Assignment 1

Leondis Evans
levans43@gatech.edu

1 INTRODUCION

The purpose of this paper is to summarize the finding of my experiments on a Mortgage Loan approval dataset and a dataset to predict chances of a stroke.

2 DATASETS

Mortgage Loan Data

This dataset is a collection of common data points used to determine if a loan request should be approved or denied.

What makes this an interesting dataset is the challenge to determine the proper weight of majority features without making minor features non relevant. A perfect example of this would be credit history. Credit history is a majority factor that is predictive of how a person is likely to manage current and future financial responsibilities.

2.1.1 Data Cleaning

The strategy used to account for missing features was to remove the entire data row from the dataset. The justification for this is it is reasonable to assume that we can enforce the requirement of none of the required feature values to be missing. While it is a valid criticism to mention Credit history could be a missing value in a real-world example the possibility does not exist in this dataset.

2.1.2 Data Transformation

The loan dataset initially was an unbalanced dataset with most of the data for approved loans. To transform the dataset to a balanced dataset Oversampling was applied to nonapproved loans.

Stroke Data

This dataset is a collection of a common health condition to predate the likelihood of a stroke occurring.

Why this data is interesting is because it has a feature which when combined with great accuracy can predict a stroke but also has anomaly conditions outside of those ranges that also point to a high probability of a stroke.

The challenge with this dataset is how do you design a solution that favors the typical features ranges, without eliminating the anomaly conditions. Another challenge is how do you obtain an efficient number of anomaly samples in order to train a model?

2.1.3 Data Cleaning

The strategy used to account for missing features was to remove the entire data row from the dataset.

2.1.4 Data Transformation

The stroke dataset initially was an unbalanced dataset with most of the data for no stroke. To transform the dataset to a balanced dataset Oversampling was applied to the data indicating a possibility of a stroke.

3 EXPERIMENTS

Metrics

3.1.1 *Learning Curve*

To determine the learning curve for each algorithm I plotted the results using Scikit Learning Curve model. The learning curve was implemented using k-fold cross validation with a k value of 5 and using accuracy as the scoring metric.

3.1.2 *Validation Curve*

To determine the training and test score from turning the algorithm hyper parameters I used Scikit Validation Curve model. The validation curve was implemented using k-fold cross validation with a k value of 5 and using accuracy as the scoring metric.

Decision Tree

From experimenting with the decisions tree, the two most influential hyper parameters I discovered are max depth of the tree and the minimal samples requested to make a new leaf.

Looking at the data from both loan and stroke dataset below it is clear the model only started to learn when most of the dataset is trained. I suspect this is due to the following factors

1. Oversampling has introduced bias. The model for when training on loan data makes the false assumption that if a person credit history is bad, they should never be approved for a loan and on the stroke dataset it assumes if the age is less than or equal to 47.5 a stroke is not possible.
2. Both datasets could benefit from adjusting the weights of features instead of making them all equal to improve the accuracy.
3. Both datasets could benefit from adjusting the weights of features instead of making them all equal to improve the accuracy.

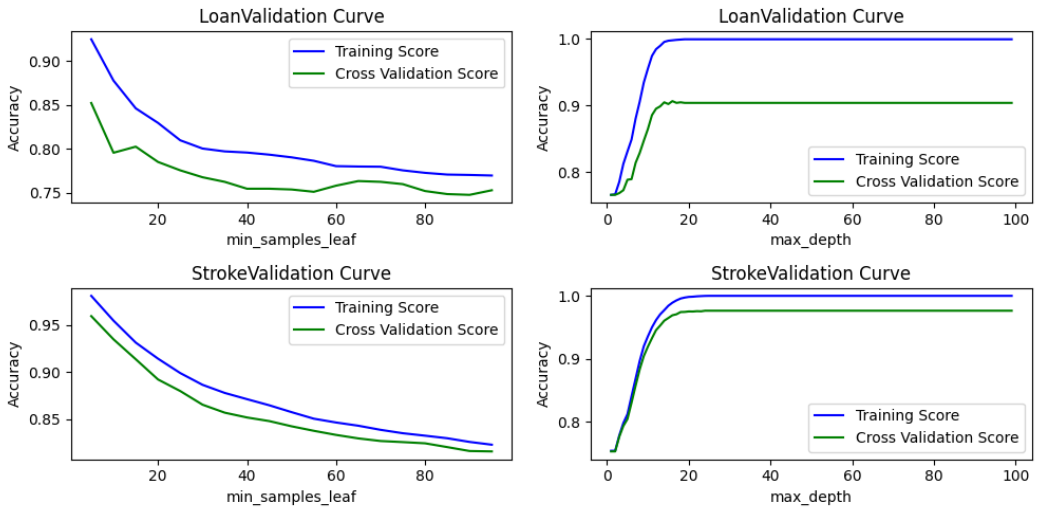
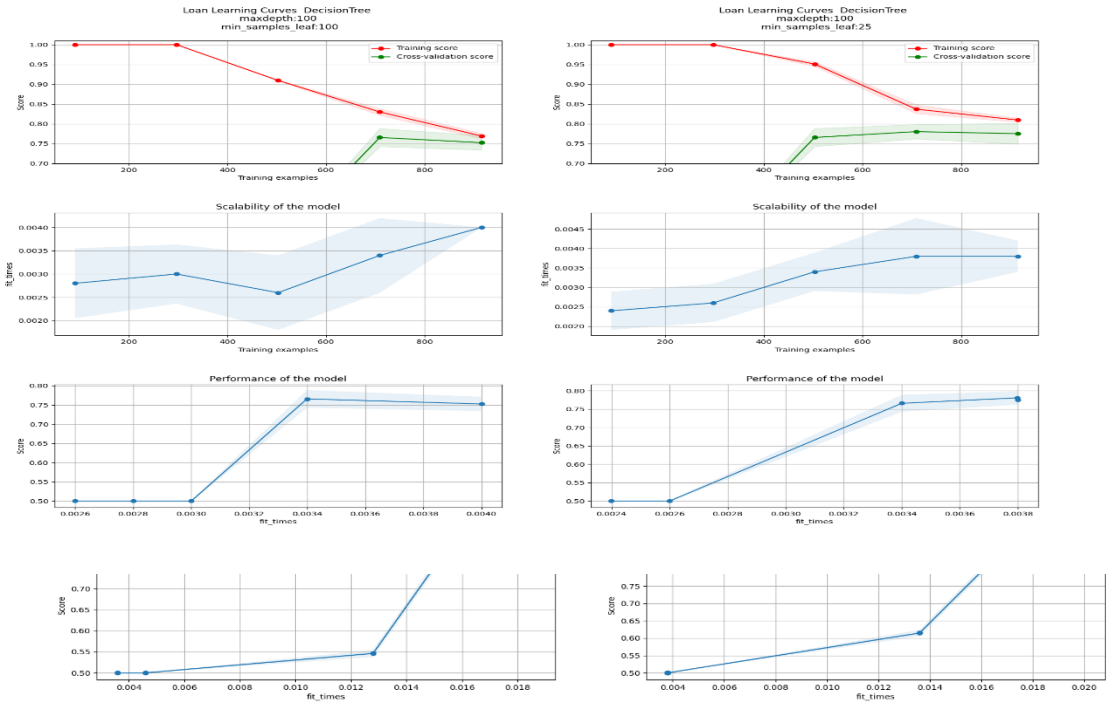


