Joshua Demeo (gtid: jdemeo3)
Supervised Learning Assignment

HIV-1 Protease Cleavage and Gamma Particle Classification

Abstract

In this work, a number of supervised learning algorithms are applied to two distinct datasets: HIV-1 protease cleavage[1] and MAGIC Gamma Telescope[2]. Model complexity and learning curve analysis was used to compare performance of algorithms (Decision Tree, AdaBoosting - Decision Trees, Multi-Layer Perceptron, Support Vector Machine, K-Nearest Neighbors) before seeing how well they generalize to a held out test set of data.

Importance of the Datasets

Each dataset is interesting for the sake of understanding the differences between machine learning algorithms. This is because neither set is able to be classified perfectly by the accuracy metric, differences between hyperparameter values are noticeable, and the differences in shape of each dataset: HIV-1 protease cleavage (2371 rows, 160 features) and MAGIC Gamma Telescope (19020 rows, 10 features) allows for checking how amount of data and number of features effects performance. Also the types of features are distinct for each set: HIV-1 protease cleavage being all one-hot encodings and MAGIC Gamma Telescope being all continuous values.

Methodology

All models were implemented with the python Scikit-Learn library.[3] The process of finding the best model before testing involves:

- Shuffling the dataset and stratify splitting the datasets into a 70/30 split for train/test sets
- Performing cross validation on the the training set with each of the default hyperparameters for each machine learning algorithm. A learning curve was plotted to see at what amount of training samples does the model negligibly improve in quality with increasing training size. For each dataset, 2-fold cross validation was sufficient for use in model complexity analysis as determined by the stated learning curve analysis.
- To find the optimal hyperparameters for each algorithm, various values of each hyperparameter were plotted against model training and validation accuracy and the ‘optimal hyperparameter’ was chosen for final learning curve analysis.
    - I would like to note that the ‘optimal hyperparameter’ for all models are determined in the same manner: the lowest value for a hyperparameter at which the model plateaus in performance on the validation set. This is a subjective choice, but I tend to lean on Occam’s razor and choose the simplest possible model with the ‘best’ performance on the validation set.
- After plotting the final models learning curves, they were tested against the held-out test set.

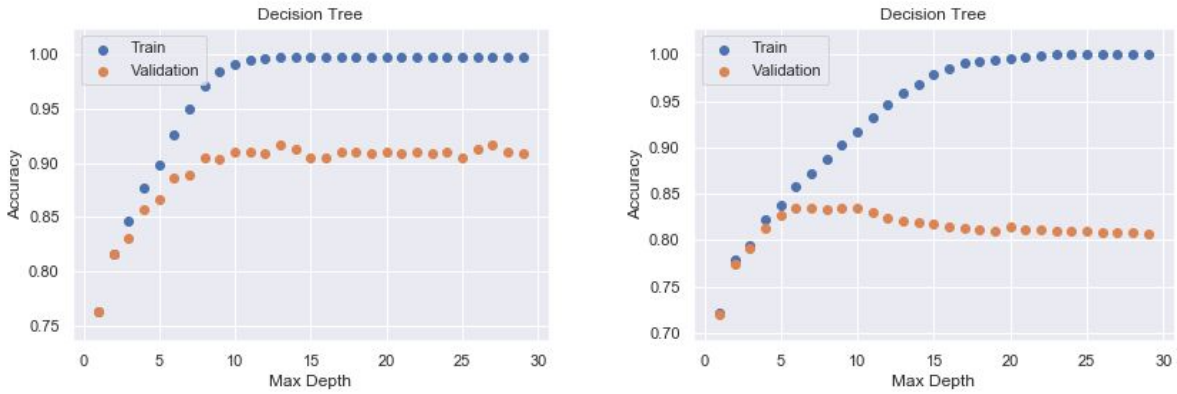
For clarity, the HIV-1 protease cleavage dataset will be noted as HIV-1 and the MAGIC Gamma Telescope dataset will be noted as Gamma in this paper.

Results and Discussion

Decision Tree

For analyzing the Decision Tree algorithm the hyperparameter of focus was it's maximum depth (how deep the tree can grow). Note that the quality of a split was determined by information gain from the change in entropy.

HIV-1 **Gamma**



* All plots are presented side by side between the two datasets (labeled above plots)
Figure 1. Decision Tree performance as a function of Tree depth

As the depth of the Decision Tree increases, the training and validation accuracy increases up to a certain depth (**Figure 1)**. The validation accuracy plateaus around a depth of 9 for the HIV-1 dataset and a depth of 5 for the Gamma dataset. The difference in the max depth for each dataset is a result of the difference in the number of features for each dataset, 160 for HIV-1 dataset and 10 for Gamma dataset. Given there are far more features in the HIV-1 dataset, there are more potential splits on features to increase information gain for the model so a greater max depth is expected as compared to the Gamma dataset. The max depths stated for each dataset are what are used as the optimal hyperparameters for later learning curve analysis and final tested models. This same logic is applied to each model discussed here after.

