Everardo Villasenor

CS 7641

Spring 2020

Assignment 1: Supervised Learning

Dataset Analysis

For this assignment I have chosen two datasets found in the UCI Machine Learning Repository. The first dataset I chose is the Poker Hand Data Set. It contains the information for a five card poker hands with a suit and number for each card, therefore the data set has 10 categorical/integer features to predict the multivariate classification of the five cards of poker. This is an interesting problem because the rules of poker are known, and it has a definite mapping to a result and a finite number of unique combinations which means we would be using inductive reasoning of machine learning to find these rules. Therefore, the problem may be applicable to other problems where machine learning may be used to find certain rules. The dataset consists of 51,250 sample of hands, and I sampled a disproportionate number of rarer events such as a Royal Flush (Poker Hand = 9) in order to train and test over all the possible scenarios more accurately.

The next dataset I chose is the default of credit card clients. This data set tries to predict whether an account will default (Yes = 1, No = 0) based on 23 features. The features include the amount of the given credit, gender, education, marital status, age, history of past payments delays, amount of previous payments and bills. There are 30,000 samples within this dataset with 78% negative samples and 22% positive samples. This is an interesting dataset because it represents a practical problem that is difficult for risk managers to forecast because of the vast number of features and potentially infinite unique situations. This problem is like the Poker problem because credit card default is a rare event much like classifying a Royal Flush or Straight Flush. It is also different to the Poker problem because there are no defined rules and it is difficult to classify for a single person and a perfect candidate to attempt to use machine learning. I did conduct some basic exploratory data analysis and found that some features are highly skewed and will probably need to transform the data in order to make use of some machine learning algorithms.

Methodology

The goal of my experimentation is to explore six different supervised learning algorithms on the datasets described above which include a decision tree classifier, a neural network classifier, an adaboost classifier, an SVM, and a KNN classifier. The first step is to separate the datasets into a train and test set. I chose to stratify and split my dataset with an 80% train and 20% test split. I had to stratify in order to preserve the proportions of the target label which is especially important in the poker dataset which has some rare events. I also chose to split the dataset because even though I will use GridSearchCV because during the hyperparameter tuning some data might leak information. After this I assemble a Pipeline that consists of an appropriate preprocessing step and the classifier. Some of the time no preprocessor is needed and can only be determined on its effect on training accuracy through experimentation. Again, the pipeline is needed to the preprocessing step does not leak information to the test set.

Then a set of parameters was assembled for each classifier and then used within sklearn's hyperparameter tuning function, GridSearchCV. Critically, the number of times to cross-validate the hyperparameters is also fed as an input. Another input is the scoring type that GridSearchCV is optimized for and I specifically use accuracy and F1 macro score. Optimizing for accuracy is obviously important but the F1 score is a good metric to optimize as well because of many reasons. First of all, accuracy may be a misleading metric especially for datasets with rare events. If a hypothetical dataset has 99 samples for a negative classification and 1 sample for a positive classification, it is conceivable to get a 99% accuracy score by just classifying all the events as negative. This is an extreme example but there is no use for machine learning if one could achieve a 99% accuracy by classifying every event as negative. I chose F1 score (2 * (Precision * Recall) / (Precision + Recall)) as a metric to optimize because it is a balance of precision and recall metrics

which means the metric can more accurately focus on false negatives and false positives and how well the model does there. This is important in imbalanced datasets where accuracy can be high because of all the true negatives.

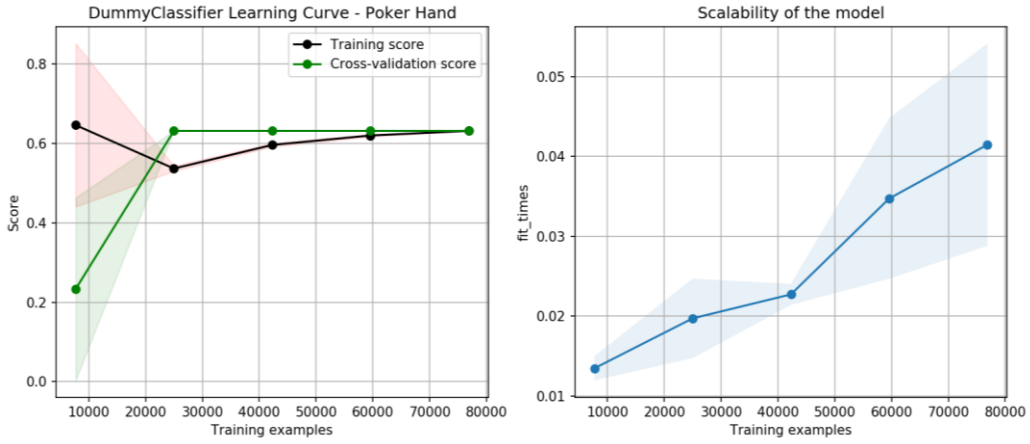
After it is tuned, a best estimator is used to generate a report detailing accuracy, recall and precision along with a confusion matrix to better see the distribution of how the classifier's distributed their predictions. In order to better see how the classifiers perform, the DummyClassifier package is used as a baseline classifier and sanity check. I would expect the DummyClassifier to be pretty good at accuracy given the imbalanced datasets but bad at the F1 metric.

