About the Data

This report investigates the performance of five supervised learning algorithms over two different classification data sets. The first data set deals with credit card default and the second data set deals with rating wine.

Credit card default

The credit card default dataset comes from the UCI Machine Learning Repository[1]. The data set contains 30,000 samples. Each sample contains 23 features. The features include:

- Amount of credit given
- Gender
- Education
- Marital Status
- Age
- 6 months of payment history (6 features)
- 6 months of bill statements (6 features)
- 6 months of previous payment amount (6 features)

The output (feature Y) is a binary value stating if the customer defaulted on the next payment. Some of the data features needed cleaning, particularly the 6 features regarding payment history. The payment history features should have values -1 for paid duly, and 1,2,3,4... for the number of months the payment was delayed. However, the payment history data contained many 0's and -2's which were converted to a -1, representing paid in full. The problem we are solving is given a customer's demographics and 6 months of credit card history, will they default on their payment next month? In the following sections we will refer to the credit card default dataset as dataset 1.

Wine quality

The wine quality dataset also comes from the UCI Repository[2]. We are looking at the white wine quality dataset which contains 4898 samples each with 12 features. The features include: fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, alcohol, and quality (score between 0 and 10). The data description shows however that only 7 classification outputs exist in the dataset as all wine fell within the 3-9 range on the rating scale. All the features are normalized in the preprocessing step using the python Standard Scalar library. This is an interesting problem because we now have a multi class classification problem as compared to the 2 classes in the binary classification problem above. In the following sections we will refer to the wine quality dataset as dataset 2.

[1] https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients
[2] https://archive.ics.uci.edu/ml/datasets/wine+quality

A Note on Bias and Variance
The following experiments with different supervised learning algorithms will comment on the learning curves generated by running the experiment repeatedly with different sizes of training data. In general, a low bias is indicated by a high training score and a high bias is indicated when the model doesn't fit the training data well. Variance is indicated by the difference in the training and testing accuracy. Typically, there is a tradeoff between bias and variance, for example, if there is a low bias and the training data can be fit well, the testing data does not fit as well, and the gap creates a larger variance in the data.

A Note on Cross Validation
All the experiments included some form of cross validation as well as data randomization when calculating the train-test split. The combination of these two methods should ensure the most generic model for all the algorithms in the event the data was presorted, or clustered into groups. Given none of the data is time series, shuffling and cross validation can only help improve the models.

Decision Trees
Each data set was run through the sci-kit learn Decision Tree Classifier and tuned on 2 hyperparameters: splitting criterion and max depth. To tune the parameters, the Grid Search function was used with 5 folds of cross validation. The options for the Grid Search for both datasets were set to test the Gini and entropy splitting criterion as well as the maximum depth of the tree testing depths from 1 to 50 inclusive. Setting the max depth is a form of pre pruning the tree, to control size and complexity.

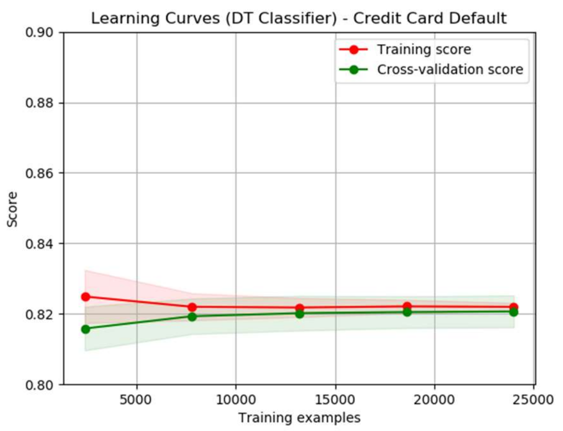
Dataset 1 performed with 81.7% accuracy. The final parameters tuned by Grid Search were Gini index for the splitting criteria and a max depth of 3. Initially all 22 features were used in the computation, however, for reasons that will described below the dataset was pared down to 6 features and maintained the same 81.7% performance with the same parameters being selected by Grid Search. The 6 features that were chosen from the dataset were the past 6 months of payment history. For a given month the value would be -1 for paid duly or 1,2,3,4, etc. for how ever many months late the payment was. These features were chosen based on the formula for credit score giving the biggest weight to payment history[3].