As the number of training samples increase, the accuracy on the validation set increases (**Figure 2**). For both datasets, the training accuracy decreases with increasing number of training samples, but the Gamma dataset drops off more dramatically. This can be attributed to the dataset covering a more diverse input space with its continuous valued features and the max depth pruning which may prevent the Decision Tree from overfitting to the training set so it loses training accuracy with increasing training sample size.

HIV-1 **Gamma**

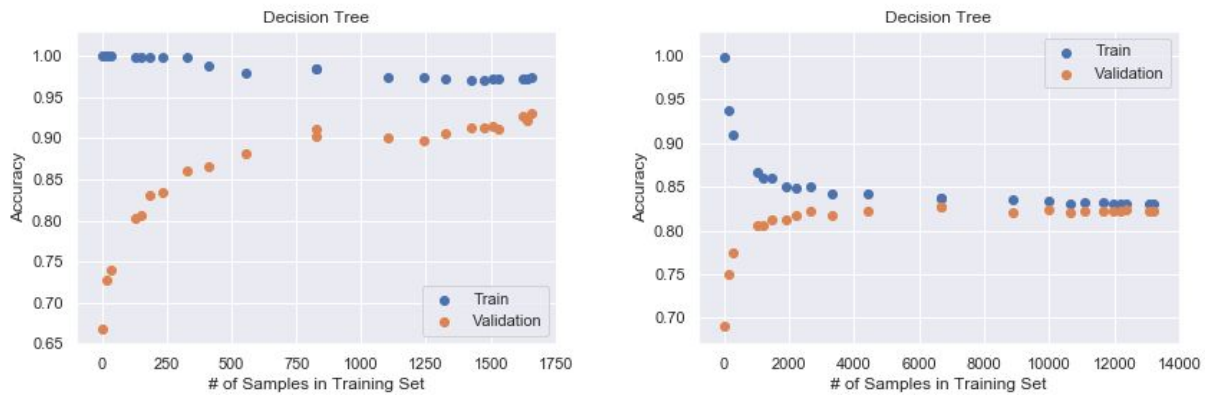


Figure 2. Optimized Decision Tree performance as a function of training sample size

Adaboost - Decision Trees

For analyzing the boosting on Decision Trees, the AdaBoost classifier with Decision Trees as its base classifier was chosen in Scikit-Learn. The primary hyperparameters of focus were the trees maximum depth and then the number of iterations (sequential boosting builds). Note that the quality of a split was determined by information gain from the change in entropy just as in the Decision Tree models.

HIV-1 **Gamma**

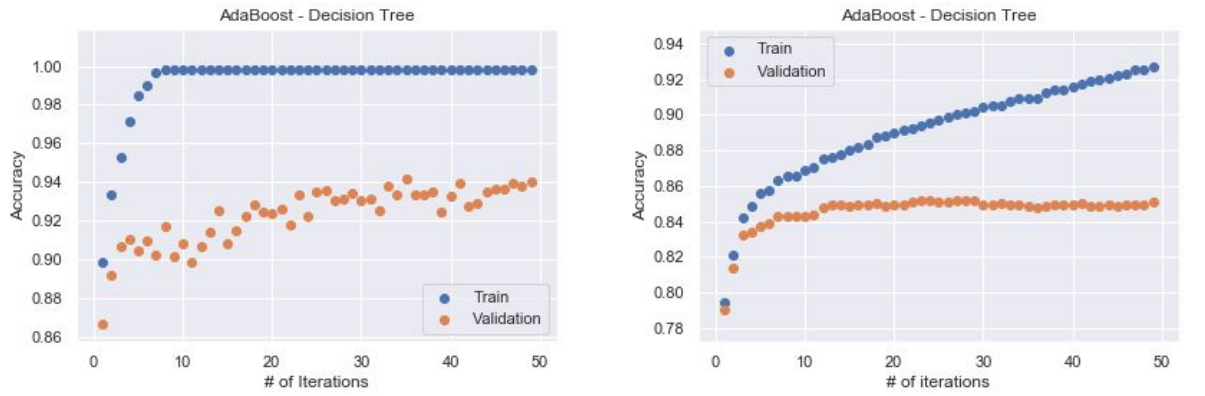


Figure 3. AdaBoost - Decision Tree performance as a function of boosting iterations

The optimal maximum depth for the base tree was 5 for the HIV-1 dataset and 3 for the Gamma dataset (*model complexity curves not shown). Using these optimal hyperparameters, the optimal amount of iterations of boosting for each model was determined to be 25 for the HIV-1 dataset and 12 for the Gamma dataset (**Figure 3**). As the number of sequential models during boosting increases, model performance on both training and validation sets increase, but each plateaus on the validation set. The training accuracy is nearly perfect on the HIV-1 dataset after 8 or 9 iterations, but the validation accuracy continuously increases in accuracy. This can be attributed to the nature of boosting where the confidence/margin for each successive boost

tends to increases for each round of boosting, which helps to prevent overfitting.[4] The same could be said about the Gamma dataset.

