CS 7641 Machine Learning Report

Asif Rehan

arehan7@gatech.edu

1. Datasets

First, the Wine Quality Dataset involves predicting the quality of white wines on a scale given chemical measures of different wine samples and labels them by quality rating of 0-10. It is a multiclass problem to predict the quality rating of the wine based on the chemical characteristics. The dataset is not balanced, and it comes with 4898 data points. There are 11 input variables. Second, Pima Indian Diabetes Dataset dataset comes with different input variable with medical condition indicators such as age, BMI, insulin etc and the Outcome column, which is 1 if the target had diabetes or 0 otherwise, making it a binary classification problem. There are 768 records with 8 variables and 1 label column.

Methodology

EDA: The two datasets were first analyzed using exploratory data analysis (EDA) techniques such as scatterplots to see outliers and correlations. Since both dataset were fairly suitable for machine learning models without much feature engineering or outlier removal, the each of the data set was split into a 70% training and 30% testing test.

Next, for each dataset, 5 classifiers were applied on the training set: Decision Tree (DT), k-Nearest Neighbors (kNN), Multi-Layer Perceptron (MLP) Neural Network, Support Vector Machine (SVM), and a Boosted Decision Tree using the first DT model.

To train each model, both training dataset was preprocessed as standard normal distribution. first a few parameters were used to see a corresponding validation curve for each dataset using a 5-fold cross validation process using f1 score which is recommended for the slightly imbalanced datasets. For Neural Network model, for a range of parameters, a Loss Learning Curve was also created to see how the model performs over the iterations.

Next, after seeing how these parameters affect the model, a hyperparameter search process was applied with stratified 5-Fold cross validation approach.

When the final tuned model was determined, the 5 models were applied on the held-out normalized test dataset for the final model comparison.

Exploratory Analysis:

Wine data: For this assignment, the quality ratings>=6 were marked as 1 and else 0 to make it a binary classification problem. After this, 3258 records (65%) become positive (quality>=6) and 1640 are negative (quality<6) (35% of data).

Pima data: This data is a bit imbalanced, there are 500 false (~65%) and 268 (~35%) true diabetes indicators. Visually no outlier was detected and all features were numerical so no feature engineering was required.

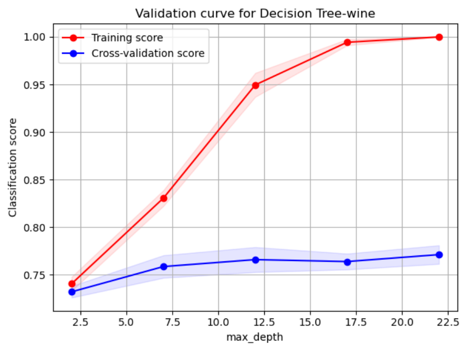
The scatterplot shows there are a few outliers handled by preprocess step with standardization.

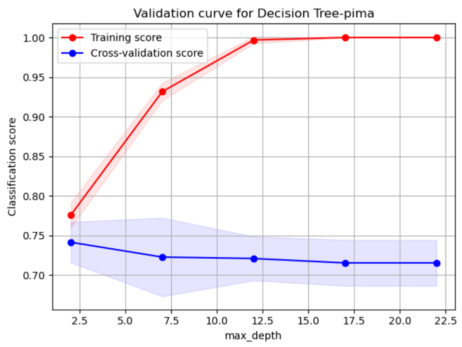
Decision Tree Training:

Firs the training data was preprocessed with Standardized by removing the mean and scaling to unit variance. For ease with the preprocessing, Scikit-Learn's Pipeline was used.

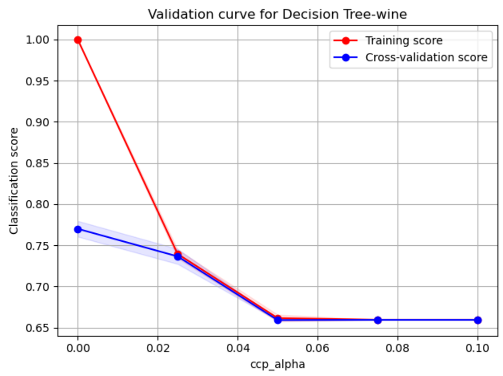
Decision Tree Validation Curves:

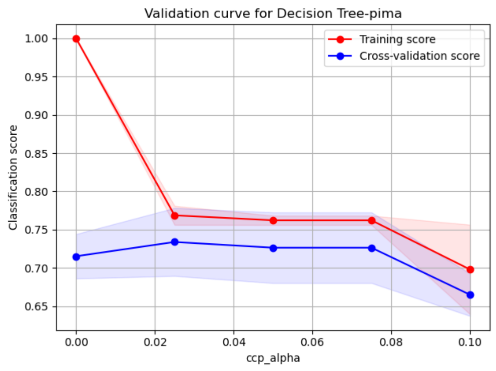
Too low max depth of the tree restricts the performance. For wine data f1 score kept improving as depth increased. But for pima data, it started to overfit with too deep tree.