The output of GridSearchCV should be a tuned and cross-validated model but it is tough to visualize the hyperparameters and their effect on the model's fit. This is important to visualize because there are times that the model may be under or overfitting, despite trying to mitigate that with generalizing the model more by using cross-validation. Therefore, I pick one hyperparameter to tune and visualize using a learning curve and validation curve. This shows the effects on training and cross-validation score as the model is fed more and more training examples. It also shows how the model chosen scales in terms of time to train and the number of training examples, which is important because right now on my local machine I am limited to 4 CPU cores, so with limited resources I have to be judicial how a model scales to train and test if I were to run it again. Then I also must consider on whether a marginal increase in score for the model is worth the additional time to train the model (performance of the model) and the graphs should better show these concepts.

Dummy Classifier – Baseline Classifier

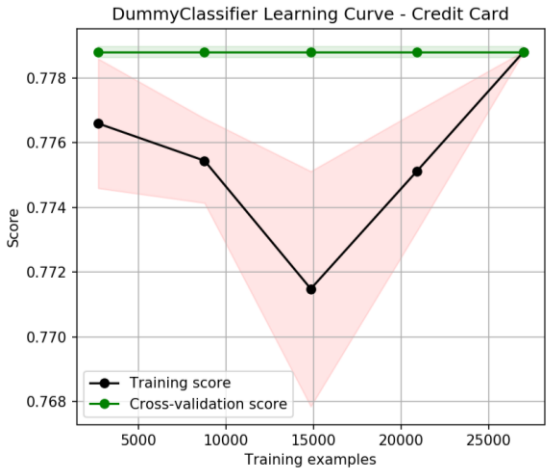
1. Poker Dataset

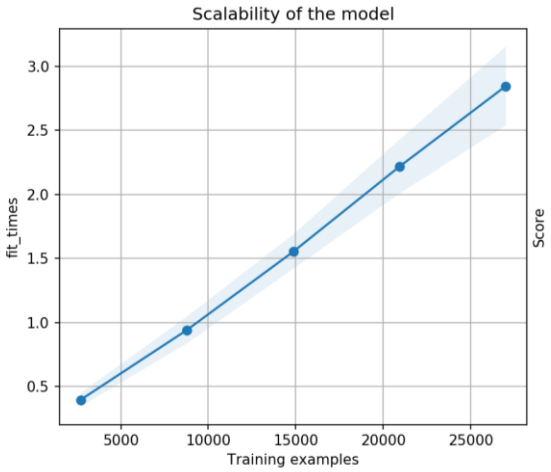
The DummyClassifier achieved an accuracy of 63% and an F1 score of 8%. By analyzing the results for the Dummy picks mainly a result of Nothing in Hand or a One Pair which are the most common poker hands (92% of all hands). The Dummy Classifier just classifies the next Poker Hand based on the previous seen.



2. Credit Card Dataset

Here, the Dummy Classifier achieved a baseline accuracy of 78% and an F1 score of 44%, which is a high bar to beat by the other models. This accuracy score concurs with the amount of true negative samples in the dataset of 78%, which is how it can achieve a high accuracy. Therefore, F1 score might be a better metric to evaluate this dataset.





Decision Trees

This type of model will try to divide the dataset with attributes that gives the most information about splitting the dataset with the maximum Gini gain which is like the information gain. The tree can become very complex and therefore it is pruned by limiting the number of features the tree may decide on or the depth of the tree can also be limited. This helps control for overfitting the training data so the model can be used more confidently on data the model has not seen. The hyperparameters I initially tested this model with are max_features, max_depth, and min_samples_leaf. These should prune the tree sufficiently to generalize the model.

1. Poker Dataset

The results by optimizing for either accuracy or F1 score yielded the same results. The optimal tree had the following hyperparameters: max_depth of 24, max_features of 9, min_samples_leaf of 1 and no preprocessing. Then a classification report is generated just for my test set and shown below.

