Introduction

For this assignment, I utilized two datasets from the UC Irvine Machine Learning Repository. The first dataset is the electric grid dataset with 13 numeric predictor variables and 1 target variable. The target variable indicated whether the electric grid network is considered to be stable or not. The predictor variables consisted of the reaction time of the participant (tau), the power consumed (p), price elasticity (g), and maximal root of the equation (stab). To process the grid dataset, I used interquartile range to define the upper and lower bounds in order to identify outliers. I ended up removing one outlier so that the grid dataset now has 9,999 rows instead of a thousand rows. Then I encoded the target variable (stabf) so that unstable is 0 and stable is 1. Next, I standardized the dataset so it is adequately scaled for the models to work. This dataset is nontrivial because the predictor variables have some influence over the grid stability variable. Running the algorithms can help to discern how well these features can help to predict the stability of systems.

The second dataset is the Wisconsin breast cancer dataset with 30 numeric predictor variables and 1 target variable. The target variable indicated whether the breast mass is benign or malignant. The dataset came with the target variable encoded so I didn't have to encode it myself. I then standardized the dataset so that it would be adequately scaled. This dataset is nontrivial because the predictor variables include various characteristics of the breast mass. These variables can directly influence whether the mass is malignant or not.

To conduct these algorithms and visualize them, I utilized Python along with its packages such as numpy, pandas, matplotlib, and more. For each dataset, I split the data into the test set (20%) and the train set. Then, I split the train set into the train set (80%) and the validation set (20%). I then conducted the decision tree model, Gradient Boosting model, k-Nearest Neighbors, Support Vector Machine model, and Neural Network model. I plotted the validation curve for the train and validation sets for each hyperparameter. I also visualized the learning curve for each model. Lastly, I evaluated the model with various metrics such as training time, prediction time, F1 score, accuracy, precision, AUC, and recall.

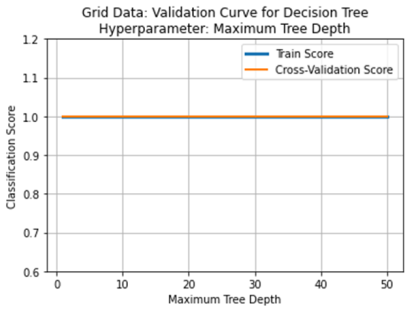
Results - Electric Grid

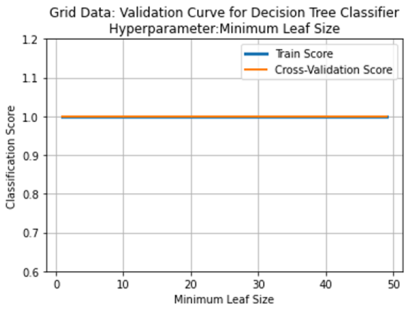
The optimal values for each hyperparameter are summarized below in the table.

Electrical Grid Dataset: Optimal Hyperparameter Values					
	Decision Tree	**Boosting**	**kNN**	**SVM**	**Neural Network**
Hyperparameter 1	Maximum tree depth	Maximum tree depth	Leaf size	C	Hidden layer size
Best Value	1	1	1	1000	1
Hyperparameter 2	Minimum leaf samples	Number of estimators	Number of neighbors	Kernel type	Learning rate
Best Value	32	10	17	Linear	0.01

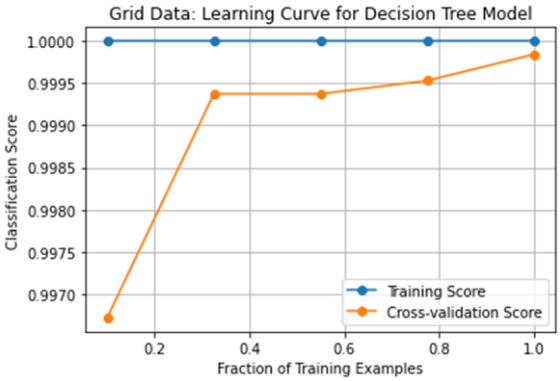
First, the decision tree algorithm with the entropy criterion was run on the train sets of the electric grid dataset. The accuracy of this model prior to hyperparameter tuning is 100%. This may be attributed to an imbalance in feature importance. Examination of the model's feature importances elucidates that the *stab* variable has an importance of 1.0, while the rest of

the variables have importances of zero. These feature importances underscores the importance of understanding the relationship between the different variables. For instance, the predictive variables with importance of 0 include the participants' reaction times, electric power consumption, and price elasticity. The *stab* variable refers to the stability analysis conducted, thus directly affecting whether or not a system is stable or unstable in the *stabf* variable.