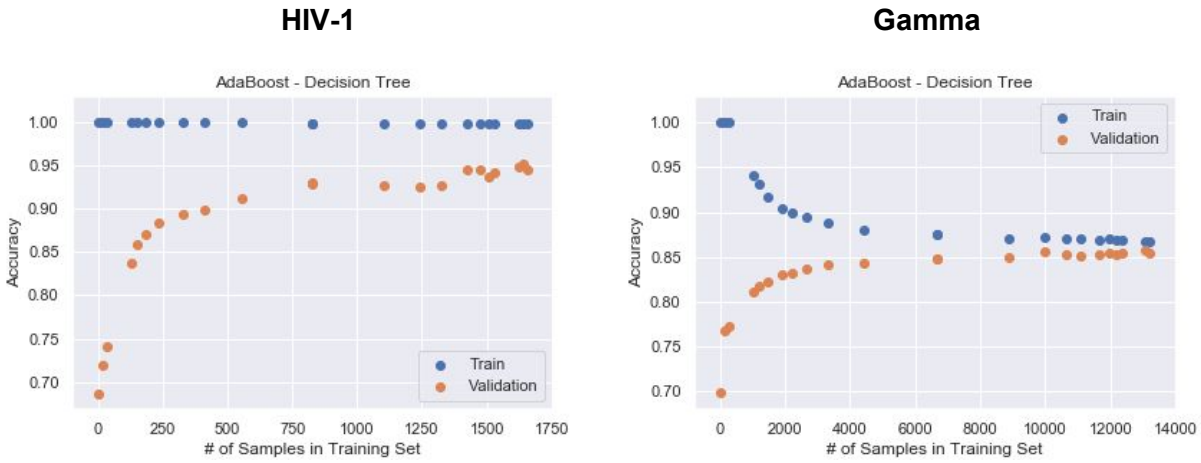


Figure 4. Optimized AdaBoost - Decision Tree performance as a function of training sample size

Each model continues to improve on the validation set as the number of training examples increases over time (**Figure 4**). For Gamma, the training accuracy decreases overtime, which might be explained by overfitting to the noise in the dataset during early training and then becoming less prominent with increasing sample size. Also the training accuracy remains near perfect for the HIV-1 dataset which may indicate that the input data has little noise in it despite the larger input size as compared to the Gamma dataset. Also to note, the boosting algorithms perform better by validation accuracy for both datasets as compared to Decision Trees alone. This showcases the ability to prevent overfitting of Decision Trees with the use of boosting.

Multi-Layer Perceptron (MLP)

The primary hyperparameters of focus for the MLP models were number of epochs (rounds of training) and learning rate. Extensive optimization of number of layers and hidden nodes was not performed as the hyperparameter space seemed too large to efficiently find the 'best' hyperparameters. Each MLP build consisted of 2 hidden layers with number of nodes equal to number of features of the dataset to capture essentially all possible functions.

Optimal hyperparameters for the HIV-1 dataset were 35 epochs and a learning rate of 0.001 while the Gamma dataset were 60 epochs and a learning rate of 0.025 (**Figure 5**). Learning rate did not affect performance on the HIV-1 dataset, but did so on the MLP dataset. The model complexity curves were not especially smooth for the Gamma dataset, which highlights again the amount of inherent noise in the dataset as compared to the HIV-1 dataset.

