SUPERVISED LEARNING

Dixon K. Domfeh

GTID: 903658462

1.0 Introduction

This report is on the implementation and performance evaluation of five supervised machine learning algorithms – Decision Trees (DT), Adaptive Boosted Decision Trees (DT AdaBoost), Artificial Neural Networks (ANN), Support Vector Machines (SVM), and k-Nearest Neighbors (k-NN). We assess the performance of five algorithms on two different data sets to solve two classification problems. The first dataset is a binary classification problem on customer churn. The second solution tackles a multi-class classification problem using time series data of stock prices. Specifically, we use the S&P 500 Index Exchange Traded Funds (ETF), which tracks the stock performance of the top 500 companies in the United States. These two datasets are chosen for several reasons, which will be discussed in the upcoming sections of this report.

The results reveal that the DT AdaBoost and NN learners are the top performers on the Churn data set. NN learners emerge as the topmost learner on the ETF data set. The remainder of this report is as follows. We describe the data sets used, their relevance, and the feature engineering steps used to generate attributes (in the case of the time series data set) in Section 2. We analyze the different algorithms used on the two data sets to evaluate their performance in Section 3. Finally, we rank them across several performance metrics in Section 4. All plots are assessed based on accuracy score for ease of comparison across both data sets.

2.0 Data Description

Our first data set is a customer churn data set obtained from Kaggle. The original task on Kaggle was to use the data to predict customer behavior and develop focused customer retention programs. For this report, we will stick with the prediction part of the task. The churn data set contains information on a company that provides home phones and Internet services to 7043 customers in California. The target label—Churn—which is binary, indicates whether a customer left or stayed with the company. Other import information included for each customer, such as demographics, account information, services received, and demographics. Figure 1(a) reports the Cramer's V as a measure of the correlation between the features in the data. The churn data set is fascinating due to the imbalanced nature of the target label. It consists of about 73% YES outcomes and 27% NO outcomes. Predicting and learning the underlying dynamics that cause a customer to leave (NO in this case) can be challenging due to the underrepresentation of this class. Prediction on the majority class (YES in this case) may be reasonably accurate out of sample due to their relatively ample size. Consequently, an imbalanced data set can lead to poor model performance. In this empirical exercise, we employ several approaches to lessen the effects of imbalance.

The second data set is thirty years of daily stock prices for the S&P 500 ETF from 1-29-1993 to 2-4-2022. The relevant entries are Open, High, Low, Close, Adjusted Close, and Volume. Stock market prediction is fraught with many challenges. One, there are countably many factors that can cause changes in stock price movement. For example, a negative tweet about a company's environmental practices, the death or resignation of a CEO, economic crisis, geopolitical conflict, just to mention a few. The stock price is an amalgamation of all information available to the public.

There is valuable information in prices that can be mined for a competitive advantage in investing and stock trading.

However, due to its time-series nature, stock prices retain complex statistical properties, and the mechanisms behind the process are unrevealed to a large extent. Moreover, stock prices and financial time series generally tend to have high noise to signal ratio. A machine learning model can overfit the noise rather than the signal, leading to poor generalization. Time series also have natural in-built memory in their data generation process. There are high levels of serial and autocorrelation in price movement. For example, the price of an Apple stock is likely to depend on its previous prices. One of the fundamental assumptions of machine learning models is the notion of independent and identically distributed (i.i.d) samples. This i.i.d assumption does not extend to time series data, so unique data curation methods must be considered. For example, when performing cross-validation on time series data, random split into training and test can lead to unreliably optimistic predictions due to data snooping. Instead, roll-forward cross-validation must be performed.

Other challenges in time series include regime shifts or changes that occur over time due to economic shocks. The price evolution of the S&P500 ETF in Figure 1 clearly shows some of these regime shifts. The Dot-com Bubble of the early 2000s, the Great Financial Crisis of 2008, and the Covid-19 Pandemic Crisis reveal a significant drop in these periods followed by rising prices indicating "boom-bust" economic cycles (see Figure 1 below). These significant abrupt changes in trends can pose challenges for machine learning models and have become an area of active research in machine learning in finance.

