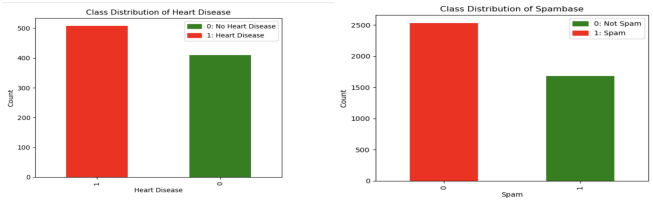
Supervised Learning

Nayana Saji
CS-7641
nsaji6@gatech.edu
903967749

***Abstract*—As part of this assignment, different algorithms such as Neural Networks, Support Vector Machine and KNN are compared and contrasted with each other using 2 datasets - Heart Disease Dataset and Spambase dataset.**

I. Datasets



(a) Validation curve - k vs F1 score (b) Validation Curve: Distance Metric vs F1 Score

Fig. 1: Class Distribution of Heart Disease and Spambase Database

A. Heart Disease Dataset

The Heart Disease dataset provides medical information on 918 patients, featuring variables like age, cholesterol levels, blood pressure, smoking habits, and whether they experience exercise-induced angina, etc. The primary goal is to predict a patient's risk of heart disease, therefore it is a binary classification problem. It is a mix of categorical and numerical features. Fig 1.a, shows that it has a balanced class distribution where 55% of the cases had heart disease and 45% did not.

This dataset is interesting because of its real-world uses in healthcare. Heart disease is one of the leading causes of death globally, and the ability to accurately predict its risk can significantly impact early diagnosis and treatments. Along with this, this dataset is a mix of categorical and numerical features, which increases complexities in the preprocessing of the data and it providing an opportunity to observe how different algorithms handle these complexities.

B. Spambase Dataset

The Spambase dataset consists of 4,601 emails, labeled as spam or non-spam, and contains 57 features that capture various characteristics of the emails, such as the frequency of specific words and characters. The primary goal is to predict, given the features whether an email is spam or not, making it a binary classification problem. Fig 1.b, shows that this dataset is on the noisier side and exhibits class imbalance, with only 40% of the cases being spam and the remaining 60% being legitimate emails.

This dataset is interesting because it is used by every single person in the world who has an email. It has direct implications for security and user experience. Along with this, this dataset has a very large number of features (57 in total), making it an excellent candidate for testing the robustness of different algorithms in handling high-dimensional data and subtle patterns that differentiate spam from legitimate emails.

II. Performance Metrics

There are several metrics, like F1 score, accuracy, precision, and recall, to evaluate algorithm performance.

For the heart disease dataset, both false positives and false negatives have serious consequences, so the F1 score was chosen as it is an average of precision and recall, minimizing both types of errors.

In the Spambase dataset, the class distribution is imbalanced, with more non-spam than spam emails. It can be seen from Fig 1.b. F1 score was chosen as it is an average of precision and recall, making it ideal for assessing the model's performance on the minority class (spam).

III. Data Pre-processing

A. Heart Disease Dataset

- There were no missing or duplicate values in this dataset, so no rows or columns were removed from this dataset.
- This dataset has 5 categorical features: Sex, ChestPainType, RestingECG, ExerciseAngina, ST_Slope. OneHotEncoder was used to convert each category into binary values. One-hot encoding helps the model understand categorical data without making it seem like there's any natural order or ranking between the categories, which could sometimes mislead the model.
- For the rest of the numerical features: Age, RestingBP, Cholesterol, FastingBS, MaxHR, Oldpeak, were standardized using StandardScaler. Standardizing numerical features with StandardScaler make sure all the values are on the same scale, which helps machine learning models work better.

B. Spambase Dataset

- There were no missing values in this dataset, so no fillings were done.
- 391 rows were duplicates, these rows were dropped from the dataset

- This dataset contained only numerical values, and all the numerical values were standardized using StandardScaler.

IV. Hypothesis

A. Heart Disease Dataset

- **K-Nearest Neighbors** - KNN is expected to perform well on the Heart Disease Dataset. Since this dataset has a balanced class distribution, it is extremely favorable for the KNN algorithm. The dataset also has fewer features thereby reducing the risk of the curse of dimensionality.
- **Support Vector Machines** - Support Vector Machines are expected to perform at a high level on the Heart Disease dataset. SVMs work well on with small to medium-sized datasets and excel in binary classification tasks. Therefore it should perform exceptionally on this dataset.
- **Neural Networks** - Neural Networks is expected to perform reasonably well, it should be able to effectively capture the relationship between all the features, however since the dataset is quite small it may over-fit and perform worse on test data, therefore it may not surpass SVM for this dataset.