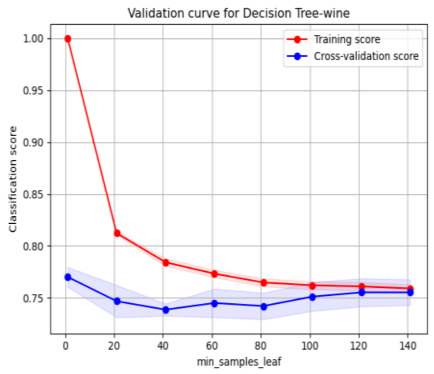


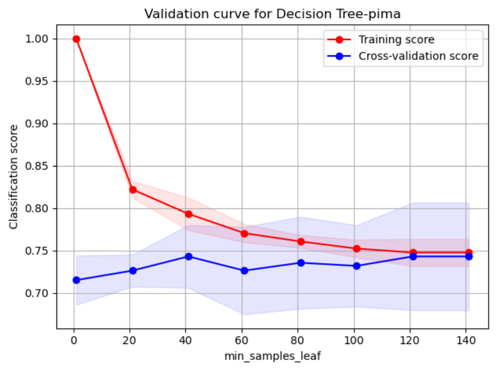
Pruning using Minimal Cost-Complexity Pruning (ccp_alpha) seemed to remove generalizability of the decision tree for both datasets. Significant pruning was not needed as the limited number of numerical features likely did not make too many branches.





With more samples required for splitting a node, the training and validation scores eventually merged above 120. This means less over-or-under-fitting.





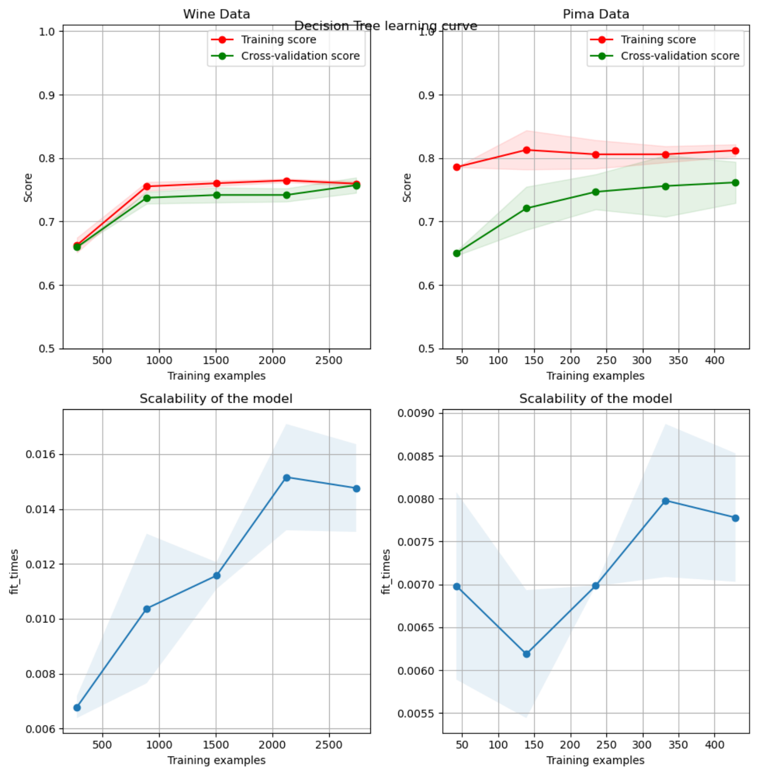
Hyperparameter Tuning:

Up next, for a range of values for pruning variable ccp_alpha, max_depth and min samples for a leaf, and min samples required for a split, we obtained a tuned model for both data. The tuned model required no pruning, smaller max depth of 4-6, and needed at least 105 samples for a leaf for wine and minimum 70 samples for a leaf node in pima data. This is consistent with the validation curve. The accuracy we 0.76 and 0.81 for wine and pima data.

Wine Data	`Tuned params: {'cfr__ccp_alpha': 0.0, 'cfr__max_depth': 4, 'cfr__min_samples_leaf': 105, 'cfr__min_samples_split': 10}` `accuracy                          0.76      3428`
Pima Data	`Tuned params: {'cfr__ccp_alpha': 0.0, 'cfr__max_depth': 6, 'cfr__min_samples_leaf': 15, 'cfr__min_samples_split': 70}` `refit train metrics=` `    accuracy                          0.81       537`

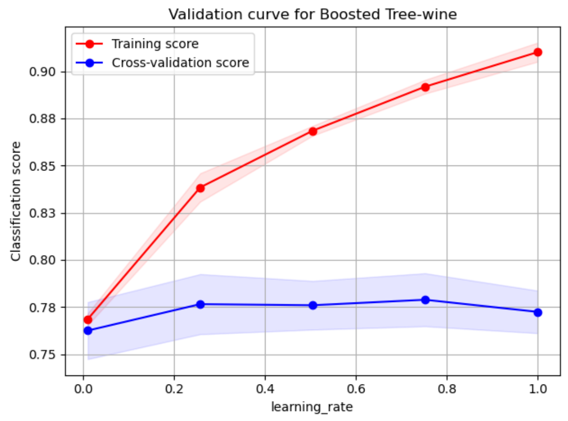
Learning Curves:

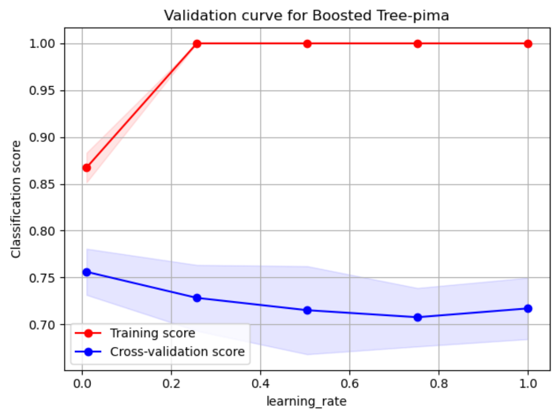
Decision Tree seems to perform better with more data. For pima, more data would be better to converge 5fold CV f1-score and that of training. The fit times generally seem to increase with more samples for wine but with a dip for pima.

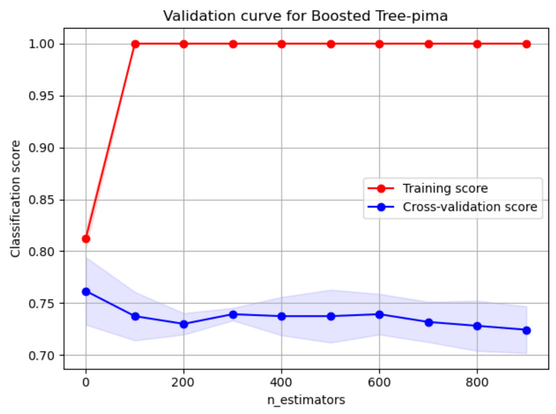


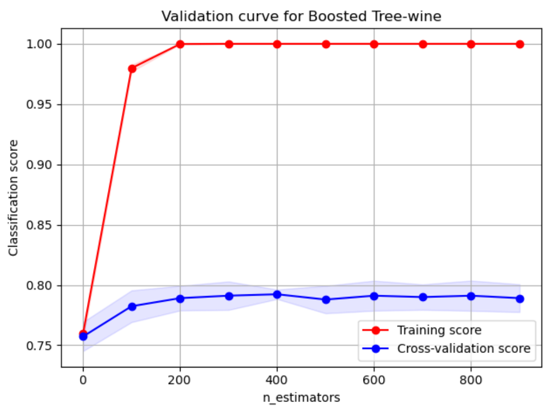
Boosted Tree Validation Curves:

Using a AdaBoost classifier, as the number of estimators increase, the models tends to overfit as expected. But with learning curve, wine data seems to perform better, but a small learning rate was better for pima data.









Hyperparameter Tuning:

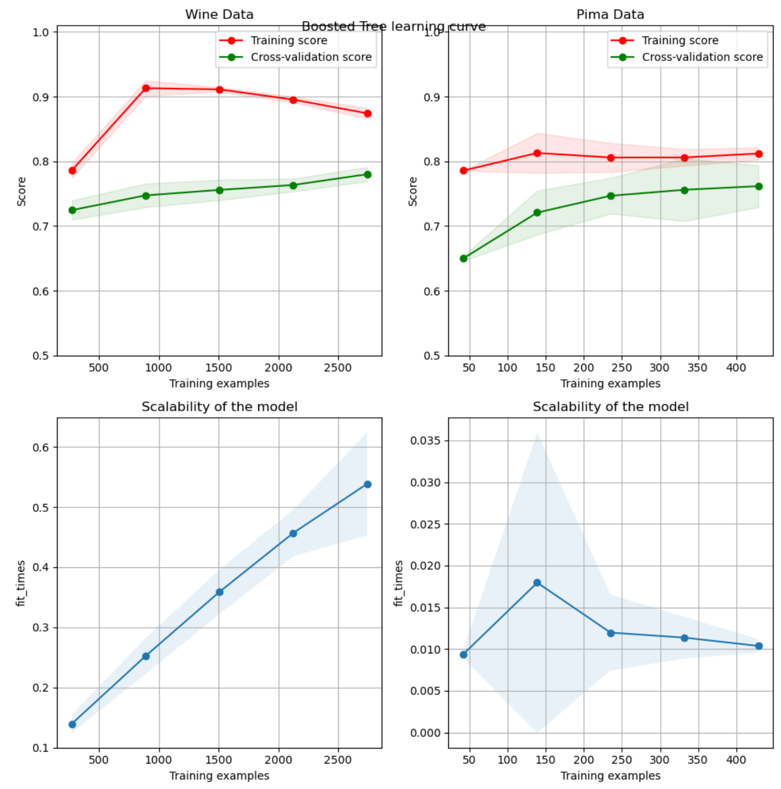
The tuned model had learning rate of 0.7525 and 41 estimators for wine data and 0.01 and 1 for pima. The decision tree did not require more than one learner for pima data as its datasize was smaller. Accuracy from precision and recall are 0.87 and 0.81, improvement over decision tree for wine data. The decision tree should have had more aggressive pruning for the decision tree to perform better for pima.

