Supervised Machine Learning Models for Pima Diabetes and Wisconsin Breast Cancer Predition

I. DATASETS

The purpose of this project is to develop, analyze, and compare several supervised machine learning models for two real-life renowned binary classification datasets for predicting diabetes (Pima) and breast cancer (Wisconsin). The following datasets were selected:

- Dataset DS1 Pima Indian Diabetes (size) [1]
- Dataset 2 DS2 Wisconsin Breast Cancer (size) [2]

Both are binary classification datasets for predicting the presence of diabetes or breast cancer. The reasons these are interesting datasets for Machine Learning classification are:

- They're both imbalanced with a 65/35 split biased towards training samples being negatively classified. However, while the Pima dataset is all numerical including outliers, and features that have "zero" (incorrect) measurements, the Wisconsin dataset has all ordinal columns.
- Despite the imbalance, it's more important to get the minority predictions right rather than simply an overall prediction accuracy since the consequences of missed positive classifications are life altering.
- Yet at the same time, cannot swing the pendulum completely that way and severely compromise negative classifications either since they can result in costs/hardship for unwarranted further medical procedures. So, deciding a balanced appropriate performance metric is itself a challenge.
- The number of training samples (700-800) is modest by machine learning standards, and could pose challenges with training, although this remains to be seen.
- Ideally, preprocessing is required since the Pima dataset contains outliers and many false "zero" values. However, only minimal preprocessing is performed since that's not a focus of this assignment, rather it's being considered as a factor of comparison between the different algorithms.

II. EXPERIMENTAL METHODOLOGY

A. Performance Metric

Due to the moderately imbalanced nature of both datasets, and the desire for a metric that summarizes the performance of the minority (positive) class in a binary classification, the Precision Recall curve or the Precision Recall Area Under Curve (PRAUC) is chosen. This is quantified using the Average Precision (AP) metric. The Receiver Operating Characteristic Area Under Curve (ROC-AUC) metric was also considered but was not selected since it can be overly optimistic with imbalanced datasets [3].

The general experimental methodology applied to each dataset and classification algorithm is summarized in Table 1.

III. RESULTS

A. Decision Tree

The initial (untuned model) learning curves of both datasets showed a problem of low bias and high variance. This is because the trees were allowed to grow fully unconstrained, creating many splits and fitting the training data too closely. Validation-set performance continued to change with different training set sizes, indicating adding more training samples would help (which however isn't an option here). To trade bias for better (lower) variance, cost-complexity pruning was performed to remove sub-trees by making a trade-off between a node's cost and its purity [4]. Feature split algorithm (Gini and Entropy) was selected as another hyperparameter and post pruning was implemented for both.

As expected, this added considerable bias and improved variance for both datasets (Figure 1). Gini post-pruning was more effective for the Pima dataset and Entropy post pruning for the Wisconsin dataset. While the preferred choice would've been Gini due to its faster computation time, the performance improvement to Wisconsin with entropy was considerable enough to pay the penalty, especially since these are smaller datasets. A hypothesis for the reason one feature split mechanism performed better than the other across the datasets is possibly due to the nature of data. Only the better performing post-pruning validation curves (Gini for Pima & Entropy for Wisconsin) are shown here, while the other set is available in the uploaded files. No grid search was needed here since the validation curves/results for the four possible pruning combinations (Gini/Entropy for DS1/DS2 each) were easily generated individually through training and cross validation, and the best combination for each dataset was selected.

The learning curves of the tuned models are shown in Figure 2. The training and validation set curves are trending toward convergence indicating that the tuned models have reduced overfitting characteristics and are able to better generalize to unseen samples. Adding more data and exploring more hyperparameters for tuning are likely to help further.

Finally, the tuned models were evaluated against the withheld (unseen) testing datasets. Table 2 summarizes the significant performance metrics, and Figure 3 shows the Precision Recall Curves. While the Decision Tree model for the Wisconsin dataset was high performing, it was very modest for the Pima dataset. Possible hypotheses for modest performance are likely due to several features with incorrect zero values and possibly multiple similar samples in the training dataset. It is expected that boosting can improve the Pima metrics. For small datasets, both training and prediction times were very small.

Step	*Detail*
Basic Data Preprocessing	• Dataset 2: removed “ID” column as it has no significance. • Dataset 2: removed 16 records with a “?” in one of the attributes. • Both: Feature scaling and standardizing using the Stand-ardScaler [5]. Required for nearest-neighbors, neural networks and SVM with Radial Bias Kernel
Split into Training/Testing Sets	• An 80/20 split with stratified splitting is used. • Training set also used for k-fold cross validation for learning and validation. • Testing set ONLY used for final metrics and measurement
Fit training data into a “base”/default model without tuning and generate learning curve	• Learning curve generated with K-fold cross validation. Pur-pose is to study default model bias/variance and inform next actions. For e.g., if low bias/high variance is observed, adding regularization may be a next step. Or if validation-set has not plateaued, more samples could help.
Identify Hyperparameters	• For this assignment, only a few hyperparameters are selected for each algorithm, along with their justification
Generate Validation Curves	• Generated for each hyperparameter individually against the base model. Its purpose is to understand the general impact of this parameter on bias/variance. • Sometimes, interplay between two hyperparameters might require multiple intermediate validation curves to understand their interaction. E.g., learning rate and number of estimators in Boosting . • Parameter ranges for “Grid Searches” will be refined/narrowed in this step, which helps in cutting down grid search training time.
Execute GridSearch with k-fold cross-validation	• Executed for the hyperparameter ranges identified to identify the most optimal parameter settings. • Detailed ranked results written to an output file. • Results analyzed and referenced with validation curves, to select optimal hyperparameter settings. Occam’s razor [8] is utilized to prefer selection of parameters that lead to simpler models while remaining reasonably close to best performance
Train model using identified hyperpa-rameter settings and generate learning curve	• Evaluate model bias/variance and over/underfitting. Rinse and repeat above steps as needed (with only K-fold cross validated training data)
Retrain tuned model with entire train-ing set	• To ensure all available training data is utilized for the final model
Evaluate finally against testing dataset	• Generate performance metrics for final evaluation on unseen data

