Supervised Learning

yrizvi3 – one drive

Introduction

This study is a comparative analysis of the performance of five machine learning algorithms: Decision Trees, Neural Networks (NN), Boosting, Support Vector Machines (SVM), and K-Nearest Neighbors (KNN) on two datasets: The Wine dataset and the Wisconsin Diagnostic Breast Cancer (WDBC) dataset. This analysis will first evaluate the performance of each individual algorithm on the wine dataset, then it will perform an analysis on the WDBC dataset in the similar manner. Additionally, this analysis will investigate the comparison between the five algorithms on the individual datasets after which it will make a comparison of the performance between the datasets.

The wine dataset, with its distinct wine classes derived from chemical analyses, and WDBC dataset, featuring characteristics of cell nuclei in breast cancer biopsies serve as representative datasets with varied complexities. In the approach of this analysis, for each algorithm on each dataset, there is first an implementation of a baseline algorithm for a benchmark, then each algorithm was tuned with hyper-parameters to optimize performance on validation sets, after which it was tested on the hold out test set. Several metrics, including precision, recall, F1 score, accuracy, cross validation score, confusion matrix and training time were explored to find a reasonable score to make a comparative analysis of these algorithms on both datasets.

Preliminary observations and exploration of the data indicated notable variations in the performance of the algorithms highlighting both strengths and weakness in the handling of the different datasets. Specifically, certain algorithms performed better on the multi-class nature of the wine dataset, while others showed strength in the binary classification context of WDBC dataset. This study aims to provide insights into the strengths and limitations of the chosen datasets to implement these supervised learning techniques on and how future work can be enhanced for the further understanding and categorization of the data.

The Datasets

1. Wine Dataset

An analysis on the dataset shows that the data has 178 entries and a total of 14 columns, without null values. The dataset consists of 13 distinct features taken from three types of wine, including various chemical properties like alcohol content, ash, malic acid, flavonoids, etc. Each of these attributes varies in its array and types, making the dataset suitable for exploring multiclassification. For this dataset "class" was dropped after separating the target variable. Since this dataset is a multi-classification problem this analysis also aims to understand the impact of class imbalance (class 0 count: 59, class 1 count: 71, class 2 count: 48) which can potentially include to model under or over fitting. This analysis also aims to question which evaluation metrics can be misleading for this type of problem.

2. WDBC Dataset

In contrast to the wine dataset, the WDBC dataset is typically popular for binary classification where it distinguishes between malignant and benign tumors for breast cancer based on a range of features. Upon initial investigation of the dataset, the WDBC is sized relatively larger than the wine dataset (569 entries versus 178 on wine). For this dataset, there was presence of unnamed, non-null values which were dropped along with "id" and after separating the target variable, "diagnosis" was also excluded.
The reason this dataset is interesting is because of the importance of scoring metrics that measure whether the model correctly identifies malignant or benign tumors due to the medical setting. For this dataset, it will be interesting to explore whether accuracy will be sufficient metrics to evaluate performance and explore with the tradeoffs of experimenting with different evaluation metrics and which metrics

could be potentially misleading or should be prioritized, while dealing with class imbalance. To note, there is a slight imbalance on the dataset (357 malignant versus 212 benign cases) and certain features seemed to have redundant, repetitive information. Although a potential path for future endeavors, resampling techniques were not used to explore the algorithms to make a fair comparison benchmark between the two datasets.

Since the two datasets have different classifications (multi and binary) and sizes, the relative complexity and nature of this setting might lead to some algorithms performing better on one dataset than the other. An aim of this analysis is to capture as many of these comparisons as possible. The approach is to preprocess both datasets in a similar manner, train all five algorithms on both datasets using default hyperparameters for a baseline model, measure the performance using metrics like precision, recall, F1 score etc. (employing macro averaged version for multi classification scenarios) on the validation set. The next step in this approach is to perform hyperparameter tuning on each dataset using grid search on the validation set before proceeding to combine the training and validation set and evaluating a final optimized model to be tested on the hold out (the test set).

Wine Dataset Implementations

1. Decision Tree

Baseline implementation

Using default parameters, the simple decision tree classifier is fit on the training data for wine data and predicts on the validation set. The evaluation metrics can be seen in the table.

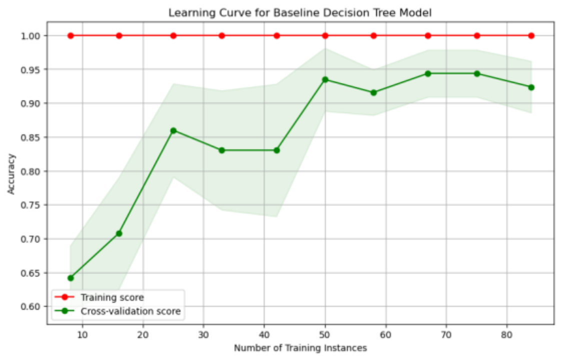
Table 1 Baseline Decision Tree Scores - Wine

Accuracy	97.22%
F1-Score	97.40%
Recall	97.22%
Precision	97.78%

Confusion Matrix:

```
[[11  1  0]
 [ 0 14  0]
 [ 0  0 10]]
```

An initial evaluation of the results shows a good performance on the validation set based off these metrics. However, these results could be a result of either the simplicity of the dataset or due to the preprocessing steps of the dataset or a combination of both. The close values of the performance metrics suggest that the classifier is balanced. However, the model shows a small difficulty in differentiating between class 0 and Class 1 based off the confusion matrix. This could be a result of some overlap in the feature space between some classes. This analysis leads to the question of how well the model will perform on the hold out set to ensure generalization and to check for fitting. For further investigation, it might be useful to look at the learning curve to examine for high bias or variance and check if the model is capturing data complexities or noise or if the data is too simple.