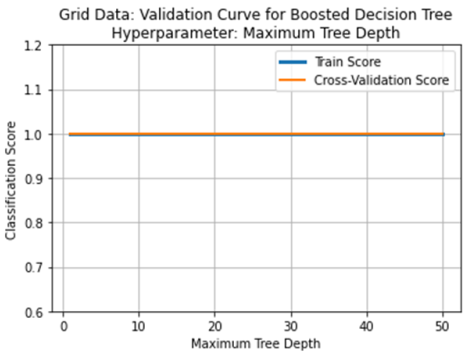


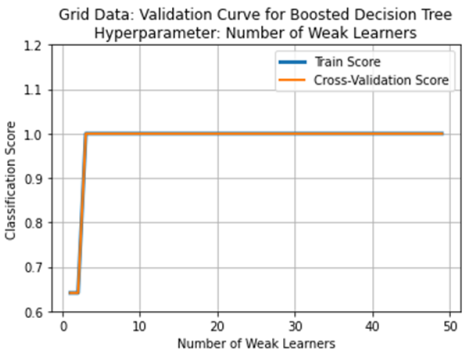
The validation curve plots for the maximum tree depth and minimum leaf size are similar in that the curves for both are constant at classification score of 1.0 for both training and cross-validation set. Again, this may be attributed to insufficient model complexity. A perfect score of 1.0 for the validation curves indicate that this model has a case of overfitting or that these hyperparameters are irrelevant to optimizing the model's performance. GridSearchCV was utilized to search for the optimal values of each hyperparameter. Finding the optimal hyperparameter values was done in 108.66 seconds. The optimal value for maximum tree depth is one, and the optimal number of samples at a leaf is 32. The accuracy of the decision tree when predicting y-values of the validation set is 100%. Below, we have the learning curve for the decision tree model.



The learning curve for the decision tree model shows that the training score holds constant across all sizes of the training set. Meanwhile, the learning curve for the cross-validation score increases up until around 30% of the training examples, stays constant until 55%, and then increases again as it approaches the training score. This learning curve reflects high variability as well. Since the training and cross-validation scores are converging at a high classification score, it is clear that the decision tree model could utilize more data points. Adding more data points can also help subdue the variability of the data.

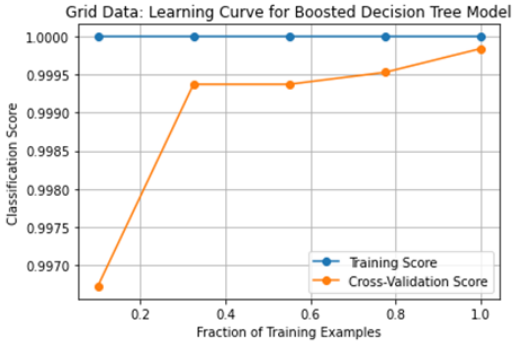
Next, the boosted decision tree algorithm for the electric grid dataset prior to hyperparameter tuning resulted in 64.12% accuracy. This is much lower than the accuracy for the decision tree model prior to hyperparameter tuning. The substantial difference in accuracy for these two algorithms may arise from how each model handles the training data. The decision tree model resulted in 100% accuracy before tuning, which may point to overfitting the data. The boosted model resulted in 64.12% accuracy, which may be attributed to how boosting generally focuses on reducing the model's bias. Reducing the model's bias results in less overfitting over the model, thus resulting in a lower accuracy score.





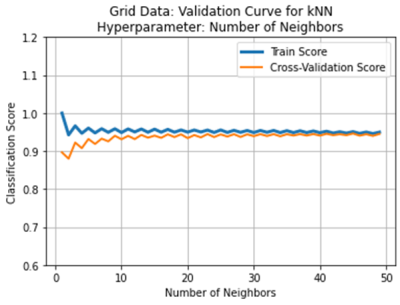
The boosted algorithm's validation curve for the maximum tree depth is the same as the validation curve for the decision tree model. The perfect classification score of 1.0 indicates that the boosted model tends to overfit the data. It could also be argued that maximum tree depth is irrelevant to optimizing the model's performance. The validation curve for the weak learners hyperparameter shows overlap between train and validation scores, which indicates that the model is able to generalize well to the unseen data of the validation set.

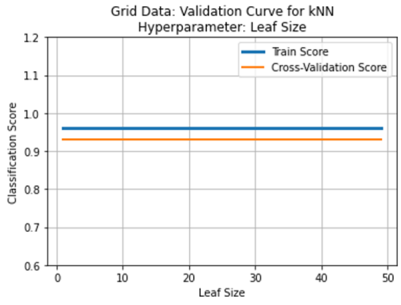
Finding the optimal hyperparameter values for the boosted decision tree model was done in 65.85 seconds. The optimal value for maximum tree depth is one, and the optimal number of weak learners is 10. The accuracy of the decision tree when predicting y-values of the validation set is 100%. While 100% accuracy could indicate correct prediction, it is possible that overfitting may be at play. Below, we have the learning curve for the boosted decision tree model.