B. Spambase Dataset

- **K-Nearest Neighbors** - KNN is expected to have low performance on this dataset. This dataset has high dimensionality(57 features) and the size of the dataset (4601 data points) due to which the KNN algorithm can have the curse of dimensionality, along with that it will be computationally expensive. Therefore KNN will underperform compared to the other algorithms.
- **Support Vector Machines** - SVM is expected to have moderate to high performance on this dataset. Using different kernels, SVMs should be able to handle high-dimensional dataset and model non-linear relationships.
- **Neural Networks** - Neural Networks are expected to perform at a high level on this dataset. They are perfect for high-dimensional datasets and they should be able to capture, complex, non-linear relationships between each of the features. Since the dataset is large, there is a reduced chance of overfitting. Neural Networks should effectively outperform all the other algorithms.

V. K-Nearest Neighbors

Hyperparameter tuning was done by generating validation curves for the number of neighbors - k , distance metrics, and weight options, and learning curves were plotted for both datasets. KNNClassifier() from sklearn was used. Validation curve for each hyperparameter vs f1 score was plotted.

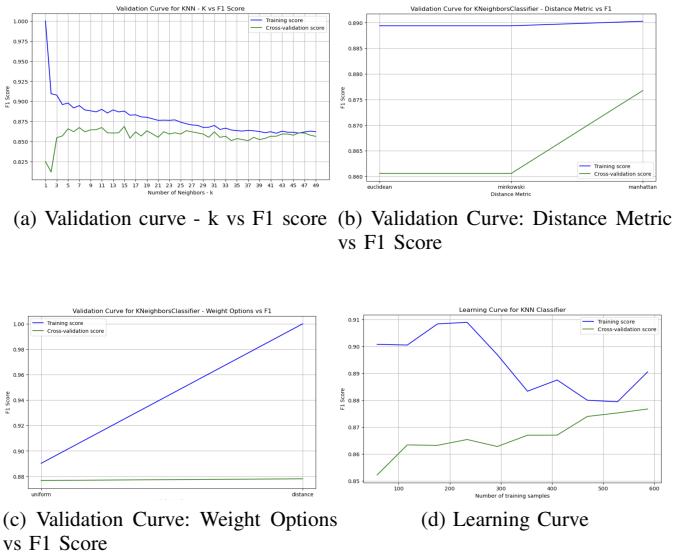
The most critical hyperparameter is k. A low k may lead to overfitting, while a high k can cause underfitting, so finding the optimal k is key.

Next, the distance metric (Euclidean, Manhattan, Minkowski) was tuned, as it defines how closeness between neighbors is measured.

Lastly, weight options (uniform, distance-based) were optimized since they affect how much influence each neighbor has on the predictions.

The hyperparameters were tuned in the same sequence as above once the optimal previous hyperparameter is found.

A. Heart Disease Dataset



(a) Validation curve - k vs F1 score (b) Validation Curve: Distance Metric vs F1 Score

(c) Validation Curve: Weight Options vs F1 Score (d) Learning Curve

Fig. 2: KNN - Heart Disease Validation Curves and Learning Curve

a) Validation curve - k vs F1 score: Fig 2.a shows that as k decreases, the model tends to overfit, this can be clearly seen when k is less than 3. And the model underfits for higher values of k. The cross-validation score stabilizes around K=11 to K=15. This range offers the best generalization to unseen data. 13 is chosen as the value of k since it fits into this range.

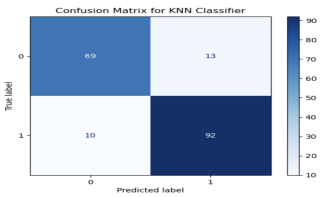
b) Validation Curve: Distance Metric vs F1 Score: The distance metrics significantly impact the performance of this model. The model's performance was evaluated based on these 3 distance metrics: Euclidean, Manhattan and Minkowski. From Fig 2.b it can be seen that Manhattan distance provides the highest cross-validation score compared to Euclidean and Minkowski. Therefore Manhattan distance is chosen as the distance metric.

c) Validation Curve: Weight Options vs F1 Score: The weight option affects the model's performance by determining how much influence each neighbor has. Two options were validated on the datsets: uniform (equal weighting) and distance-based (closer neighbors have more influence). Figure 2.c shows that distance-based weighting improved the training score, but it had little effect on the cross-validation score, meaning it didn't help generalization. Therefore, it's better to use the default uniform option, as the other option doesn't improve generalization.

d) Learning Curve Analysis: The learning curve shows how the number of training samples affects KNN. In Fig 2.d,

the training score starts high at 0.9 and fluctuates, mostly due to the dataset's complexity. The cross-validation score starts low at 0.86 but improves as number of samples increase. The initial gap between training and cross-validation scores indicates overfitting with fewer samples. As more data is introduced, this gap decreases, showing the model generalizes better and overfits less with a larger dataset.

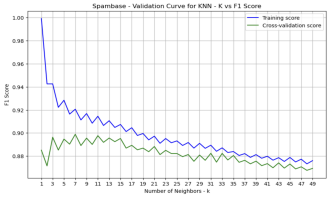
*e) **Hypothesis Evaluation**:* Fig 3 shows the confusion matrix for KNN on the testing data. There are 92 true positives, 69 true negatives, 13 false positives, and 10 false negatives. It has an F1 score of 0.8747, which is a reasonably good score, it indicates a balanced precision and recall performance. The confusion matrix shows a reasonable balance in predicting both classes and a low number of false negatives and false negatives indicating that the model is not biased toward either of the classes. The **initial hypothesis holds true**. This can be seen from the confusion matrix and the F1 score. The balanced class distribution and smaller dataset size seem to contribute well to the KNNs performance. And since there are only 11 features, the KNN did not suffer too much from the curse of dimensionality and performed well.