The training score at a constant line at 1.00 on the learning curve suggests that the baseline decision tree model is perfectly fitting the training data, capturing outliers and noise which could be due to the nature of decision tree model without any pruning. Though this may mean it may not generalize well on unseen data. The cross-validation score suggests that as number of training instances increase, the model gets better at generalizing. The slight dips and fluctuation may indicate that there is slight over-fitting as the tree grows.

Hyperparameter Tuning

The method chosen for tuning is to Grid SearchCV to exhaustively try every combination of hyperparameters, this approach was chosen due to wine data set being relatively small. The hyperparameters chosen to tune and incorporate pruning were max_depth and ccp_alpha and the code uses 5-fold cross validation.

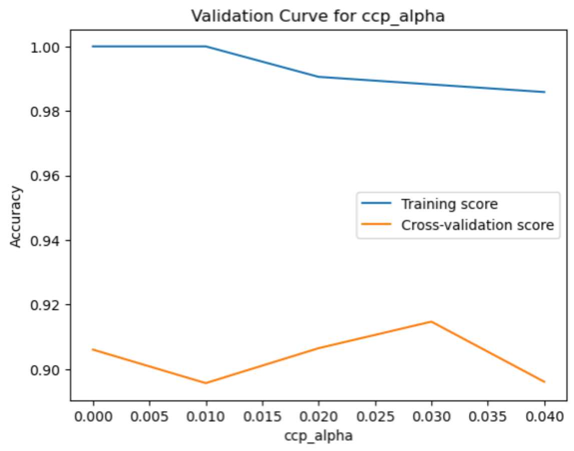
Table 2 Tuning Decision Tree Scores - Wine

Accuracy	97.22%
F1-Score	97.40%
Recall	97.22%
Precision	97.78%

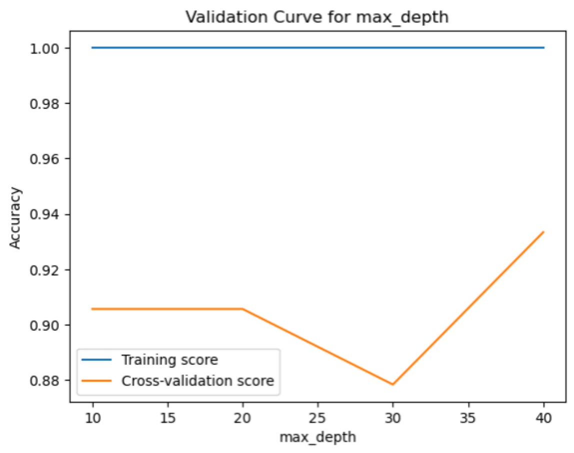
```
Results:
Fitting 5 folds for each of 25 candidates, totalling 125 fits
Best hyperparameters: {'ccp_alpha': 0.0, 'max_depth': Non
e}
CV Score on Training Data: 0.9238
Confusion Matrix:
[[11  1  0]
 [ 0 14  0]
 [ 0  0 10]]
```

The best hyperparameters found suggest that t he best performing tree is a fully grown tee with out any pruning and the cross-validation score (0.9238) suggests that the model performs well with the given hyperparameters. The performa nce metrics haven't changed from the baseline decision tree which makes sense considering it also didn't have any pruning. However, the best hyperparameters offer room for concern. If the best hyperparameter is a full-grown tree, it mig ht be fitting very closely to the training data and the validation performance is also high.

Below are the resulting validation curves to evaluate how different hyper parameters affect training and validation accuracy.



Validation curve for ccp_alpha: a training score curve shows accuracy starting at 1 suggests the model is fitting the training data perfectly. This might be due to the nature of decision tree models without any constraints, as they grow deep enough to classify perfectly. As ccp_alpha increases, the training accuracy decreases – this is expected due to penalty it imposes on the complexity of the tree causing it to prune back and fit data less perfectly. The cross-validation score curve indicates that the unconstrained tree generalizes well to unseen data, and an increase over time in the score suggests that a little pruning enhanced the generalization of the model. However, over time, CV accuracy decreases indicating that too much pruning has a downward effect on unseen data.



The validation curve for tree depth displays that training score has a perfect accuracy indicating it can grow deep and fit data irrespective of the constraint. The curve for cross-validation score for this hyperparameter has an almost opposite curve to the one of ccp_alpha. This might also suggest that the natural depth of the tree without constraints is likely less than the minimum max_depth in the code. The cross-validation score suggests that the model initially generalizes well however the dip suggest that there might be overfitting. But since the training score is high it might be due to the data structure and split within cross validation. Furthermore, the curve returning to its original value may indicate that a deeper tree, after a certain point does not impact the model's generalization negatively.

Therefore, regarding ccp_alpha, it may be that the tree can be pruned up to 0.03 without sacrificing performance on the cross validation set. Pruning might simplify the model and make it more interpretable and generalize to unseen data.