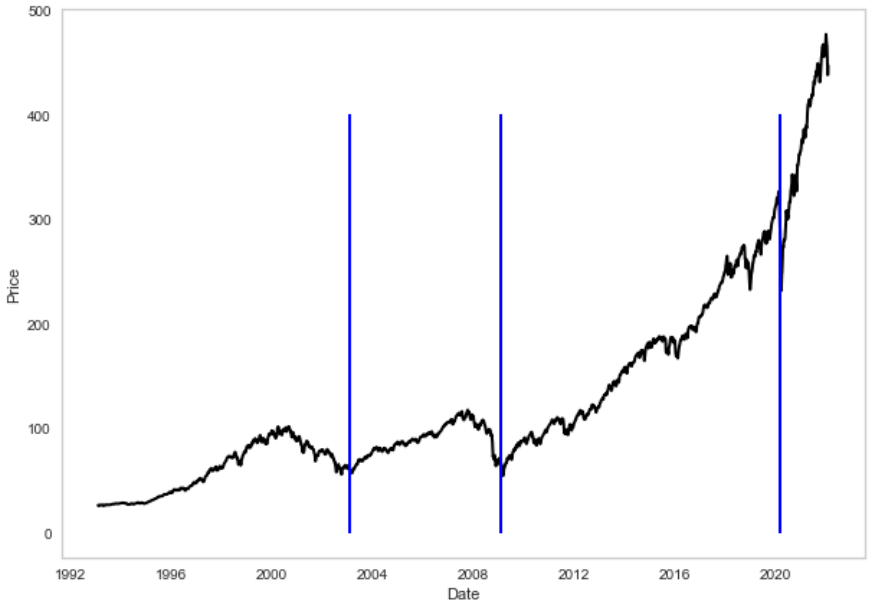


Figure 1: Price evolution of S&P500 ETF from 1993-2022.[1]

Using our domain in stocking trading, we frame the classification problem by defining a long-short trading strategy, generating discrete target labels, and engineering relevant features. The discrete labels represent actionable trading decisions to BUY, SELL or do NOTHING. We

[1] Major economic shocks are identified in blue vertical lines

generate trading signals from stock returns by dividing the one-day future returns into predefined quantiles (0.66 and 0.34). Future returns greater than the value in the 0.66 quantile are considered a BUY (+1) opportunity. Future returns less than the value in the 0.34 quantile are deemed a SELL (-1) opportunity. Anything else is considered as do NOTHING. Therefore, the target series then becomes $\{-1, 0, 1\}$. The features engineered are in the form of technical indicators and lagged technical indicators, lagged returns, and date-time features.

3.0 Analysis of Results

Decision Trees typically suffer from overfitting and need to be pruned to get optimal performance. As such, we first explore the effect of pruning in the form of a cost complexity penalty – α. Specifically, we show the impact of α on the number of nodes, tree depth, and model accuracy (see Figure 3 below). In the left panel of Figure 2, we observe that both tree depth and node numbers decrease as we continually increase the penalty parameter α. In terms of model performance, the trade-off in train and validation is evident in the right panel of Figure 2. Pruning the model during training causes in-sample accuracy to reduce (as we increase α) but leads to a better model generalization on the validation in terms of improved accuracy score.

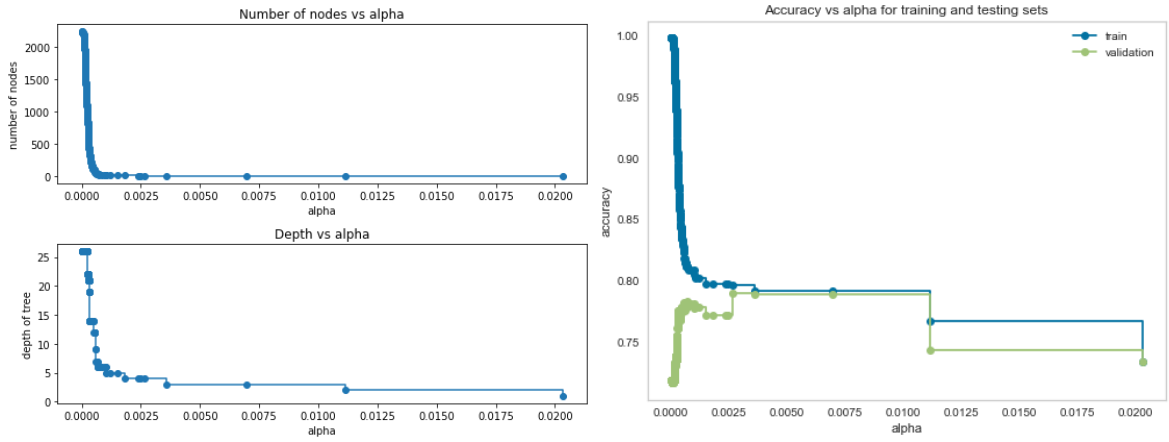


Figure 2: Effects of pruning in the form cost complexity penalty parameter[2] (α)

The churn dataset is highly imbalanced in terms of the labels; therefore, we examine its effect on the model's performance. First, we fit a decision tree with un-optimized hyperparameters on the imbalanced data. Second, we adjust the model to account for the fact that the labels are imbalanced and measure the performance. Thirdly, we oversample the minority class label, fit the model with optimized hyperparameters, and measure performance. Table 1 reports F1 scores on the test set.