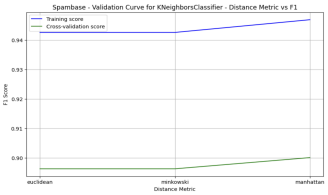
(a) Confusion Matrix

Fig. 3: KNN - Heart Disease and Confusion Matrix

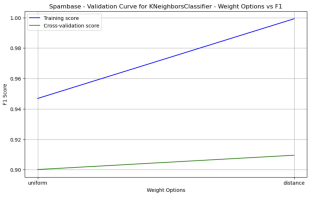
B. Spambase Dataset



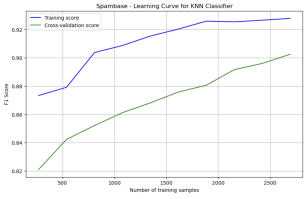
(a) Validation curve - k vs F1 score



(b) Validation Curve: Distance Metric vs F1 Score



(c) Validation Curve: Weight Options vs F1 Score



(d) Learning Curve

Fig. 4: KNN - Spambase Validation Curves and Learning Curve

a) Validation Curve - k vs F1 score: Fig 4.a This dataset also follow a trend similar to that off the heart disease dataset as expected. For lower values of k the model overfit and for high values of k the model underfits. The F1 score for the cross-validation set stabilizes from k = 7 to k = 15, indicating that a k in this range will offer the best generalization on unseen data. k = 11, is selected as the optimal value since it balances between overfitting and underfitting.

b) Validation Curve: Distance Metric vs F1 Score: In Fig 4.b the validation curve shows the effect of different distance metrics on the model's performance. The three distance metrics evaluated—Euclidean, Manhattan, and Minkowski. Fig 4.b shows that Manhattan distance performs best, providing the highest cross-validation and training score. Therefore, the Manhattan distance metric is chosen.

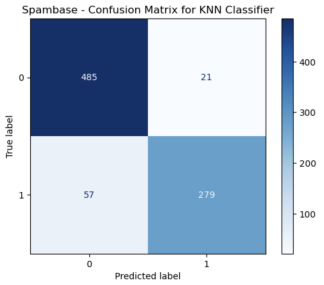
c) Validation Curve: Weight Options vs F1 Score: Fig 4.c this curve shows the impact of different weight options (uniform, distance) on the F1 score. Distance-based weighting improves the training score significantly, but the cross-validation score shows minimal improvement, indicating that the model does not generalize better with distance-based weighting. Therefore it is better to stick to the uniform weight options since they balance both the training and cross-validation performance.

d) Learning Curve Analysis: Fig 4.d shows the learning curve for the KNN: Number of sample vs F1 score. The training score starts at a 1.0, indicating overfitting when the number of samples are lower. The cross-validation score starts around 0.85 and gradually improves as the number of training samples increases. The gap between the training and cross-validation scores decreases with as the number of samples increases, indicating better generalization. Therefore, increasing the number of samples improves the model's performance and generalization.

*e) **Hypothesis Evaluation**:* Fig 5 shows the confusion matrix for the KNN classifier on the test set. The matrix shows 487 true negatives, 279 true positives, 21 false positives, and 57 false negatives. The results indicate the model performs well in classifying both classes and is not biased toward either. It also has an F1 score 0.90. Therefore, the **initial hypothesis is proven false**, it was hypothesized that KNN is expected to have low performance on this dataset since the dataset has high dimensionality, and therefore KNN may suffer from the curse of dimensionality. This could be because this dataset has a large number of samples, which might have help the model overcome the curse of dimensionality, allowing KNN to better capture the underlying patterns. Along with this the distance metric that was chosen was Manhattan distance, it computes the absolute difference in each dimension and then sums it up. This leads to a gradual increase in distance as dimensionality grows, therefore also help KNN overcome the curse of dimensionality.

VI. Support Vector Machine

Multiple validation curves were generated for kernel types, regularisation parameter C, degree of polynomial kernels, and



(a) Confusion Matrix

Fig. 5: KNN - Spambase Confusion Matrix

gamma values for hyperparameter tuning. SVC() from sklearn was used.

The most important hyperparameters here are the kernel function and the regularisation parameter C. This is because the kernel function and the regularization parameter work together to create the decision boundary in a Support Vector Machine. Multiple validation curves were generated with a combination of different kernel types and different C values against F1 score.

Once the optimal kernel function and C is determined, the next hyperparameter is the degree of the polynomial. It determines the complexity of the decision boundary formed by the polynomial kernel. A higher degree may lead to overfitting, while a lower one leads to underfitting. This validation curve was only plotted for the heart disease dataset since it used the polynomial kernel.