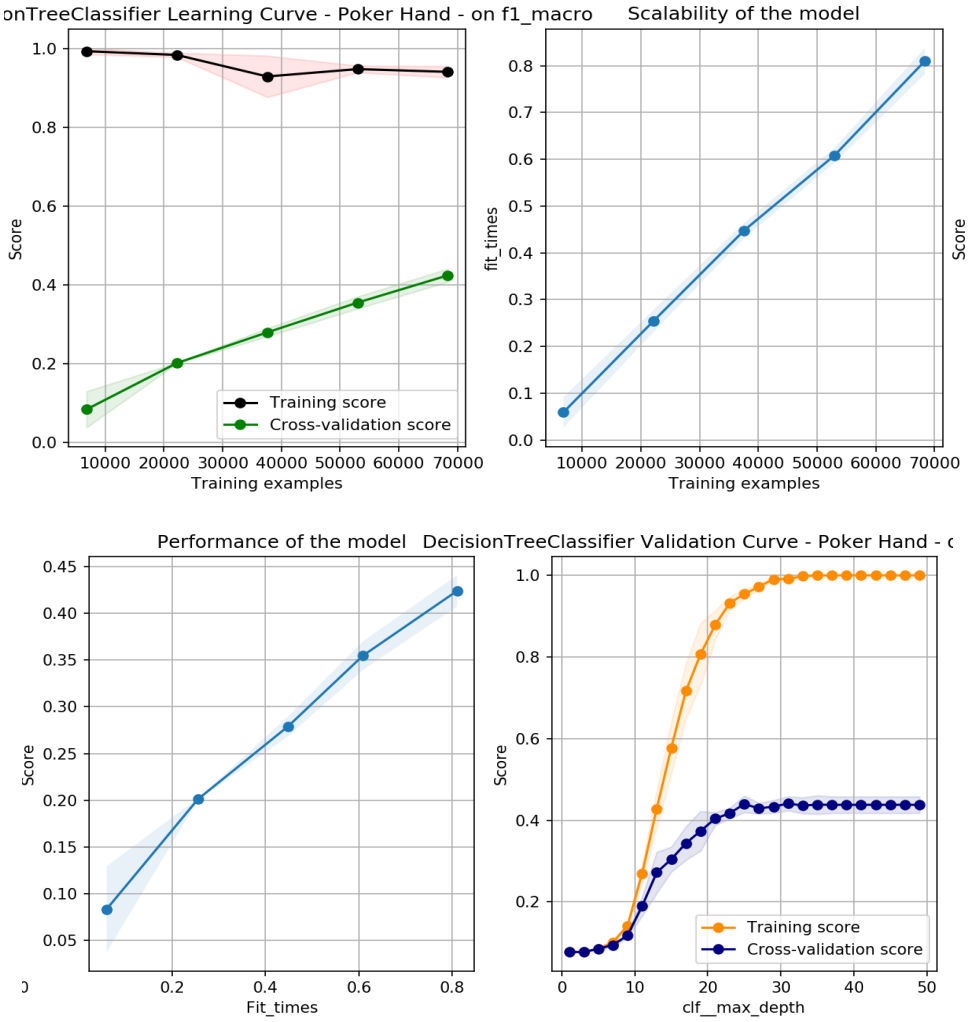
Decision Tree on the Poker Hand Dataset Classification Report

Class	Precision	Recall	F1-Score	Support
0	0.0627	0.450	0.11	678
1	0.928	0.333	0.491	18021
2	0.107	0.574	0.180	891
3	0.107	0.564	0.180	590
4	0.147	0.559	0.232	136
5	0.107	0.560	0.179	91
6	0.134	0.423	0.203	78
7	0.244	0.667	0.357	15
8	0.091	1.0	0.167	1
9	0.0	0.0	0.0	0

Accuracy			0.358	
Macro-Avg	0.193	0.513	0.210	20501
Weighted-Avg	0.827	0.358	0.451	20501

This classification report shows that a more complex story than just using the cross-validated accuracy and F1 score. If a random classifier were to try and predict the class one would expect a 1/10 chance of accuracy and this had a better accuracy with 35.8%. Then, in terms of the precision and recall both the macro-average and weighted-average resulted better than the Dummy Classifier. Meaning that the model tried to predict on certain rules instead of just picking the most likely (class 0 or 1). Note that the precision tries to say if the proportion of positives are positive and recall notes of the actual positives are correctly classified. Note that for this problem predicted 92.8% of the Pair Hands (Class 1) are Pair hands. But the recall for that class is particularly low where only 33.3% of the total Pair Hands were classified. I would argue that this decision tree model did an ok job at trying to figure out the rules of Poker but just heavily missed classifying hands that had no value (Class 0) and obviously the rarer Royal and Straight Flushes.

In terms of evaluating the model on how well it learned the following curves were generated. The learning curve is interesting because it shows that the training accuracy was close to perfect, but the cross-validation accuracy kept steadily increasing. This shows that even more samples of the dataset could be used to get a nearly linear increase of a cross-validation score. But it comes at a high cost of times to fit as the model linearly scales. These two concepts are illustrated in the performance curve. The hyperparameter curve shows that increasing the depth and not pruning a tree creates a divergence in training and cross-validation scores which shows that around a value of max_depth = 10 the model starts to overfit. This makes sense because there are of course only 10 features to poker and the rules are not too complex.