TABLE I. EXPERIMENTAL METHODOLOGY

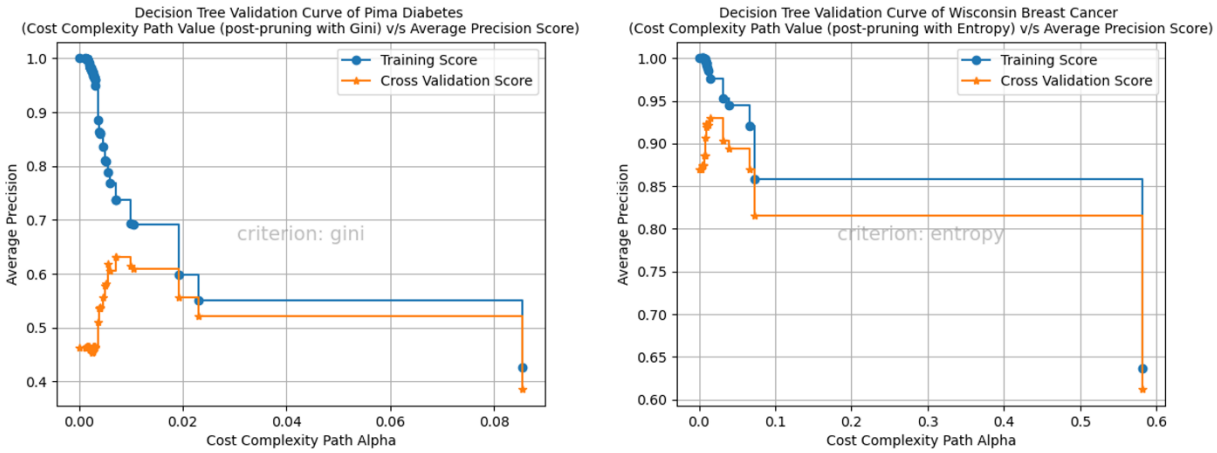


Figure 1. Decision Tree Cost Complexity Pruning Validation Curves

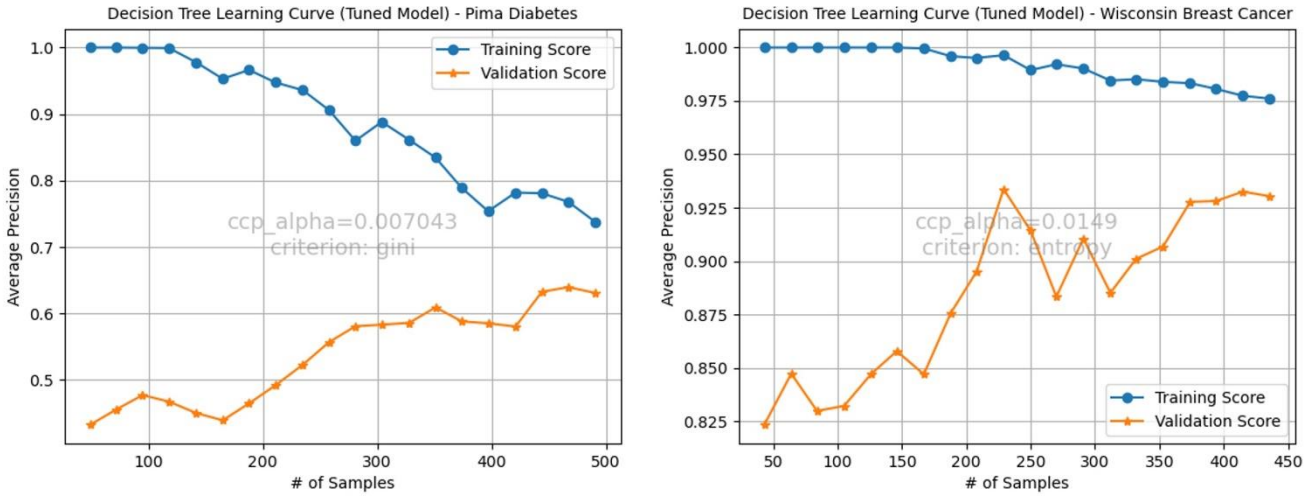


Figure 2. Decision Tree Learning Curves (tuned model)

Dataset	Average Precision	Confusion Matrix		Training Time	Prediction Time
Pima	0.70	76	24	0.0033s	0.0001s
		13	41		
Wisconsin	0.92	87	2	0.0013s	0.0001s
		4	44		

Table 2 Decision Tree Performance Metrics

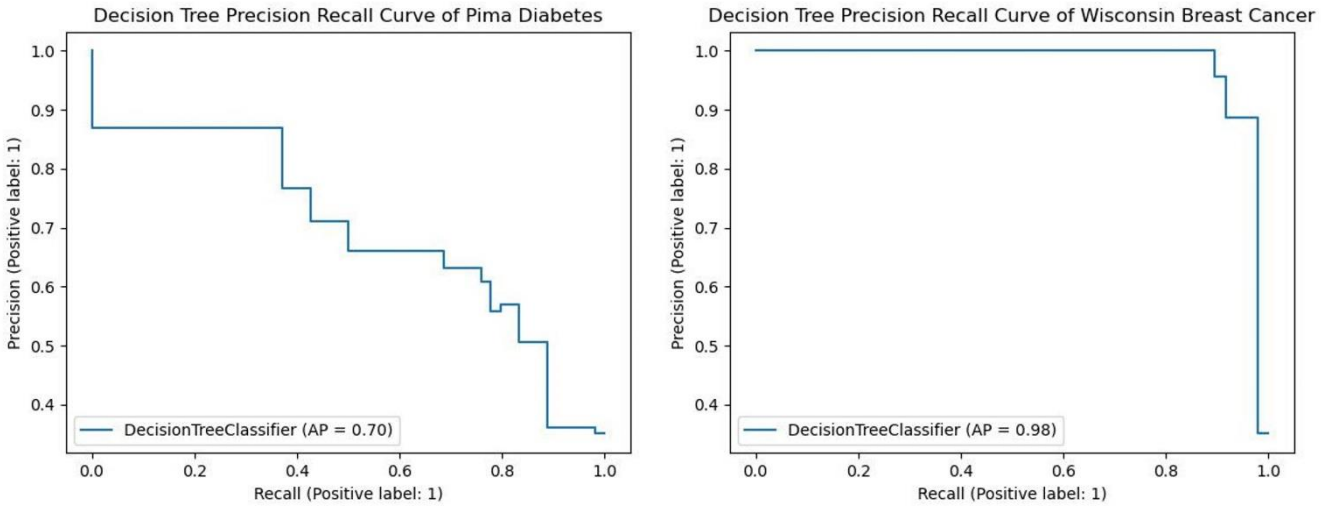


Figure 3 Decision Tree Precision Recall Curve

B. Boosting

Boosting was performed with the AdaBoost [9] using weak Decision Tree "stumps", with default of 50 weak estimators. Even with no tuning, the Pima booster exhibited low bias, while the Wisconsin booster had zero bias. This demonstrates the ability of boosting to learn from performance errors in one "round" and sequentially train new trees with