[2] The output is from the churn dataset.

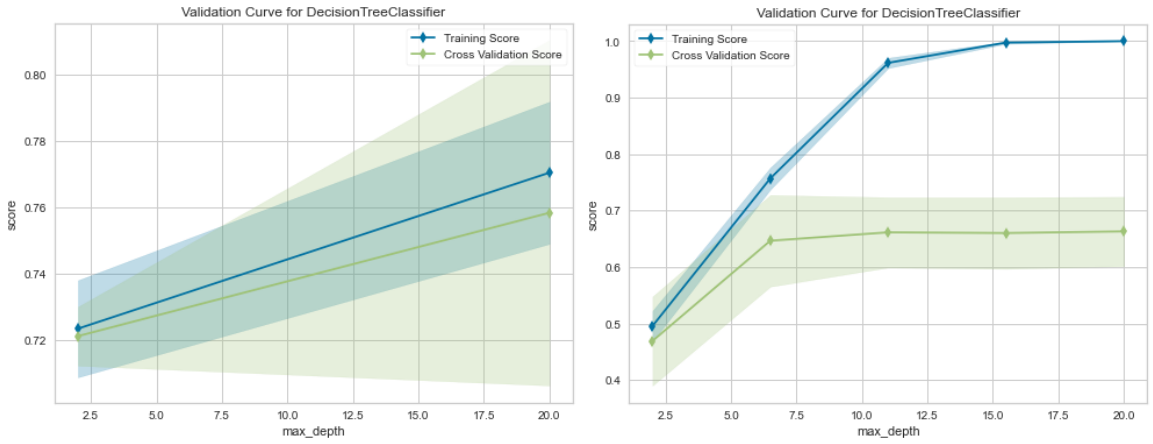
F1-Score	Decision Tree (Imbalanced & Unpruned)	Decision Tree (Balanced & Pruned)	Decision Tree (Oversampled & Pruned)	Support
Yes	0.81	0.81	0.78	1033
No	0.48	0.49	0.56	374
Macro Average	0.64	0.65	0.67	1477

Table 1: Imbalanced data effect of DT performance

We can observe that the performance of the decision tree is partly due to the data set used. Table 1 shows that the F1-score on "False" class labels is significantly higher than those of "Yes" due to the unproportionally many data points in the support. We see marginal improvement in the performance of the "No" labels and overall performance when we oversample the minority class to increase its occurrence in the training data set.

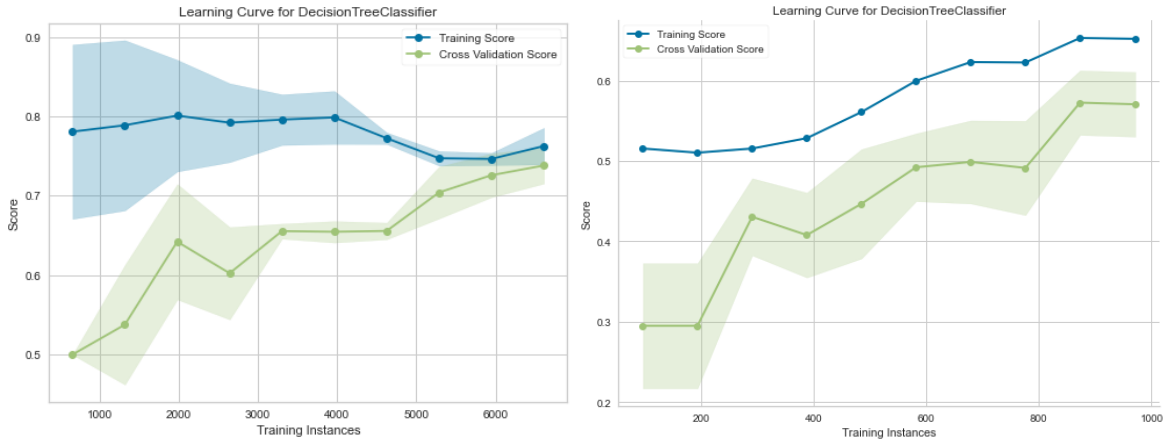
Hyperparameters such as maximum tree depth can raise overfitting or underfitting for decision trees. We investigate the tree depth on both data sets (see Figure 3(a)-(b) below). Across both data sets, we observe that with an increase in tree depth, in-sample performance increase with a consequent trade-off in worse cross-validation score. Due to the complexity of an underlying data generation process, more data is needed for a model to learn patterns in the data. Evidence shows that the decision tree can benefit from better generalization when we increase training instances. Figure 4 (a)-(b) reports the learning curves for the optimized and pruned DT. Notably, we see the training score suffer a bit (at 4000 instances) while the cross-validation score still increases with the addition of more training examples. This indicates that after around 4000 samples, the model starts generalizing more and recovering from overfitting.

