 13, 15, 5], Regularization Param: 0

Regularization helps prevent overfitting by penalizing large weights in the network. In this case, architectures with a regularization parameter of 0.1 outperformed those with lower regularization values, indicating that stronger regularization can lead to better generalization. When comparing, architecture 1 and architecture 2 - we can see a clear increase in accuracies and f1 scores. Increasing the number of layers in this dataset has resulted in a decline in performance. When comparing architectures 2, 3, and 4, it's evident that as we increase the number of layers, there's a consistent decrease in performance. This suggests that for this particular dataset, architectures with fewer layers and more neurons tend to outperform deeper architectures with fewer neurons. A comparison between architectures 2 and 4 highlights a significant drop in both accuracy and F1 score with architecture 4, indicating that there's a point where constructing and training more complex networks doesn't enhance performance and may even degrade it. This phenomenon is likely attributed to overfitting, where the model becomes too specialized to the training data, hindering its ability to generalize to unseen data.

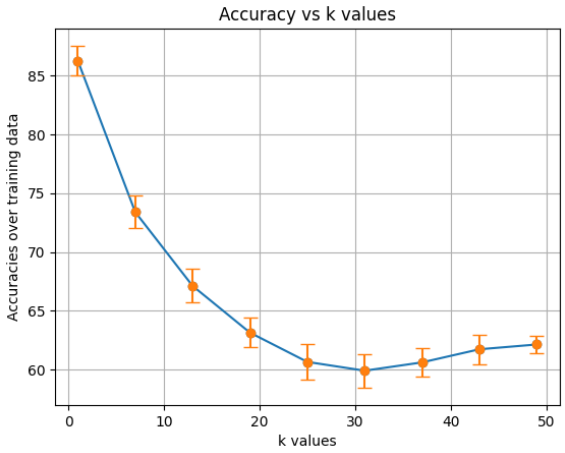


Figure 39: Heart Dataset - Training Accuracy

In light of these observations, architecture 2 emerges as the preferred choice for deployment in real-life scenarios. Architecture 2 consistently demonstrates the highest average accuracy and F1 score among the provided architectures, striking a balance between depth and width to achieve improved performance without unnecessary complexity. Furthermore, architecture 2 exhibits consistent performance across both accuracy and F1 score metrics, suggesting its stability and reliability across various evaluation criteria.

I have displayed the learning curve using architecture 2.

1. Figure 46,47,48
2. From the figure we can see that as we train more and more instances the cost reduces and eventually flattens. The batch size used here is 120 and learning rate is 0.1

6 Results

Most of these datasets have a class imbalance which explains why neural networks performs the best. KNN can be sensitive to class imbalance because it relies on majority voting among nearest neighbors. In datasets with imbalanced classes, the majority class may dominate the decision-making process, leading to biased predictions. Random Forest is less affected by class imbalance compared to KNN because it builds multiple decision trees and averages their predictions. However, it can still be influenced by class imbalance, especially if one class is significantly underrepresented. Neural Networks are generally robust to class imbalance due to their ability to learn complex relationships from data. They can automatically adjust their weights during training to focus more on minority class instances if necessary.

K-Nearest Neighbors (KNN) faces challenges when applied to categorical datasets, particularly when datasets have multiple classes and significant class imbalance. This is because KNN relies on measuring the similarity between instances, which might be less effective in high-dimensional and categorical feature spaces. However, KNN can still demonstrate reasonable performance on