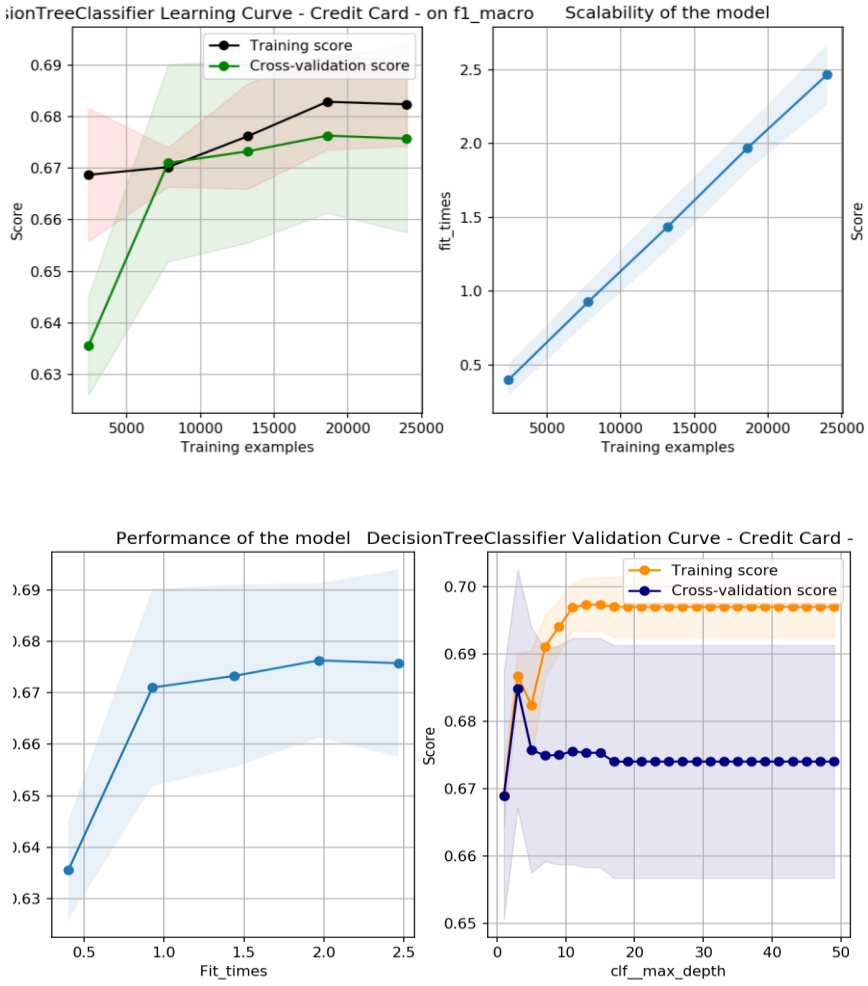
2. Credit Card Dataset

Once again, the results of optimizing for accuracy and F1 score yielded the same results. The optimum Decision Tree found was using the Gini method, a max_depth of 5, max_features of 23, and a min_samples_leaf of 51. This was found with pipeline of transforming the dataset with the Yeo-Johnson method, and robustly scaling the dataset. The cross-validation accuracy was 82% and an F1 score of 68%, which both are an improvement of a Dummy Classifier. It had a precision of 75% and a recall of 65%.

The following is the confusion matrix from this model. Note that in the context of this problem the costliest decision generated by the model is predicting negative and the account defaulting. At least with predicting positive a trained risk manager could delve deeper into the situation and classify it as a false negative.

	True Negative	True Positive
Predicted Negative	4434	239
Predicted Positive	858	469

The actual results of the model are very intriguing because just as before the model scales in a linear fashion, but the scores plateau after several training examples and therefore the performance also plateaus. The max_depth of the decision tree is also curious because it seems that any more than 2-3 nodes in the decision tree and the model overfits. Curiously, we could print out the decision tree and see which attributes are the most important.



Neural Networks

The neural network that I decided to use is a multi-layer perceptron (MLP) classifier that optimizes across its neurons using stochastic gradient descent. I chose to optimize across the hidden layer sizes and experimenting with different number of neurons and layers. Then when validating, I chose to visualize how the alpha penalty term would affect the scores.