Although the model with best features aligns mostly with training scores -it is interesting due to the validation curves. The accuracy is relatively high so a trade-off might still be considered for interpretability without sacrificing too much on performance such as higher ccp_alpha or more restrictive max_depth.

Modeling with ccp_alpha set to 0.01 and max_depth set to 10 resulted the same performance metrics as the baseline model and the best parameters. However, the key difference that accuracy increased for the cross-validation score for both hyperparameters even though the training score remained the same - indicating better generalization.

Optimized Model

Table 3 Performance Metrics on Optimized Model - Decision Tree:

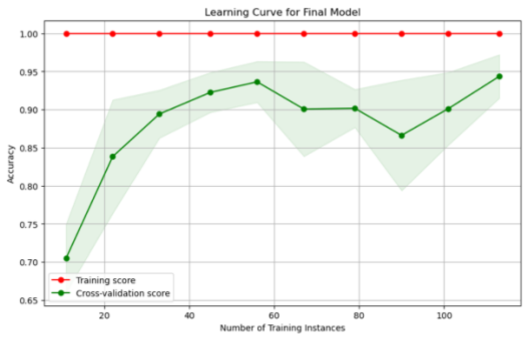
Accuracy	94.44%
F1-Score	95.00%
Recall	95.00%
Precision	95.00%

Confusion Matrix:

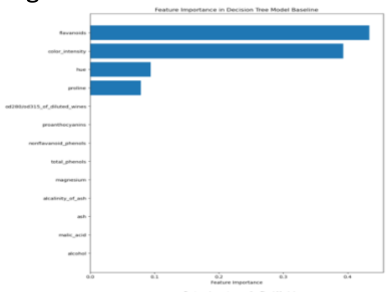
```
[[11  1  0]
 [ 1 14  0]
 [ 0  0  9]]
```

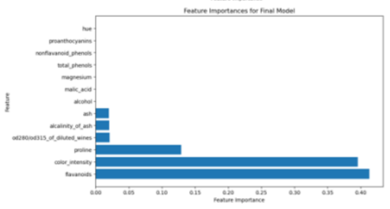
Training Time on Dataset: 0.00 seconds

With the best parameters found, as well as with ccp_alpha=0.01 set to max_depth=10, the evaluation metrics and learning curve produced the same results. An interesting observation is, although the purpose of ccp_alpha and max_depth is to add regularization and with the knowledge of simple models typically performing worse on training sets and better on new data, the model still performs worse on the hold-out set which is unexpected. A key question to consider is whether the baseline model is overfitting data, since that would be an explanation to the lower performance scores on the unseen test set. It may be worth investigating more hyperparameters for the future.



The learning curve also suggests that training data with a constant training score at 1 may be overfitting, persisting with a larger training size. The gap between the two curves suggests a high variance scenario and a potential for regularization as further work for the future.

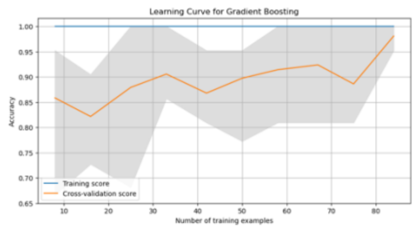




It is interesting to look at the way feature importance changes with the baseline model, hyper tuning model on the validation set and final model -particularly ash and alkalinity of ash. As the complexity or depth changes, it may be that the feature provides more information in decision making and valuable splitting. For example, Hue became less valuable; this may have been a contribution to overfitting in the baseline model.

2. Boosting

Baseline model



The confusion matrix and the learning curve for the baseline model suggests that there may be overfitting due to the perfect score on the training set and misclassifications on the matrix. However, the cross-validation score tries to converge towards the curve of the training score, suggesting that the model becomes better at generalizing. Regardless, since gradient boosting iteratively is building trees to correct error, it still may be overfitting when observing other performance metrics below.

Table 4 Boosting - Baseline Model Performance metrics

Accuracy	77.78%
F1-Score	77.78%
Recall	77.46%
Precision	78.52%

Training Time: 0.2533 seconds
Confusion Matrix:

```
[[ 8  3  1]
 [ 2 12  0]
 [ 2  0  8]]
```

Hyperparameter Tuning

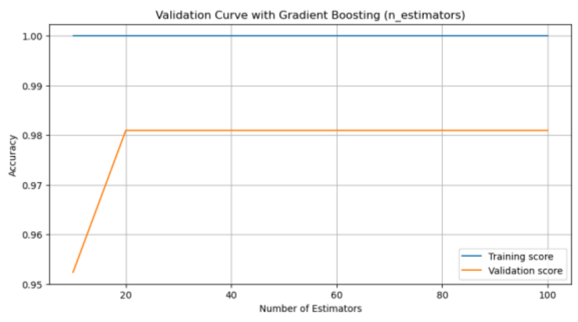
GridSearchCV was used to search for the best parameters for n_estimators and learning_rate.

