CS7641 A1: Supervised Learning

Scott Schmidl
sschmidl3@gatech.edu

I. Introduction - Dataset Explanation

The two datasets used in this paper were chosen with two very specific ideas in mind. Both of these datasets are centered around the health and wellness of human beings.

For the first dataset, the idea was to help people understand the features that most affect Cardiovascular Disease (CVD) with the hopes of reducing the number of people with CVD. We attempt to predict if a person has Cardiovascular Disease or not given their height in centimeters, weight in kilograms, systolic blood pressure in millimeters of mercury, diastolic blood pressure in millimeters of mercury, and age in years. Preprocessing was performed to remove columns that were not needed or that exhibited multicollinearity, reduce the amount of samples for modeling, and scale the data so that it is standardized. This data is relatively balanced.

The metric chosen to represent the model performance for dataset one is recall. Recall helps the analyst see how the model performs with true positives and false negatives. The score will increase if true positives increase or false negatives decrease. In this case, we want false negatives to decrease, because this means that we are less likely to predict that a person doesn't have CVD, when they really do.

For the second dataset, the idea was to help people understand how foods are grouped based on the nutrition elements they contain. We attempt to predict food groups given their level of protein in grams, carbohydrate in grams, cholesterol in milligrams, water in grams, and sodium in milligrams. Similar preprocessing was performed on this dataset to remove columns that were not needed, that exhibited multicollinearity, or that had too much data missing. Additional preprocessing steps were to reduce the amount of samples for modeling, correct and rename column, fill in some missing data, and scale the data so that it is standardized. The classes are relatively imbalanced with the majority class occurring 44%, class two occurring 32%, class three occurring 13%, and class four occurring 11% of the time.

The metric chosen to represent the model performance for dataset two is balanced accuracy (BA). BA does a decent job identifying how the model performs with imbalanced classes. The score will increase if true positives increase or false negatives decrease. In this case, we want false negatives to decrease, because this means that we are less likely to predict that a person doesn't have CVD, when they really do.

II. Decision Trees

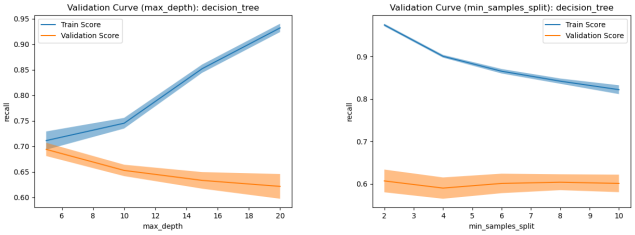
Decision Trees are built by a simple set of decision rules. These decision rules ask true or false questions about the features of the data. The objective is to build a tree that can predict the value of some dependent variable. The deeper the tree grows, the more complex and the better the fit of the model. This can, however, be prone to overfitting.

A. Dataset 1

In figure 1, no pruning was used and the model has learned all of the training data, but has not generalized well. This lack of generalization has led to overfitting. It's interesting to note that at around 5,200 samples the model begins to perform better on the validation set. The training score has a greater than 95% recall, while the validation score stays around 60%. It is worth exploring with more data.

Fig. 1: Cardiovascular Disease - Learning Curve

In figure 2(a), the hyper-parameter Max Depth is analyzed. Max Depth is a form of prepruning that chops the tree a some depth. At a max depth of five, we see the train and validation scores are quite close - around 70%. After this point, the model begins to overfit quite rapidly. This overfit is due to forcing the model to be to complex - the level of depth - and is therefore memorizing too much of the training data while not generalizing. This dataset would likely perform better with a max depth less than five.



(a) Max Depth (b) Min Samples Split

Fig. 2: Cardiovascular Disease Validation Curves

In figure 2(b), the hyper-parameter Min Samples Split is analyzed. Min Samples Split is a form of "mid" pruning that required a certain number of samples at a node in order for it to split. The default for this model is a minimum sample requirement of two, however at this value we are overfitting. As we increase the amount of minimum samples to 10, the training score decreases with a slight increase in validation score. At 10 samples, the model has potential to get better, but would probably benefit the most from being combined with another hyper-parameter, perhaps setting a Max Depth value of five. For training, the Recall score starts at 90% and drops to 80%, while the validation score starts almost constant with slight decreases and increases around 60%.