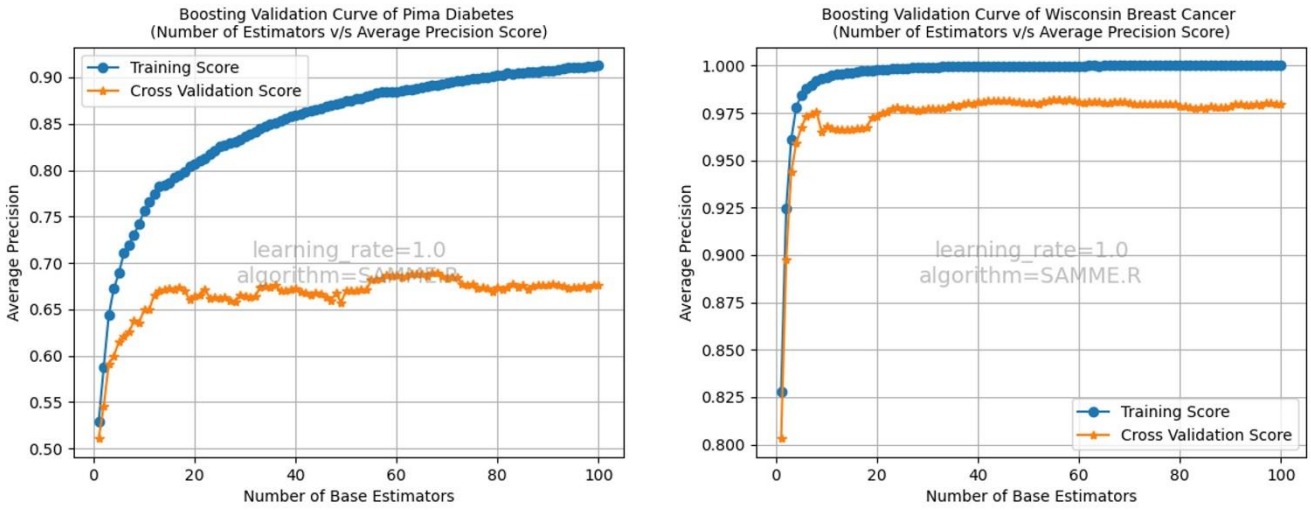


Figure 4. Boosting Validation Curves for Number of Estimators

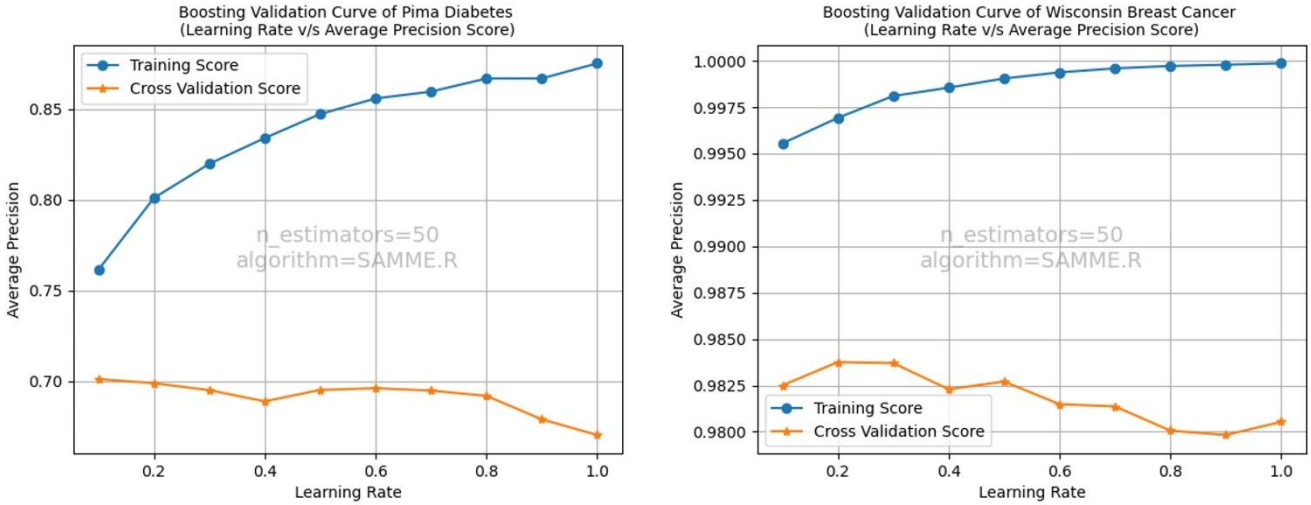


Figure 5. Boosting Validation Curves for Learning Rate

continual improvement. This is also evidenced by comparing the untuned learning curves of the previous Decision Tree and this Boosted Tree which indicates a 25 and 12 basis point improvement in validation set performance of the Pima and Wisconsin datasets respectively. The untuned learning curves are not shown here but are available in the uploaded files. For both datasets, the next step was to trade bias for improvement in variance. Two hyperparameters chosen were number of base **estimators** and **learning rate**. Validation curves for number of estimators while retaining default learning rate (of 1.0) (Figure 4) showed that validation set performance plateaus, while that of training continues to rise, indicating the onset of overfitting. Validation plots for learning rate (Figure 5) while retaining default number of estimators (50) show similar behavior which suggests that an interplay exists between the two. Intuition suggests that the two parameters are inversely related since a high number of estimators can afford a slow rate of learning from errors, whereas lower the estimators, the faster the ensemble would need to learn to achieve similar performance. This was validated by generating intermediate sets of validation curves with each set holding a different (but constant) learning rate while the number of estimators was varied. (Table 3). Grid search was executed to select the best combination, and in-line with Occam's razor [9] principle. From a wall clock perspective, it is expected that higher estimators will impact training and prediction times. Learning curves of both tuned models (Figure 7) show convergence of the training/validation sets. Performance metrics on the unseen testing set are listed in Table 4. Compared to Decision Tree, significant improvements in performance metric are noticed in both datasets to the extent of about **8-9%**, however the training times are almost 2 orders of magnitude higher. This trade-off will be important to consider before deciding to use it in practice.

Table 3 – Learn Rate v/s Performance at **15** Estimators

	Average Precision Score with 15 Estimators	
	Learn Rate = 0.1	**Learn Rate = 0.4**
Pima	0.67	0.70
Wisconsin	0.965	0.977

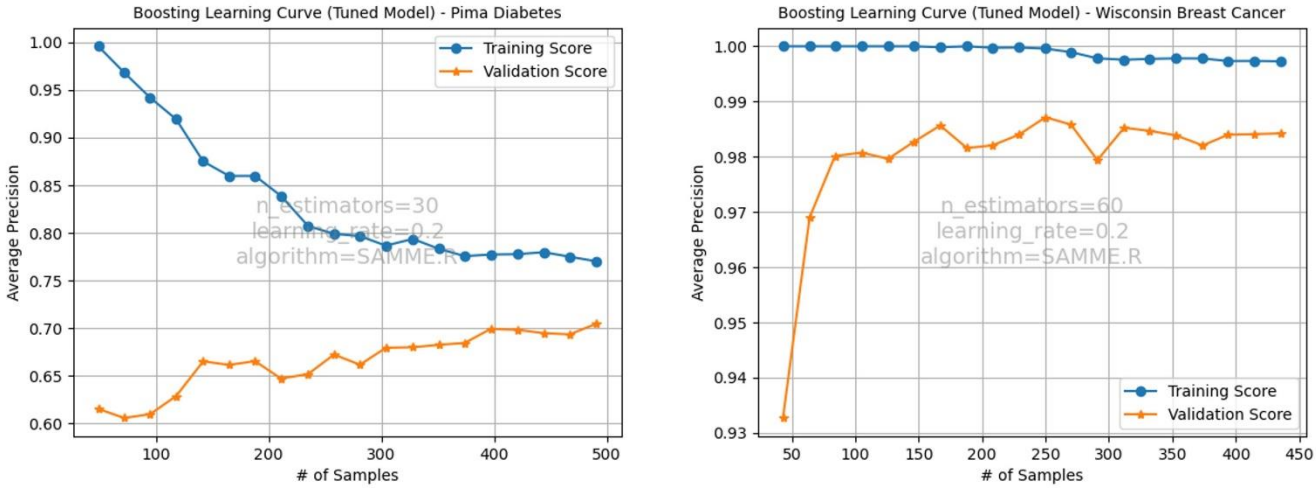


Figure 6. Boosting Learning Curves – Tuned Model

Dataset	Average Precision	Confusion Matrix		Training Time	Prediction Time
Pima	0.76	92	8	0.0726s	0.004s
		27	27		
Wisconsin	1.00	89	0	0.0994s	0.008s
		2	46		