Table 5 Boosting Tuning Performance Metrics - Wine

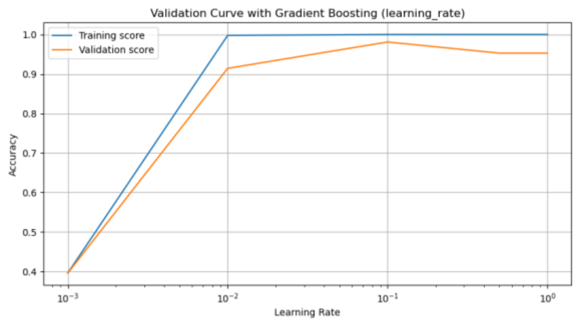
CV accuracy	0.9810 (+/- 0.0233)
CV recall	0.9821 (+/- 0.0220)
CV precision	0.9838 (+/- 0.0203)
CV f1	0.9818 (+/- 0.0224)

Training time: 29.40 seconds
Best parameters found: {'learning_rate': 0.1, 'n_estimators': 20}
Confusion Matrix:

```
 [[ 8  3  1]
 [ 1 13  0]
 [ 2  0  8]]
```

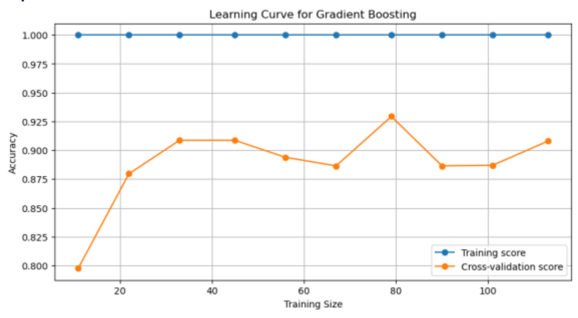


These results suggest high performance for the model. A inspection is needed for over-fitting even though the validation score attempts to rise for n_estimators. The validation curve for n_estimators indicates an overall reasonable score. The validation curve clearly agrees with the finding of n_estimators at 20 estimators.



Although there is not a relatively large gap between both scores curves on both plots, a sharp spike may indicate that the corresponding values may be too conservative for the model to learn meaningfully, especially considering the misclassifications on the confusion matrix. The tradeoff is that larger values may output suboptimal results. A future approach is to consider different sampling techniques due to class imbalance to investigate this.

Optimized model



The model performs relatively well on the hold out test set, misclassifying one sample on the third class- the smallest class. As the training size gets larger, as the learning curve displays, as the training size gets larger the performance increases till a period where adding samples doesn't improve the score- perhaps due to noise or complexity. However, there is an overall rising trend suggesting larger dataset helps the model to generalize better.

Table 6 Boosting - Optimized Model Performance Metrics - Wine

Accuracy	94.44%
F1-Score	94.50%
Recall	93.51%
Precision	96.07%

Training time: 0.07 seconds

```
[[ 12  0  0]
 [ 0 15  0]
 [ 0  1  8]]
```

3. Neural Network (NN)

Baseline model

The baseline model offers an impressive evaluation however the perfect scores are a cause for concern about data leakage or overfitting.

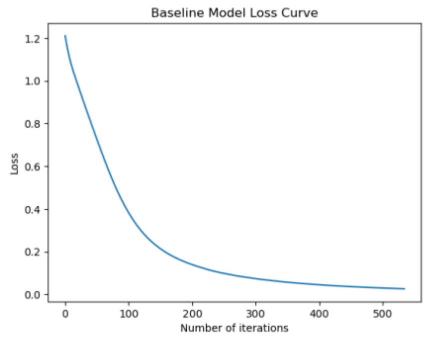
Table 7 Baseline NN Performance Metrics -Wine

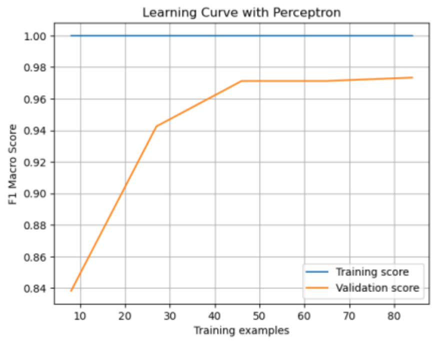
Accuracy	100%
F1-Score	100%
Recall	100%
Precision	100%

Training time: 0.23 seconds

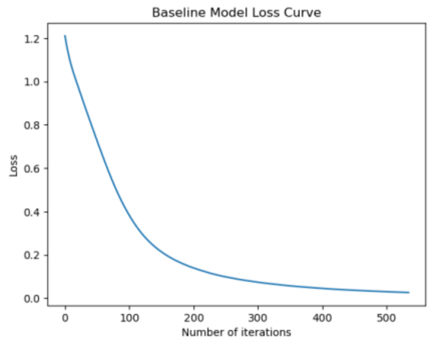
Confusion Matrix:

```
[[12  0  0]
 [ 0 14  0]
 [ 0  0 10]]
```





The learning curve is plotted against F-1 Macro Score, however, for future endeavors it should be noted that for consistency, accuracy should have been used. Since F-1 score is relatively more important than accuracy for imbalanced dataset, this metric was chosen due to the perfect scoring on the baseline model, hoping for a more insightful analysis.



The loss curve suggests a successful training, it indicates there is unlikely chance for over-fitting.

Tuning:

Since there were already optimal results on the baseline, it is interesting to see what hyperparameter tuning will offer.