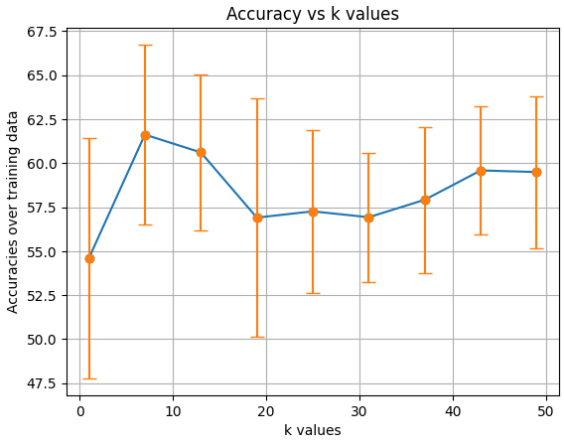


Figure 40: Heart Dataset - Testing Accuracy

	Trees	Accuracy	Recall	Precision	F1 Score
0	1	54.887213	0.045000	0.055667	0.047351
1	5	59.569684	0.010000	0.006667	0.008000
2	10	57.213362	0.045667	0.070580	0.051410
3	20	57.826868	0.000000	0.000000	0.000000
4	30	58.838362	0.010000	0.010000	0.010000
5	40	57.868534	0.010000	0.020000	0.013333
6	50	57.565374	0.000000	0.000000	0.000000

Figure 41: Heart Dataset - Hyperparameter 1

numerical datasets, especially when the feature space is not too high-dimensional like parkinson dataset.

On the other hand, Random Forest exhibits robust performance on categorical datasets due to its ability to handle high-dimensional feature spaces and capture nonlinear relationships effectively. It can efficiently model complex interactions between categorical variables without extensive preprocessing. Moreover, Random Forest can also perform well on numerical datasets, particularly when nonlinear relationships exist between features. It can seamlessly handle both numerical and categorical features without requiring extensive preprocessing steps.

In contrast, Neural Networks excel in capturing intricate patterns and interactions within categorical datasets. They possess the capability to automatically learn hierarchical representations from categorical data without the need for manual feature engineering. Additionally, Neural Networks are well-suited for numerical datasets, where complex nonlinear relationships and intricate patterns may exist. They can adapt to various data distributions and effectively learn from numerical features.

	Trees	Accuracy	Recall	Precision	F1 Score
0	1	49.496408	0.050667	0.033333	0.036333
1	5	56.518678	0.044833	0.056471	0.048606
2	10	53.445402	0.000000	0.000000	0.000000
3	20	59.893678	0.070250	0.063627	0.064805
4	30	58.783046	0.010000	0.020000	0.013333
5	40	59.127874	0.066500	0.075556	0.065994
6	50	59.162356	0.010000	0.020000	0.013333

Figure 42: Heart Dataset - Hyperparameter 2

	Trees	Accuracy	Recall	Precision	F1 Score
0	1	46.150246	0.000000	0.000000	0.000000
1	5	54.732941	0.000000	0.000000	0.000000
2	10	60.226604	0.132167	0.144912	0.128476
3	20	56.218348	0.056083	0.067333	0.057647
4	30	58.388252	0.064000	0.064444	0.063824
5	40	59.265818	0.064000	0.085238	0.065393
6	50	59.656851	0.016667	0.025000	0.019048

Figure 43: Heart Dataset - Hyperparameter 3

7 Distribution

1. Handwriting and Titanic Dataset have been done using *Shreya's* codes.
2. Loan, Parkinson and Heart Datasets have been done using *Vaishnavi's* codes.

	Handwriting		Titanic		Loan		Parkinson		Heart	
	Accuracy	F1-Score	Accuracy	F1-Score	Accuracy	F1-Score	Accuracy	F1-Score	Accuracy	F1 Score
KNN	97.76	0.973	77.34	0.760	79.16	0.694	83.61	0.693	61.62	0.365
Random Forest	96.66	0.965	82.85	0.818	80.40	0.728	91.39	0.871	60.22	0.128
Neural Network	97.44	0.974	79.91	0.785	100.0	1.0	78.09	0.694	94.35	0.758