B. Dataset 2

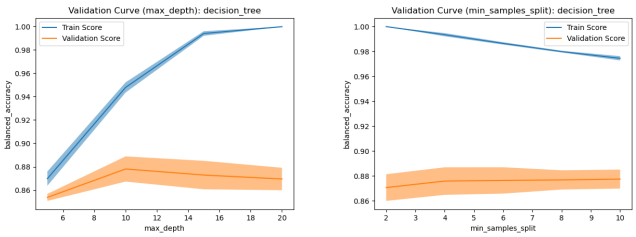
Due to the class imbalance of this dataset, the class weight hyper-parameter was set to balanced for both learning and validation curves, below.

In figure 3, no pruning was used and the model has learned all of the training data just as before, but has not generalized well. This lack of generalization has led to overfitting. It's interesting to note that at around 3,400 samples the model begins to perform better on the validation set. The training score has a balanced accuracy at 100%, while the validation score is between 86% and 88%. It is worth exploring with more data.

In figure 4(a), the hyper-parameter Max Depth is analyzed. At a max depth of five, we see the train and validation scores are quite close - around 86%. After this point, the model begins to overfit quite rapidly. This overfit is due to model complexity and is therefore memorizing too much of the training data while not generalizing. This dataset would likely perform better with a max depth less than five. As the depth increases from five to 20 we see an increase in overfitting until the training score reaches almost 100%.

In figure 4(b), the hyper-parameter Min Samples Split is analyzed. At a value of two, we are overfitting with a train score of 100% and validation score of 87%. As we increase the amount of minimum samples to 10, the training score decreases with a slight increase in validation score. At 10 samples, the model has potential to get better, but would benefit the most from being combined with another hyper-parameter - perhaps setting a

Fig. 3: Food Group - Learning Curve



(a) Max Depth (b) Min Samples Split

Fig. 4: Food Group Validation Curves

Max Depth value of five - or increasing its value. For training, the Recall score starts at 100% and drops to 97%, while the validation score is mostly constant with slight decreases and increases around 87%.

III. KNN

K Nearest Neighbors is an instanced-based, lazy learning model. This means that the model only stores the training data. In our case of classification, a majority vote of the nearest neighbors is performed. The nearest neighbors are calculated by taking the Euclidean distance.

A. Dataset 1

In figure 5, as the number of samples increases the validation score decreases. This is mean that the model isn't complex enough and that even more data is required. Once again the model is slightly overfit which a training score around 76% and validation score around 66%.

In figure 6(a), the hyper-parameter N Neighbors is analyzed. At a neighbor count of three, both train and validation scores approach each other. This suggests that the model is learning to generalize. There is an inflection point at five, at which point the scores begin to level out but continue in there trajectory. This suggest that with more neighbors we could get even less overfitting. After

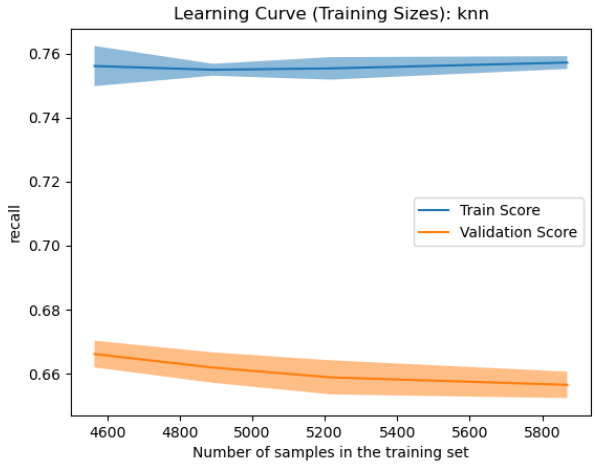


Fig. 5: Cardiovascular Disease Learning Curve

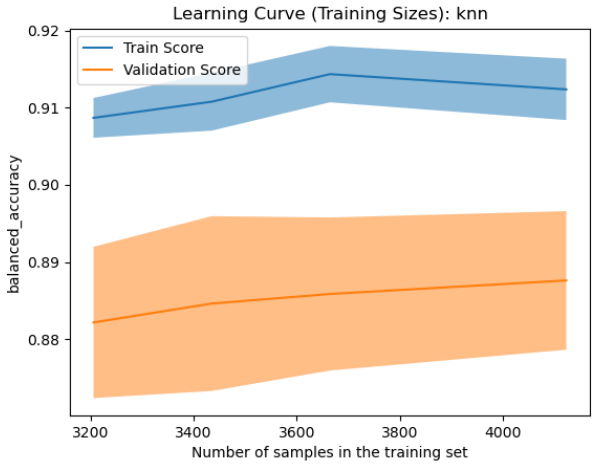
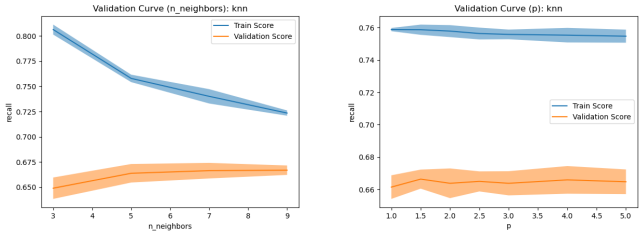