Wine Data	`Tuned params: {'cfr__learning_rate': 0.7525, 'cfr__n_estimators': 41}`
	`refit train metrics=`
	`    accuracy                        0.87      3428`

Pima Data	`Tuned params: {'cfr__learning_rate': 0.01, 'cfr__n_estimators': 1}`
	`refit train metrics=`
	`    accuracy                        0.81       537`

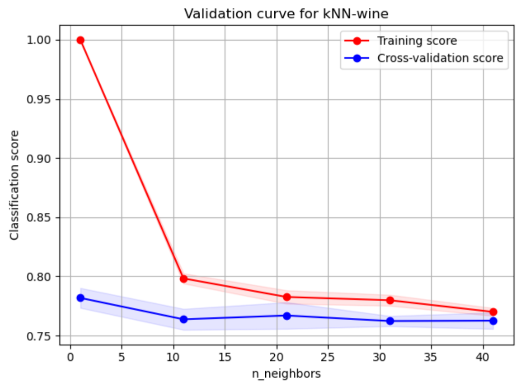
Learning Curves:

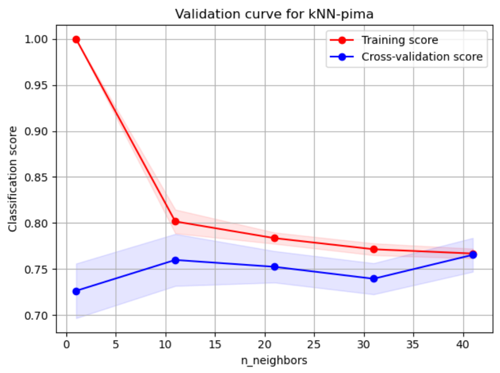
More data helped Boosting improve f1 score. The time to learn also increased significantly for wine data. Apparently more learner tree added run time. But for pima, training time did not change much after some 150 samples.



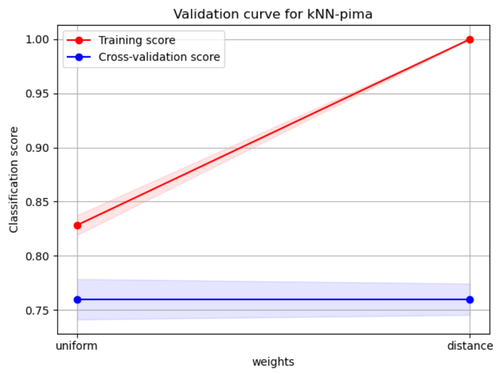
KNearest Neighbor Validation Curves:

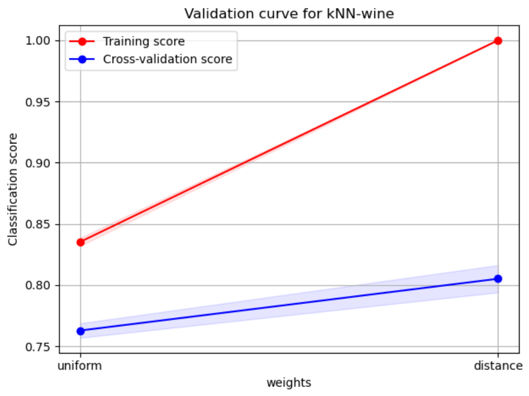
For k in kNN, higher number of neighbors tend to provide best model. Apparently the training and testing f1 scores merged at 41 nieghbors.





Among the two kinds of weights, `distance' based weight for the neighbors was better than flat uniform weight as expected. Also the power parameter for the Minkowski metric did not seem to matter much. But p=1 (Manhattan distance) was better.





KNearest Neighbor Hyperparameter Tuning:

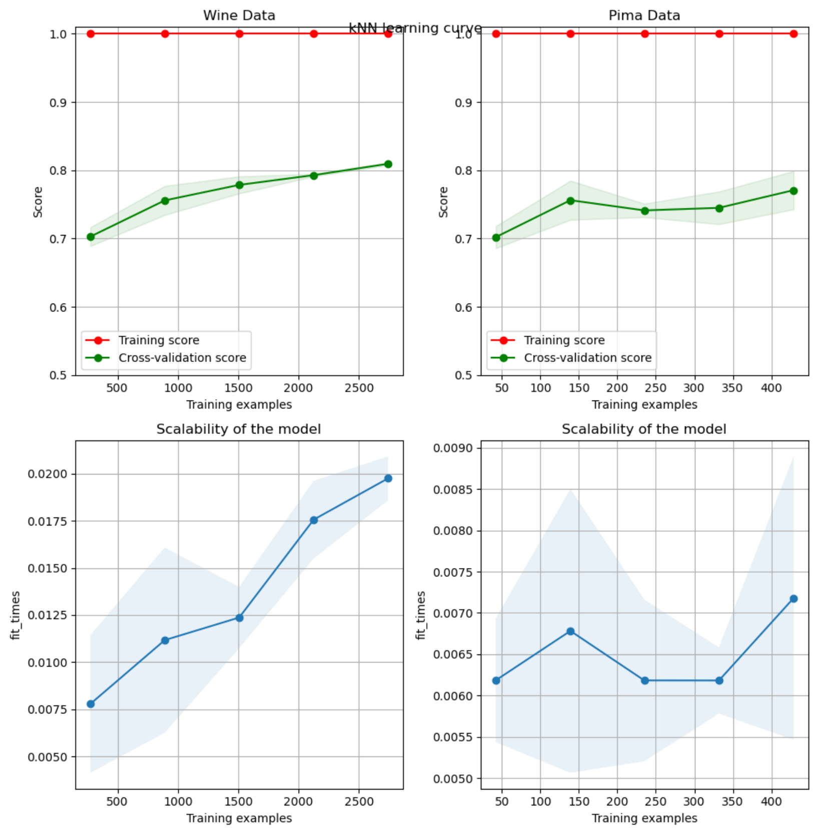
After the tuned model was refit into the training data, the accuracy was 1 which might mean high overfitting with k=4 and p=1 for wine and p=3 for pima.

Wine Data	`Tuned params: {'cfr__n_neighbors': 4, 'cfr__p': 1.0, 'cfr__weights': 'distance'}`
	`    accuracy                           1.00      3428`

Pima Data	`Tuned params: {'cfr__n_neighbors': 4, 'cfr__p': 3.0, 'cfr__weights': 'distance'}`
	`    accuracy                           1.00       537`

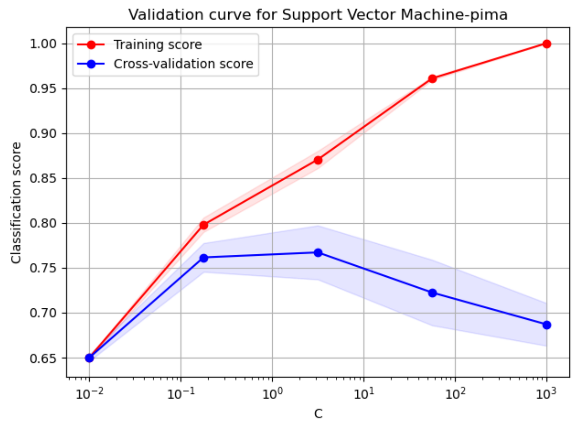
KNearest Neighbor Learning Curves:

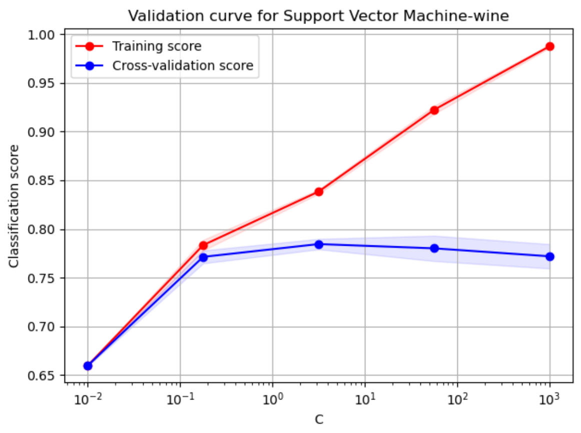
kNN is a lazy model and it performs better with more data, training time increases linearly with sample.



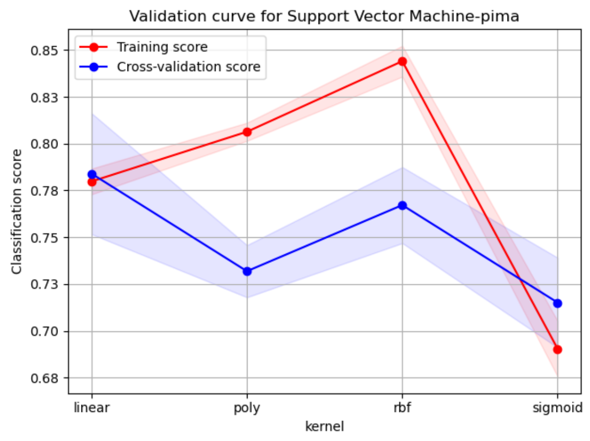
Support Vector Machine Validation Curves:

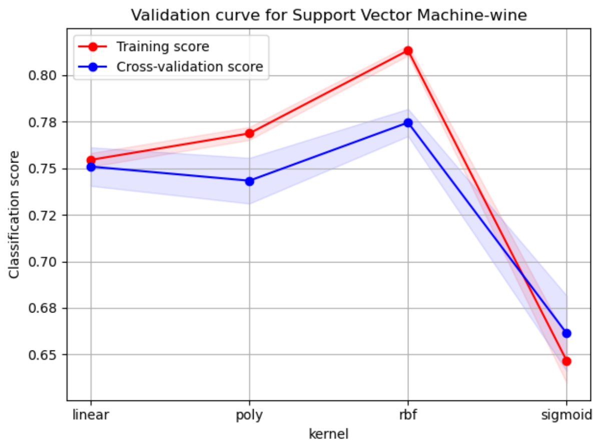
SVM validation curve shows, with higher regularization of C parameter, the model overfits. It is necessary to find the optimal C.