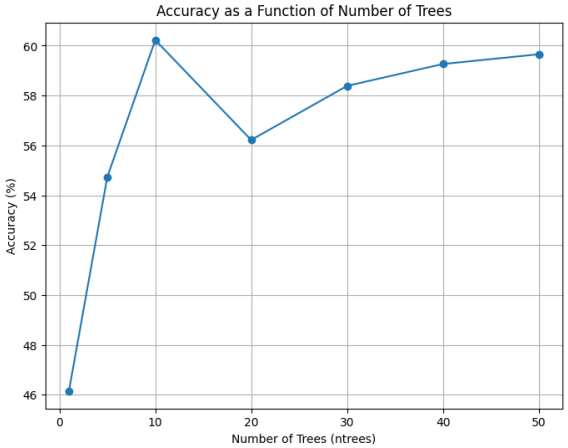


Figure 44: Heart Dataset - Hyperparameter 3 Accuracy Graph

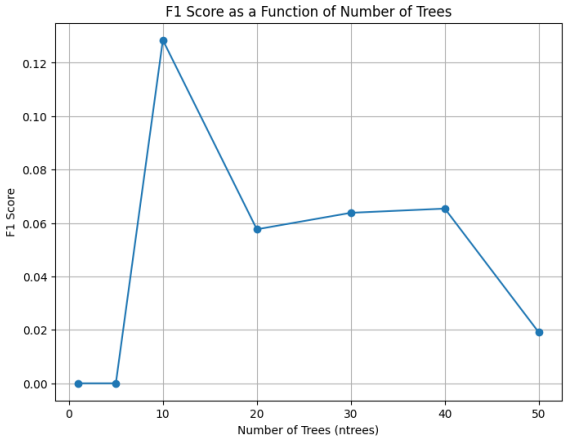


Figure 45: Heart Dataset - Hyperparameter 3 F1 Score Graph

	Architecture 1	Architecture 2	Architecture 3	Architecture 4	Architecture 5	Architecture 6
0	96.551724	96.551724	62.068966	55.172414	55.172414	62.068966
1	96.666667	96.666667	66.666667	53.333333	53.333333	53.333333
2	83.333333	96.666667	66.666667	53.333333	53.333333	63.333333
3	93.548387	93.548387	67.741935	51.612903	51.612903	51.612903
4	93.333333	93.333333	60.000000	56.666667	60.000000	53.333333
5	90.322581	87.096774	61.290323	51.612903	51.612903	51.612903
6	96.551724	96.551724	75.862069	75.862069	55.172414	55.172414
7	93.333333	93.333333	70.000000	53.333333	53.333333	60.000000
8	93.103448	93.103448	72.413793	55.172414	55.172414	55.172414
9	90.000000	96.666667	73.333333	53.333333	53.333333	73.333333
Mean	92.674453	94.351872	67.604375	55.943270	54.207638	57.897293

Figure 46: Heart Dataset - Accuracy

	Architecture 1	Architecture 2	Architecture 3	Architecture 4	Architecture 5	Architecture 6
0	0.771429	0.771429	0.247876	0.142222	0.142222	0.230116
1	0.793939	0.781818	0.282828	0.139130	0.139130	0.139130
2	0.626190	0.793939	0.282828	0.139130	0.139130	0.257857
3	0.778555	0.771429	0.293498	0.136170	0.136170	0.136170
4	0.731717	0.727273	0.226306	0.193333	0.232381	0.139130
5	0.709596	0.682035	0.284103	0.136170	0.136170	0.136170
6	0.771429	0.784615	0.327273	0.332857	0.142222	0.142222
7	0.738555	0.742857	0.401905	0.139130	0.139130	0.225564
8	0.766667	0.753247	0.311111	0.142222	0.142222	0.142222
9	0.667949	0.777778	0.320000	0.139130	0.139130	0.321569
Mean	0.735603	0.758642	0.297773	0.163950	0.148791	0.187015

Figure 47: Heart Dataset - F1 Score

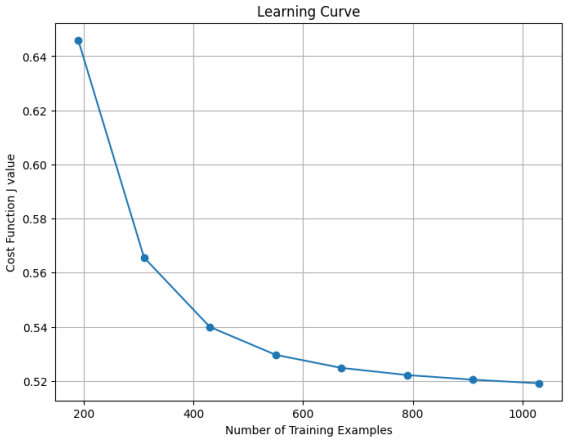


Figure 48: Heart Dataset - Learning Curve