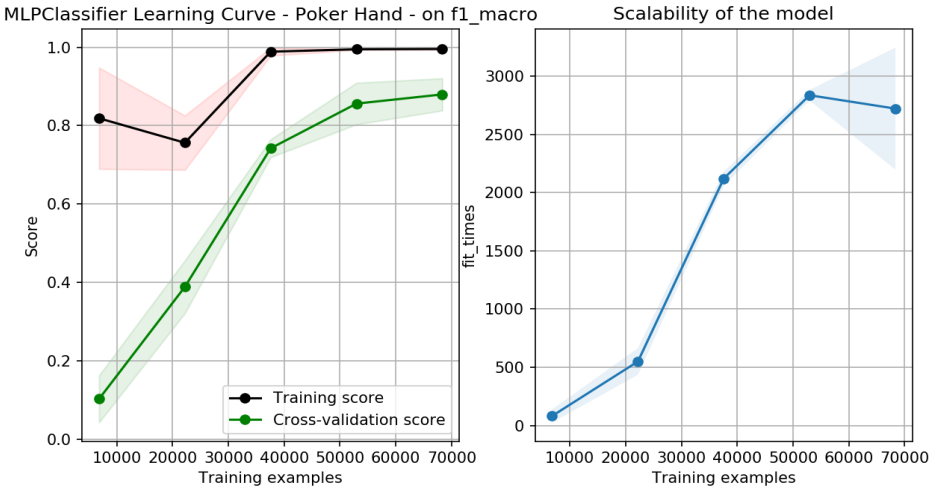
1. Poker Dataset

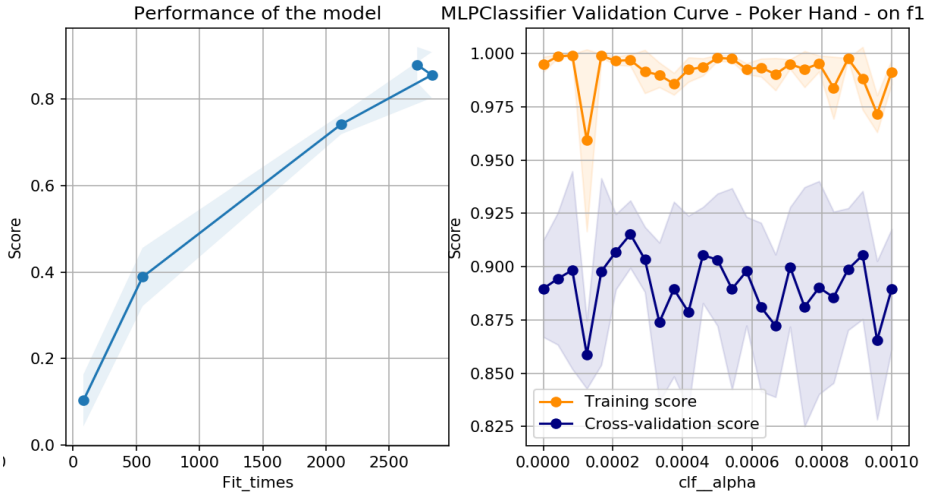
For this model, the resulting metrics are found in the classification report below. The parameters that were found to be best were 200 neurons and no hidden layers, using stochastic gradient descent, an activation function of the rectified linear unit function and an alpha of 1e-05. The accuracy of 96% and an F1 score of 75% are significantly better than the Dummy and Decision tree. Analyzing the results, it seems that it also had trouble correctly identifying No Poker Hand or the Royal Flush. This is probably due to the low training samples for the two categories especially in comparison to the One Pair. A training dataset with more balance would probably achieve an even higher accuracy and F1 score.

MLP Classifier on the Poker Hand Dataset Classification Report

Accuracy			0.959	
Macro-Avg	0.698	0.887	0.752	20501
Weighted-Avg	0.971	0.959	0.963	20501

One can see the improvement of scores with respect to training sizes is significant at around 40,000 samples. The model also scales in a curious manner as it is nearly exponential until it plateaus which is probably due to my local computer's limited resources. The validation of the alpha parameter and its effect on the score is very noisy and seems to have no true effect on the score at all.





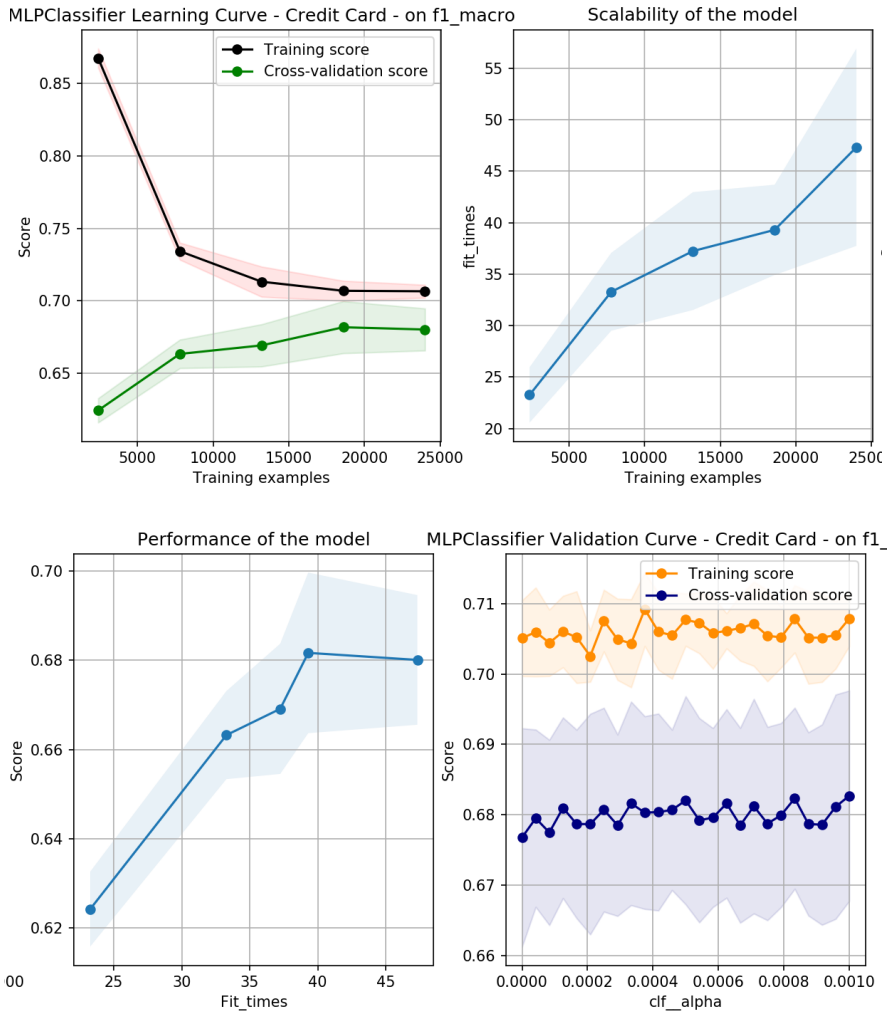
2. Credit Card Dataset

The results of training an optimal neural network are found below. The best neural network found was with 50 neurons, using stochastic gradient descent, a rectified linear unit function and an alpha of 1e-05. Just like before heavy preprocessing was done to transform and scale the dataset beforehand. The results were an accuracy of 82%, F1 score of 68%, average recall of 66%, and precision of 75%. The confusion matrix is found below. This model was significantly better than the Dummy classifier and the Decision Tree but at the cost of being tougher to explain how it predicts the outcomes and computation costs.