On the other hand, the ETF data set increases both training and cross-validation scores with an increase in training instances without convergence. For the model to learn the underlying complexity of the data and generalize well, more data is needed. Given the limited ETF data set, another alternative was to reduce the feature space through dimensionality reduction techniques such as Principal Component Analysis (PCA). This approach extracts only the relevant features that can aid in learning. The curse of dimensionality is directly linked with the feature space via an exponential relationship. As the number of features increases, the amount of data needed to generalize grows exponentially.



(a) Churn data (b) ETF data

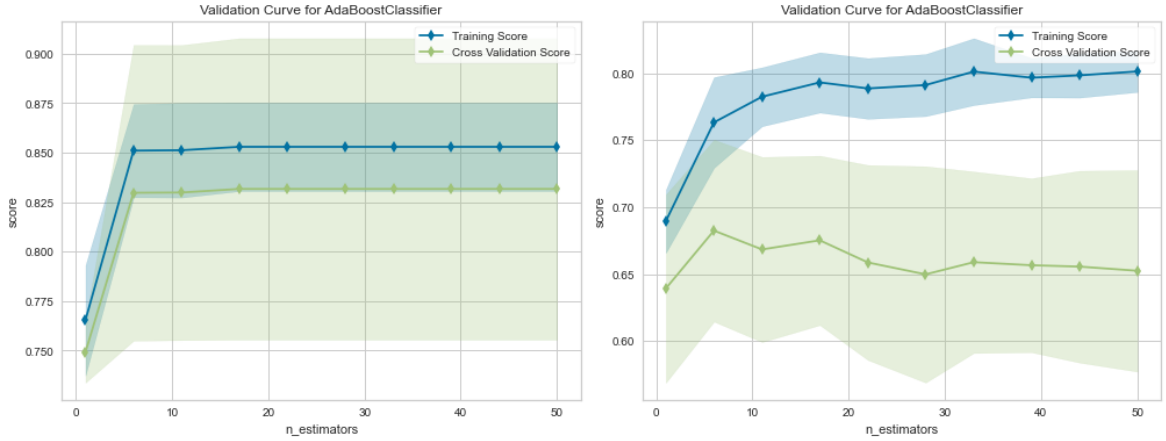
Figure 3: Validation curve for DT Classifier on the two data sets



(a) Churn data (b) ETF data

Figure 4: Learning curves for DT Classifier on the two data sets

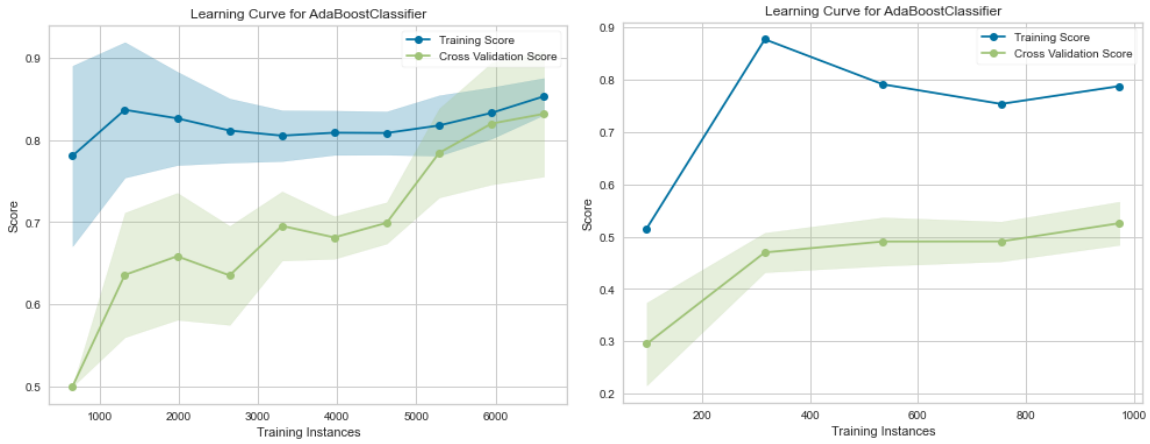
Adaptive Boosted Decision Trees are sequential ensembles of multiple decision trees. Boosting helps overcome the problem of underfitting but can be sensitive to noisy data. The ensemble prediction is guaranteed to get better in each successive build of new trees. In adaptive boosting, there is "actual" learning through the feedback loop of constantly updating sample weight to increase the probability of picking an incorrectly classified example in subsequent iterations. This process is akin to backpropagation in neural networks. We report a marked improvement in score for the boosted trees. To reduce overfitting, we apply an aggressive pruning penalty. The validation curves in Figures 5 (a)-(b) show that an ensemble of 6 decision trees is optimal to obtain the highest score for both Churn and ETF data sets.