The max depth being 3 makes sense for this dataset since the output is binary and only 6 features were used. The learning curve for dataset 1 in Figure 1 shows convergence after about 10000 samples meaning the addition of more data will not improve accuracy.

Dataset 2 did not perform as well as dataset 1 with a 59.2% accuracy. Grid Search chose Gini as the splitting criteria for this dataset and a max depth of 27. All 11 input features were used. The learning curve for this dataset shown in Figure 1 shows a high accuracy for the training data and a low but increasing accuracy for the testing data. This shows that the addition of more data to

[3] https://www.wellsfargo.com/financial-education/credit-management/calculate-credit-score/

the model would continue to improve the model accuracy. This makes sense since there are way fewer samples than the credit card dataset and more classification outputs.



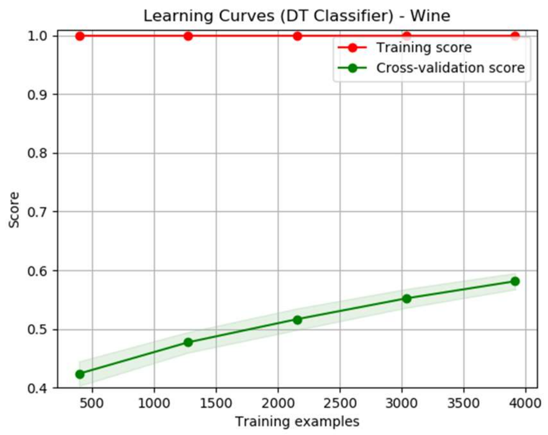


Figure 1. Learning curve plots for Decision Tree Classifiers.