	True Negative	True Positive
Predicted Negative	2216	120
Predicted Positive	420	244

The training and cross-validation scoring graphs begin to converge as more samples are added until about 20,000 samples which is approximately what I used to train with. The model also does not scale that well and minimal performance gains are found around that level. Again, the alpha parameter seems to be not as important and is noisy. I should have used a much larger space to better see the parameter's effects. It is interesting to note that the performance gain for the credit card dataset was not as large as the performance gain for the poker set. In the poker

dataset it is completely worth using a neural network as opposed to most models, whereas for this dataset I am not convinced that the complexity is worth the accuracy, recall, and precision.



Ensemble Learners

The ensemble learners I chose to optimize were either using the Adaboost algorithm with a Decision Tree or a Random Forest Classifier. I varied the number of estimators used, learning rate or the depth of the trees and minimum samples contained at each leaf. I chose to evaluate both because I was curious on comparing their performance and advantages and disadvantages of very similar models.

1. Poker Dataset

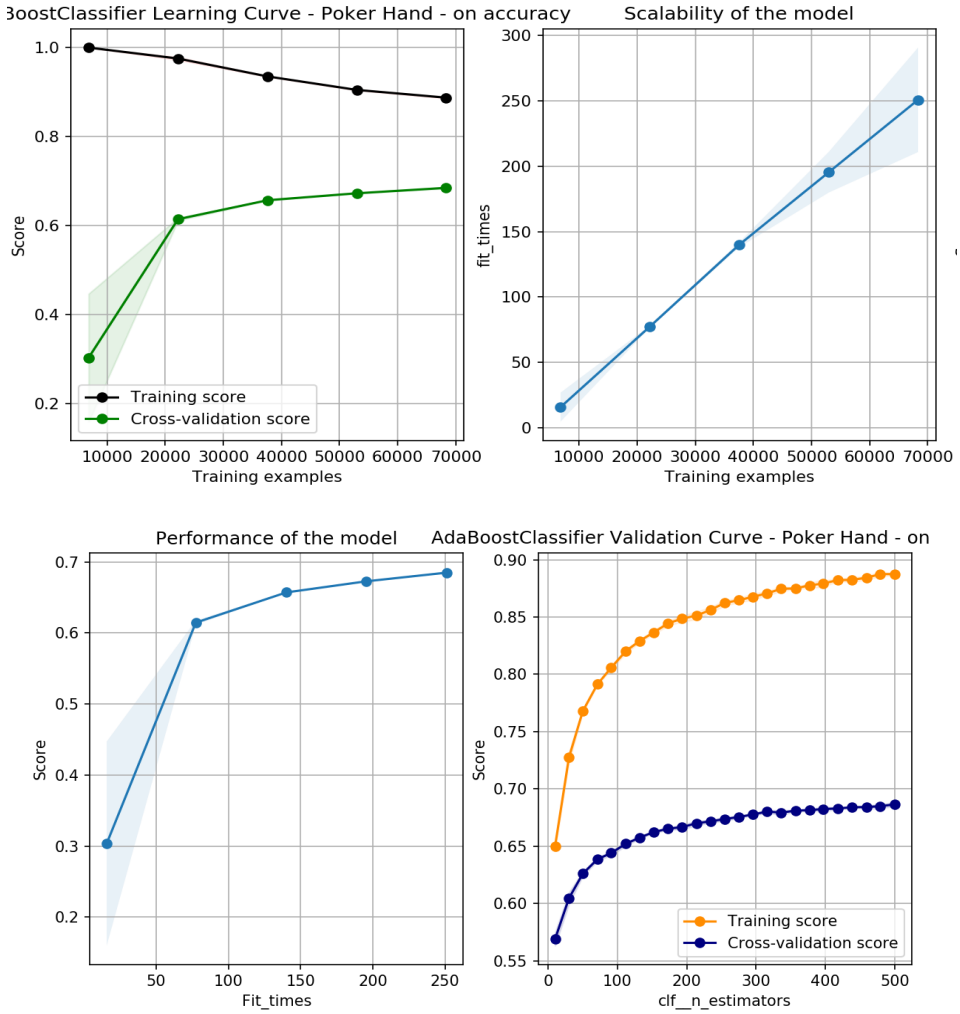
For this dataset I chose to use the AdaBoost algorithm along with the optimized Decision Tree classifier found previously. Therefore, I found an optimal learning rate of 1.0 and 500 estimators along with a decision tree with a max_depth of 10 and min_samples_leaf of 1. The metrics found below reflect an improvement over the weak learner we found which is what is expected.