(a) Churn data (b) ETF data

Figure 5: Validation curves for Boosted DT Classifier on the two data sets

In terms of the learning capabilities of the boosted DT, the learning curve on both data sets suggest similar observations in decisions in Figure 6. The model can benefit from more data.



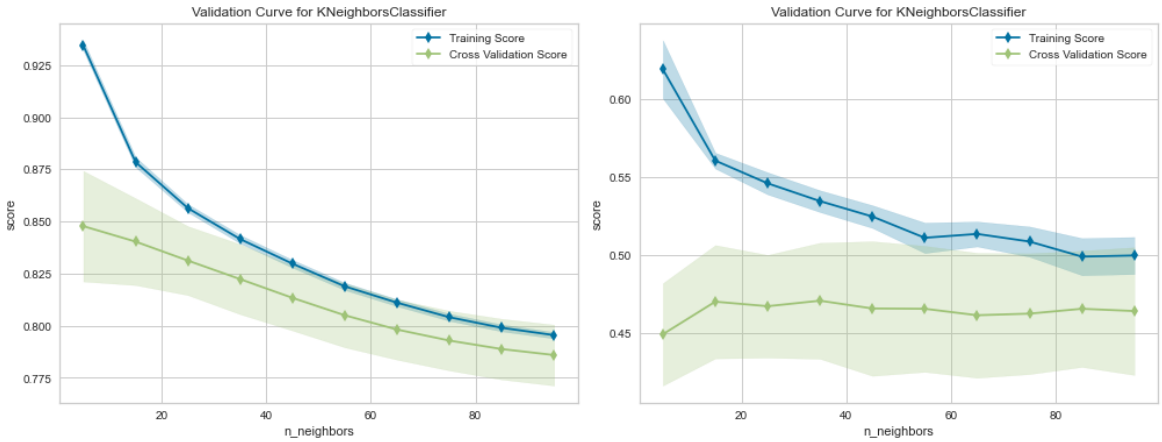
(a) Churn data (b) ETF data

Figure 6: Learning curves for Boosted DT Classifier on the two data sets

k-NN Classifier is a lazy algorithm that postpones learning to the actual query time for a prediction. In its training phase, the k-NN algorithm just stores the data. At query time, k-NN constructs local approximations of functions specifically tailored to the new instance been queried. In other words, k-NN never forms an explicit general hypothesis regarding the target function.[3] Due to its delayed learning, query time is usually longer.

We tried two distance metrics (Euclidean, Manhattan) for the Churn data set, distance weights (uniform or inverse distance), and a varying number of nearest neighbors. The optimized model obtained via grid search uses a "Manhattan", "distance" and k=35. We explore another distance metric for the ETF data — Dynamic Time Warping (DTW)—which is more suitable for evaluating the similarity between time-series data. Euclidean distance is very sensitive to small changes in the time axis, leading to wrong measurements. The DTW overcomes this problem making optimal correspondence for two time series by a non-linear stretching and shortening of the time axis. Given two time-series of length *N* and *M*, the DTW distance has a disadvantage due to its order *O(NM) calculation*. The validation curves in Figures 7 (a)-(b) reveal that with a decrease in the number of neighbors, the k-NN algorithm tends to overfit.