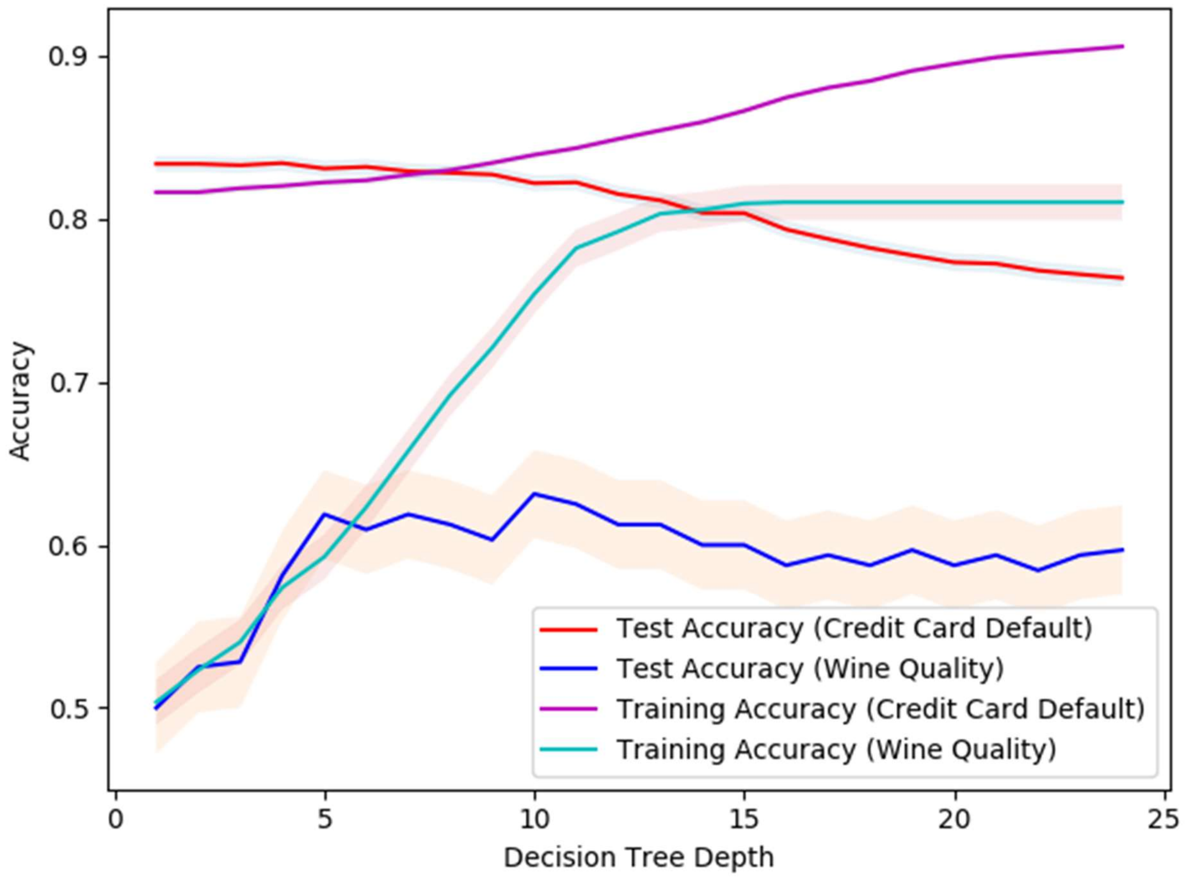


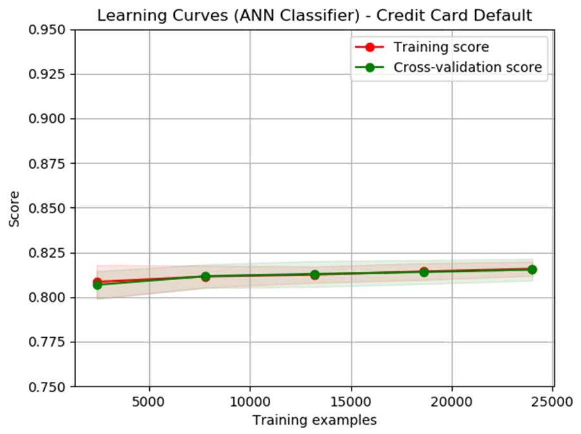
Figure 2. Comparison of accuracy vs decision tree depth for dataset 1 and 2.

Neural Networks

In this experiment each data set was run through the sci-kit learn Multi-layer Perceptron (MLP) classifier and tuned on the alpha and momentum hyperparameters. The MLP classifier used the stochastic gradient descent solver along with 1 hidden layer containing 10 nodes. The stochastic gradient descent is an approximation function for gradient descent optimization, so rather than calculating the entire gradient it calculates an estimate based on a subset of the data. The tradeoff is a slightly lower convergence rate for faster runtime. This was necessary for this experiment in order to perform an exhaustive Grid Search over many hyperparameter test values. 10 nodes were chosen based on the number of features for both datasets being less than 10 and only 1 hidden layer was used to ensure simplicity. From my research, it is rare than more than a few hidden layers are necessary and can cause overfitting. A few experiments were run with 2-3 hidden layers and there was not an improvement, so it made sense to do the hyperparameter tuning with 1 hidden layer. Again, to tune the parameters, the Grid Search function was used with 5 folds of cross validation. The options for the Grid Search for both datasets were alpha and momentum with alpha having a range of [0.0001, 0.5] and momentum having a range of 0.1 to 1.0. Momentum affects the updating of the weights by adding a term that is found by multiplying the previous weight by a constant in the range [0,1). Alpha is an L2 regularization term which adds $\frac{1}{2}\alpha * w^2$ to the error function for each weight in the network. This term causes the weights to decay linearly and encourages more of the inputs to be factored in. A larger alpha will cause the weights to decay faster

Dataset 1 performed with 81.4% accuracy. The final parameters tuned by Grid Search were 0.235 for alpha and 0.9 for momentum. The learning curve dataset 1 in Figure 3 shows immediate convergence for the training and testing data and a slight increase in accuracy as the number of samples used in the model increases. This indicates a low variance since the training and testing sets converge but a high bias since the training data does not achieve a high accuracy. More data would not improve the accuracy of the model in this case. The higher value for alpha means the parameters were considered more equally which makes sense since we are looking at 6 months of payment history.