Ada Boost with Decision Tree on the Poker Hand Dataset Classification Report

Accuracy			0.526	
Macro-Avg	0.674	0.564	0.544	20501
Weighted-Avg	0.846	0.526	0.623	20501

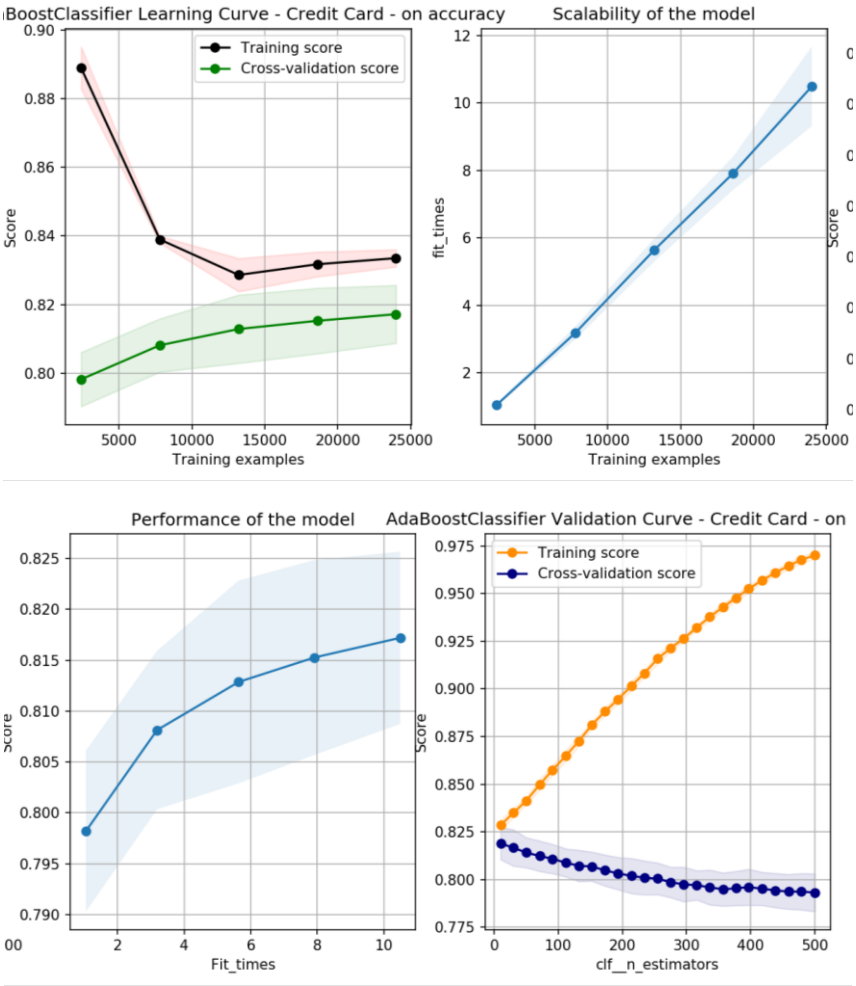
This algorithm made a big improvement on the decision tree as it more accurately classified rarer events and One Pair classes. The exception is the No Hand and Royal Flush, which the algorithm could be improved by increasing the number of training samples for each class. It does not make an improvement on the Neural Network Model but is trained faster

and reaches peak performance faster than the Neural Network. One can see that the cross-validation score plateaus after 20,000 samples along with the plateau of the performance. On one hand, the validation curve is a good visualization how the model's score improves with the more estimators and one can clearly see the diminishing returns.



2. Credit Card Dataset

For this problem I got the same accuracy and F1 scores for the Ada Boost algorithm method and Random Forest Classifier. I will present the Ada Boost algorithm because it ran faster than the Random Forest. The preprocessing steps taken were to transform the data with the Yeo-Johnson method to normalize it and Robustly Scale the data. The learning rate of 0.5 and number of estimators used was also just 25. The decision tree used had a max_depth of 5 and min_samples_leaf of 51. The cross-validation accuracy was found to be 82%, F1 score of 68%, precision of 75%, and recall of 64%. This method had the exact same statistics as the Decision Tree method. The confusion matrix can be found below. This was a bit shocking that it cannot be improved and is highlighted in the validation curve, where, as the number of estimators increase the worse the performance gets. Therefore, between a decision tree or this algorithm I would choose the tree for its simplicity.

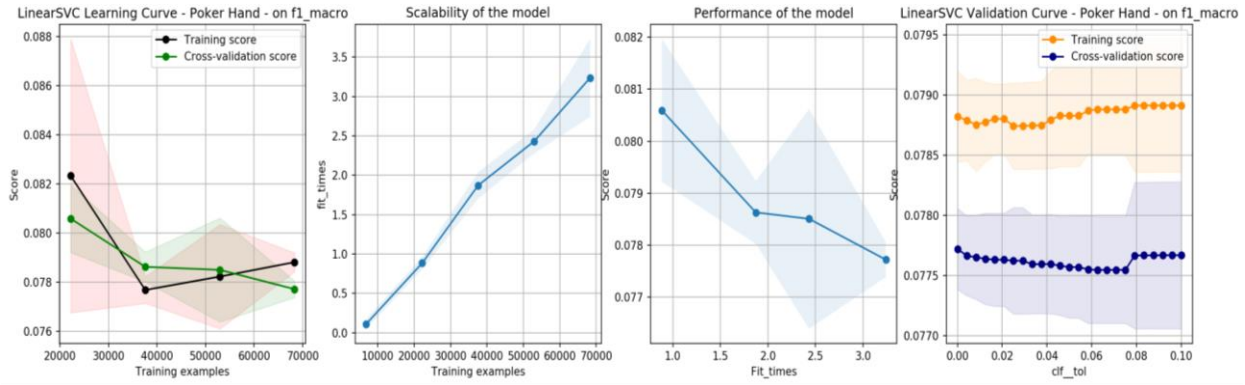


Support Vector Machines

The support vector machine models were the trickiest models to tune in my opinion. I had trouble tuning a linear and a radial basis function kernel. I decided to tune the gamma or kernel coefficient and tolerance for stopping criterion. Heavy preprocessing had to be done, I investigated scaling methods and using the Nystroem kernel method to speed up the model. Note tolerance for stopping criteria does not seem to affect the model much like the Neural Network.