Precision: 1.0
Recall: 1.0
Accuracy: 1.0
F1 Score: 1.0
Confusion Matrix:

```
[[12  0  0]
 [ 0 14  0]
 [ 0  0 10]]
```

Cross-validation F1 Score: 0.9717 (+/- 0.0233)

Optimized model:

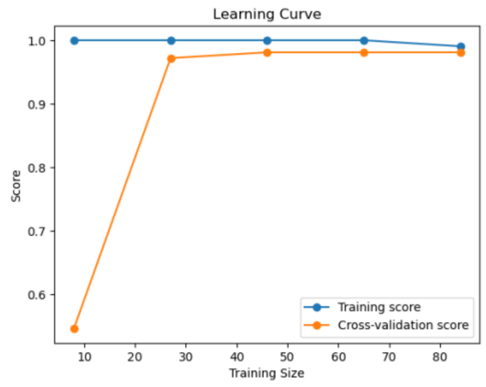
Training time: 0.05 seconds
Accuracy: 0.8333
Macro Recall: 0.8315
Macro Precision: 0.8635
Macro F1 Score: 0.8391

```
[[11  1  0]
 [ 3 12  0]
```

[1 1 7]]

4. SVM (Support Vector Machines)

Baseline Model

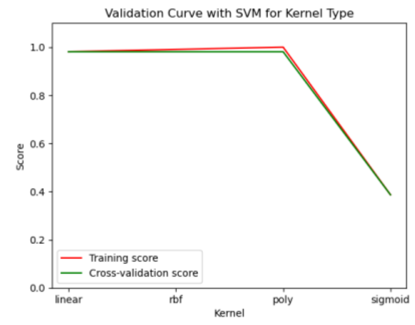


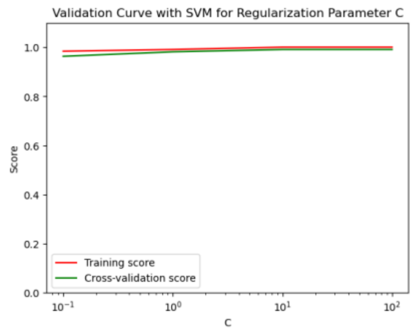
Baseline Precision: 1.0 Recall: 1.0
Accuracy: 1.0. F1-Score: 1.0
Training Time: 0.0021979808807373047
Confusion Matrix:

```
 [[12  0  0]
 [ 0 14  0]
 [ 0  0 10]]
```

These results show a perfect score.

Hyper parameter tuning





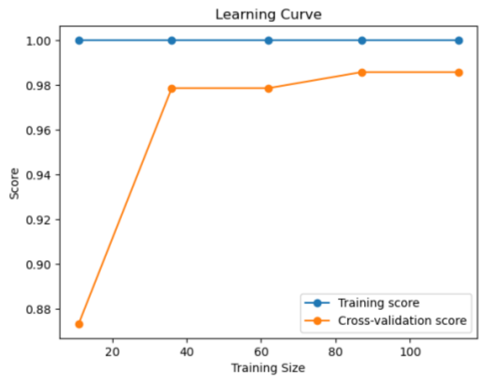
Precision: 1.0 Recall: 1.0 Accuracy: 1.0
F1-Score: 1.0 Training Time: 0.00045180320739746094
Confusion Matrix:

```
 [[12  0  0]
 [ 0 14  0]
 [ 0  0 10]
```

Best Parameters: {'C': 10, 'kernel': 'rbf'}
Cross Validation Score: 0.9904761904761905

Optimized model:

The performance on the hold out set shows that accuracy remains high and misclassification is low.

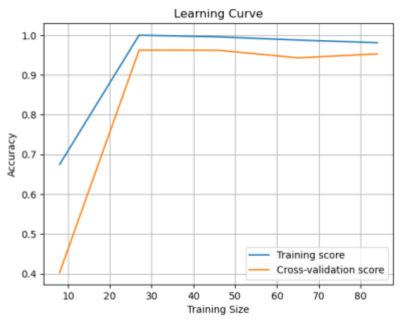


Training time: 0.0020549297332763367 seconds
Accuracy: 0.9722222222222222
Recall: 0.9629629629629629
Precision: 0.9791666666666666
F1 Score: 0.9696394686907022
Confusion Matrix:

```
[[12  0  0]
 [ 0 15  0]
 [ 0  1  8]]
```

5. KNN (K-Nearest Neighbors)

Baseline



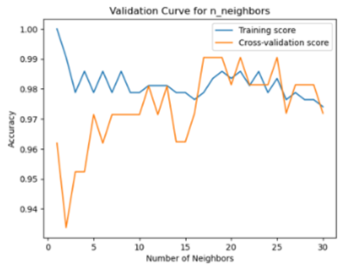
Training Time: 0.00 seconds
Accuracy: 0.9444, F1 Score: 0.9452, Recall: 0.9524,
Precision: 0.9441. Confusion Matrix:

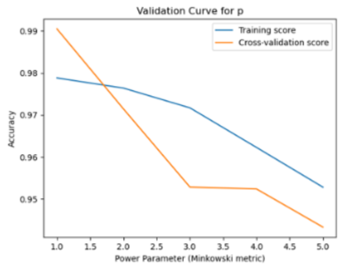
```
[[12  0  0]
 [ 1 12  1]
 [ 0  0 10]]
```