Fig. 7: Food Group Learning Curve

nine neighbors the train score is about 72%, while the validation score is 67%.

In figure 6(b), the hyper-parameter P is analyzed. P is the power parameter for the Minkowski (Euclidean) distance metric, which is used by default. The train score and validation score stay around 76% and 66% respectively. This suggests that adjusting this power parameter, alone, is not adding value. This most likely means that a different distance metric or weights parameter should be used.



(a) N Neighbors (b) Power Parameter

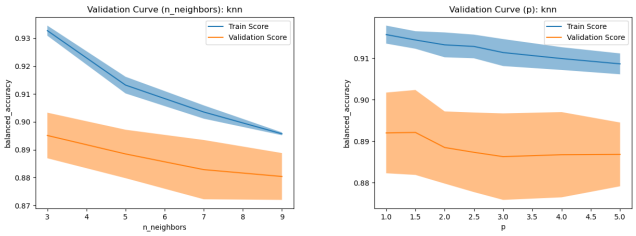
Fig. 6: Cardiovascular Disease Validation Curves

B. Dataset 2

In figure 7, the training score is around 91%, while the validation score is between 88% and 89%. KNN performs quite well as the model with dataset two shows little signs of overfitting. Upon deeper inspection, this model begins to perform better on validation score but worse on training score. The conclusion here is that as we gather more data the model continues to generalized better.

In figure 8(a), the hyper-parameter N Neighbors is analyzed. At a neighbor count of three, both train and validation scores approach each other. This suggests that the model is learning to generalize. There is an inflection point at five for the training score, at which point the slope of the score becomes smaller. This suggests that with more neighbors we could get an even better fit. After nine neighbors the train score is about 90%, while the validation score is 88%.

In figure 8(b), the hyper-parameter P is analyzed. P is the power parameter for the Minkowski (Euclidean) distance metric, which is used by default. The train score and validation score stay around 91% and 89% respectively. This suggests very little overfitting, if any, but that adjusting this power parameter, alone, is not adding much value. This most likely means that a different distance metric or weights parameter could be used, if a little more performance is required.



(a) N Neighbors (b) Power Parameter

Fig. 8: Food Group Validation Curves

IV. Boosting

Gradient Boosting Classifier is an ensemble algorithm like decision tree but in this case it attempts to minimize some cost, or loss, function using gradient decent. The model also has the ability to performing resampling without replacement.

A. Dataset 1

In figure 9, boosting immediately begins to perform poorly. The train score is 69%, while the validation score around 67%. However, at around 5,200 samples the model begins to improve in both train and validation

scores. The are no signs of overfitting, however more data is likely needed for this model.

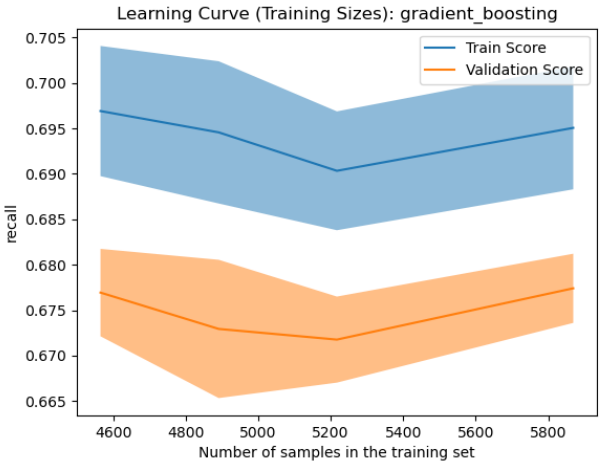
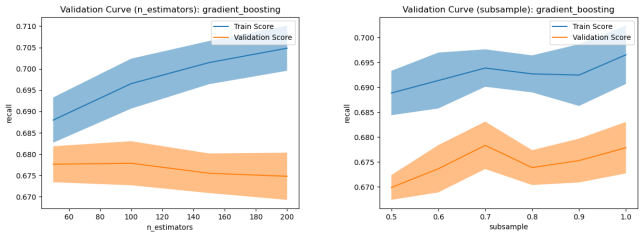