For different kernel options, between linear, 3rd degree polynomial, Radial Basis Function (RBF) and sigmoid, RBF is best as the data is numerical so Gaussian RBF is well suited, so did linear kernel. Sigmoid is worst as probably because of the stepping characteristics of the function.





Support Vector Machine Hyperparameter Tuning:

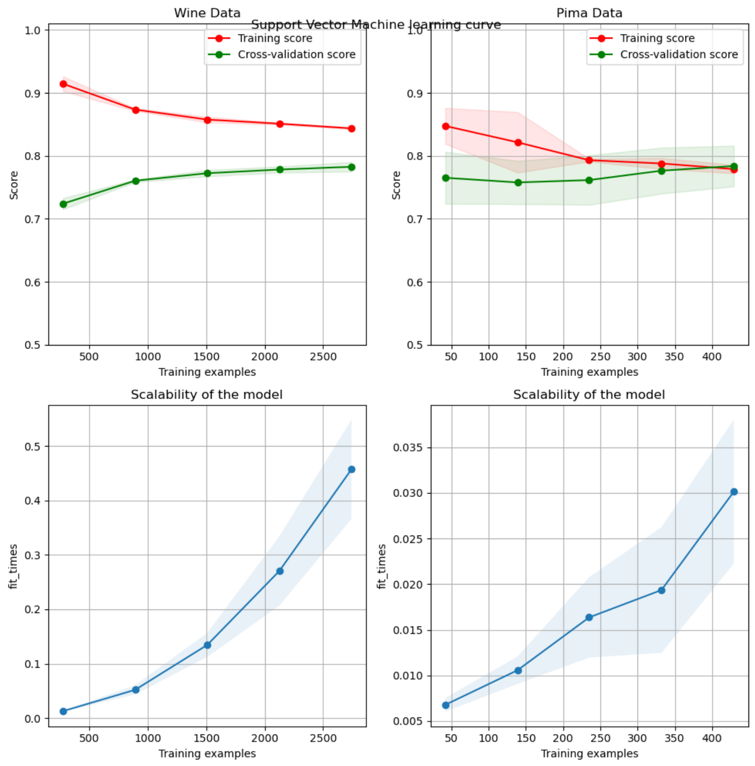
The tuned SVM model had k=3.16 for both daa, with different gamma. The kernel for wine was RBF, but for Pima, a linear kernel was found to be optimum.

Wine Data	`Tuned params: {'cfr__C': 3.1622776601683795, 'cfr__gamma': 0.1, 'cfr__kernel': 'rbf'}`
	`    accuracy                          0.85      3428`

Pima Data	`Tuned params: {'cfr__C': 3.1622776601683795, 'cfr__gamma': 1e-06, 'cfr__kernel': 'linear'}`
	`    accuracy                          0.78       537`

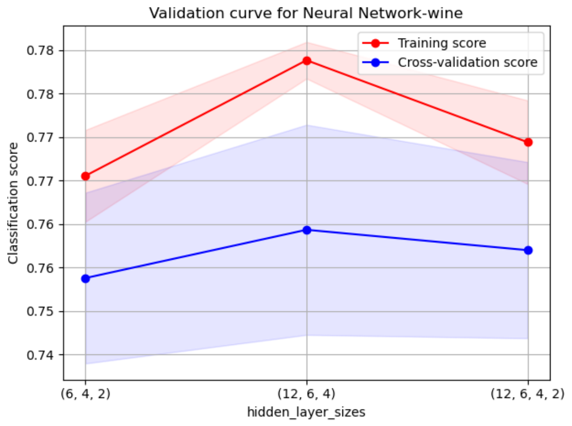
Support Vector Machine Learning Curves:

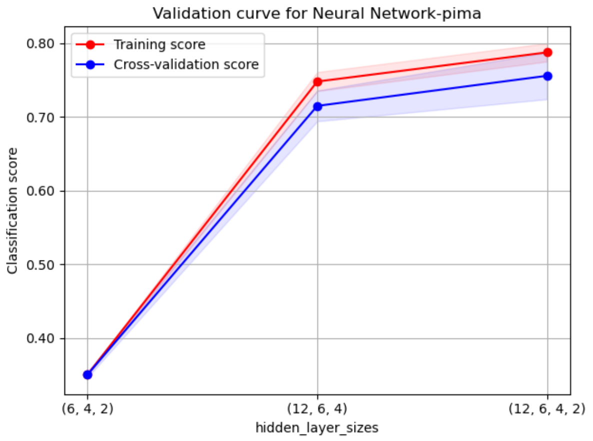
SVM tends to benefit from data as it can learn better hyper plane in a higher dimension projected by the kernel. As the training sample grows, the training time grows exponentially.