The final hyperparameter that was tuned was gamma. Gamma controls how much influence a single training example has. A low gamma means a larger influence and could lead to underfitting. While a high gamma means a more localized influence and could lead to overfitting. This validation curve was only plotted for the heart disease dataset since it used the polynomial kernel.

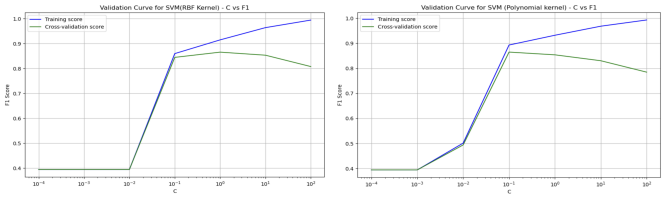
SVM has a lot of hyperparameters, so a few hyperparameter such coef0 which is important in the case of heart disease dataset was found using the grid search, it is set to be 1.

A. Heart Disease Dataset

a) Validation Curve - Kernel and C value vs F1 score: Fig 6 show the validation curve of different kernel types and ranges of C vs F1 score.

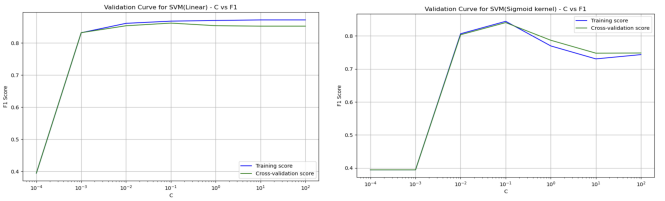
RBF Kernel and C vs F1 score: Fig 6.a shows that after C = 0.1, the model starts overfitting, with the gap between training and cross-validation scores increases. At higher C values, the model overfits by focusing too much on correct classifications. When C is less than 0.1, both scores are low due to high regularization, causing underfitting. For the RBF kernel, C = 0.1 is ideal, giving a balanced F1 score of around 0.85 without overfitting or underfitting.

Polynomial Kernel and C vs F1 score: Fig 6.b is similar to RBF Kernel and C vs F1 score. Both start to overfit post C = 0.1 and underfit below C = 0.1. However, this kernel



(a) Validation curve - RBF Kernel and C vs F1 score

(b) Validation Curve: Polynomial Kernel and C vs F1 score



(c) Validation Curve: Linear Kernel and C vs F1 score

(d) Validation Curve: Sigmoid Kernel and C vs F1 score

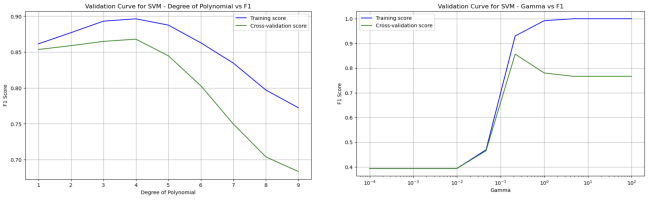
Fig. 6: SVM - Heart Disease - Kernel and C value vs F1 score

gives a better F1 score of about 0.89 for training and 0.87 for cross-validation data.

Linear Kernel and C vs F1 score: Fig 6.c shows that the appropriate C for this kernel is 0.1 as well. For C lower than 0.001 the model underfits. And for C greater than 0.1 the model somewhat overfits, but not as much as RBF or Polynomial. It can be seen that F1 score for training and cross-validation sets remains at around 0.85.

Sigmoid Kernel and C vs F1 score: Fig 6.d shows that the appropriate C for this kernel is 0.1 as well. For C lower than 0.001 the model underfits. And for C greater than 0.1 the model overfits significantly causing the training and cross-validation scores to decrease significantly. It can be seen that the F1 score for training and cross-validation sets remains at around 0.85.

Based on the observations from Fig 6 and the F1 score. The polynomial kernel, with C = 0.1 was chosen.



(a) Validation Curve - Degree of Polynomial vs F1 score

(b) Validation Curve - Gamma vs F1 score

Fig. 7: SVM - Heart Disease Validation Curves - Degree of Polynomial and Gamma

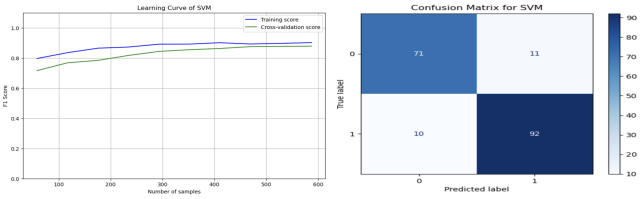
b) Validation Curve - Degree of Polynomial vs F1 score: Fig 7.a shows that post degree 4, the F1 scores of both training and cross-validation scores decrease. This could be because at higher degrees the polynomial kernel produces

an overly complex model that reduces the margin between support vectors, leading to less generalization. Degrees 3 and 4 show a similar cross-validation and training F1 score. However, at degree 4, this is a slightly larger gap between the cross-validation and training scores, therefore degree 3 is chosen.

c) Validation Curve - Gamma vs F1 score: Fig 7.b shows that post a gamma of 0.01 both the F1 scores of cross-validation and training set start increases. However, post gamma of 0.1008, the training score increases while the cross-validation score decreases, which indicates that the model has started to overfit. This is because, with a higher gamma, each training sample has a more localized influence, leading to overfitting and less generalization.

d) Learning Curve Analysis: Fig 8.a shows that the training scores remain consistently high, increasing slightly as the number of samples increases. This indicates that the model is learning from the data. The cross-validation scores increase as the number of samples increases. Therefore, as the number of samples increases the model is able to generalize and learn better.