The learning curve for the boosted decision tree model above shows that the training score holds constant across all sizes of the training set. Interestingly, the learning curve plot for the boosted model looks very similar to the learning curve plot for the decision tree model. This may be attributed to how the boosted model and decision tree model may adopt similar "learning" processes. In the plot above, the learning curve reflects high variability. Since the training and cross-validation scores are converging at a high classification score, it is clear that the decision tree model could utilize more data points. Adding more data points can also help subdue the variability of the data.

The k-Nearest Neighbors (kNN) model prior to hyperparameter tuning resulted in 96.2% accuracy, which indicates that the default hyperparameters are likely effective for the electric grid dataset. Despite this, I went ahead with hyperparameter tuning to achieve optimal performance.

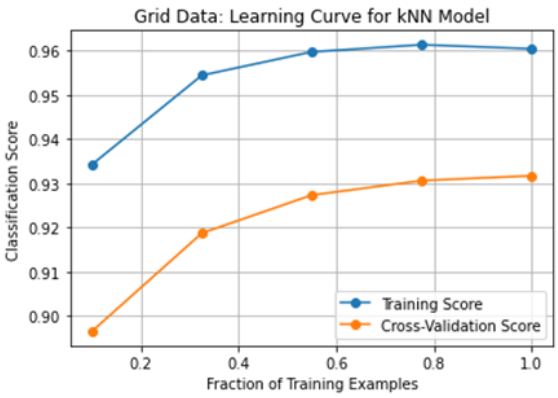




The validation curve plot for the number of neighbors parameter above depicts fluctuating curves for both the train and cross-validation classification scores. These rather rapid fluctuations may be attributed to the model's behavior in being overly sensitive to changes in this hyperparameter. Overfitting the data can also prompt the model to be unstable. These curves mostly have classification scores that range from 0.88 to 1.0. Both the train and cross-validation curves converge and even begin to overlap at around 35 neighbors. The similar model performance on both the training and validation sets indicate that the kNN model has a high level of bias. The validation curve plot for the leaf size hyperparameter remains constant as the

leaf size increases. This indicates that this hyperparameter does not have a meaningful impact on the performance of the kNN model. Again, the performance of the model is similar for both training and validation sets so there is high bias in the kNN model.

Finding the optimal hyperparameter values for the kNN model was done in 780.22 seconds. The optimal value for leaf size is one, while the optimal number of neighbors is 17. This makes sense as the classification score remained constant across all leaf sizes for both the training and validation sets in the validation curve plots. The accuracy of the decision tree when predicting y-values of the validation set is 93%, which is slightly lower than the model's accuracy prior to hyperparameter tuning. This small drop in accuracy of 3.2% may be attributed to the GridSearchCV function as it searches for the optimal hyperparameter values that may not necessarily match in accuracy. Below, we have the learning curve for the kNN model.

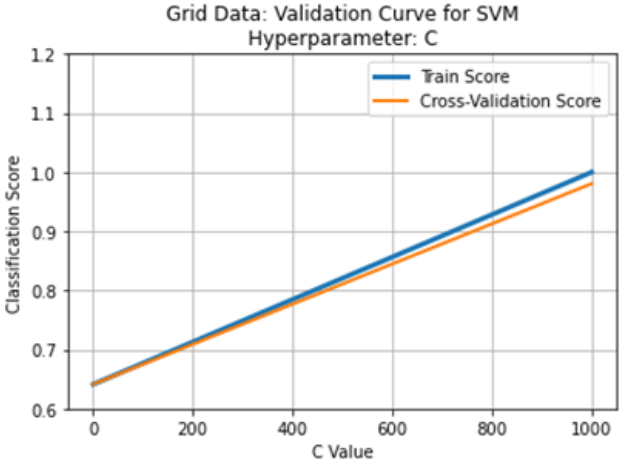


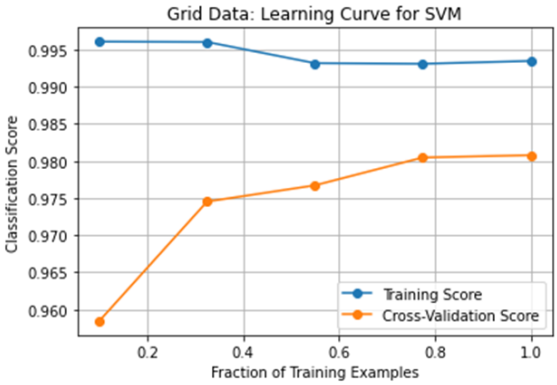
The learning curve for the kNN model shows classification scores greater than 0.89 across all training sizes for both the training and cross-validation scores. The training score increases until about 78% of the training size before decreasing slightly back to 0.96. The cross-validation score continually increases as the training size gets larger. There is a fairly small gap between the training score and cross-validation score across all the training sizes, which indicate that the model has optimal fit.