Neural Network Validation Curves:

Multilayer Perceptron was used for the 5fold CV validation. Among multiple hidden layer architectures, 6,4,2 nodes in the three hidden layers performed the best for wine, but for pima, 4 hidden layers were even better as more hidden nodes were added to capture a more complex hyperplane.





Neural Network Hyperparameter Tuning:

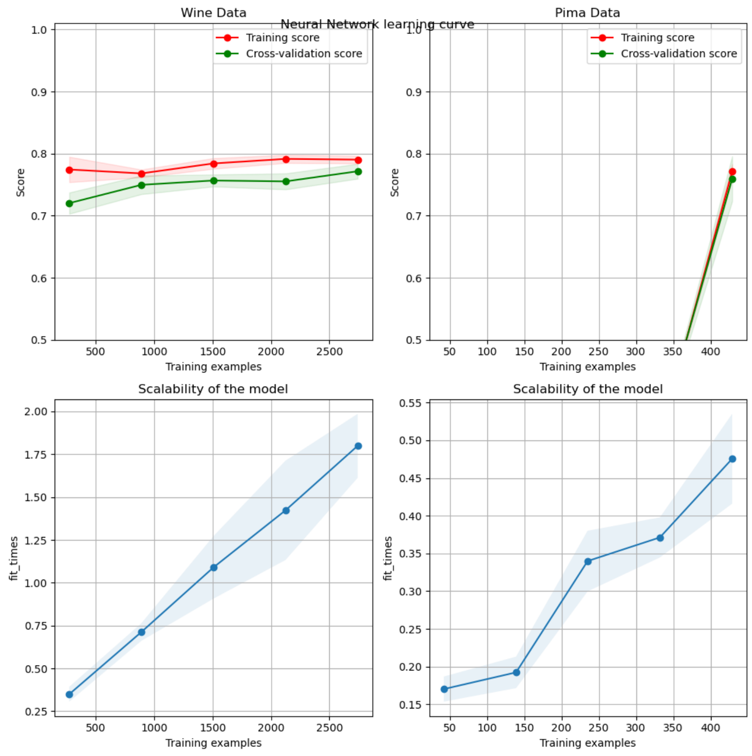
The tun models had Tan hyperbolic activation function and small alpha values, and the 3 and 4 hidden layers for wine and pima. Accuracy on the whole training data is 0.79 for both. Allowed 100 iterations max_iter=100, and a tolerance of 0.001 to enable quick convergence. A more generous parameter would improve performance with Adam solver.

Wine Data	`Tuned params: {'cfr__activation': 'tanh', 'cfr__alpha': 0.01, 'cfr__hidden_layer_sizes': (12, 6, 4)}`
	`accuracy                                   0.79       3428`

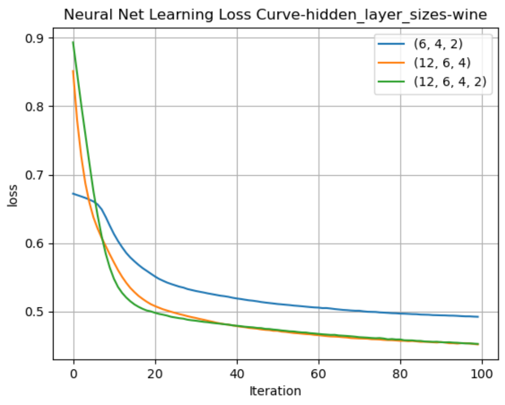
Pima Data	`Tuned params: {'cfr__activation': 'tanh', 'cfr__alpha': 0.15848931924611134, 'cfr__hidden_layer_sizes': (12, 6, 4, 2)}`
	`accuracy                                   0.79        537`

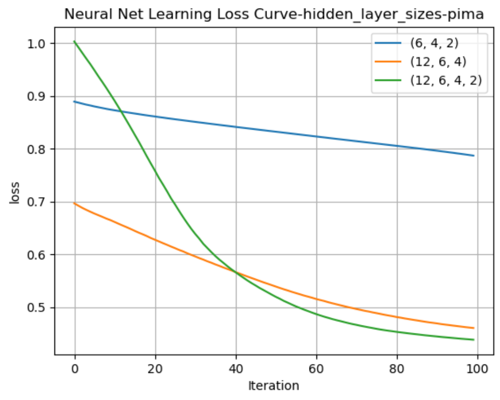
Neural Network Learning Curves:

With more data, Neural Network runtime increased linearly with data but performance remained same for wine but improved for pima. Probably the wine data was not challenging enough.



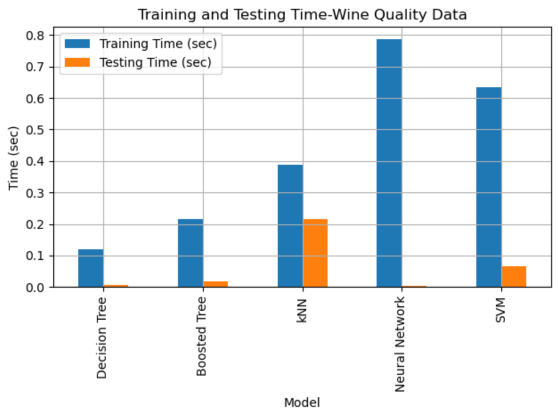
With different hidden layer architecture the loss function had different slopes over iterations. More simpler model, (6,4,2) hidden layer architecture, the slope is slow. But with more complex models with more nodes and layers, the model was complex enough to learn faster as the loss curves show.