Dataset 2 performed worse in ANN than any other algorithm with an accuracy of 53.9%. The final parameters tuned by Grid Search were 0.0001 for alpha and 0.9 for momentum. All 11 input features were used. The learning curve for this dataset shown in Figure 3 shows a decreasing variance as more sample are added and a high bias throughout. Unlike the decision tree, the training data performs poorly for the ANN. More data will not improve the accuracy of the model for this dataset either. This model selected the lowest value for alpha which signifies that a few parameters were heavily weighted, and the other parameters were less relevant.



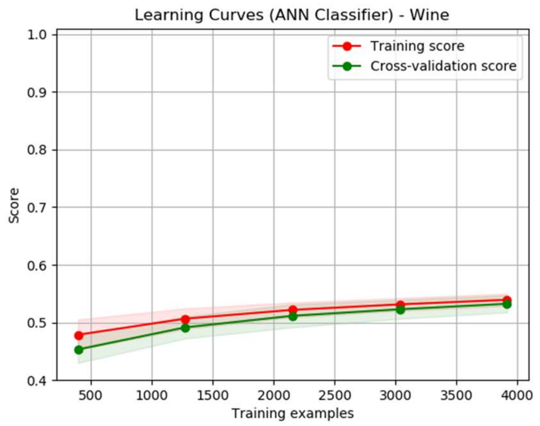


Figure 3. Learning curve plots for Artificial Neural Networks.

Boosting

Each data set was run through the sci-kit learn Adaboost Classifier containing Decision Trees tuned from the previous experiment. The decision trees in this ensemble learner are of depth 3 and used the Gini splitting criterion. The hyperparameters that are used and tuned are the number of estimators and the learning rate. An exhaustive Grid Search was used to tune the hyperparameters. In theory, the boosting algorithm should outperform the classic decision tree algorithm, but the learning rate and number of estimators could influence the accuracy and complexity of the model.

The best accuracy score for dataset 1 was 81.7% with a learning rate of XX and XX estimators. The learning curve in Figure 4 for this dataset shows an increasing bias and a decreasing variance as the number of samples increases. The training and testing accuracy are nearly converged after 24,000 training examples. There is room for a slight increase in accuracy with more training examples. Boosting in this dataset had the same performance as the non-boosted decision tree, but the learning curve trajectory shows that with more samples the Boosting algorithm would outperform the classic Decision Tree since the classic Decision Tree's learning curve converged between 15,000 and 20,000 samples.

The best accuracy score for dataset 2 was 51.0% with a learning rate of XX and XX estimators. The learning curve in Figure 4 for this dataset shows an increasing bias and a decreasing variance as the number of samples increases, however, the cross validated score stops increasing at 1500 samples. The model would not improve with any more data. The Boosting algorithm heavily underperformed the classic Decision Tree. This is due to the multi-class nature of dataset 2. In the classic Decision Tree, we were able to make a tree of much greater depth, but in the Boosting classifier, the trees were limited to a depth of 3. Perhaps the performance could be increased with less estimators of increased depth.