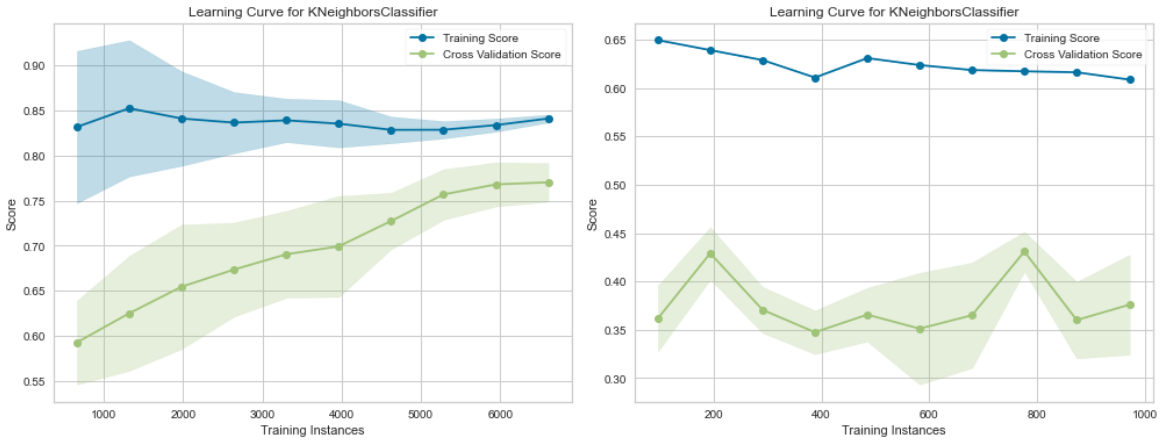
[3] Mitchell, T. M. (1997). Machine learning. 1997. *Burr Ridge, IL: McGraw Hill*, *45*(37), 870-877. pp. 232



(a) Churn data (b) ETF data

Figure 7: Validation curves for k-NN Classifier on the two data sets

The learning curves of k-NN are reported in Figures 8 (a)-(b). In the case of the churn data set, model performance can be improved considerably with an increased number of training instances. The learning curve for the ETF data set reveals a severe case of underfitting. The accuracy score of the training set remains almost flat with increasing training instances. Furthermore, the cross-validation score also gets worse. There's a high variance in performance scores between the training and validation set, suggesting that the model does not have the suitable capacity for the complexity of the ETF data set.



(a) Churn data (b) ETF data

Figure 8: Learning curves for k-NN Classifier on the two data sets

SVM Classifier chooses a line or a point that maximizes the distances/margins to the nearest point in either class. The maximum margin is only valid when the data is linearly separable. If data is not linearly separable, a kernel function can be used. A kernel function quantifies the similarity between two observations by mapping the data from low dimensional to high dimensional space. This report uses the Gaussian, Polynomial, and Sigmoid kernels.

Two critical hyperparameters affect the model's decision boundary in all three kernels -- C *and* γ. C calculates the trade-off between the proper classification of the training set and a smooth decision boundary. Large C means more support vectors, leading to an un-smoothed

decision boundary, overfitting, and high variance. Small C implies few support vectors with smooth decision boundary and reduced overfitting. The hyperparameter γ measures how far the influence of a single training instance reaches. High γ means decision boundary will depend on points closer to the boundary and ignores points farther away from it, leading to overfitting. Low γ considers faraway points when determining decision boundaries. Consequently, near point has low weights. Figures 9 (a)-(b) report the validations curves for C *and* γ on the churn data set using a sigmoid kernel.

Figure 9: Validation curves for SVM Classifier on the Churn data set

The learning curves in Figures 10 (a)-(b) show that the SVM model on the Churn data set can benefit from more training instances because both training and cross-validation scores tend to increase towards convergence as training instances increase. On the other hand, the SVM model has severe underfitting for the ETF data set. The SVM classifier cannot capture the complexity of the ETF data set.

(a) Churn data (b) ETF data

Figure 10: Learning curves for k-NN Classifier on the two data sets

Artificial Neural Networks: We use a Multi-Layer Perceptron (MLP) as a neural network architecture in this analysis. A simple MLP starts with random weights for each feature in the training example modifies the perceptron weights whenever it classifies an example. It repeats this

process iteratively until the perceptron classifies every training example correctly. Gradient descent determines the vector of weights which minimizes the classification error. Due to the feedback loop in the forms of weight updating, MLPs tend to perform better, given that they can learn from their mistakes and improve. An appropriate activation function (squashing function) is needed to map input features to discrete target labels for a classification problem. In the case of the Churn data set, we used the logistic function due to their binary labels. A hyperbolic tangent function (tanh) bounded between -1 and 1 is used for the ETF data set.