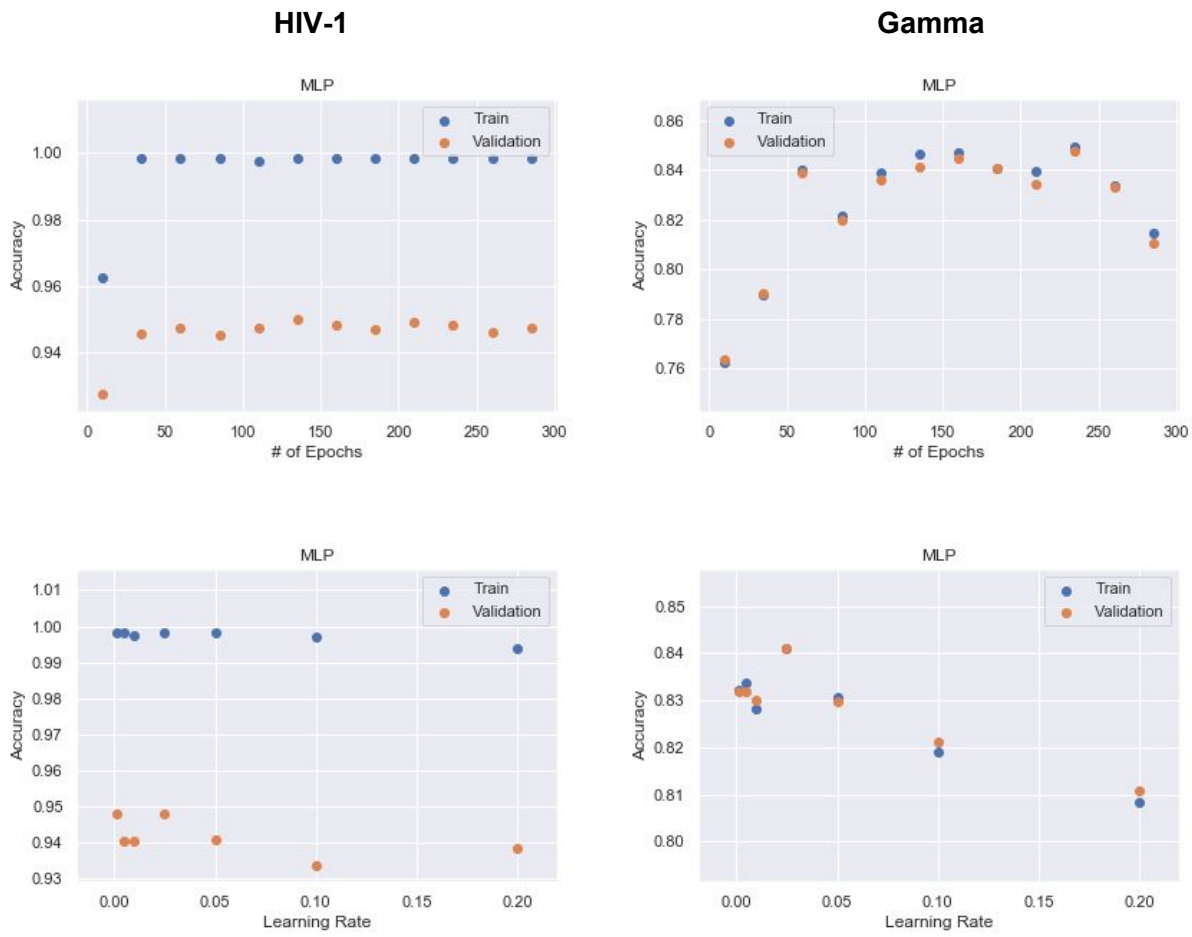


Figure 5. Multi-Layer Perceptron performance as a function of number of epochs and learning rates

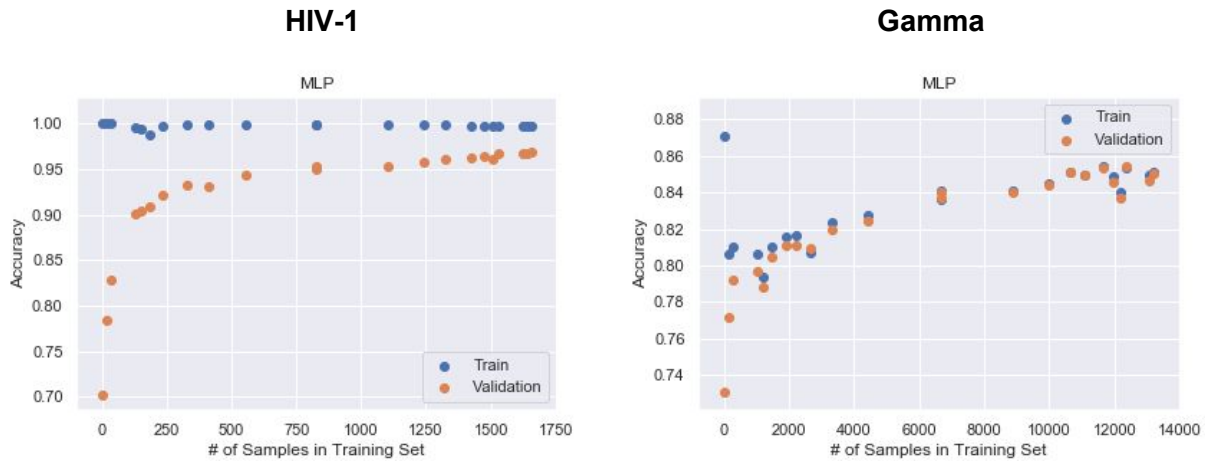


Figure 6. Optimized Multi-Layer Perceptron performance as a function of training sample size

Looking at the learning curves for each model (**Figure 6**), the models continue to improve on the validation set as the number of training examples increases over time. For the HIV-1 dataset, the training accuracy stays near perfect as the validation accuracy increases with increasing number of training samples. It looks like the model is not overfitting on the training

data as the validation accuracy increases with increasing training sample size, which either indicates the dataset is not especially noisy and/or the hyperparameter choices have tempered the possibility of overfitting. For the Gamma dataset, the training accuracy stays very close to validation accuracy, which indicates the algorithm is not overfitting the training set even at small training sample sizes and also that the dataset may be overly noisy so overfitting is not possible.

Support Vector Machine (SVM)

The primary hyperparameters of focus for the SVM models were the models kernel and a particular hyperparameter distinctive to that kernel.