Table 4 Boosting Performance Metrics

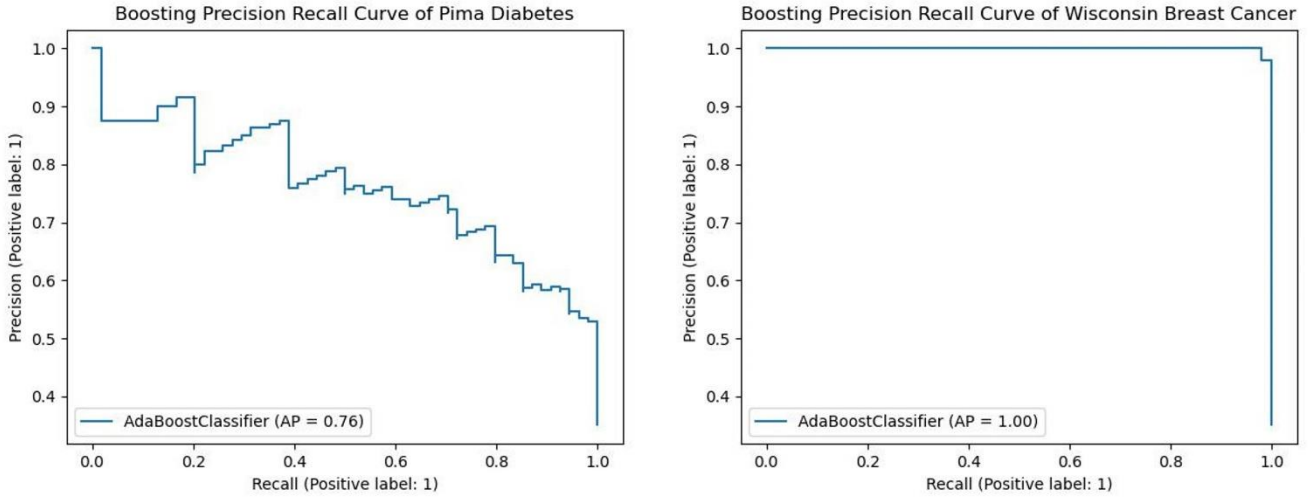


Figure 7 Boosting Precision Recall Curve

C. -Nearest Neighbors

The initial (untuned model) learning curves of the Pima dataset showed moderate bias and high variance, whereas the Wisconsin dataset showed low bias and low variance using k-fold cross validation. Therefore, it appears (for the Wisconsin set) that although the default hyperparameters appear well tuned, the learning curve (Figure 8) for the cross validation set still looks erratic for different training sizes. A hypothesis is because the default model has a low value of k (5), and therefore could be excessively influenced by a few samples, particularly if they happened to be outliers. This was confirmed by validation curve for hyperparameter "K" (Figure 9), where convergence between the training and validation-set curves was seen at higher values of K for both datasets, to almost the point of irreducible error [10]. However, the behavior with different training sizes remains to be validated. The subsequent thought

process was to attempt to lower bias without compromising variance and thereby improve performance.

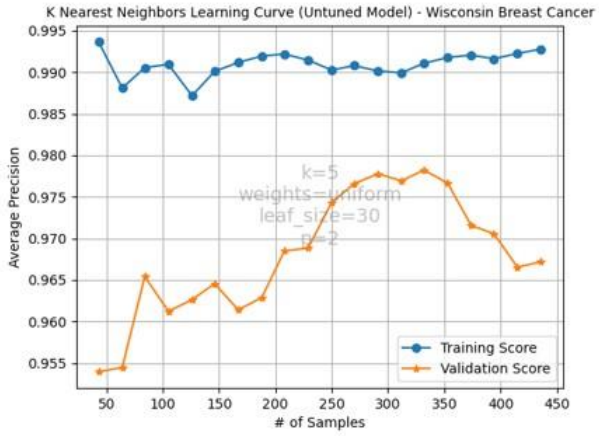


Figure 8 KNN Untuned Learning Curve Wisconsin

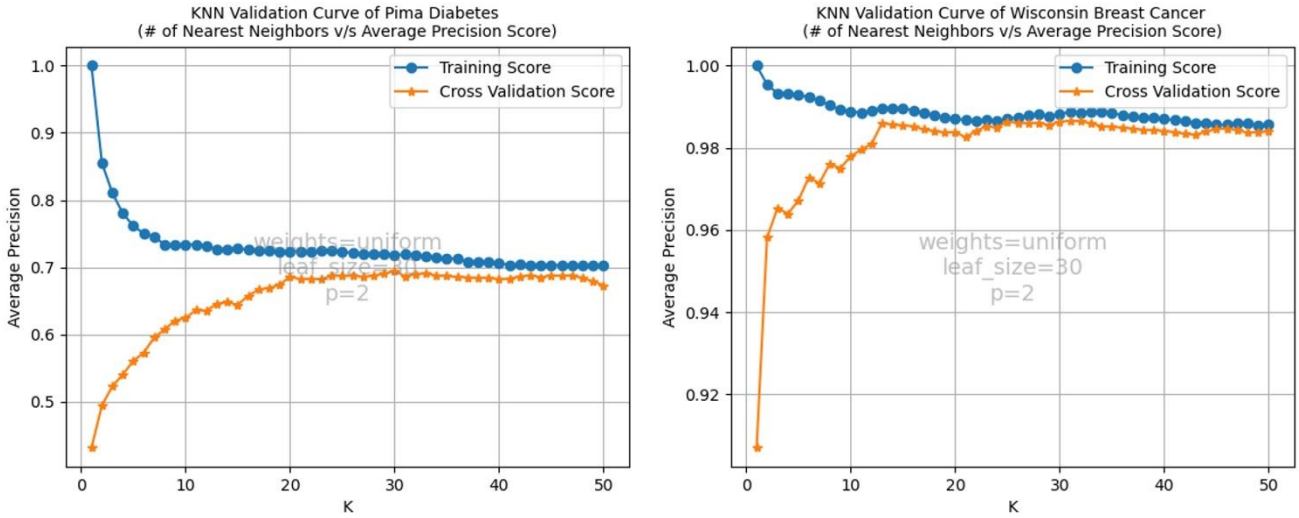


Figure 9 KNN Validation Curves for "K"

Since both datasets are imbalanced, the KNN weighting function was chosen as a hyperparameter. The intuition is that class imbalance can be especially problematic in KNN due to the majority class dominating during classification voting. Therefore, the hypothesis is that the technique of weighting samples based on distance could be beneficial [reference]. However, validation testing for both datasets showed significant increase in bias as shown in Table 5.

	UNIFORM		**WEIGHTED**	
	Train	Test	Train	Test
PIMA	0.7177	0.6946	1.0	0.7003
WISCONSIN	0.9870	0.9838	1.0	0.9867

Table 5Average Precision Scores for Weighting

Further research [11] suggested that distance weighting increases bias as seen above, therefore uniform weighting was chosen. No grid search was necessary since the values of optimal hyperparameters (K and distance) were both obtained through validation. Learning curves of the tuned models were generated (Figure 10). While the tuned model was able to generate good performance on the Wisconsin dataset, the Pima dataset was still somewhat inconsistent in performance with different training sizes. Performance metrics are shown in Table 6 and the Precision Recall curve plotted (Figure 11). The confusion matrix confirms that the model is only able to predict about **46%** of true positive opportunities. It is hypothesized therefore that KNN had trouble with the Pima dataset due to the presence of noise, outliers and invalid feature values. Wall clock times are expectedly very fast for training due to KNN's "lazy" training approach, but prediction time is slower.