Fig. 9: Cardiovascular Disease Learning Curve

In figure 10(a), the hyper-parameter N Estimators is analyzed. At an estimator count of 60, both train and validation scores are quite close, but drifting apart, suggesting that the model is beginning to overfit. There's a slight inflection point at 100 estimators but the model continues to overfit. With less estimators, we could have less overfitting. This likely means that we don't see so many boosting stages and that this model is too complex for the data. A simpler model suggested by Occam's razor could be more beneficial.

In figure 10(b), the hyper-parameter Subsample is analyzed. Subsample is the fraction of samples used to fit the learners. The train score and validation score stay around 69% and 67% respectively. This suggests very little overfitting, if any, but that adjusting Subsamples, alone, is not adding much value. When smaller than one, Stochastic Gradient Boosting is used. There is an inflection point at 0.7, and drop in performance until 0.8, which likely means that the model got stuck in a local minima and therefore wasn't able to minimize the loss.



(a) N Estimators (b) Subsamples

Fig. 10: Cardiovascular Disease Validation Curve

B. Dataset 2

In figure 11, boosting perform quite well. The train score is 94%, while the validation score around 89%. The training score has a constant negative slope while the validation score remain mostly constant. This suggests that the model is underfitting or that the model is beginning to generalize better. There are minimal signs of overfitting, however more data is likely needed for this model.

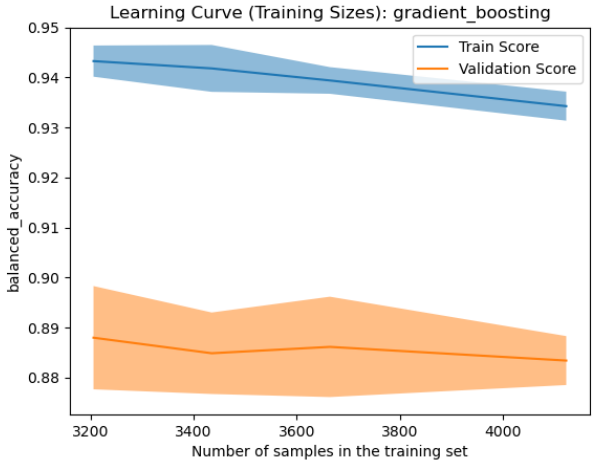


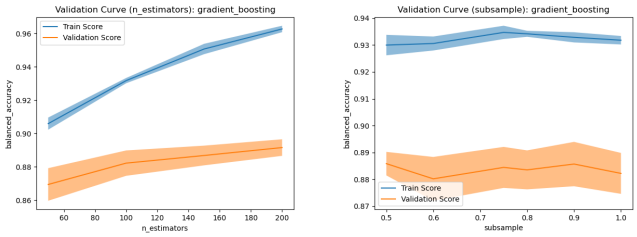
Fig. 11: Food Group Learning Curve

In figure 12(a), the hyper-parameter N Estimators is analyzed. At an estimator count of 60, both train and validation scores are quite close and not much overfitting can be seen. Between 60-100 esitmators the model is showing some signs of overfitting. There's a slight inflection point at 100 estimators but the model continues to overfit (train and validaiton are drifting apart). With less estimators, we could have less overfitting, but the difference is so small that it's worth investigating the affect with more estimators. The train and validation scores range from 90% - 96% and 87% - 88% respectively.

In figure 12(b), the hyper-parameter Subsample is analyzed. The train score and validation score stay around 93% and 88% respectively. This suggests very little overfitting, but adjusting Subsamples, alone, is not adding much value. When smaller than one, Stochastic Gradient Boosting is used. There are a few inflection points along with drops in performance, which likely means that the model got stuck in a local minima and therefore wasn't able to minimize the loss.

V. Neural Networks

Multilayer Perceptron is a neural network algorithm that uses some function and gradient decent to calculate gradients using backpropagation. The model accepts an input layer (independent data), intermediate layer(s), and an output layer (dependent data).



(a) N Estimators (b) Subsamples

Fig. 12: Food Group Validation Curves

A. Dataset 1

For this dataset, the neural network has a default initial learning rate equal to 0.01, to help the model converge.