HIV-1 **Gamma**

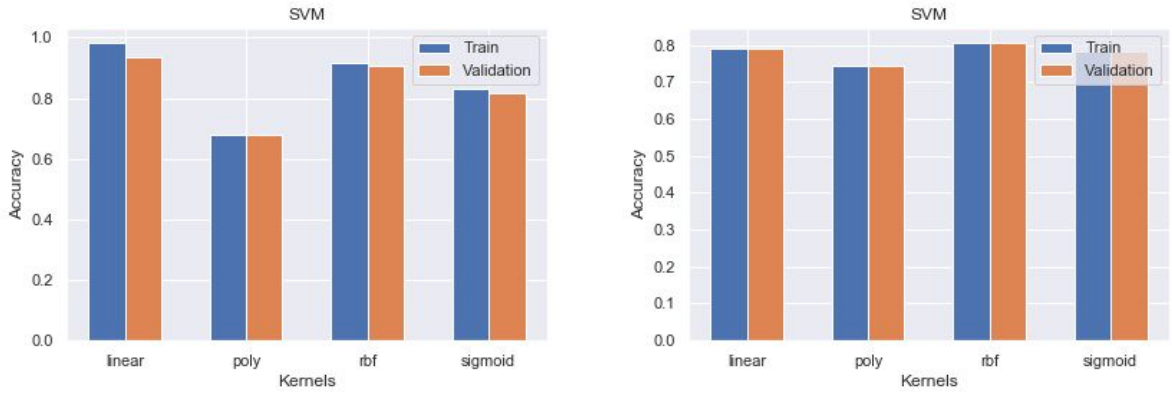


Figure 7. Support Vector Machine performance as a function of kernel used

Looking at validation performance (**Figure 7**), a 'linear' kernel is most appropriate for the HIV-1 dataset while an 'rbf' kernel performs the best on training and validation for the Gamma dataset. The optimal gamma is 2.0 for the Gamma dataset and the optimal penalty parameter C is 1.0 for both models (*model complexity curves not shown).

HIV-1 **Gamma**

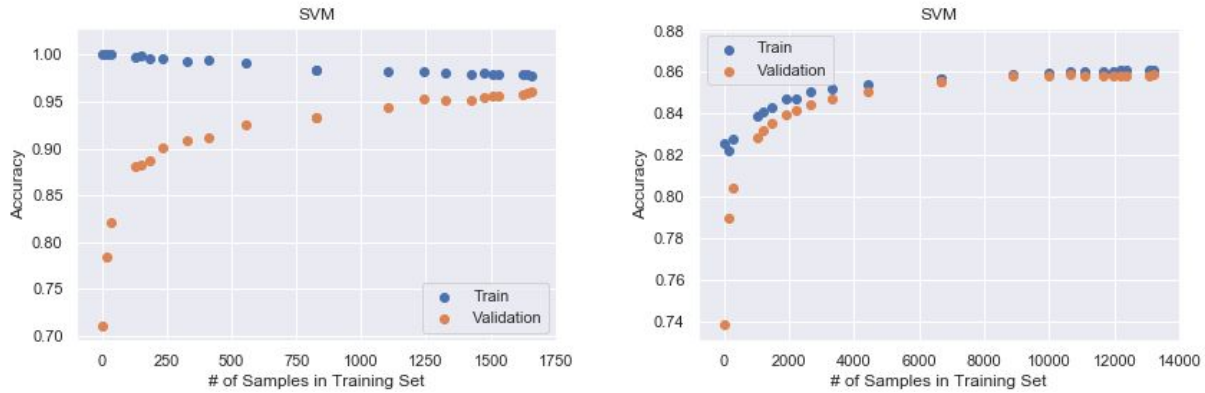


Figure 8. Optimized Support Vector Machine performance as a function of training sample size

Looking at the learning curves (**Figure 8**), the SVM models continue to improve on the validation set as the number of training examples increases. For the HIV dataset, there doesn't seem to be an issue of overfitting as the validation accuracy still improves with increasing number of training samples, which is likely due to the dataset having low levels of noise. Again the Gamma dataset seems to be complex and noisy enough such that the model even fails to accurately classify on the training set much better than the validation set even at small sample sizes.

K-Nearest Neighbors (KNN)

The primary hyperparameter of focus for the KNN models was the number of neighbors. The distance metric used was the Minkowski distance.