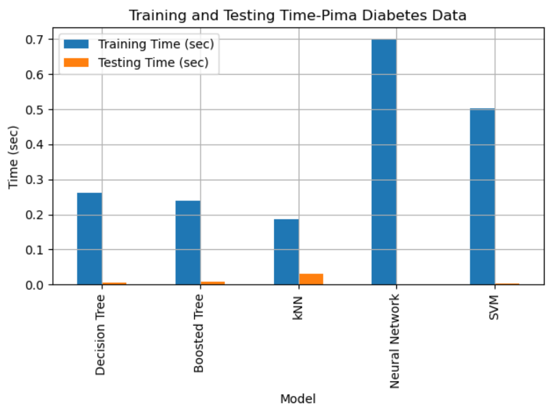




Model Comparison:

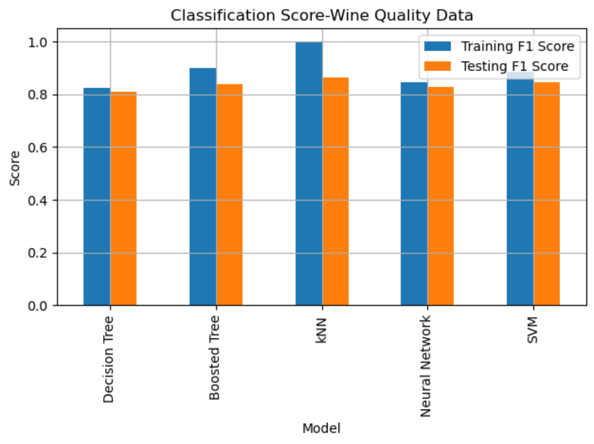
Training time took longest for Neural Network and then SVM. This is because the learning process is complex. But the prediction time on testing data is smaller for both. For kNN, wine data took longer due to more samples. SVM test in wine took longer than in pima.

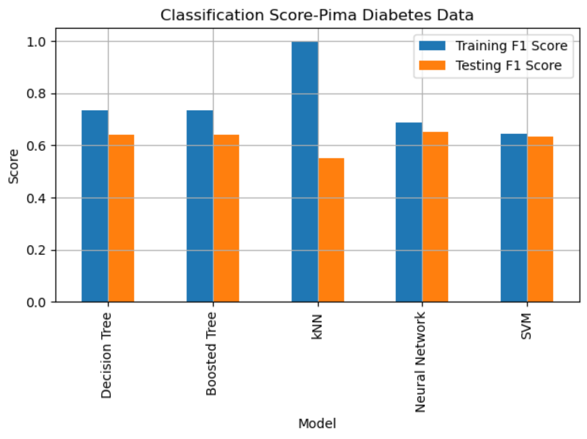




Pima data					
		Training Time (sec)	Training F1 Score	Testing Time (sec)	Testing F1 Score
Decision Tree	0	0.262163	0.736041	0.005983	0.64
Boosted Decision Tree	1	0.238697	0.736041	0.007971	0.64
kNN	2	0.185302	1	0.029923	0.551282
Neural Network	3	0.697823	0.688347	0.001995	0.650602
SVM	4	0.503441	0.646526	0.00399	0.632911
Wine data					
		Training Time (sec)	Training F1 Score	Testing Time (sec)	Testing F1 Score
Decision Tree	0	0.118957	0.823706	0.006983	0.810251
Boosted Decision Tree	1	0.216125	0.90087	0.017984	0.840436
kNN	2	0.389102	1	0.21444	0.865327
Neural Network	3	0.785926	0.844473	0.002993	0.826772
SVM	4	0.633183	0.88588	0.064861	0.84789

kNN had the perfect training f1 score but performed poorly in pima test data due to overfitting even. But it did well for wine. Domain knowledge to tweak model function would be better. Pima data was more challenging for all models than wine data.





Conclusion:

Reference:

1. Wine Quality Data: UCI dataset. http://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-white.csv
2. Pima Indian Diabetes dataset: kaggle.com: https://www.kaggle.com/uciml/pima-indians-diabetes-database?select=diabetes.csv
3. Sklearn tutorial on Learning Curve plotting: https://scikit-learn.org/stable/auto_examples/model_selection/plot_learning_curve.html
4. Scikit-Learn Decision Tree Classifier. https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
5. Scikit-Learn kNN Classifier https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html#sklearn.neighbors.KNeighborsClassifier
6. Scikit-Learn MLP Classifier https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html?highlight=mlp classifier#sklearn.neural_network.MLPClassifier
7. Scikit-Learn SVM Classifier https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html?highlight=svc#sklearn.svm.SVC
8.