Next, the Support Vector Machine (SVM) model was run on the train sets of the electric grid dataset. The accuracy of this model prior to hyperparameter tuning is 99.34%. Such a high accuracy value may be attributed to an imbalance in feature importance, as described similarly earlier in the discussion about the decision tree model.

Electric Grid Dataset: Hyperparameter - Kernel Type		
Kernel Type	**Training Classification Score**	**Validation Classification Score**
Linear	1	0.99
Poly	0.98	0.96
RBF	0.99	0.99
Sigmoid	0.95	0.95

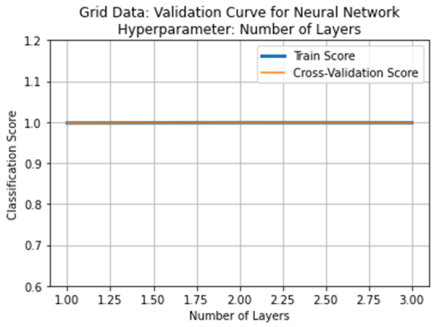
The table above depicts the classification scores for the training and validation sets when the kNN model utilizes each type of kernel. The classification scores reflect that each kernel type exhibits high accuracy for both the training and validation sets. The linear kernel type demonstrated a better performance on both sets compared to the other kernel types. The target variable *stabf* refers to whether the system is stable or unstable. This affirms that the dataset is a binary classification problem as there are two choices present: stable or unstable. Therefore, this dataset can be identified as linearly separable. The linear kernel type is best suited for linearly separable datasets, so it makes sense that the linear kernel type was found to have the best performance with this dataset.

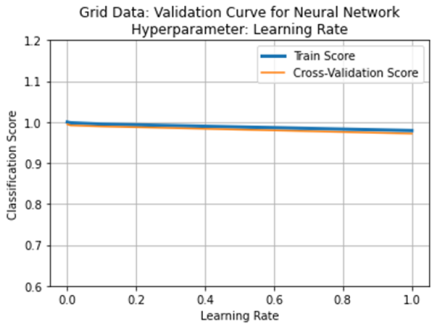




Here we have the validation curve for the C value hyperparameter. The train and validation scores overlap until about C = 400. After this point, the gap between train and validation scores slightly increases. Increasing C generally weakens the regularization of the SVM model, thus fitting the data better. This might explain why the validation scores overlap until C = 400 and not when C is greater than 400.
For the SVM model, the optimal kernel type is linear and the optimal C value is 1,000. Finding these optimal hyperparameter values was done in 28.78 seconds. The accuracy of the SVM model was found to be 98.63%, which is a 0.72% decrease from the accuracy of the model prior to hyperparameter tuning. The learning curve for SVM reflects a decrease in training score with an increase in validation score, which indicates that the SVM model is underfitting the data.

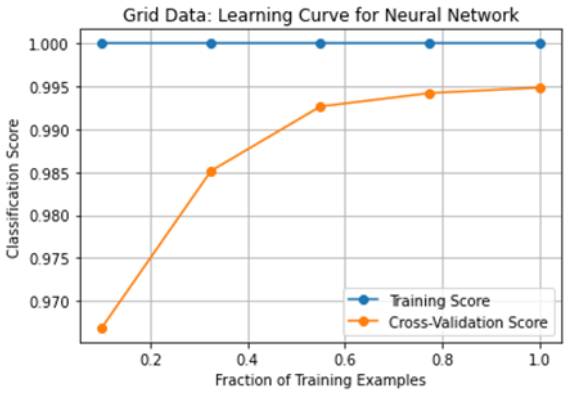
Lastly, the neural network (NN) algorithm for the electric grid dataset prior to hyperparameter tuning resulted in 100% accuracy. This may be attributed to the linear separability of the dataset or that the model is overfitting the data.





The validation curve for the number of layers hyperparameter is constant at 1.0 across different numbers of layers, which may indicate overfitting of the data or that the number of layers is irrelevant to optimizing the NN model. The validation curve for the learning rate hyperparameter decreases for both the training and validation sets as the learning rate increases. The learning rate parameter controls how quickly the model adapts to the data. A higher learning rate can produce more rapid and hasty changes, thus lowering the accuracy. Therefore, it makes sense that accuracy is higher at the lower values of the learning rate hyperparameter.