HIV-1 **Gamma**

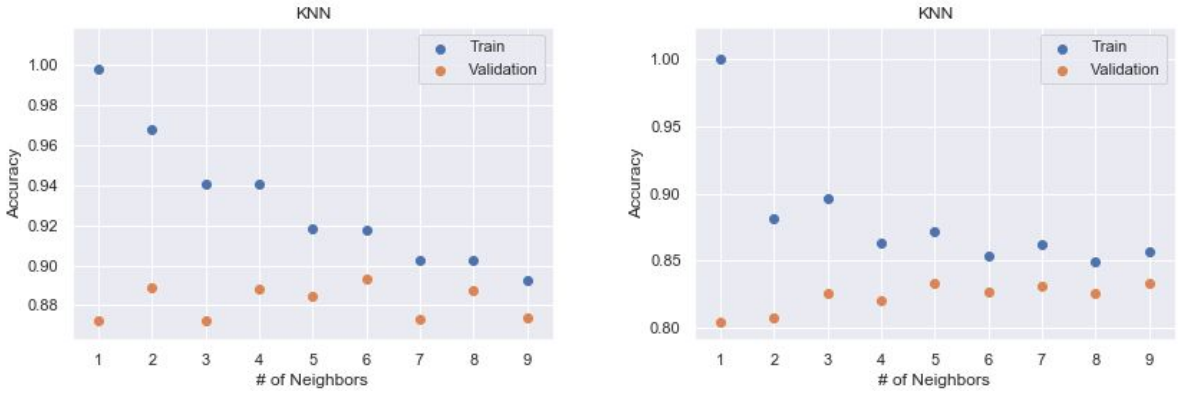


Figure 9. KNN performance as a function of number of neighbors

The optimal number of neighbors for the HIV-1 model is 1 while for the Gamma model is 3 which was chosen based on weighing the simplest interpretable model with the best validation score (**Figure 9**). The HIV-1 and Gamma models perform worse on the training set with increasing number of neighbors, which shows that the model becomes more biased with increasing number of neighbors. Validation accuracy was relatively constant for each k-value.

Looking at the learning curves, each model continues to improve on the validation set as the number of training examples increases over time (**Figure 10**). The complexity of the data for the Gamma dataset is apparent by even having the low number of training samples resulting in low training accuracy versus the HIV-1 dataset where it's nearly perfect classification on the training set with constant improvement in validation accuracy.

HIV-1 **Gamma**

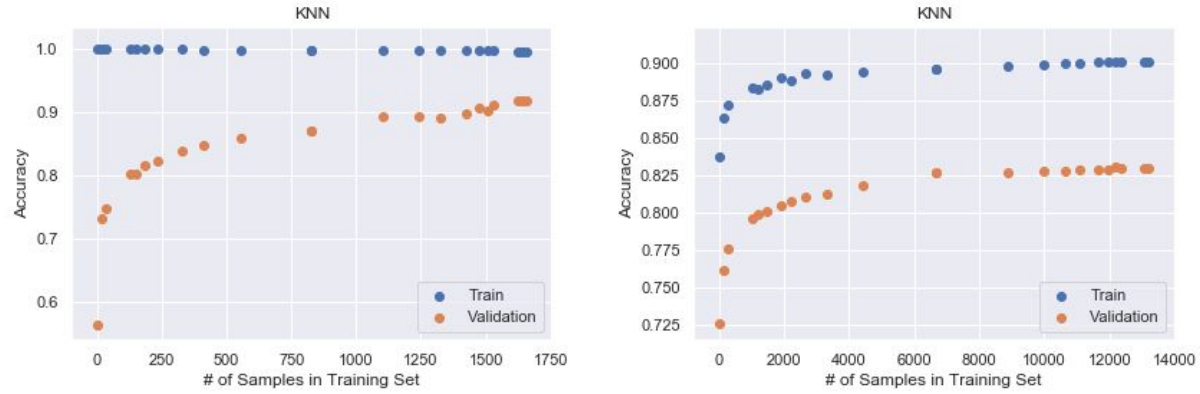


Figure 10. Optimized K-Nearest Neighbors performance as a function of training sample size

Final Model Performance

Using the optimal hyperparameters for each model described in the previous sections, they were tested on the held-out test set to see how well they generalize to unseen test data. The final 'best' models were determined by which performed best during cross validation.

Surprisingly for the HIV-1 dataset (**Table 1**), each model performed better on the test set relative to the validation set. The best performing model on the validation set was the Mutli-Layer Perceptron, which also performed the best on the test set. This was slightly surprising considering neural networks can overfit their data without more significant regularization. Possibly tempering the number of epochs helped to prevent overfitting. KNN performed the worst, which may indicate that Minkowski distance was not an appropriate criterion for classification for this task. Boosting did prove to be beneficial in improving Decision Tree performance both during validation and generalizing to unseen data.

For the Gamma dataset, all models validation accuracy was relatively the same as their test accuracy (**Table 2**). The best performing model on the validation set was the Support Vector Machine, which also performed the best on the test set, but only marginally better than the AdaBoost - Decision Trees model. SVM performing the 'best' indicates that using the kernel trick did help to change the dimensionality of the input data in a beneficial way. Boosting did prove to be beneficial in improving Decision Tree performance both during validation and generalizing to unseen data.