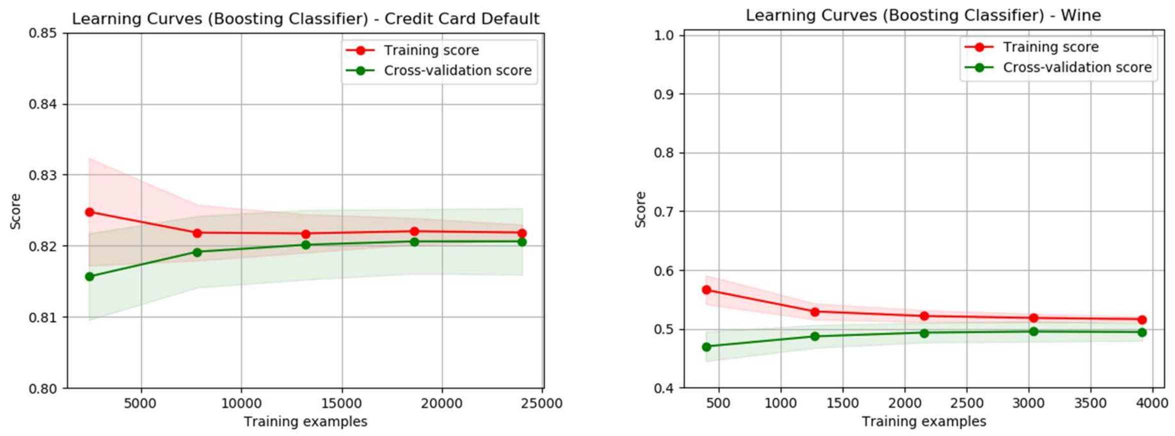


Figure 4. Learning curve plots for Boosting algorithm.

Support Vector Machines (SVM)

Each data set was run through the sci-kit learn Support Vector Machine Classifier and tuned on the kernel function and the kernel coefficient, gamma, hyperparameters. Cross validation was used for both datasets. Some tradeoffs had to be made when running this experiment due to the runtime complexity of SVM. Only 50% of the data was used for dataset 1, which was pulled from a randomized subset of the data using the train test split sci-kit learn module. Each dataset was run through 2 folds of cross validation to reduce experiment runtime. Lastly, instead of an exhaustive Grid Search for parameter tuning, the Randomized Search was used. Initially when running an exhaustive search, the experiment would not complete within a 12-hour window. As Figure 8 shows, the training time for dataset 1 was over 100 seconds and thus the need to reduce the amount of data, features, and cross validation splits used. The exponential runtime of the SVM is due to the kernel functions needing to repeatedly calculate the distance functions between data points.

The best accuracy score for dataset 1 was 81.9% using the Radial Basis Function (RBF) kernel function with a gamma of 0.102. The learning curve in Figure 5 for this dataset shows a high bias and low variance. The SVM took about 6000 samples to converge on the accuracy between the training and testing data. This shows that using a randomized subset of the dataset did not affect the accuracy of the model.

The best accuracy score for dataset 2 was 59.8% also using the RBF kernel function with a gamma of 0.826. The learning curve in Figure 5 for this dataset shows an increasing bias and decreasing variance as more samples are added. The training and testing accuracy did not converge as the number of samples increased, so there is room for improvement in accuracy as more samples are added, however, given the trajectory of the learning curves the model will probably converge around the 75% accuracy range.

Both datasets favored the RBF kernel function over a polynomial and sigmoid kernel function. Linear data would favor a polynomial function, so due to the non-linearity of both datasets the RBF kernel function has the best performance.