Finding the optimal hyperparameter values for the NN model was done in 73.6 seconds. The optimal value for hidden layer size is 1, while the optimal learning rate is 0.01. After hyperparameter tuning, the accuracy of the model lowers by 0.25%. The learning curve for the NN model below indicates that the model has optimal fit in that the curves converge and don't overlap.



The various metrics to evaluate all five algorithms on the test set are depicted in the table below. The NN model had the slowest model training time, which can be attributed to a low learning rate of 0.01. The decision tree model had the fastest training time, which can be attributed to the relatively simple algorithm. This model by nature does have a simpler structure compared to the other four models, which makes it easier and faster to train the model. The simplicity of the decision tree model can also help it predict data faster than the other four models, as indicated in the table below. The kNN model was found to have the slowest predicting time, which may be because the optimal number of neighbors is 17. The computational complexity increases when the number of neighbors increases as more distances need to be calculated and sorted.

Electric Grid Dataset: Model Evaluation Metrics for Test Dataset					
Metrics	**Decision Tree**	**Boosting**	**kNN**	**SVM**	**Neural Network**
Model training time (seconds)	0.03526	0.10794	0.03988	0.43999	1.73237
Model prediction time (seconds)	0.00052	0.00928	0.6421	0.01337	0.00539
F1 score	1	1	0.86	1	1
Accuracy	1	1	0.9	1	1
Precision	1	1	0.88	1	1
AUC	1	1	0.89	1	1
Recall	1	1	0.83	1	0.99

The accuracy score is 100% for all of the models except for kNN. The 100% accuracy value may be attributed to the data being a binary classification problem and linearly separable. Moreover, it could be a sign of potential overfitting of the data.

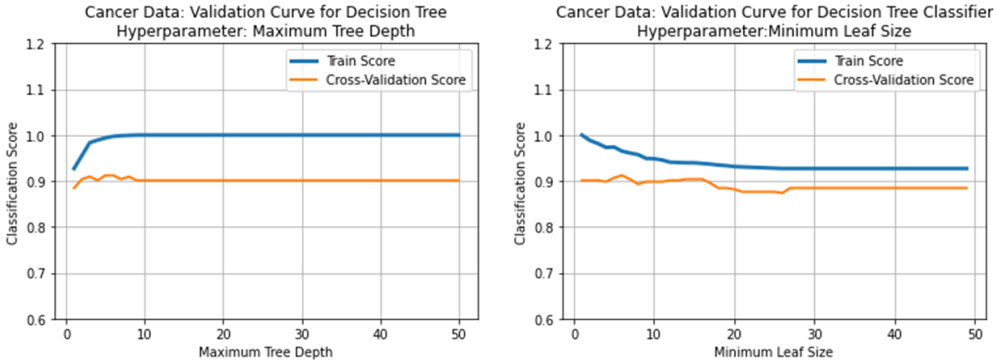
Results - Wisconsin Breast Cancer

The optimal values for each hyperparameter are summarized below in the table.

Wisconsin Breast Cancer Dataset: Optimal Hyperparameter Values					
Algorithm:	**Decision Tree**	**Boosting**	**kNN**	**SVM**	**Neural Network**
Hyperparameter 1	Maximum tree depth	Maximum tree depth	Leaf Size	C	Hidden layer size
Best Value	10	1	1	1000	1
Hyperparameter 2	Minimum leaf samples	Number of estimators	Number of neighbors	Kernel type	Learning rate

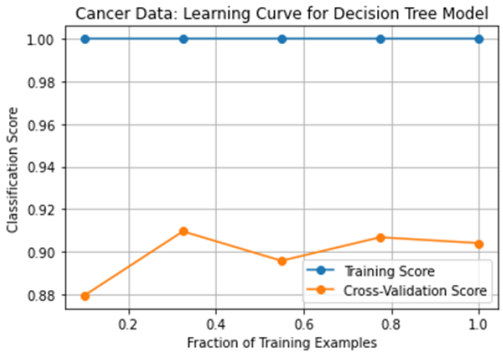
Best Value	6	50	3	Linear	0.01

First, the decision tree algorithm with the entropy criterion was run on the train sets of the breast cancer dataset. The accuracy of this model prior to hyperparameter tuning is 100%. This may be attributed to the linear separability of the dataset as the target variable classifies each breast mass as either benign or malignant.