Table 1. HIV-1 - Summary Statistics

Model	Hyperparameters	2-fold CV Accuracy	Test Accuracy
Decision Tree	max_depth=9	0.913	0.920
AdaBoost - Decision Trees	max_depth=5,	0.929	0.965

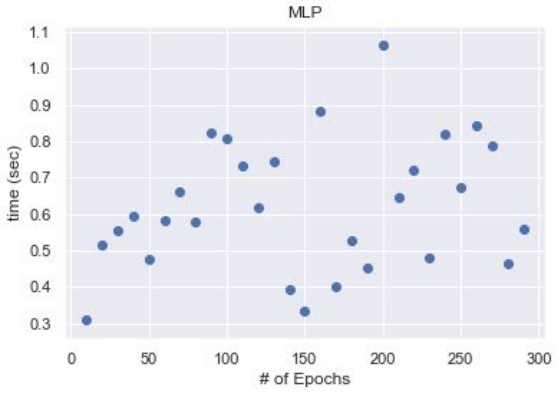
	n_estimators=25		
Multi-Layer Perceptron	max_iter=35, learning_rate_init=0.001	**0.946**	**0.976**
Support Vector Machine	kernel='linear'	0.937	0.961
K-Nearest Neighbors	n_neighbors=1	0.872	0.909

Table 2. Gamma - Summary Statistics

Model	Hyperparameters	2-fold CV Accuracy	Test Accuracy
Decision Tree	max_depth=5	0.827	0.819
AdaBoost - Decision Trees	max_depth=3, n_estimators=12	0.848	0.857
Multi-Layer Perceptron	max_iter=60, learning_rate_init=0.025	0.841	0.852
Support Vector Machine	kernel='rbf', gamma=2.0	**0.855**	**0.859**
K-Nearest Neighbors	n_neighbors=3	0.826	0.826

Time Complexity

In addition to optimizing hyperparameters it is important to understand the run times of these algorithms both during training, testing, and number of iterations/steps for an algorithm. Models that may perform similarly may have quite different runtimes and as result one algorithm may be more favorable especially as the size of the dataset increases. Models presented are using their respective hyperparameters optimized for the particular dataset and only the Gamma dataset plots (**Figure 11, 12**) are presented because the sample size is much larger than the HIV-1 dataset and the HIV-1 dataset yielded very similar plots regardless.



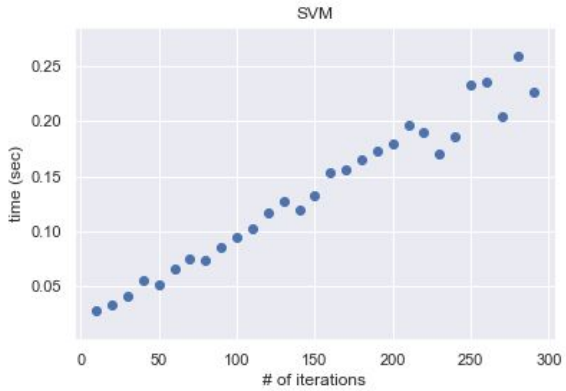


Figure 11. Runtime on the training set with respect to a models number of steps/iterations/epochs

Looking at AdaBoost with a Decision Tree, Multi-Layer Perceptron, and SVM we can understand their respective runtimes on the full training set with respect to the iterative parts of their algorithms (**Figure 11**). The number of boosting rounds runs about linearly with time for the AdaBoost model. For SVM, the number of iterations (optimization steps) increases about linearly in time. For MLP, the number of epochs (rounds of training) is quite variable with time, which is likely due to use of stochastic gradient descent to minimize the loss function so the time to reaching a minimum of the loss function can vary greatly depending on what random sampling of the training samples is chosen during each optimization step.

Now looking at the run times of each algorithm on the training and validation sets for the with respect to each sets sample size, the training time is about constant for each of the algorithms (**Figure 12**). Multi-Layer Perceptrons are quite variable in this regard likely due to a highly variable path for optimization of the loss function via stochastic gradient descent. The slowest of the algorithms for training is SVM but it is slightly odd that it is in constant time with samples size considering fit time complexity is more than quadratic with the number of samples during training.[5] The reason for not observing quadratic training time might be due to the sample size not being large enough to see significant differences.