D. Neural Networks

The default Neural Network (NN) models in Scikit Learn builds a 100-node single hidden layer network. The approach used here is to start with a simple network and only add complexity if that proves insufficient. Figure 12 shows the validation curves for the network's hidden layers hyperparameter, which are the culmination of an iterative

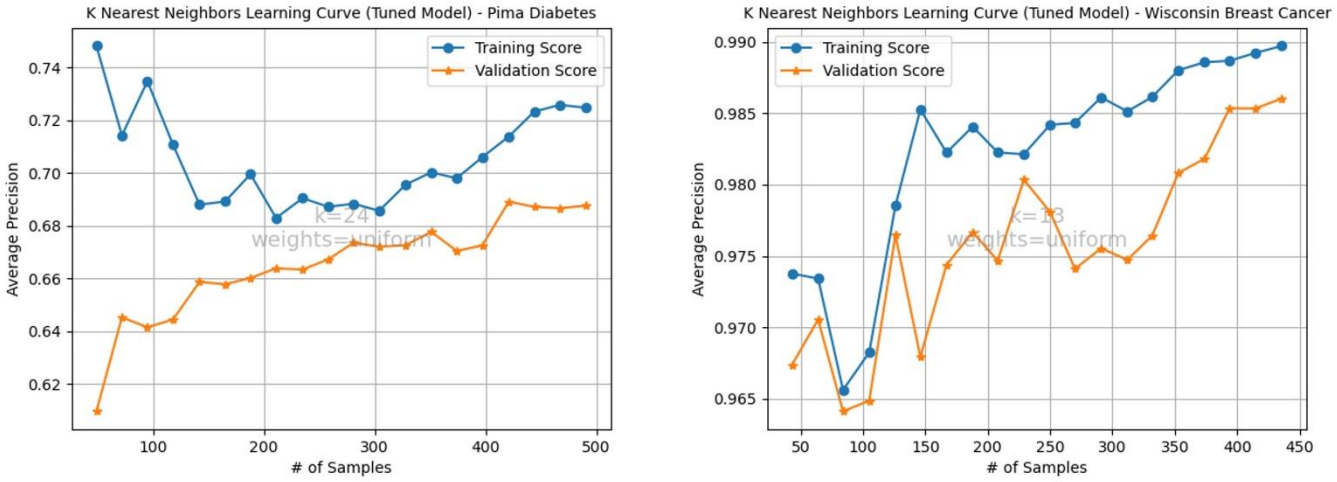


Figure 10 KNN Learning Curves of Tuned Models

Dataset	Average Precision	Confusion Matrix		Training Time	Prediction Time
Pima	0.69	89	11	0.0009s	0.0126s
		29	25		
Wisconsin	1.00	88	1	0.0008s	0.0095s
		2	46		

Table 6 KNN Performance Metrics

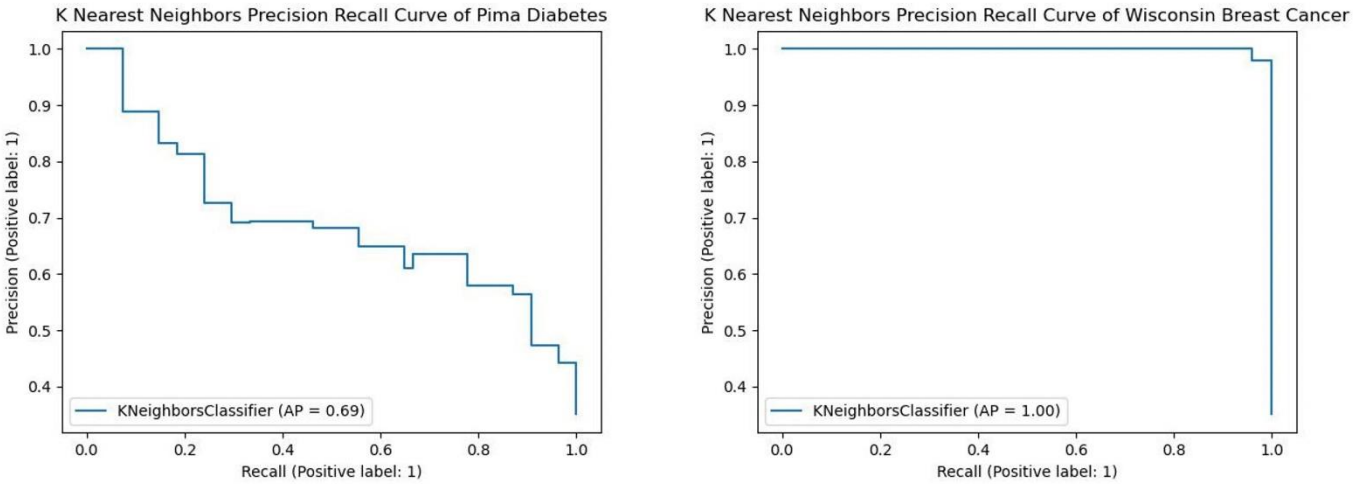


Figure 11 KNN Precision Recall Curve

process (intermediate validation curves not shown). For both datasets, with a size of only 700/800, batch size was considered an important hyperparameter since its tuning can significantly affect generalization and training time [12], as shown in Figure 13. Other hyperparameters explored were the L2 Regularization for the Pima dataset (which has previously shown a tendency to overfit), (to keep feature weights low), and the learning rate parameter for the Wisconsin dataset. Loss curves for both datasets are shown in Figure 14, which indicate optimal learning rates. Since both loss curves plateau at