*e) **Hypothesis Evaluation**:* Fig 8.b shows the confusion matrix for SVM on the testing data. There are 92 true positives, 71 true negatives, 10 false positives, and 11 false negatives. It has an F1 score of 0.8857, which is a reasonably good score, it indicates a balanced precision and recall performance. The **initial hypothesis does not hold true**. This is because it was initially hypothesized to perform at a high level. Though SVM does give a reasonably good F1 score, it performs better than KNN and Neural networks but only marginally. This could be because SVM was unable to capture the dataset's complexity and create a solid boundary to distinguish between the classes. The polynomial kernel was either too complex or wasn't complex enough to map the data into a higher-dimensional space where it could be better separated



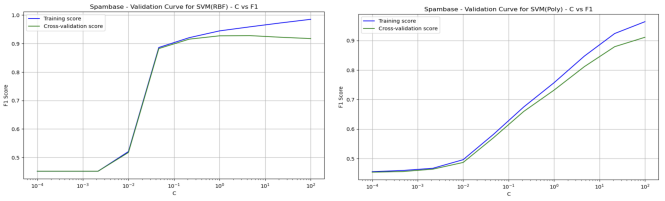
(a) Learning Curve (b) Confusion Matrix

Fig. 8: SVM - Heart Disease Learning Curve and Confusion Matrix

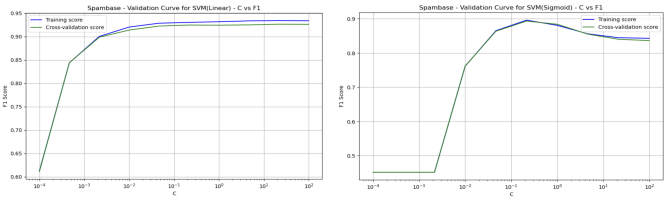
B. Spambase Dataset

a) Validation Curve - Kernel and C value vs F1 score: Fig 9 show the validation curve of different kernel types and ranges of C vs F1 score.

RBF Kernel and C vs F1 score: Fig 9.a shows that after C = 1, the model starts overfitting, with the training F1 score nearing 0.9 and a increasing gap between training and



(a) Validation curve - RBF Kernel and C vs F1 score (b) Validation Curve: Polynomial Kernel and C vs F1 score



(c) Validation Curve: Linear Kernel and C vs F1 score (d) Validation Curve: Sigmoid Kernel and C vs F1 score

Fig. 9: SVM - Spambase - Kernel and C value vs F1 score

cross-validation scores. Higher C values lead to overfitting by focusing too much on correct classification. Below C = 0.015, the model underfits due to high regularization, causing low F1 scores. For the RBF kernel, C = 0.15 is ideal, giving an F1 score close to 0.92 without underfitting or overfitting.

Polynomial Kernel and C vs F1 score: Fig 9.b shows that the polynomial kernel's F1 score improves as C increases, with both the training and cross-validation scores increasing consistently. However, it can be seen that as C increases the gap between the training and cross-validation scores increases, indicating potential overfitting.

Linear Kernel and C vs F1 score: Fig 9.c presents a linear kernel's performance as a function of C. The linear kernel provides a more stable performance, achieving high training and cross-validation scores even at lower values of C = 0.1. This suggests that for this dataset, a linear decision boundary may be sufficient, as increasing C does not lead to overfitting.

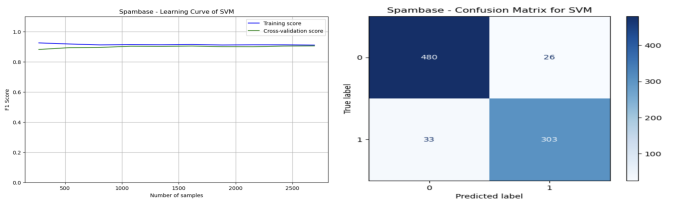
Sigmoid Kernel and C vs F1 score: Fig 9.d shows the sigmoid kernel performance across different C values. Similar to other kernels, both scores improve with increasing C, but model loses its ability to generalize at high C values which causes the training and cross-validation scores to drop down. For this kernel, C = 0.15 is ideal since it gives high training and cross-validation scores

Based on the observations from Fig 9 and the F1 score. The linear kernel, with C = 0.1 was chosen.

b) Learning Curve Analysis: Fig 10.a shows the SVM learning curve on the Spambase dataset. The training score stays high near 1.0, while the cross-validation score gradually improves and stabilizes around 0.9. The small gap between them indicates good generalization without overfitting. The minimal improvement with more data suggests the model reaches its generalization early. This could be because the linear kernel was able to create a clear boundary between

different classes.

c) Hypothesis Evaluation: Fig 10.b shows the confusion matrix for the SVM classifier using the RBF kernel on the test set. There are 480 true negatives, 303 true positives, 26 false positives, and 33 false negatives, resulting in an F1 score of 0.9298. The model performs well in classifying both spam and non-spam messages, with relatively low misclassification. Therefore, the **initial hypothesis holds true**. It was hypothesized that SVM would have moderate to high performance on this dataset due to its ability to handle high-dimensional data effectively, which is reflected in the confusion matrix and F1 score. This is because the linear kernel was able to successfully create boundaries between the two classes.