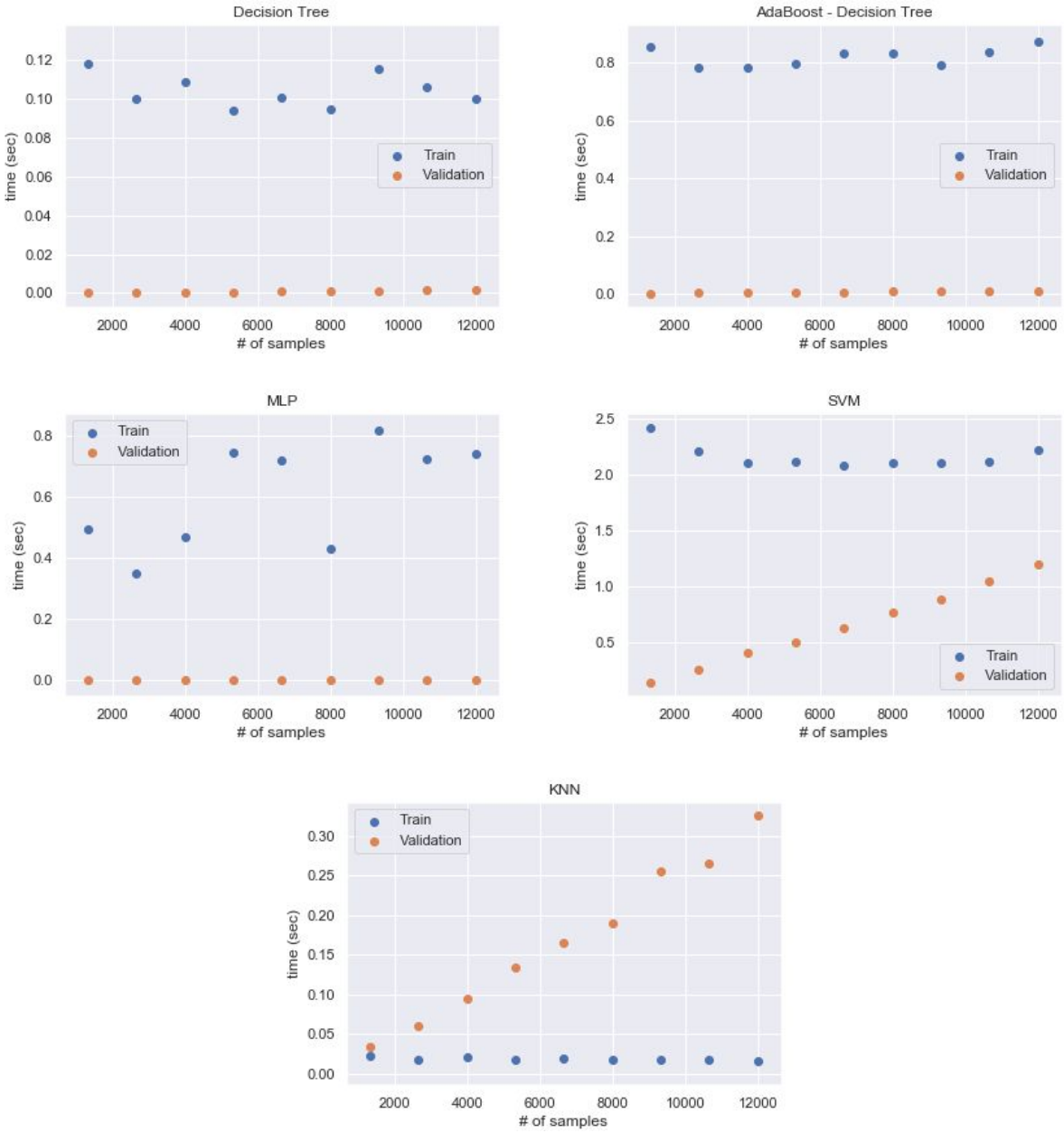


Figure 12. Runtime on the training set and validation set with respect to the number of samples in each set

As for validation after the model has been trained, Decision Tree, AdaBoost with Decision Tree, and MLP all run in constant time, but it likely should be in linear time. This is because the fitted function has already been made during training so only the test samples inputs need to be inputted into the function so should scale with the number of samples. SVM and KNN run times increase with increasing number of samples during validation. This is the case for KNN because only during validation does the algorithm query from the training samples to find the nearest neighbors and all neighbors distances must be calculated before classifying that one test example. As for SVM, the increase in runtime with increase in number of samples

can be explained by how the calculated classification of all the test samples is just the dot product with the classification hyperplane calculated during training so you would expect testing time to increase with increasing test size. Because the sample size seems too low to affect runtime significantly, the runtime for the algorithms wouldn't be a determining factor when choosing between two similarly performing algorithms. I would like to note that some of these runtimes seem intuitively odd, but may be because of how Scikit-Learn has optimized the algorithms with respect to different processors (my processor: 2.7GHz quad-core Intel Core i7).

Conclusions

In this work, a number of supervised learning algorithms were applied to two distinct datasets: HIV-1 protease cleavage[1] and MAGIC Gamma Telescope[2]. Model complexity and learning curve analysis was used to compare performance of algorithms. The best performing model on the validation set for the HIV-1 dataset was the Mutli-Layer Perceptron, which also performed the best on the test set. The best performing model on the validation set for the Gamma dataset was the Support Vector Machine, which also performed the best on the test set, but only marginally better than the AdaBoost - Decision Trees model. Differences in algorithm performance on each dataset could be marginal in some cases and their runtime complexity would not be a factor in choosing the best algorithm for each dataset since the sample size and feature size were not large enough to affect practical runtimes.

References

1. HIV-1 protease cleavage data set. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml/datasets/HIV-1+protease+cleavage#
2. MAGIC Gamma Telescope Data Set. UCI Machine Learning Repository. https://archive.ics.uci.edu/ml/datasets/magic+gamma+telescope
3. Scikit-Learn, Machine Learning in Python. http://scikit-learn.org/stable/
4. Schapire R.E. (2013) Explaining AdaBoost. In: Schölkopf B., Luo Z., Vovk V. (eds) Empirical Inference. Springer, Berlin, Heidelberg
5. sklearn.svm.SVC. Scikit-Learn. http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html