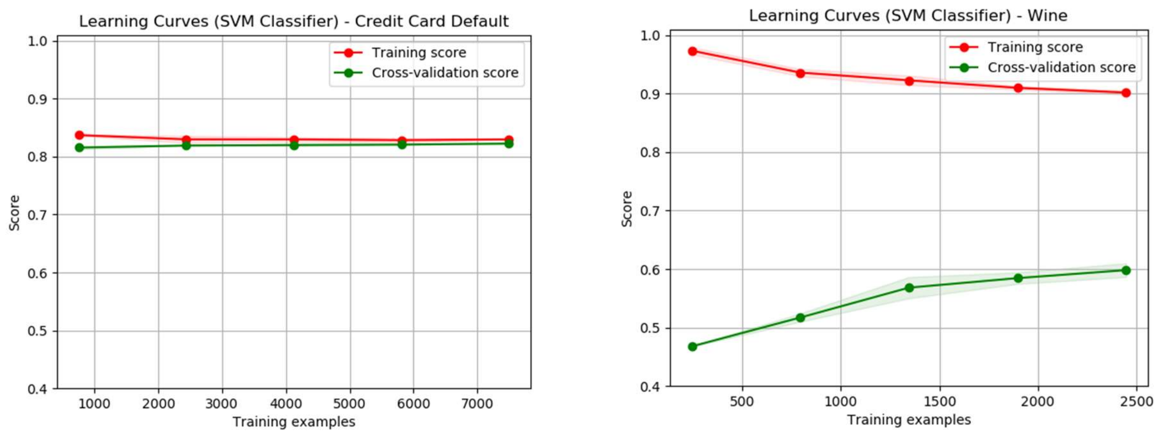


Figure 5. Learning curve plot for Support Vector Machines.

Kernel	Dataset	Mean Train Time (s)	Max Accuracy Score
RBF	Credit Card Default	0.739	0.819
	Wine Quality	0.453	0.596
Polynomial	Credit Card Default	253.842	0.808
	Wine Quality	5.184	0.525
Sigmoid	Credit Card Default	0.998	0.778
	Wine Quality	0.280	0.501

Figure 5.1. Comparison of SVM kernel functions

k-Nearest Neighbors

Each data set was run through the sci-kit learn K Neighbors classifier and tuned on the value of k. Cross validation was used for both datasets in the learning curve plots in Figure 7, but not for the plots in Figure 6 where the algorithm was tuned. We can see based on the learning curves in Figure 7 that the accuracy of the testing model did not vary when cross validated with the "best" value of k. The results for both datasets are shown in Figure 6 for values of k in the range [1, 25].

The score for dataset 1 increased generally with the value of k with the best accuracy of 83.1% with k = 23. This experiment, again, only considered the 6 features relating to 6 months of payment history. The high value of k indicates that the features were all equally relevant since they are all normalized. The learning curve plot in Figure 7 shows a low variance since the training and testing sets converge early but a high bias since the training data does not achieve a high accuracy. More data would not have achieved a higher accuracy, and the model would have the same accuracy with only 7500 samples which is 25% of the total amount of samples. Thus, this model was overly complex. This experiment showed the highest accuracy over all the other models.

The score for dataset 2 was best for k=1 with a 62.1% accuracy score. Figure 6 shows a drop off in performance for this dataset beyond k=1. This shows that 1 feature was strongly favored in comparison to the other features. All the input features were normalized so, when more neighbors were considered this will ultimately include more features from the feature set. The decrease in accuracy as k increases shows the lack of correlation of the output feature with most of the input features. Looking at the learning curve shows a high variance and a low bias. Again, this model would have been improved with the addition of more training samples.