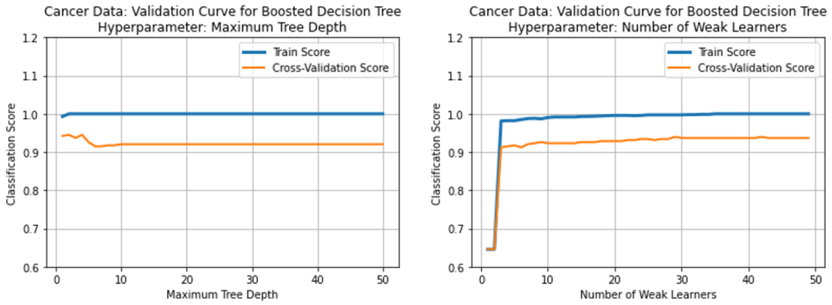
The validation curve plot for the maximum tree depth has some fluctuation from 0 to 10 for the validation set. This indicates that the validation set is very sensitive when the maximum tree depth ranges from 0 to 10. The train and validation scores don't seem to converge, which could be attributed to differences between the training and validation datasets. For instance, it's possible that the training set has features with much less variance than the validation set. The validation curve plot for the minimum leaf size has fluctuations in the validation score up until when the minimum leaf size is about 28. These fluctuations may be attributed to the model being sensitive to the leaf size or the variability of the data.

Finding the optimal hyperparameter values was done in 47.72 seconds. The optimal value for maximum tree depth is 10, and the optimal number of samples at a leaf is 6. The accuracy of the decision tree when predicting y-values of the validation set is 89.01%. Below, we have the learning curve for the decision tree model. The training score and validation score do not converge, which may indicate that there is substantial variance and more data points may need to be added.

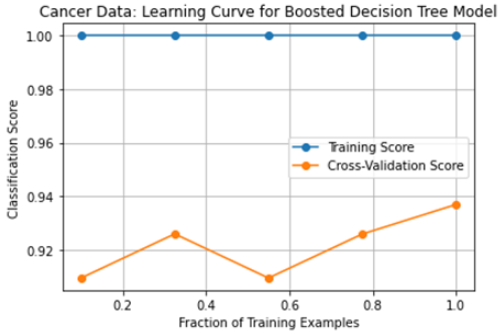


Next, the boosted decision tree algorithm for the breast cancer dataset prior to hyperparameter tuning resulted in 100% accuracy. Similar to the decision tree model, this may be occurring due to the binary classification nature of this problem. The validation curve plots below for both maximum tree depth and number of weak learners doesn't depict convergence

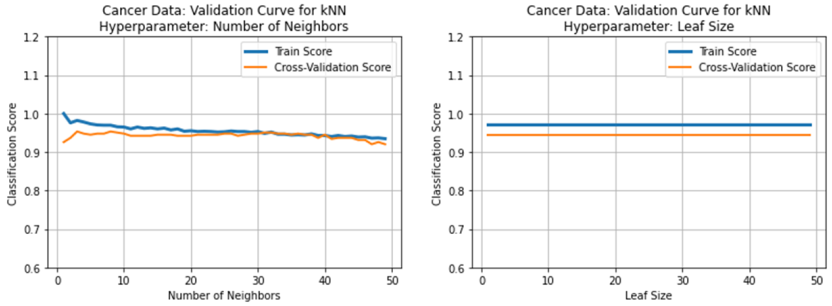
between the train and validation scores. This may be attributed to the range of the hyperparameters as it only ranges from 0 to 50. It is possible that convergence may occur when the hyperparameters are set to values greater than 50.



Finding the optimal hyperparameter values for the boosted decision tree model was done in 44.02 seconds. The optimal value for maximum tree depth is one, and the optimal number of weak learners is 50. The accuracy of the decision tree when predicting y-values of the validation set is 92.31%. Although the accuracy dropped from 100% before hyperparameter tuning to 92.31% after tuning, it is still a better model after tuning because 92.31% is still quite high. Moreover, 100% accuracy isn't necessarily better as it may indicate the model overfitting the data. Below, we have the learning curve for the boosted decision tree model. The validation scores experience fluctuations at smaller training sizes, because smaller batch sizes tend to introduce noise.

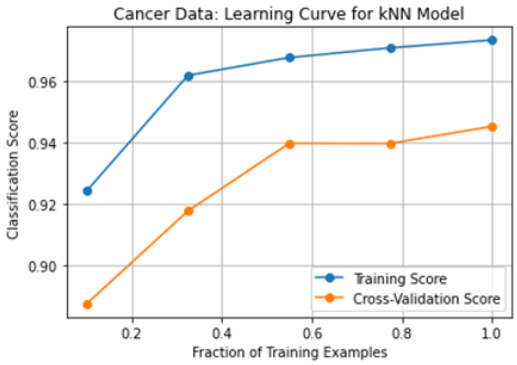


The k-Nearest Neighbors (kNN) model prior to hyperparameter tuning resulted in 98.08% accuracy, which indicates that the default hyperparameters are likely effective for the electric grid dataset. The validation curve plot for the number of neighbors hyperparameter converges and overlaps starting around 30 neighbors. This indicates that when the number of neighbors is greater than 30, there is a high level of bias. The validation curve plot for leaf size remains constant at high accuracy levels for both train and validation scores. This plot indicates that varying the leaf size hyperparameter may not impact the kNN model's fit on the data.