(a) Learning Curve (b) Confusion Matrix

Fig. 10: SVM - Spambase Learning Curve and Confusion Matrix

VII. Neural Networks

Since neural networks have multiple hyperparameters, all of which cannot be explored, a random search was initially performed to determine the appropriate solver, number of iterations, hidden layers, alpha, and activation. The values found were then used as a starting point for fine-tuning the hyperparameters. The solver was found to be SDG, the number of iterations as (100,50), and activation to be reLu. MLPClassifier from sklearn was used.

One of the critical hyperparameters here is alpha, which represents the regularization strength. Regularization helps prevent overfitting by penalizing larger weights. Alpha basically controls the amount of regularization applied during training. A low alpha may result in overfitting while a higher one may result in underfitting.

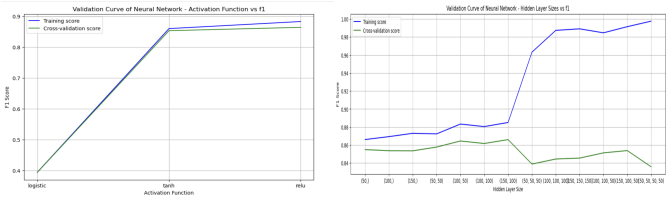
Once an appropriate alpha is found the next parameter to be tuned is hidden layer size. It defines the architecture of the model, thereby controlling its complexity. A model with a very small number of hidden layer may underfit while too many hidden layers may cause the model to overfit.

After determining the optimal alpha and hidden layer size, another important hyperparameter is the activation function. Activation functions control how neurons process the data and it also introduces non-linearity into the model. The activation function (logistic, tanh, ReLU) determines how the weighted sum of inputs is transformed before being passed to the next layer. Non-linear activation functions help the model learn complex patterns.

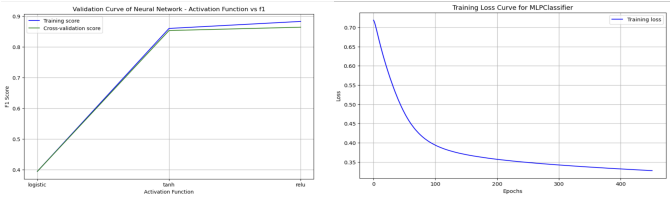
The final hyperparameter is the number of epochs. It defines the number of times the model iterates over the entire dataset. If the number of epochs is too low, then the model may not have enough time to learn the underlying patterns of the dataset and may underfit. Too many epoch may lead to overfitting, since the model may become specialized in the training data.

A. Heart Disease Dataset

For this dataset, the initial random search revealed the following values for the hyperparameter as a starting point: the solver was found to be sdg, the number of iterations as 300, hidden layers size as (100,50) and activation to be relu.



(a) Validation Curve - alpha vs F1 score (b) Validation Curve - Hidden Layers Size vs F1 score



(c) Validation Curve - Activation Function vs F1 score (d) Training Loss Curve

Fig. 11: Neural Network - Heart Disease Validation Curves and Training Loss Curve

a) Validation Curve - alpha vs F1 score: Fig 11.a shows that hows alpha affects the models generalization ability. Till at around 1.05, the training and cross-validation remain consistently at the same value, only increasing slightly as alpha increases. However, at around 1.05, both the training and cross-validation scores bump up. But after alpha = 1, both training and cross-validation score bump down, indicating over-regularization or underfitting.

b) Validation Curve - Hidden Layers Size vs F1 score: Fig 11.b presents the effect of different hidden layers on the model's performance. Post the hidden layer size of (100,50), it can be seen that while the training score increases the cross-validation score drops, indicating that as the hidden layers size increases the model tends to overfit. This hold true because heart disease dataset is a small dataset with less than 1000 data points.