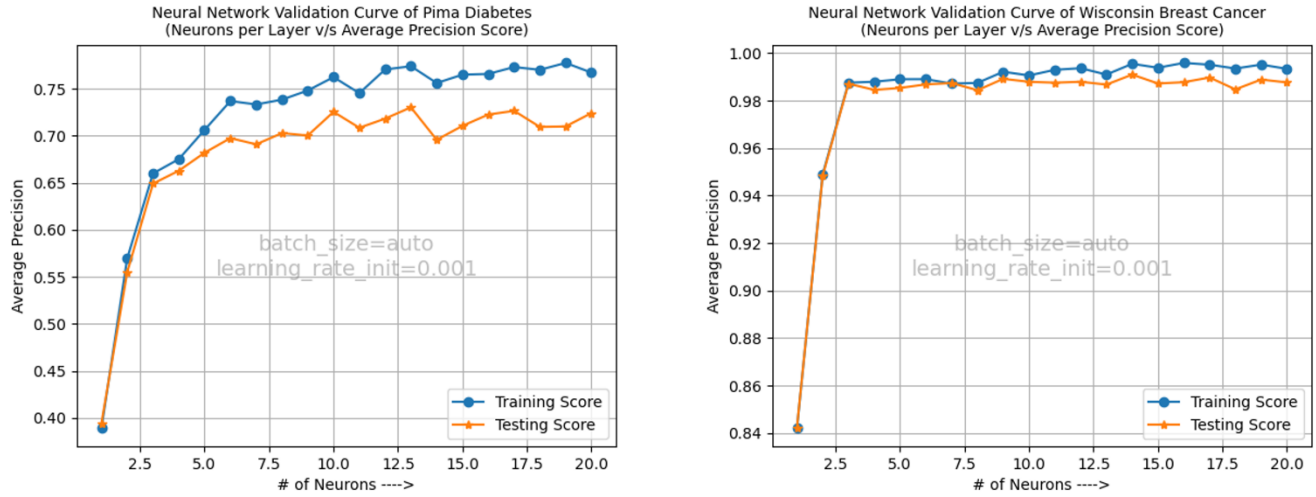


Figure 12 Neural Network Validation Curves for Hidden Layer Nodes

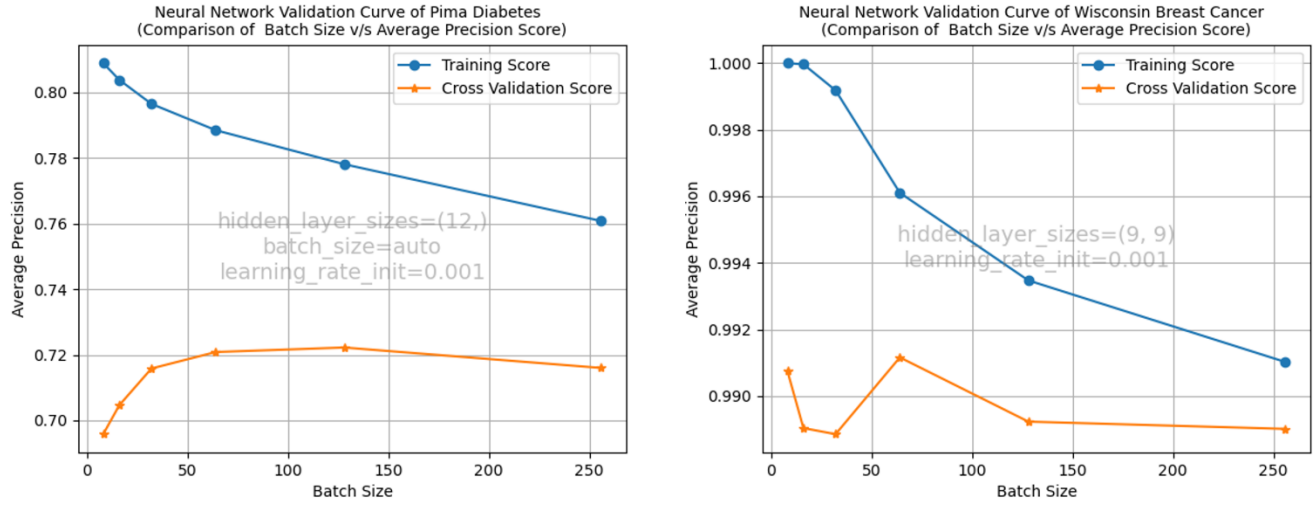


Figure 13 Neural Network Validation Curves for Batch Size

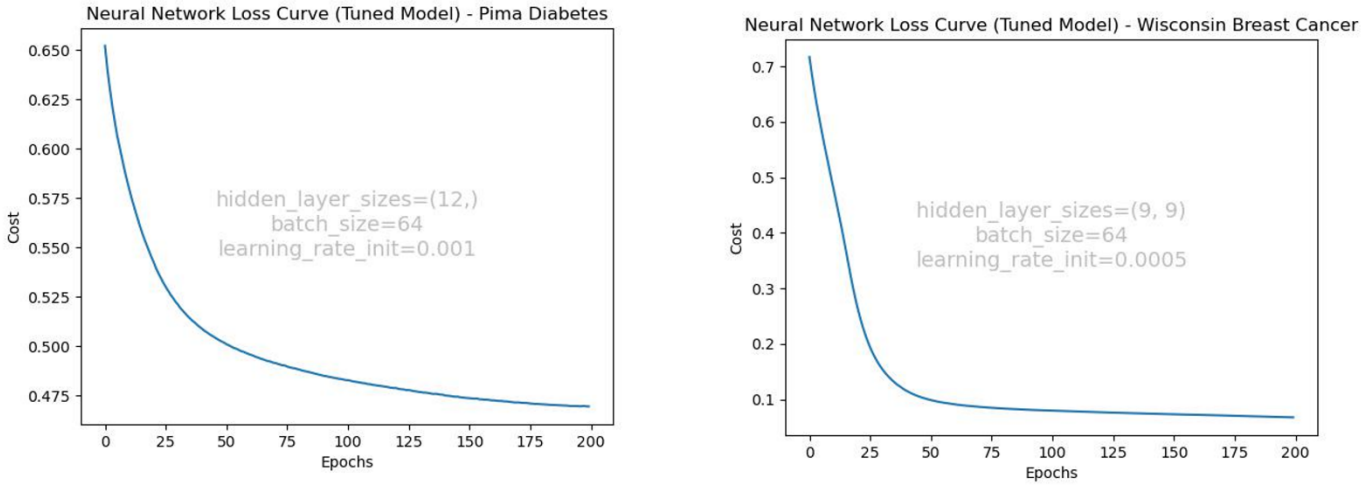


Figure 14 Neural Network Loss Curves (*training curve only)

Dataset	Average Precision	Confusion Matrix		Training Time	Prediction Time
Pima	0.72	87	13	0.7806s	0.0002s
		20	34		
Wisconsin	1.0	88	1	0.9786s	0.0002s
		0	48		

Table 7 Neural Network Performance Metrics

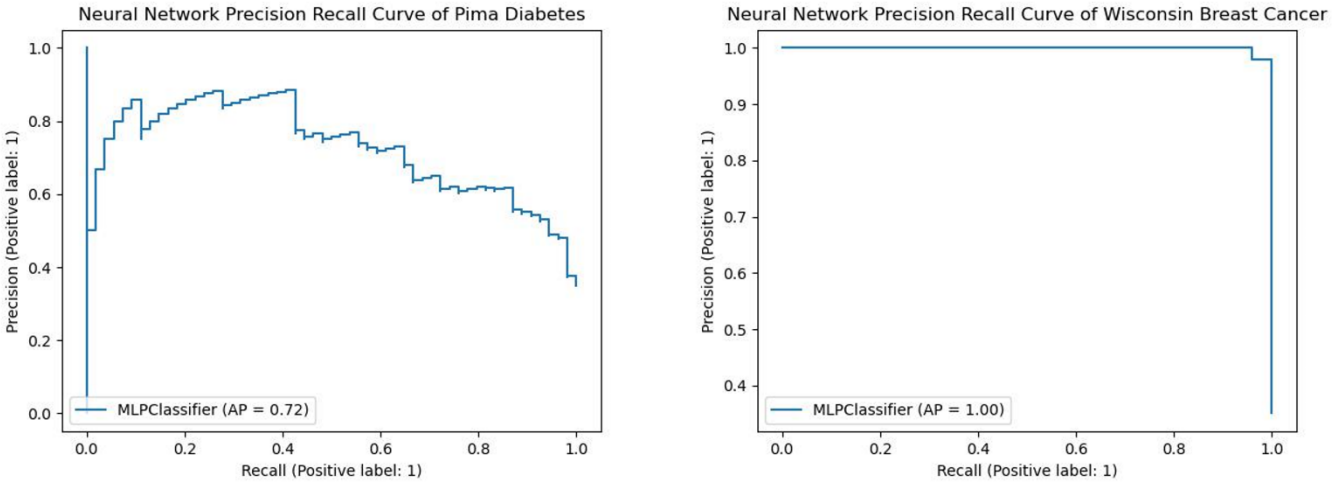


Figure 15 Neural Network Precision Recall Curves

around 175 epochs, no further adjustments to the default 200 iterations were necessary. Performance results and the confusion matrix are shown for both datasets in Table 7, and the precision recall curve in Figure 15. Even with only a single- or two-layer network with a handful of nodes, training performance is the costliest so far of all models due to the iterative forward and backward propagation. It is hypothesized that the network's inability to perform better for the Pima dataset is likely due to noise and invalid entries in the underlying data.