Finding the optimal hyperparameter values for the kNN model was done in 20.57 seconds. The optimal value for leaf size is one, while the optimal number of neighbors is three.

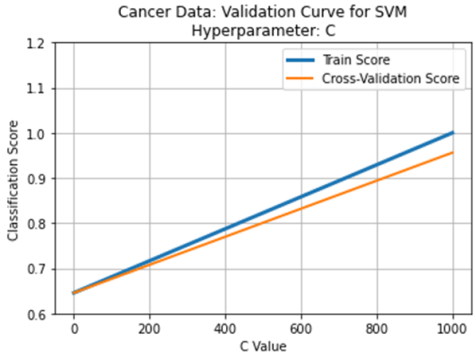
This makes sense as the classification score remained constant across all leaf sizes for both the training and validation sets in the validation curve plots. The optimal number of neighbors being 3 also makes sense as the accuracy and fit of the model declines as the number of neighbors increases as seen in the validation curve plots above. The accuracy of the decision tree when predicting y-values of the validation set is 97.8%, which is slightly lower than the model's accuracy prior to hyperparameter tuning. The learning curve plot below demonstrates high accuracy for both training and validation sets, as well as minimal gap between the two curves. This indicates that this model has optimal fit for the dataset.



Next, the Support Vector Machine (SVM) model was run on the train sets of the breast cancer dataset. The accuracy of this model prior to hyperparameter tuning is 98.63%.

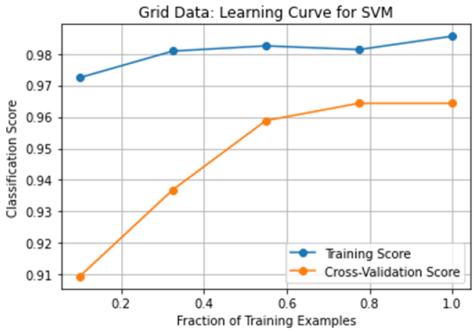
Wisconsin Breast Cancer Dataset: Hyperparameter - Kernel Type		
Kernel Type	**Training Classification Score**	**Validation Classification Score**
Linear	0.99	0.96
Poly	0.9	0.91
Rbf	0.99	0.98
Sigmoid	0.96	0.95

The classification scores above reflect that each kernel type exhibits high accuracy for both the training and validation sets. The linear and RBF kernel types performed better than the other kernel types. The target variable in the breast cancer dataset refers to whether the mass is malignant or benign, thus setting up a binary classification problem. Since this dataset is linearly separable, it might be better to utilize the linear kernel type. It was surprising to see the RBF kernel type perform so well for linearly separable data. This might be attributed to the other kernel hyperparameters at play, such as gamma.

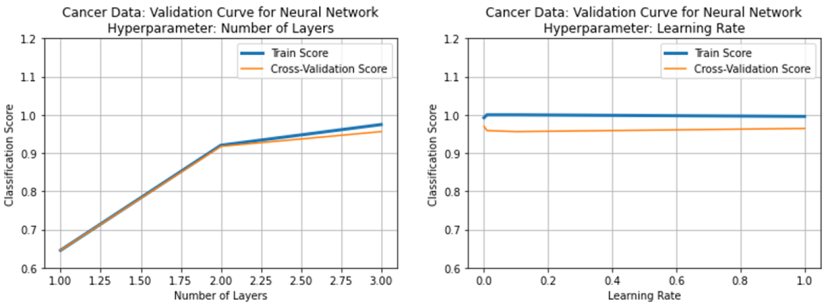


Here we have the validation curve for the C value hyperparameter. The train and validation scores overlap until about C = 200. After this point, the gap between train and validation scores increases. Increasing C generally weakens the regularization of the SVM model, thus fitting the data better. This might explain why the validation scores overlap until C = 200 and not when C is greater than 200.

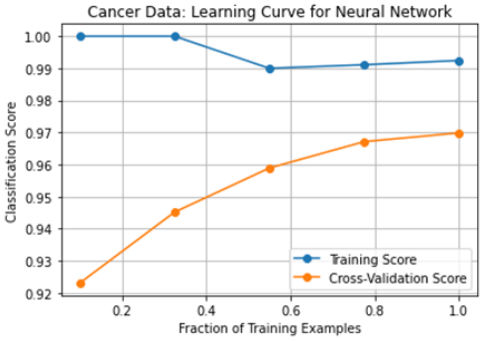
For the SVM model, the optimal kernel type is linear and the optimal C value is 1,000. Finding these optimal hyperparameter values was done in 0.53 seconds. The accuracy of the SVM model was found to be 97.8%, The learning curve for SVM below shows that the training score continues to increase for most of the training sizes. This is interesting as the curves haven't converged yet. The increase for the training score may be attributed to overfitting the data, so it's clear that we shouldn't use 100% of the training examples.