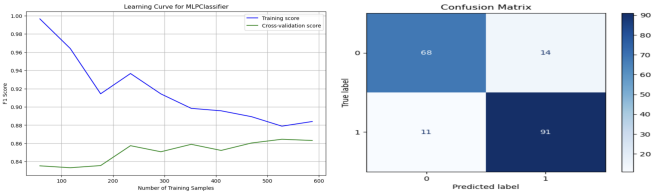
c) Validation Curve - Activation Function vs F1 score: Fig 11.c different activation functions (logistic, tanh, and relu) were tested. tanh and ReLu functions performed far better than logistics function. This could be because during backpropagation, smaller gradient lead to very slow weight updates, increasing the training time and sometimes even

prevently the network from learning effectively. This is likely what lead to this result. Since ReLu gives the best training and cross-validation score. It is chosen

d) Training Loss Curve: Fig 11.d shows the training loss as a function of the number of epochs. Initially, the training loss is high, which is expected at the beginning of training due to random weight that are initialized. As the model trains, the loss decreases in the early epochs, indicating that the model is learning and optimizing its weights. Eventually, the curve begins to flatten as the epochs increase, suggesting that the model makes smaller adjustments to minimize the loss. The final part of the curve shows convergence, where the model has minimized the training loss, reaching a stable point. Also since the loss does not increase, it can be implied that the model is not overfitting and is generalizing.

e) Learning Curve: In Fig 12.a it can be seen that for lower number of samples, the training scores are high while the cross-validation scores are lower. This shows that the model overfits at lower number of samples. However, as the number of samples increase the training score decreases while the cross-validation score increasing. Eventually the gap between the starts decreasing, indicating that with higher number of sample the model is able to generalize better.

*f) **Hypothesis Evaluation**:* Fig 12.b shows the confusion matrix for Neural Network on the testing data. There are 92 true positives, 71 true negatives, 11 false positives, and 10 false negatives. It has an F1 score of 0.86, which is a reasonably good score, it indicates a balanced precision and recall performance.The **initial hypothesis holds true**. The Neural Network does perform reasonably well. However its F1 score is lower than that of KNNs and SVMs this could be because of the small size of the dataset, which might be causing the model to slightly overfit leading to a lower F1 score.



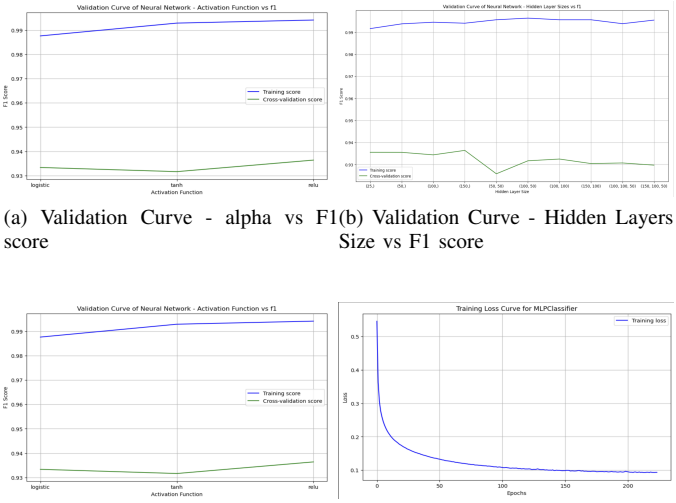
(a) Learning Curve (b) Confusion Matrix

Fig. 12: Neural Network - Heart Disease Learning Curve and Confusion Matrix

B. Spambase Dataset

For this dataset, the initial random search revealed the following values for the hyperparameter as a starting point: the solver was found to be adam, the number of iterations as 300, hidden layers size as (150) and activation to be reLU.

a) Validation Curve - alpha vs F1 score: Fig 13.a shows that how alpha affects the model's generalization ability. The training score remains high and stable for lower values of



(a) Validation Curve - alpha vs F1 score (b) Validation Curve - Hidden Layers Size vs F1 score

(c) Validation Curve - Activation Function vs F1 score (d) Training Loss Curve

Fig. 13: Neural Network - Spambase Validation Curves and Training Loss Curve

alpha but starts to decrease as alpha increases, while the cross-validation score remains relatively flat, with a little decrease as alpha increases. For low values of alpha about 0.001, there is high training and cross-validation score. While at alpha greater than 1, the performance of both degrades due to too much regularization. A good alpha will be in the range of 0.001 to 0.1. 0.01 is chosen as the value of alpha since it provides a good balance between bias and variance.

b) Validation Curve - Hidden Layers Size vs F1 score: Fig 13.b presents the effect of different hidden layers on the model's performance. The training score remains consistently high across all hidden layer configurations, with minimal variations. The cross-validation score shows fluctuations, though the range is relatively small. As the hidden layer size increases, the cross-validation score remains stable for smaller sizes, such as (50,) and (150,). There is a dip in the score at hidden layer sizes (50, 50), (150, 100), and other more complex configurations, which suggests that increasing the hidden layer size results in overfitting, since the training score remain high while the cross-validation score drops. (150,) is chosen as the hidden layer size since it will give a good balance between generalization and performance.