Figures 11 reports the validation curves for the MLP model on the Churn data sets. We observe that as we increase the number of times each data point is being used in training, the accuracy score increases in the case of the churn data set. The learning rate controls the step size in updating the weights. We observe that by increasing the initial learning rate (i.e., the step-size of weight updating) at a constant rate, the training and cross-validation accuracy scores decrease. We observe the same trend in validation curves when using an adaptive learning rate.

The learning curves in Figures 12 (a)-(b) echo similar observations from other models discussed. Performance can be improved with more training instances for the Churn data set. The ETF data set shows that the MLP cannot learn the complexity in the data. Lastly, we look at the loss curves in Figures 13 (a)-(b). The loss curve for the churn data set exhibits an irregular surface which may indicate an issue of unrepresentative train data set. An unrepresentative data does not contain sufficient information to learn the problem at hand. For instance, in the Churn data set, there are relatively few examples of "NO" target labels for the model to learn from. As a result of this imbalanced data set, we see training and validation curves showing improvement, but a significant gap remains between the curves. On the other hand, the ETF data set offers a good fit and reveals that continued training is likely to result in overfitting.

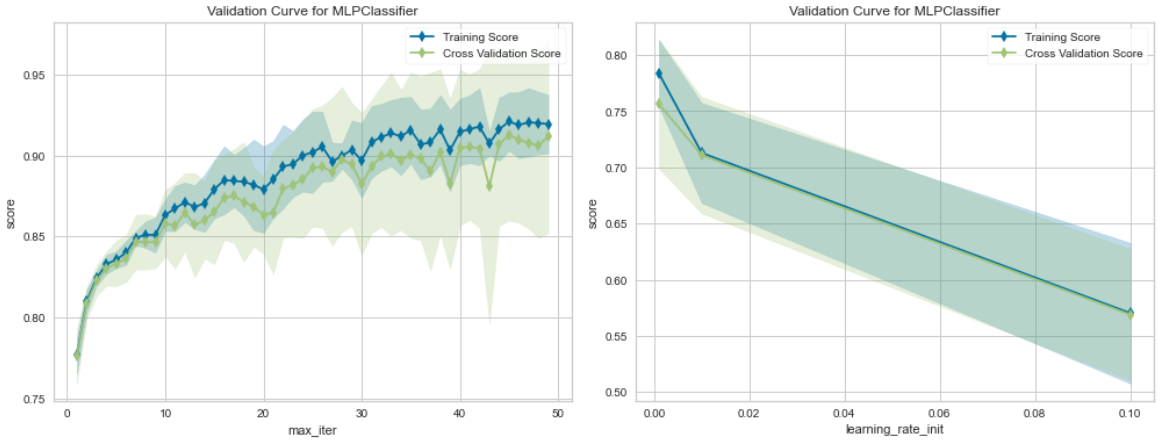
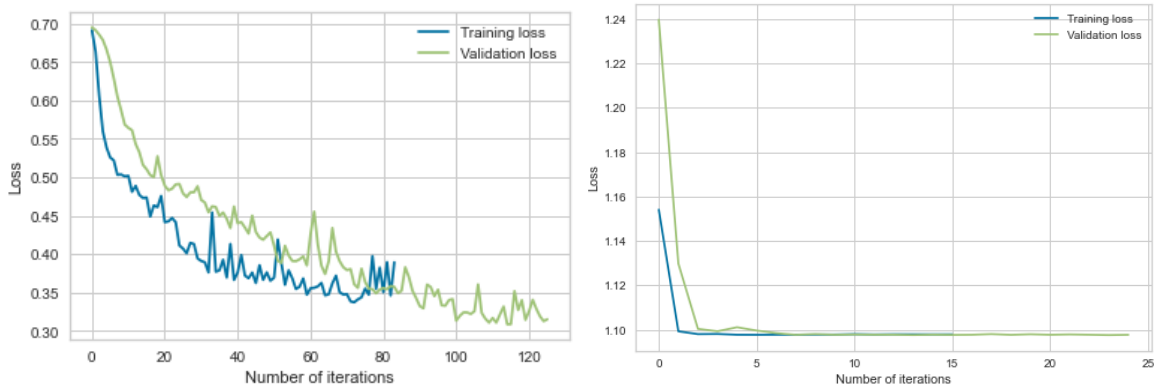


Figure 11: Validation curves for MLP Classifier on the Churn data set

(a) Churn data (b) ETF data

Figure 12: Learning curves for MLP Classifier on the two data sets



(a) Churn data (b) ETF data

Figure 13: Loss curves for MLP Classifier on the two data sets.