1. Poker Dataset

Strangely, this model with a penalty of l2 and squared hinge with a preprocessing step of robust scaling got worse with the more samples it was fed. The accuracy was also very poor with 13% at best, meaning this was worse than the dummy classifier.



2. Credit Card Dataset

This model achieved similar results with more training time than any other dataset. It was a linear kernel with an l2 penalty and it achieved an 82% accuracy.

K-Nearest Neighbors

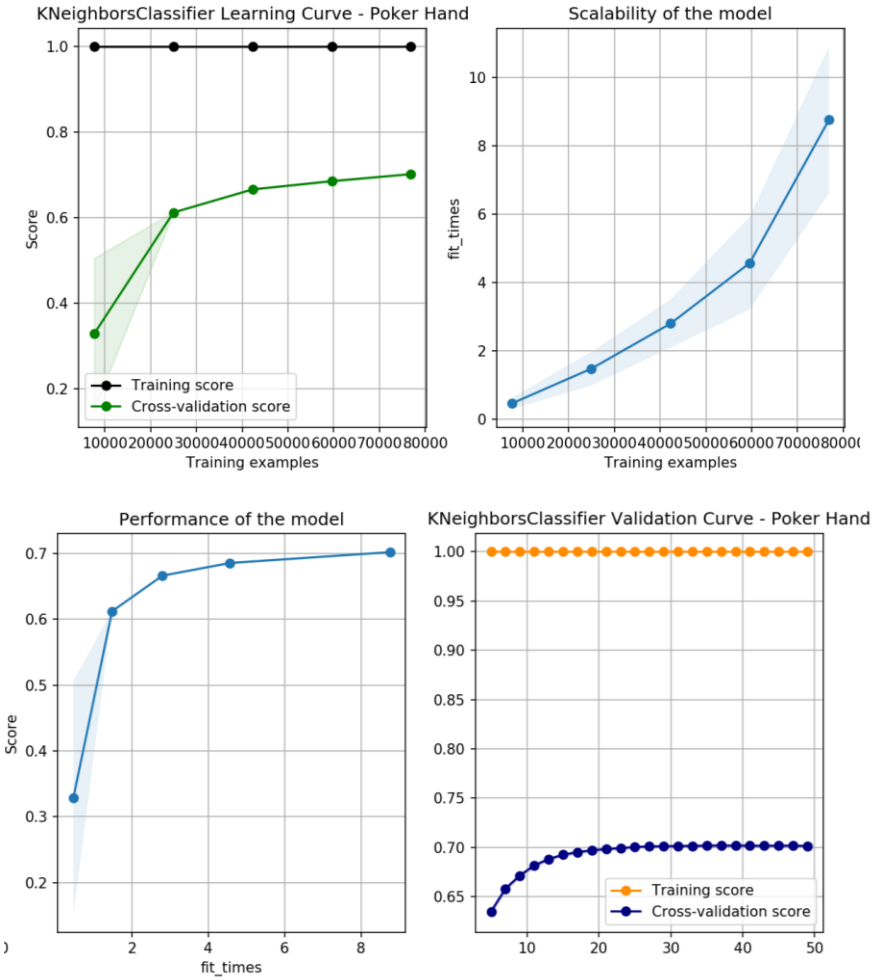
The nearest-neighbors method was optimized using different distance calculations and the number of neighbors used to fit the data. Notice that the training score is perfect for this model which makes sense.

1. Poker Dataset

For this dataset, I found that the optimal number of neighbors is 45 using distance formula as opposed to a uniform one. The metrics achieved can be found below. It should be noted that this simplistic model performed similarly to a decision tree with an accuracy of 31.5%, and F1 with 32.4%, but worse than the neural network. It is understandably slow to train so it scales poorly in that sense although it is faster to query. The number of neighbors' plateaus around 18 neighbors.

KNN on the Poker Hand Dataset Classification Report

Accuracy			0.315	
Macro-Avg	0.412	0.436	0.324	20501
Weighted-Avg	0.826	0.315	0.406	20501



2. Credit Card Dataset

Using 32 neighbors and a distance weighted function I found the optimal KNN led to an accuracy of 81% and F1 of 67%, which is about the same as the Adaboost function. It is faster to query and slower to train so the appropriate model

must be selected depending on the use/priority. It seems like most models worked up to the threshold of around 82% accuracy and 69% F1 score. Note that this model requires very little training samples and the curse of dimensionality is not seen.

	True Negative	True Positive
Predicted Negative	2220	116
Predicted Positive	442	222