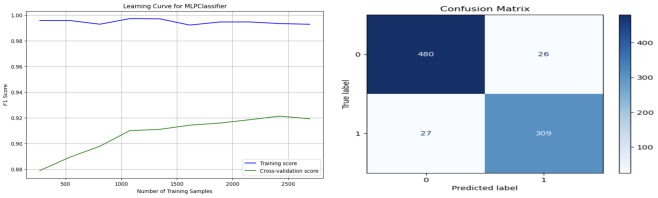
c) Validation Curve - Activation Function vs F1 score: Fig 13.c different activation functions (logistic, tanh, and relu) were tested. The was very little difference in the training and cross-validation scores across different activation function. This indicates that the model can fir the training data well regardless of the activation function used. ReLU is selected as the activation function since it gives a slightly better training and cross-validation score compared to tanh and logistic.

d) Training Loss Curve: Fig 13.d shows the training loss as a function of the number of epochs. Initially, the training loss starts high at around 0.6, indicating that the model is mak-

ing significant errors, this is because of the random weights that are initially set. As the number of epochs increases, the loss decreases rapidly during the first 50 epochs, signifying that the model is learning and improving its predictions effectively. After about 100 epochs, the training loss continues to decrease but at a much slower rate, indicating diminishing returns in learning as the model converges. Around epoch 150, the curve flattens, showing that the model has reached a state of convergence where further training does not significantly reduce the error. This implies that the model has achieved a good balance between fitting the training data and preventing overfitting.

e) Learning Curve: Fig 14.a it can be seen that the training score remains consistently high at around 0.99, indicating that the model performs extremely well on the training data and is likely overfitting. While, the cross-validation score starts at around 0.88 with fewer samples, then gradually increases as the number of training samples grows. It stabilizes around 0.93 as more samples are used, showing that with more data, the model is better able to generalize.

*f) **Hypothesis Evaluation**:* Fig 14.b shows the confusion matrix for Neural Network on the testing data. The matrix shows 480 true negatives, 309 true positives, 26 false positives, and 27 false negatives. With an F1 score of 0.94, the model shows good performance, balancing precision and recall effectively. The relatively low number of false positives (26) and false negatives (27) indicates that the model performs well in identifying both the positive and negative classes, without much bias towards either. Therefore **initial hypothesis holds true**. The Neural Network does perform extremely well on this dataset beating both KNN and SVM. This is because the neural network is better able to capture the relationship between these 57 features and generalize better.



(a) Learning Curve (b) Confusion Matrix

Fig. 14: Neural Network - Spambase Learning Curve and Confusion Matrix

VIII. Clock time

Fig 15 shows that Neural Networks take the longest to train in both datasets. This is due to backpropagation, which updates weights across layers and requires multiple epochs to learn complex patterns. SVM also takes time, especially on the larger Spambase dataset, as it solves complex optimization problems during training. KNN takes the least time because it's a lazy learner, meaning most of its computational cost happens during prediction, not training.

```
Heart Disease Training times
SVM: 0.054599 seconds
k-NN: 0.014547 seconds
Neural Network: 8.624016 seconds

Heart Disease Testing times
SVM: 0.005916 seconds
k-NN: 0.004937 seconds
Neural Network: 0.000446 seconds
```

(a) Heart Disease Clock Times

```
Spambase Training times
SVM: 8.461263 seconds
k-NN: 0.001904 seconds
Neural Network: 8.968058 seconds

Spambase Testing times
SVM: 0.031741 seconds
k-NN: 0.036680 seconds
Neural Network: 0.001340 seconds
```

(b) Spambase Clock Times

Fig. 15: Clock Times

From Fig 15, it can be seen that the with respect to testing times, in both the datasets, KNN takes the longest testing time because, as a lazy learner, it performs all calculations (finding neighbors, using distance and weight metrics) during prediction. SVM is faster compared to KNN since it only needs to check which side of a pre-defined decision boundary the new point falls on. Neural Networks take the least time, as they just pass the data through the trained network with matrix multiplications and activation functions which has less computational costs.

Conclusion

For heart disease dataset, SVM performed better than KNN and Neural Networks as hypothesised though marginally. This was SVM was able to come up a good enough decision boundary that could distinguish the different classes. KNN did perform well but was mostly affected by the curse of dimensionality and could not find the appropriate patterns in the dataset. While Neural networks was constrained by the small size of the data set perform worse that SVM and KNN though it was marginal.

All the algorithms performed very well in Spambase dataset. With all of them getting an F1 score greater than 0.90. This could be for this dataset, all the algorithms could figure out the actual patterns to distinguish spam from actual mails. As hypothesised Neural Networks gave the best F1 score, since this was large multidimensional features. The neural networks was able to effectively figure out the underlying pattern without overfitting. SVM was able also able to figure out a good decision boundary using the linear kernel, suggesting that for this dataset there is a simple linear separability. KNN also gave good performance unexpectedly, since it was able to overcome curse of dimensionality using the larger number of samples and the Manhattan distance metrics.

References

[1] *Library Reference.* Scikit-learn. https://scikit-learn.org.

[2] *Concepts Reference* Geeks For Geeks. https://www.geeksforgeeks.org/what-is-machine-learning/?ref=shm.

[3] *GPT 4o* Minor Code References

[4] T. M. Mitchell, Machine Learning, Indian Edition. New York, NY, USA: McGraw-Hill, 1997.

Overleaf link: https://www.overleaf.com/read/krtvyzvzxcrs#aa093c