4.0 Performance Comparison

We evaluate and rank the five algorithms' performance on both data sets across several metrics. Specifically, we use the ROC AUC score are the primary metric for assessing performance. We also consider other metrics such as generalization error (cross-validation), training, and query time. Due to the imbalanced nature of the Churn data set, we also evaluate each algorithm's ability to classify the minority labels in the data. From Table 2[4] below, we rank the AdaBoost Decision Tree as the best performer across metrics. The MLP classifier is equally good, but we assign it a second-best performer due to its longer training time. It appears that the learners who use an updating rule or feedback loop have better performance (i.e., ensemble decision tree and neural nets).

[4] Best performer as in bold face.

Metric	DT	adaBoost DT	K-NN	SVM (L1)	MLP(Logistic)
Yes (F1-score)	0.78	**0.85**	0.82	0.86	**0.85**
No (F1-score)	0.56	**0.59**	0.58	0.57	**0.59**
Accuracy (cross validation)	0.78	**0.84**	0.78	0.85	**0.84**
ROC AUC	0.808	**0.832**	0.80	0.81	**0.833**
Wall time (training)	31.7ms	**508ms**	136ms	31.2s	**7.96s**
Wall time (query)	4.68ms	**33.3ms**	1.61s	1.72s	**13.5ms**

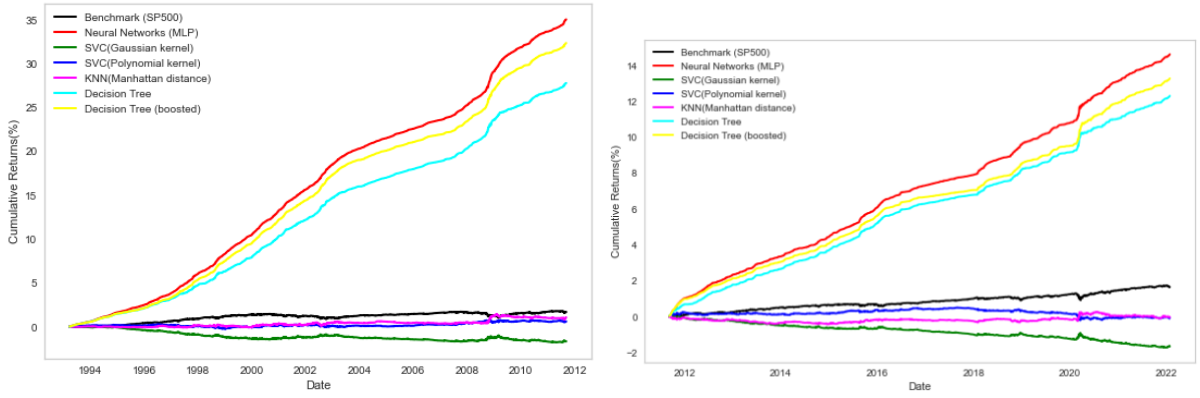
Table 2: Performance of all learners on the Churn data set

In Table 3 below, we present the performance of different configurations of the five algorithms on the ETF data set. Notably, we observe that the difference in performance does not change much between k-NN (L1 or Manhattan distance) and k-NN DTW. However, the disadvantage of the DTW as a distance measure becomes apparent in the query time. It takes the k-NN DTW 5 hours to query, which is not practical.

Metric	DT	adaBoost DT	k-NN (L1)	k-NN (DTW)	SVM (Poly)	SVM (RBF)	MLP (tanh)
Accuracy (cross validation)	0.628	0.67	0.45	0.45	0.62	0.66	**0.69**
ROC AUC	0.808	0.808	0.64	0.64	0.70	0.88	**0.93**
Wall time (training)	139ms	698ms	19.9ms	287ms	4.41s	1.43s	**8.8s**
Wall time (query)	13.9ms	5.06ms	359ms	4h 59m	177ms	861ms	**4.2ms**

Table 3: Performance of all learners on the ETF data set

Finally, based on our domain knowledge of the ETF data set, we also evaluate the learners based on cumulative returns they can generate relative to the benchmark index (S&P 500 ETF). The problem for the ETF dataset was to develop a stock market prediction that would perform better than the benchmark. Figure 15 (a)-(b) show the cumulative returns both in-sample and out-sample. Ignoring transaction cost, we would make more money by trading out strategy with top learners (Neural Networks, Boosted Decision Trees, and Decisions Trees) than holding a position in the S&P 500 ETF.



(a) Training period (b) Testing period

Figure 14: Cumulative returns relative to benchmark for all learners