In figure 13, the neural network performs quite poorly. The train score is 69%, while the validation score around 67%. The training score has a dip around 5,200 samples while the validation score has two changes in direction. This suggests that the model is underfitting, but then begins to overfit. More data and a more complex model are likely needed to help increase our metrics.

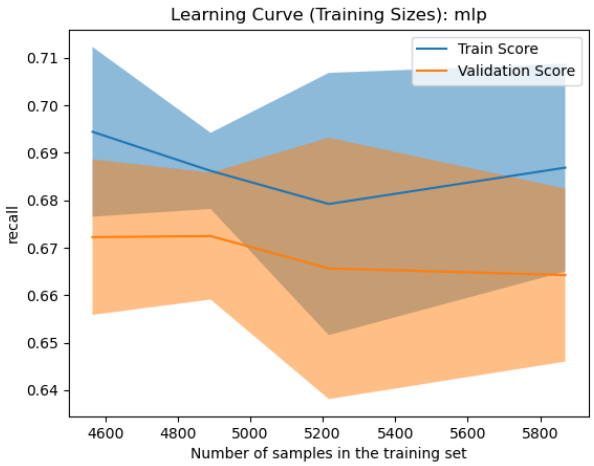
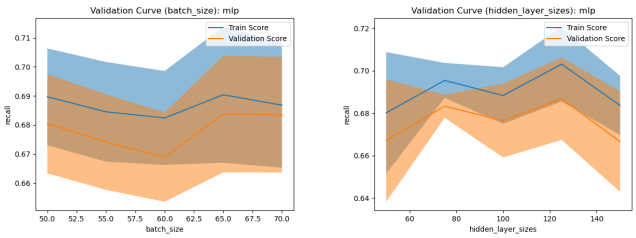


Fig. 13: Cardiovascular Disease Learning Curve

In figure 14(a), the hyper-parameter Batch Size is analyzed. The Batch Size hyper-parameter determines the number of samples used for each set of training. At a size of 50, both train and validation scores are quite close and overfitting is not suspected. There is little fluctuation in the scores, but a batch size of 70 returns the best results from the entire range. There are a couple inflection points with slight drops in performance. This could be the results of getting stuck at a local minima. The train and validation scores both stay around 69% and 68%, respectively.

In figure 14(b), the hyper-parameter Hidden Layer Size is analyzed. The train score and validation score stay around 68% and 67% respectively. This suggests very little overfitting, but adjusting hidden layer sizes, alone, is not adding much value. There are a few inflection points along with drops in performance, which likely means that the model got stuck in a local minima and therefore wasn't able to minimize the loss. Due to the few inflection points, more investigation should be done into this parameter to identify if there is a good value or if it continues to be erratic.



(a) Batch Size (b) Hidden Layer Sizes

Fig. 14: Cardiovascular Disease Validation Curves

In figure 15, analysis is performed at Max Iterations. The train log loss and validation log loss stay around 0.56 and 0.54 respectively. This suggests very little overfitting, but adjusting iteration, alone, is not adding much value. The model appears to converge at 120 iterations after which point there is no more improvements to the model. The number of iterations are helping the model to converge, but not providing much in performance results.

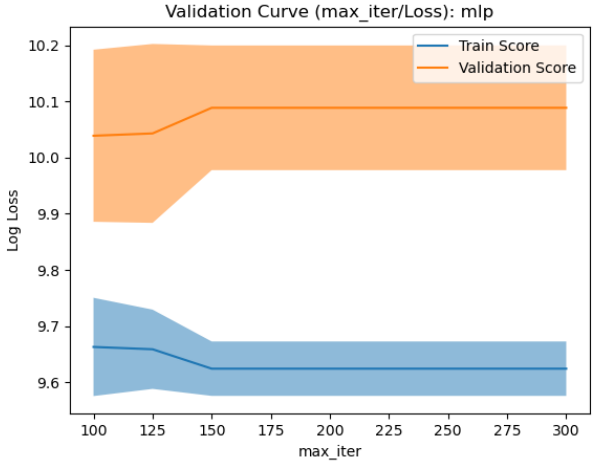


Fig. 15: Cardiovascular Disease - Loss VS Epochs

B. Dataset 2

For this dataset, the neural network has a default initial learning rate equal to 0.01 and a max iteration value of 300, to help the model converge.