E. Support Vector Machines

As with other models, learning curves for both datasets in default configurations were plotted (not shown), and indicated moderate bias (Pima) and low bias (Wisconsin), and moderate (in Wisconsin) to high (in Pima) variance. Since important hyperparameters depend on kernel type, it was first selected by a simple cross validation performance as shown in Table 8.

	RBF	POLY	LINEAR	SIGMOID
PIMA	0.7296	0.6634	0.7105	0.5566
WISCONSIN	0.9728	0.9920	0.9883	0.9592

Table 8 Kernel Performance on Cross Validation Set

For Pima, hyperparameters "gamma" (which controls the radius of a sample's influence and "C" a regularization parameter that determines the penalty for misclassifications were chosen since the untuned model showed overfitting, and both play important parts in controlling it. Similarly, for Wisconsin, "C" and the polynomial "degree" were chosen since the amount of overfit was already only moderate. For Pima, C and Gamma have interplay between them - higher values for both have the effect of overfitting due to shortening decision boundaries or increasing penalty for misclassifications; but for higher gamma values, the impact of C becomes limited. So, multiple intermediate validation curves for gamma were plotted with different (but constant) C values, one of which is shown in Figure 16. For Wisconsin, validation curves for C and degree were plotted, one of which is shown (also in Figure 16). As expected, the curves confirmed that increasing values of gamma and C eventually resulted in overfitting, but also provided insight for grid search ranges. Occam's Razor principle was used to select the ideal hyperparameters and confirmed through learning curves (Figure 17), that both showed good convergence. Performance metrics are shown in Table 9 and Precision Recall Curves in Figure 18.

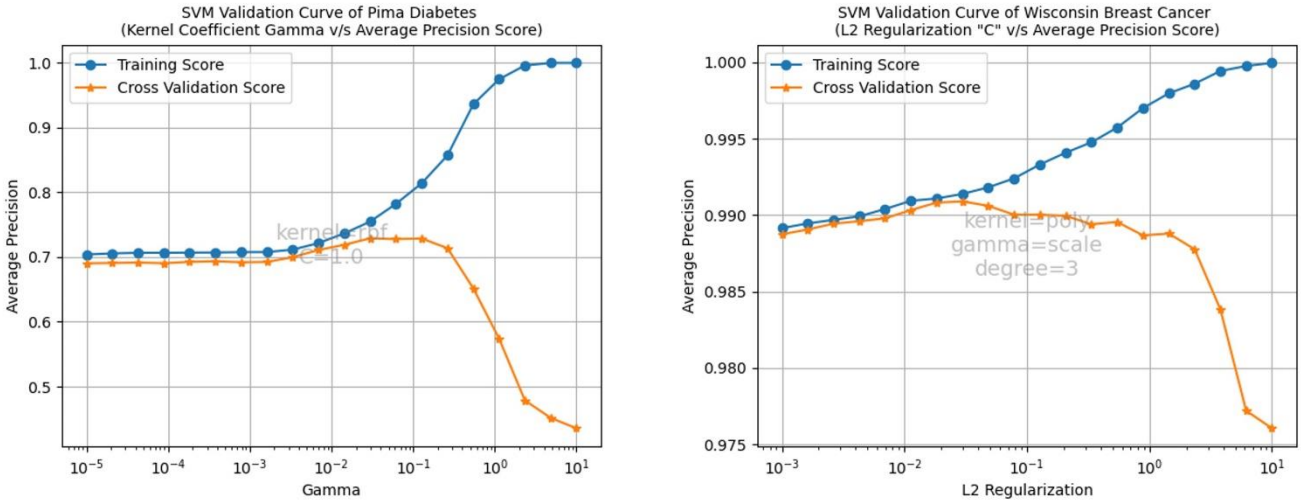


Figure 16 SVM Validation Curves for Gamma and C

.

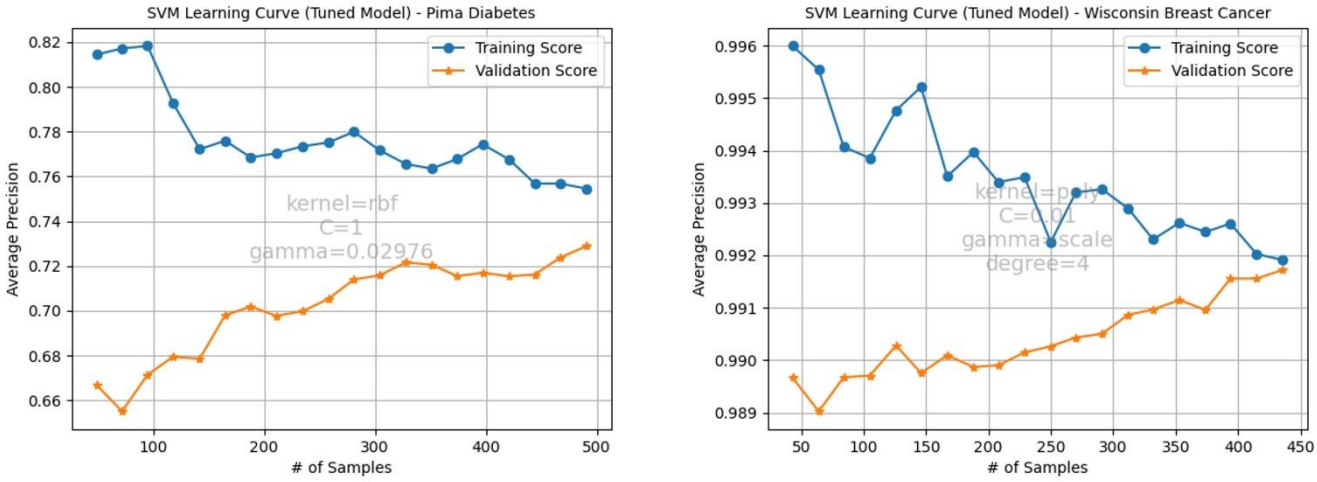
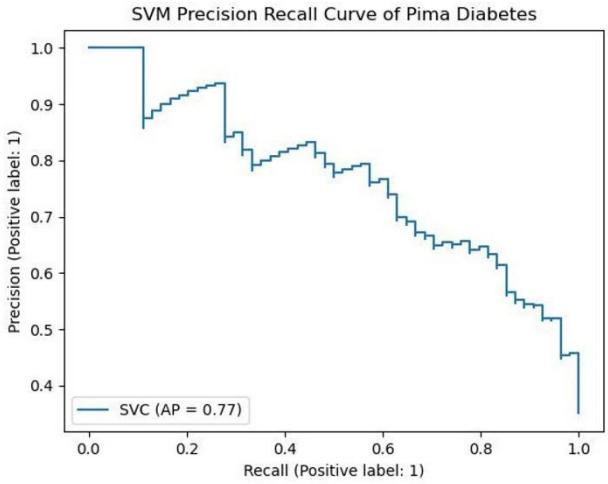