Tuning

Best hyperparameters: {'n_neighbors': 17, 'p': 1}
Mean Cross-Validation Score of Best Model: 0.9905
Grid Search Execution Time: 0.36 seconds
Performance Metrics on Validation Set:
Accuracy: 0.9167
F1-score: 0.9185
Recall: 0.9286
Precision: 0.9221
Confusion Matrix:

```
[[12  0  0]
 [ 2 11  1]
 [ 0  0 10]]
```

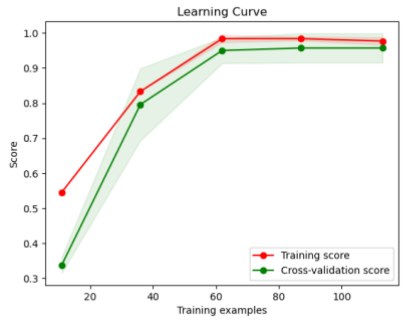




Optimized Model

Performance Metrics on Test Set:
Accuracy: 0.9722, F1-score: 0.9752
Recall: 0.9778, Precision: 0.9744
Confusion Matrix:

```
[[12  0  0]
 [ 1 14  0]
 [ 0  0  9]]
```



Comparison Between Algorithms in The Wine Dataset

An analysis on all 5 algorithms shows that NN and SVM had the best performance in the scenario that there was no overfitting. However, the validation curves of the SVM model are more difficult to evaluate for overfitting due because a loss curve wasn't visualized.

WDBC Dataset Implementations

1. Decision Tree

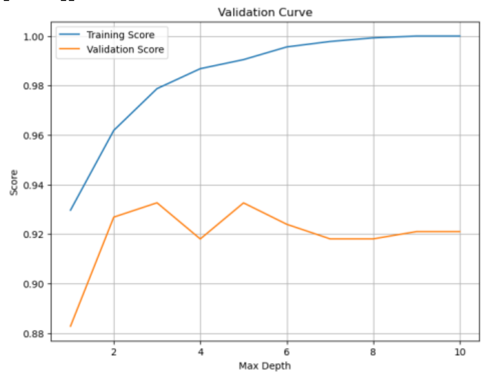
Baseline Model

Validation Set Metrics: Accuracy: 0.94 Precision: 0.97 Recall: 0.86 F1 Score: 0.91
Completed Training Time (Validation): 0.01 seconds

Tuning

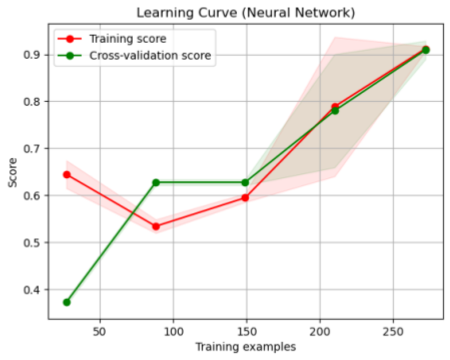
Completed training in 0.249300 seconds
Best parameters set for decision tree found on development set:
{'max_depth': 3}
Inference time on validation data: 0.000777 seconds
Accuracy: 0.9385964912280702, Precision: 0.9736842105263158, Recall: 0.8604651162790697, F1 Score: 0.9135802469135803
Confusion Matrix:

```
[[70  1]
 [ 6 37]]
```



2. Neural Network (NN)

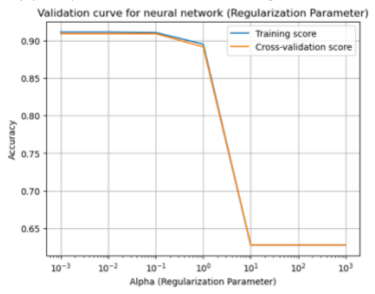
Baseline Model:

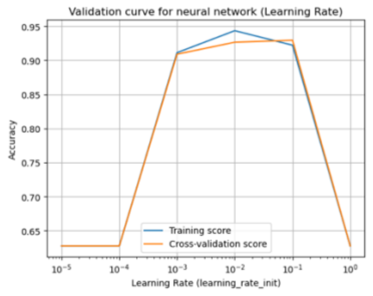


Validation Set Metrics:
Accuracy: 0.91, Precision: 0.97, Recall: 0.79, F1 Score: 0.87
Confusion matrix

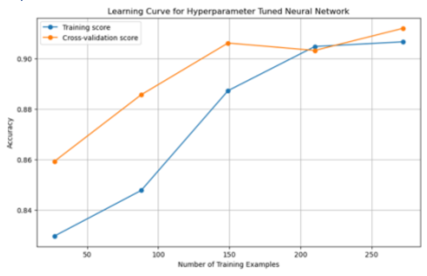
```
[[70  1]
 [ 9 34]]
```

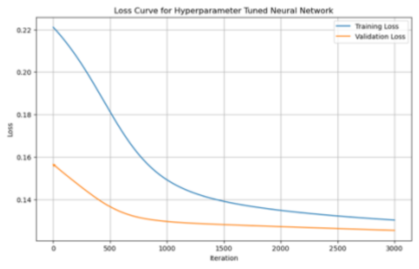
Hyperparameter Tuning:





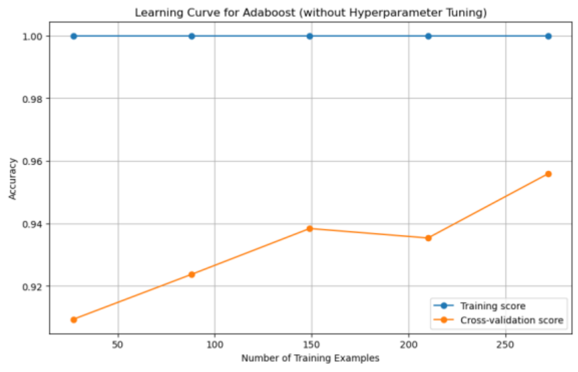
Optimized Model:



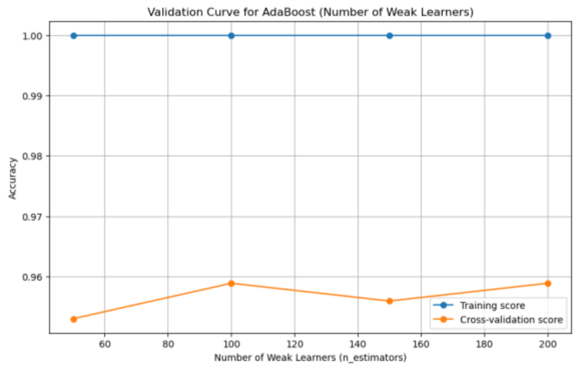


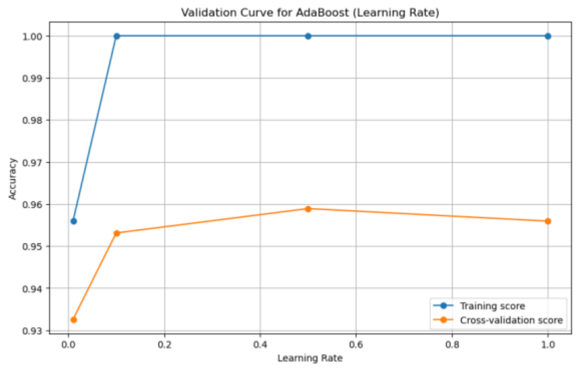
3. Boosting

Baseline Model:

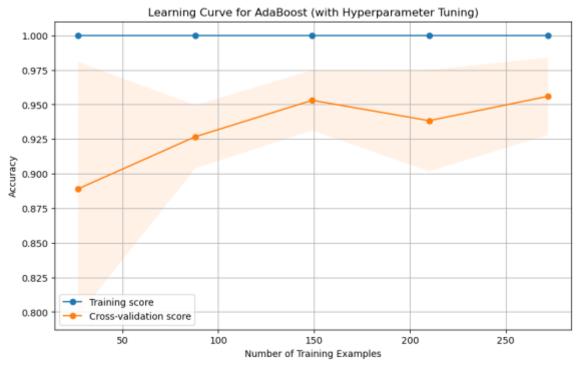


Hyperparameter tuning:



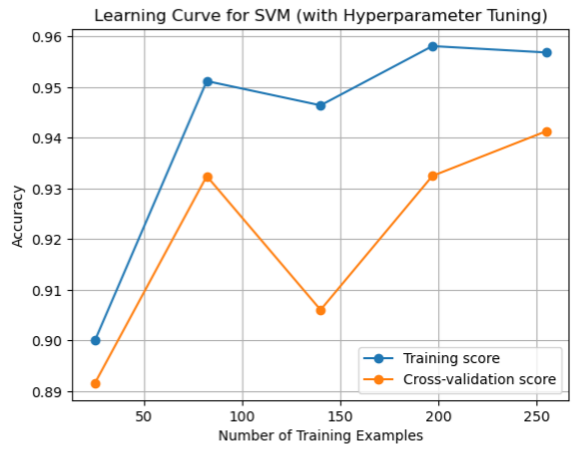


Optimized model:

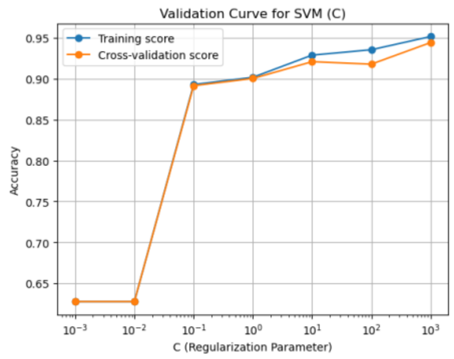


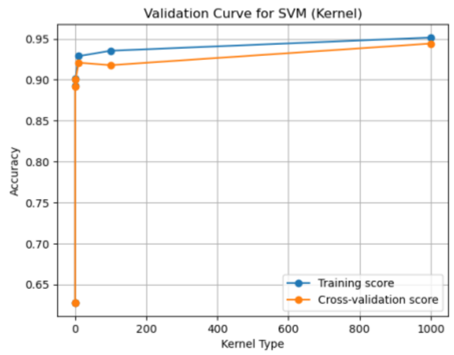
4. SVM (Support Vector Machines)

Baseline Model:

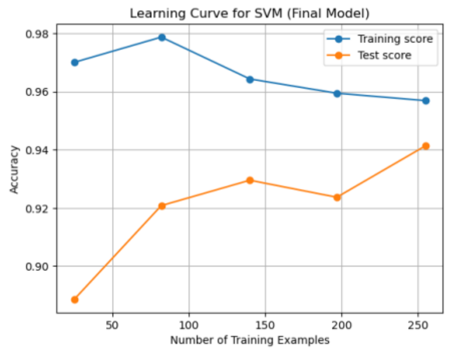


Hyperparameter tuning:



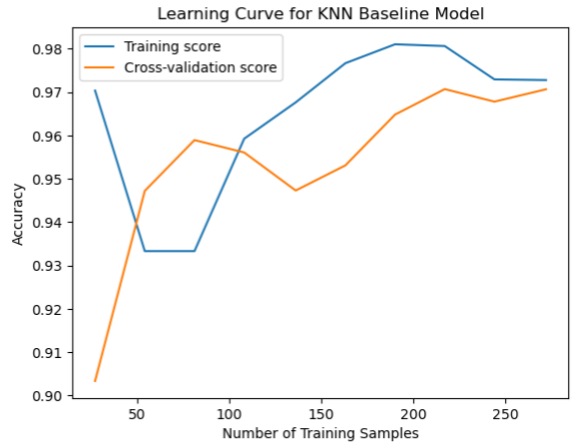


Optimized Model:

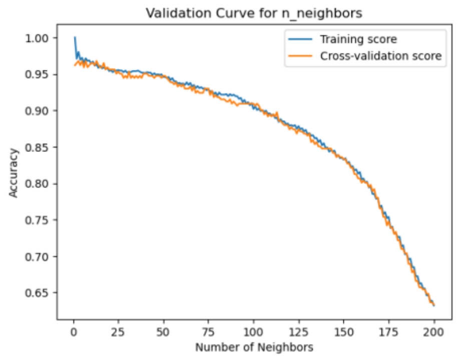


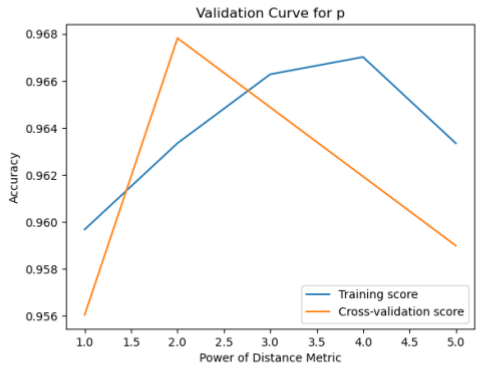
5. KNN (K-Nearest Neighbors)

Baseline model:

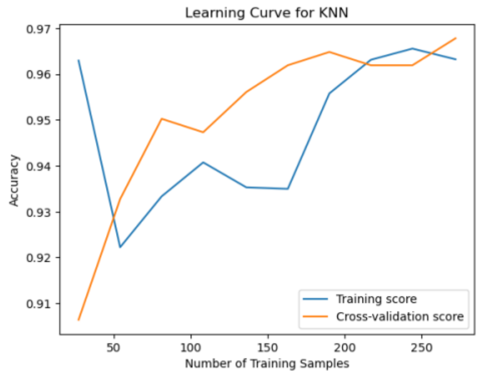


Tuning:





Optimized model:



Comparison of Algorithm Performance Between Datasets (Wine and WDBC)

Overall, the analysis on the wine dataset proved to be much more insightful attributed to the approach in maintaining relatively more consistency. There could be overfitting on my decision tree for wine, however overall, there was better performance on the WDBC dataset on the final model due to its binary nature. In contrast, for boosting on the final models, there is better performance on wine. SVM and KNN had similar performance on both datasets however, further analysis. NN was overall better on wine data.

Conclusion

The two datasets presented significantly different classification tasks, each with varying sizes. The wine dataset primarily serves as an academic tool for experimenting with different techniques, while the WDBC dataset holds more significance in real-world applications.

Initially, it was expected that SVM, Gradient Boosting, and NN would yield the best results for the Wine dataset. However, conducting direct comparisons between algorithms on each dataset proved intriguing. Analyzing both datasets together presented challenges due to their distinct classification nature (multiclassification for one versus binary for the other). Nevertheless, the results remained compelling, especially considering the substantial size difference between the datasets.

Furthermore, it was intriguing to observe the impact of different algorithms on these two unique classification problems. The importance of evaluation metrics varied accordingly. For instance, the measure of training time may not be as critical in an academic setting, but in real-world applications, such as classifying tumors in the WDBC dataset, timely decisions can be crucial. Additionally, GridSearchCV might not be the most efficient method for hyperparameter tuning due to its computational expense.

Exploring the effects of preprocessing, such as encoding, on both datasets also proved to be enlightening. Furthermore, when combining training data with validation sets to train the optimized model on a holdout test set, it's essential to maintain consistency in the method used for combining these sets. Maintaining consistency in evaluation metrics across different models proved challenging, given the diverse nature of the algorithms. Nonetheless, efforts were made to select scoring metrics that offered the most insightful evaluations for each specific algorithm.

References:

Wolberg,William, Mangasarian,Olvi, Street,Nick, and Street,W.. (1995). Breast Cancer Wisconsin (Diagnostic). UCI Machine Learning Repository. https://doi.org/10.24432/C5DW2B.

Aeberhard,Stefan and Forina,M.. (1991). Wine. UCI Machine Learning Repository. https://doi.org/10.24432/C5PC7J.

OpenAI. (2023). ChatGPT (Mar 14 version) [Large language model]. https://chat.openai.com/chatatGPT