In figure 16, the neural network performs quite well. The train score is 92%, while the validation score is between 88% and 90%. The training score increases until around 3,700 samples where it turns to have a negative

slope. The validation score continues to increase until 3,700 sample, at which point it also has a negative slope. This suggests that the model is underfitting. The are minimal signs of overfitting. More data and a more complex model could prevent the model from beginning to underfit at 3,700 samples and therefore increase performance.

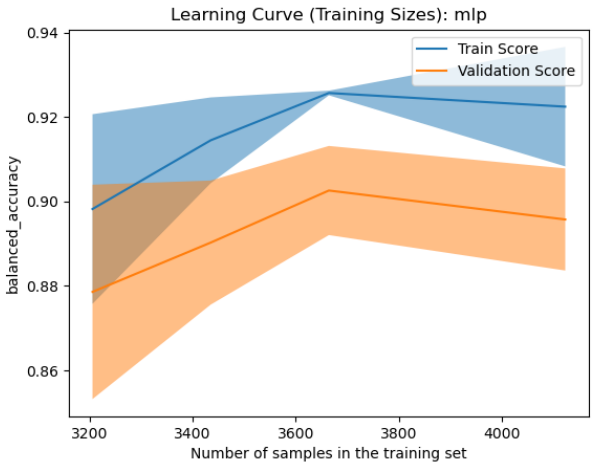


Fig. 16: Food Group Learning Curve

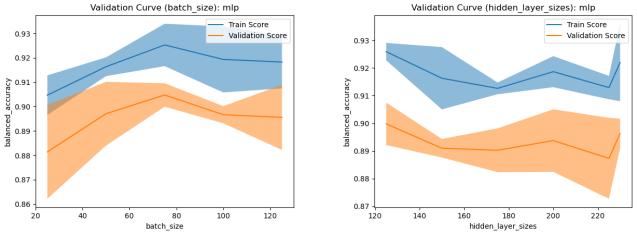
In figure 17(a), the hyper-parameter Batch Sizes is analyzed. At a size of 30, both train and validation scores are quite close and overfitting is not suspected. There is little fluctuation in the scores, but a batch size of 70 returns the best results from the entire range. There are a couple inflection points with slight drops in performance. This could be the results of getting stuck at a local minima. The train and validation scores both stay around 91% and 89%, respectively.

In figure 17(b), the hyper-parameter Hidden Layer Size is analyzed. The train score and validation score stay around 92% and 90% respectively. This suggests very little overfitting, but adjusting hidden layer sizes, alone, is not adding much value. There are a few inflection points along with drops in performance, which likely means that the model got stuck in a local minima and therefore wasn't able to minimize the loss. Due to the few inflection points, more investigation should be done into this parameter to identify if there is a good value or if it continues to be erratic. At a size of 225, we see the model begin to improve on both training and validation.

In figure 18, analysis is performed at Max Iterations. The train log loss and validation log loss stay around 0.22 and 0.14 respectively. This suggests slight overfitting, but adjusting iteration, alone, is not adding much value. The model appears to converge at 200 iterations. The number of iterations are helping the model to converge, but not providing much in performance results.

VI. SVM

Support Vector Machines is a model that operates in high-dimensional space to find a hyperplane. Utilizing



(a) Batch Size (b) Hidden Layer Sizes

Fig. 17: Food Group Validation Curves

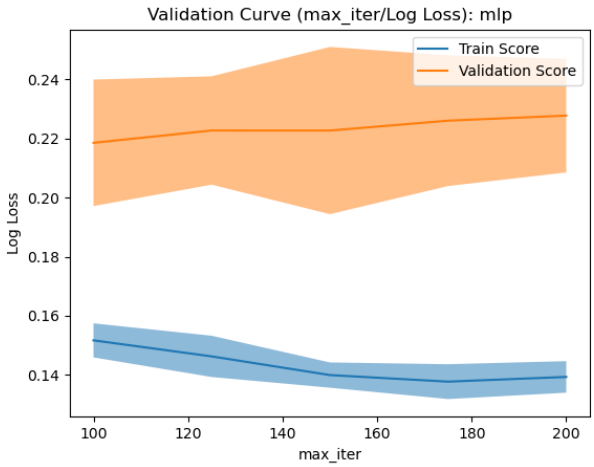


Fig. 18: Food Group - Loss VS Epochs

this hyperplane, SVMs attempt to maximize the margin between classes while simultaneously minimizing the number of misclassifications.

A. Dataset 1

For this dataset, the SVM uses a default kernel equal to RBF to help with non-linearity in the data.