Figure 17 SVM Learning Curves of Tuned Models

Dataset	Average Precision	Confusion Matrix		Training Time	Prediction Time
Pima	0.77	90	10	0.0641s	0.0032s
		21	33		
Wisconsin	1.0	89	0	0.0223s	0.0009s
		3	45		

Table 8 SVM Performance Metrics



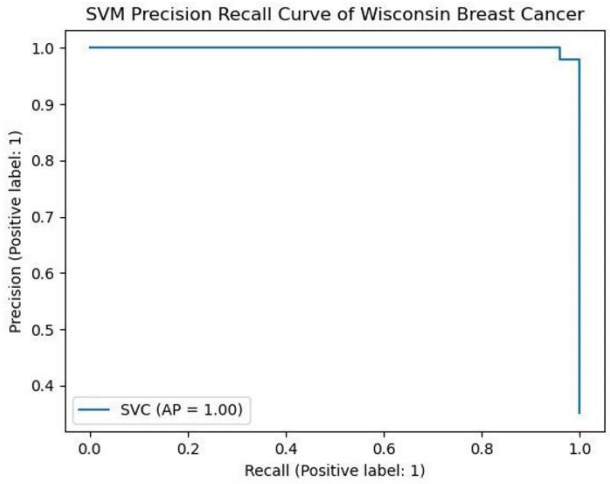


Figure 18 SVM Precision Recall Curves

F. Comparison of Algorithms and Datasets

	Pima					Wisconsin				
	Decision Tree	**Boosting**	**KNN**	**Neural Networks**	**SVM**	**Decision Tree**	**Boosting**	**KNN**	**Neural Networks**	**SVM**
Avg. Precision	0.70	0.76	0.69	0.72	0.77	0.92	1	1	1	1
Training Time (s)	0.0033	0.072	0.0009	0.7806	0.064	0.0013	0.0994	0.0008	0.9786	0.0223
Testing Time (s)	0.0001	0.004	0.012	0.9786	0.003	0.0001	0.008	0.0095	0.0002	0.0009

Table 9 Summary of Metric Performance and Wall Clock Performance

Table 9 summarizes the important metrics across algorithms and datasets. In general, all algorithms found it harder to predict the Pima dataset compared to Wisconsin from a performance metric perspective. The first hypothesis is that it's in part due to several features having invalid values (e.g., zeroes in features like Blood Pressure that's not possible). Data cleaning, which was not the focus of this assignment, should help to some extent. However, other ML practitioners have also been challenged by the dataset despite extensive data preprocessing [13], suggesting intrinsic data characteristics that are challenging to machine learning paradigms. Boosting required as many as 30 estimators, and while it was eventually able to achieve a 10% improvement, however it was at an order of magnitude or higher in training time. The imbalanced nature of the dataset requires counterbalancing techniques that were not performed in this assignment. One hypothesis for the SVM-RBF kernel performing the best is due to its ability to overcome the space complexity of KNN with the "Kernel trick" [14].

The Wisconsin dataset on the other hand scored high across most algorithms except for Decision Trees, even without data preprocessing, other than Standardization. One hypothesis is that in contrast to the Pima dataset, Wisconsin has only constrained categorical data and less noise. Further, a minimal amount of preprocessing to remove rows with question marks had been performed. The polynomial kernel was the only SVM capable of achieving a good fit for the Wisconsin dataset. Specifically, the "coef0" independent term was required to be set to a positive value during tuning. The intuition is that adding this coefficient further emphasizes the higher degree transformation between samples (of course still with the kernel trick), which would be helpful if local patterns exist and need to be recognized [15]. So, the hypothesis is that the Wisconsin dataset contains such local patterns.

Another observation was that for an imbalanced dataset in the health domain, correctly predicting the true positive cases was more critical. However, with a view to maintaining a balance with False Positives, Average Precision was chosen as the metric. But in the process, it was noticed in the confusion matrix of many models (for Pima) such as SVM, KNN, etc., that the TPR for the minority sample was significantly compromised, even if the Average Precision was decent. In hindsight, choosing "Recall" as the metric and maximizing models for it would have been a better choice.

G. Conclusion

This journey of this assignment has been an eye-opener in terms of solidifying a thorough understanding of the strengths and weaknesses of several supervised learning methods. Overfitting

and the bias/variance tradeoff were central and evident in all algorithms. Occam's Razor [8] was particularly beneficial, since in most models, the "top" result did not generalize well when applied to the unseen testing set, but without additional data, it was not possible to "go back" and adjust the model as that would violate machine learning principles. However, the lesson was learnt early in this assignment and applied to subsequent models. This was also particularly evident with Neural Networks where adding model complexity with more hidden layers resulted in significant increase in training times and compromised on generalizability. With the Nearest Neighbor model, changing the distance metric (from "uniform" to "distance") was significant, as discussed by Professor Isbell in the KNN lecture lessons. Similarly, choice of the kernel function in SVM proved to be the difference between what seemed like an impossible fit (for Wisconsin) to a solid one.

REFERENCES

[1] Smith, J.W., Everhart, J.E., Dickson, W.C., Knowler, W.C., & Johannes, R.S. (1988). Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In Proceedings of the Symposium on Computer Applications and Medical Care (pp. 261--265). IEEE Computer Society Press Kaggle: Pima Indian Diabetes Database https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database

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

[3] Dataquest: Tutorial: Learning Curves for Machine Learning in Python. https://www.dataquest.io/blog/learning-curves-machine-learning/

[4] Stathis Kamperis, Decision Trees: Gini index vs entropy https://ekamperi.github.io/machine%20learning/2021/04/13/gini-index-vs-entropy-decision-trees.html

[5] Scikit – StandardScaler Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011. https://scikit-learn.org/stable/auto_examples/preprocessing/plot_all_scaling.html#standardscaler

[6] Jeff Hale, TowardsDataScience: Scale, Standardize, or Normalize with Scikit-Learn: https://towardsdatascience.com/scale-standardize-or-normalize-with-scikit-learn-6ccc7d176a02

[7] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, Édouard Duchesnay; 12(85):2825−2830, 2011

[8] A.I. FOR ANYONE Occam's razor: https://www.aiforanyone.org/glossary/occams-razor#:~:text=In%20AI%2C%20the%20principle%20of,most%20likely%20to%20be%20correct

[9] Scikit – StandardScaler Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html#sklearn-ensemble-adaboostclassifier

[10] Dataquest: Tutorial: Learning Curves for Machine Learning in Python. https://www.dataquest.io/blog/learning-curves-machine-learning/

[11] Zhan Shi 2020 IOP Conf. Ser.: Mater. Sci. Eng. 719 012072: Improving k-Nearest Neighbors Algorithm for Imbalanced Data Classification

[12] Devansh, TowardsDataScience: How does Batch Size impact your model learning: https://medium.com/geekculture/how-does-batch-size-impact-your-model-learning-2dd34d9fb1fa

[13] Smith, J.W., Everhart, J.E., Dickson, W.C., Knowler, W.C., & Johannes, R.S. (1988). Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In Proceedings of the Symposium on Computer Applications and Medical Care (pp. 261--265). IEEE Computer Society Press.: https://www.kaggle.com/code/walidkw/pima-indians-diabetes-ml-model-selection-83

[14] Drew Wilimitis TowardsDataScience: The Kernel Trick in Support Vector Classification: https://towardsdatascience.com/the-kernel-trick-c98cdbcaeb3f

[15] [ChatGPT Response] *OpenAI GPT-2.5. Accessed Sep-2023.