Lastly, the neural network (NN) algorithm for the electric grid dataset prior to hyperparameter tuning resulted in 99.18% accuracy. The validation curves for the number of layers are increasing and overlapping until it reaches 2 layers. For layers greater than 2, the train and validation scores exhibit a gap between them.



Finding the optimal hyperparameter values for the NN model was done in 15.48 seconds. The optimal value for hidden layer size is 1, while the optimal learning rate is 0.01. The accuracy for the model is 97.8% after hyperparameter tuning. The learning curve for the NN model below indicates that the model underfits the data since the training score decreases and validation score increases.



The various metrics to evaluate all five algorithms on the test set are depicted in the table below. The kNN model appears to have the fastest training time, which can be attributed to how the kNN model is fundamentally an instance-based algorithm. As an instance-based algorithm, it is not computationally complex to train the data as it just stores the training set in memory so it can be accessed during prediction. The kNN model does have the slowest

prediction time, which may be attributed to being more computationally complex as it needs to search and retrieve the nearest neighbors by calculating distances.

Wisconsin Breast Cancer Dataset: Model Evaluation Metrics for Test Dataset					
Metrics	**Decision Tree**	**Boosting**	**kNN**	**SVM**	**Neural Network**
Model training time	0.01356	0.13636	0.00182	0.01314	0.31627
Model prediction time	0.00056	0.00103	0.01503	0.00054	0.00073
F1 score	0.97	0.98	0.96	0.97	0.97
Accuracy	0.96	0.97	0.96	0.96	0.96
Precision	0.97	0.96	0.93	1	0.97
AUC	0.96	0.97	0.95	0.97	0.96
Recall	0.97	1	1	0.94	0.97

All the models exhibit high accuracy less than 100%, which indicates that the models have optimal fit for the breast cancer data.

Conclusion

The grid and cancer datasets were similar in that they both had numeric predictor variables and a binary target variable. The target variable for the grid dataset indicated whether the system was stable or unstable, while the target variable for the cancer dataset indicated whether the breast mass was malignant or benign. Since the two datasets are similar, it hinted as to the similarity in optimal hyperparameter values. For instance, the optimal hyperparameters for both datasets for the SVM model were C = 1000 and the linear kernel type. High C values generally weakens the regularization of the SVM models, which allows the models to fit the data better. Moreover, the linear kernel type is optimal for binary classification problems such as the grid and cancer datasets.

The grid dataset had an optimal tree depth of 1 for the decision tree model, while the cancer dataset had an optimal tree depth of 10. The difference in optimal maximum tree depth values for these two datasets may be attributed to the difference in number of predictor variables. For instance, the grid dataset had 13 predictors while the cancer dataset had 30 predictors. The cancer dataset has far more predictors, which may have contributed to the need for a deeper decision tree to capture the more complex relationship with the target variable.

This assignment was insightful in understanding how crucial it is to optimize hyperparameters for each model and each dataset. In order to successfully do this, one must fully understand the dataset and conduct exploratory analysis if needed. It would be interesting to explore other types of Boosted models, for instance, or other hyperparameters for each algorithm.

References

H*ow to know if a learning curve from SVM model suffers from bias or variance?*, *Cross Validated*. Available at: https://stats.stackexchange.com/questions/220827/how-to-know-if-a-learning-curve-from-svm-model-suffers-from-bias-or-variance/220852#220852 (Accessed: 24 September 2023).

Decision trees, random forests and gradient boosting: What's the difference? (2022) *Leon Lok*. Available at: https://leonlok.co.uk/blog/decision-trees-random-forests-gradient-boosting-whats-the-difference/#:~:text=Unlike%20random%20forests%2C%20the%20decision,this%20concept%20is%20called%20boosting. (Accessed: 24 September 2023).

Learning curve (no date) *Learning Curve - Yellowbrick v1.5 documentation*. Available at: https://www.scikit-yb.org/en/latest/api/model_selection/learning_curve.html (Accessed: 24 September 2023).

Seven most popular SVM kernels (2020) *Dataaspirant*. Available at: https://dataaspirant.com/svm-kernels/#t-1608054630727 (Accessed: 24 September 2023).

Validation curve (2023) *GeeksforGeeks*. Available at: https://www.geeksforgeeks.org/validation-curve/ (Accessed: 24 September 2023).

VanderPlas, J. (2023) *Python Data Science Handbook Essential Tools for working with data*. Bejing: O'Reilly.