In figure 19, the SVM performs quite poorly. The train score is 65%, while the validation score is around 64%. The training score decreases until around 4,900 samples where it then begins to level out with one more slight turn at 5,200 before it changes to a zero slope. The validation score continues to decrease until 5,200 samples, at which point it begins to have a positive slope. This suggests that the model is underfitting and there are no signs of overfitting. More data and a more complex model could prevent the model from underfitting and therefore increase performance.

In figure 20(a), the hyper-parameter C is analyzed. The C hyper-parameter is the regularization parameter. With regularization at 0.6, both train and validation scores are quite close and overfitting is not suspected. There is little fluctuation in the scores, but they are trending up. I suspect that increasing the value of C even more could only help the model. However, the fact that it performs better with high values of C suggests that my

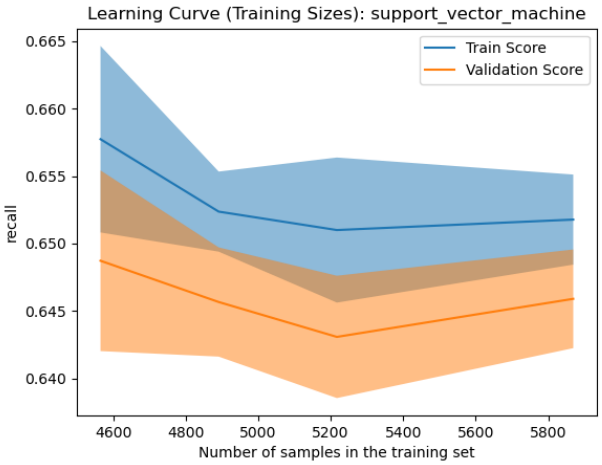
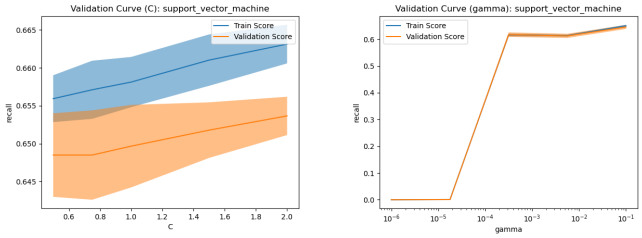


Fig. 19: Cardiovascular Disease Learning Curve

data doesn't have a ton of noise but is also not doing that well since higher values give a higher penalty for misclassifications. This could also mean that my data are good predictors and linearly separable. The train and validation scores both stay around 66% and 65%, respectively.

In figure 20(b), the hyper-parameter Gamma is analyzed. The Gamma hyper-parameter is the kernel coefficient. The train score and validation score stay together from 0% and 60%. This suggests no overfitting, and adjusting Gamma values greater than 10^{-1} could help improve the model. There are two massive inflection points with a massive spike in performance from 10^{-5} to 10^{-4}. This spike suggests that this model is well-balanced, has good generalization, stable performance, and has a good balance between bias and variance. This behavior is likely due to a smoothing effect of the decision boundary.



(a) C - Regularization Parame- (b) Gamma - Kernel Coefficient ter

Fig. 20: Cardiovascular Disease Validation Curve

B. Dataset 2

For this dataset, the SVM uses a default kernel equal to Poly which uses a default degree of three. This will create polynomial features which may give a better fit given the data. Due to imbalances in the data, the model also uses a default class weight of balanced.

In figure 21, the SVM has a decent performance. The train score is 82%, while the validation score is also about 82%. The training score decreases until around 3,400 samples where it begins to increase. The validation score has a slight negative slope until 3,4700 sample, at which point it also has a positive slope. This suggests that the model is performing well with no overfiting. This performance suggests that with more data the model would continue to improve.