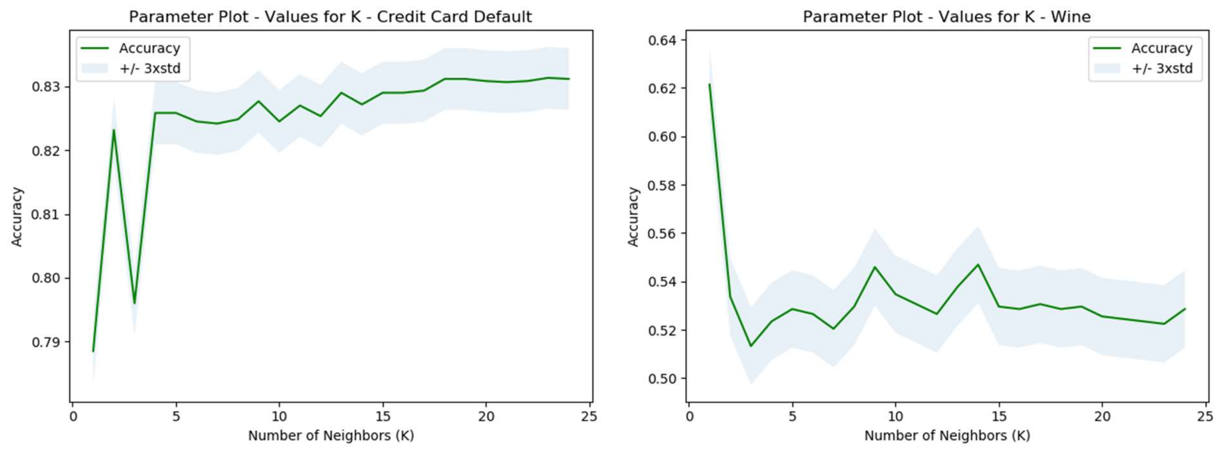


Figure 6. Value of k vs test dataset performance in KNN.

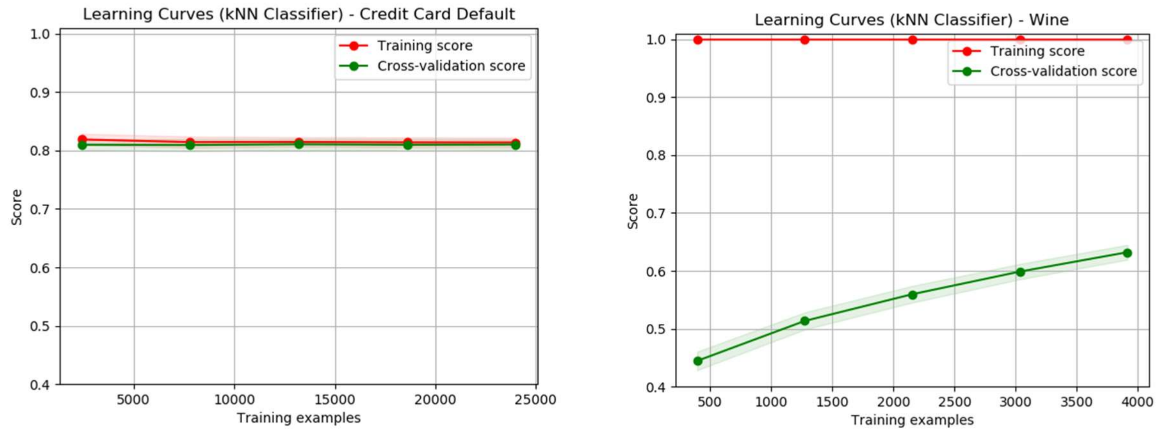


Figure 7. Learning curve plot for k-Nearest Neighbors.

Performance Comparison

Figure 8 shows an overall performance comparison of the different algorithms in terms of training time, query time, and accuracy score. The classic Decision Tree and kNN algorithms performed the best overall when considering the accuracy score and model complexity. Both of those algorithms had relatively simple models that could be trained and tested quickly, with the Decision Tree being the quickest to query. Although kNN is a lazy learner, the training still took longer for the larger dataset than the Decision Tree. The SVM performs best on a smaller dataset and it shows the best performance overall on the wine quality dataset in terms of accuracy score while still maintaining reasonable testing and training times. SVM did not perform well overall on the larger dataset with an exponential increase in training time. Still being an eager learner, SVM can perform queries quickly once a model has been created.

Algorithm	**Dataset**	**Train Time (sec)**	**Query Time (sec)**	**Accuracy Score**
Decision Tree	Credit Card Default	0.013	0.002	0.818
	Wine Quality	0.057	0.001	0.592
ANN	Credit Card Default	4.072	0.004	0.815
	Wine Quality	2.587	0.002	0.540
Boosting	Credit Card Default	2.860	0.242	0.817
	Wine Quality	1.390	0.086	0.500
SVM	Credit Card Default	102.058	0.439	0.819
	Wine Quality	1.085	0.168	0.598
k-NN	Credit Card Default	0.415	0.979	0.831
	Wine Quality	0.002	0.079	0.529

Figure 8. Comparison of training time, query time, and score for each model.