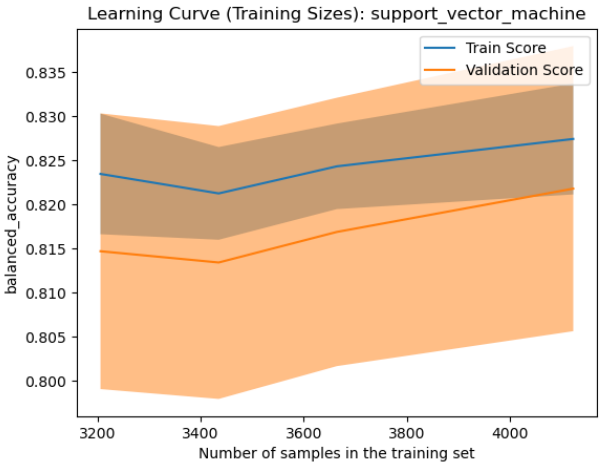


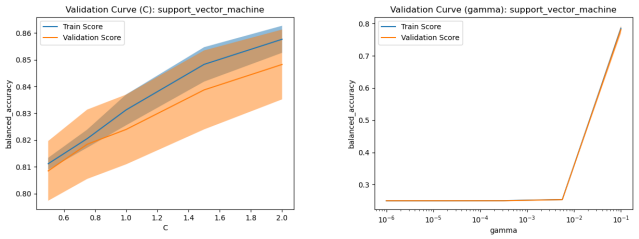
Fig. 21: Food Group Learning Curve

In figure 22(a), the hyper-parameter C is analyzed. With regularization at 0.6, both train and validation scores are quite close and overfitting is not suspected. There is a constant increase in both scores. I suspect that increasing the value of C even more could only help the model. However, the fact that it performs better with high values of C suggests that my data doesn't have a ton of noise but is also not doing that well since higher values give a higher penalty for misclassifications. This could also mean that my data are good predictors and linearly separable. The train score ranges from 81% to 85%. The validation score ranges from 81% to 84%.

In figure 22(b), the hyper-parameter Gamma is analyzed. The train score and validation score stay together from 20% and 80%. This suggests no overfitting, and adjusting Gamma values greater than 10^{-1} could help improve the model. There are two massive inflection points with a massive spike in performance from 10^{-2} to 10^{-1}. This spike suggests that this model is well-balanced, has good generalization, stable performance, and has a good balance between bias and variance. This behavior is likely due to a smoothing effect of the decision boundary.

VII. Conclusion

In this project, we conducted an in-depth analysis using two distinct datasets and applied five different machine learning algorithms to each dataset. Our primary objective was to explore the performance of these



(a) C - Regularization Parame- (b) Gamma - Kernel Coefficient ter

Fig. 22: Food Group VALIDATION CURVES

algorithms across different hyper-parameters and training sizes utilizing Recall and Balanced Accuracy scores.

Throughout our analysis, we observed interesting trends in the behavior and effectiveness of the algorithms. The diversity in dataset size, class distribution, and feature complexity, played a significant role in shaping the performance of the algorithms.

Regarding the Cardiovascular Disease dataset, we found that Decision Tree has overall better performance on training but similar on validation. SVM, Gradient Boosting, Multilayer Perceptron, and K Nearest Neighbors exhibited worse training performance maintaining in the 60% to 70% range. While the performance could be better, I do see hope for improving these models with better tuning.

On the other hand, the Food Group dataset exhibited much better performance overall on the same algorithms. While each model performed above 80% the Gamma parameter of SVM was only able to reach a score of 60%. Out of the five algorithms, Decision Trees struggled to generalize effectively and therefore overfit the most.

The exercise of hyper-parameter tuning provided valuable insights into improving the performance of the algorithms, highlighting the importance of fine-tuning parameters to achieve optimal results.

In conclusion, this project underscores the importance of understanding dataset characteristics and selecting appropriate machine learning algorithms tailored to the specific problem. Moving forward, further research could focus on exploring additional algorithmic approaches, grid search to aid in hyper-parameter tuning, or feature engineering techniques to enhance predictive performance.

VIII. Resources

[1] *API Reference.* Scikit-Learn. https://www.python.org/.
[2] *API Reference.* Pandas. https://pandas.pydata.org/.
[3] *API Reference.* Jupyter. https://jupyter.org/.
[4] *API Reference.* IPython. https://ipython.org/.
[5] *API Reference.* MatPlotLib. https://matplotlib.org/stable/.
[6] *API Reference.* NumPy. https://numpy.org/.
[7] *API Reference.* Seaborn. https://seaborn.pydata.org/index.html.
[8] *API Reference.* OpenPyXl. https://openpyxl.readthedocs.io/en/stable/tutorial.html.
[9] *API Reference.* Scikit-Learn. https://scikit-learn.org.
[10] Nakamura, K. (2023). *ML LaTeX Template*.
[11] *Data Source* Nutrition Facts Database Tools and Spreadsheet. https://tools.myfooddata.com/nutrition-facts-database-spreadsheet.php.
[12] *Data Source* Cardiovascular Disease. https://www.kaggle.com/datasets/colewelkins/cardiovascular-disease.