Figure 1 — Loan Data

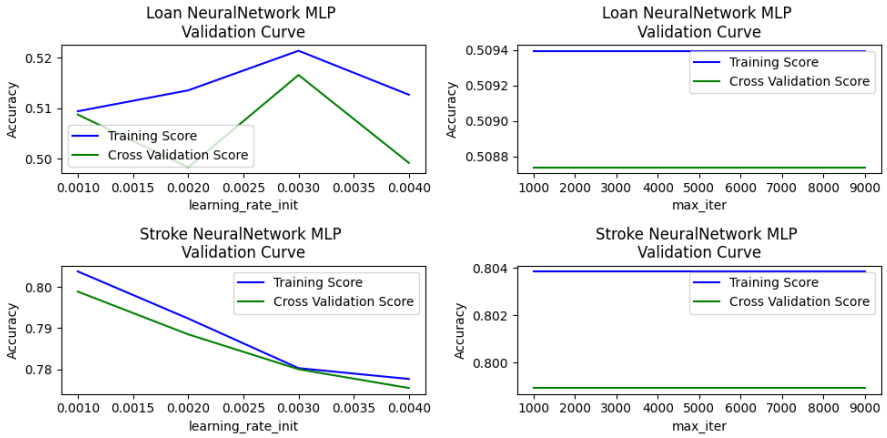


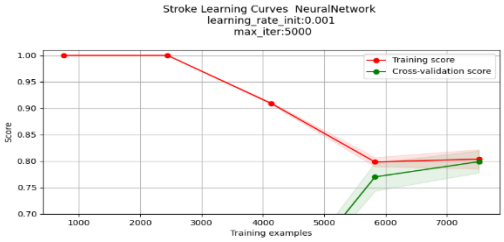
Neural Networks

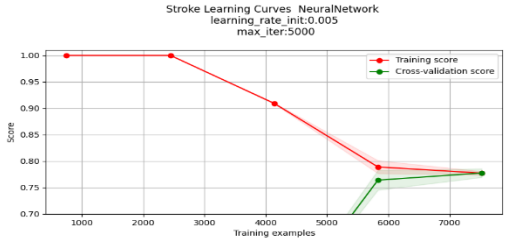
I decided to use a Multilayer perception for my Neural Network implementation using scikits's MLPClassifier classifier. The MLPClassifer is a Muti-layer Perceptron classifier, and I am using a stochastic gradient descent as the solver for log-loss function to (Scikit-learn, n.d.)

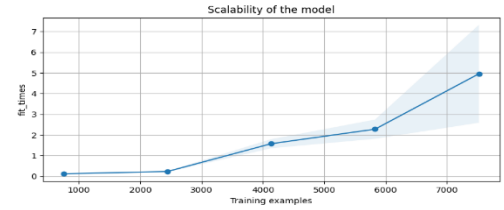
The results I got resulted in an unfitted model. Due to computation cost I had to end the model after 5k iterations, and I suspect this a major cause of the unfitting model. I plan to try experimenting with using a StandardScaler to help reduce

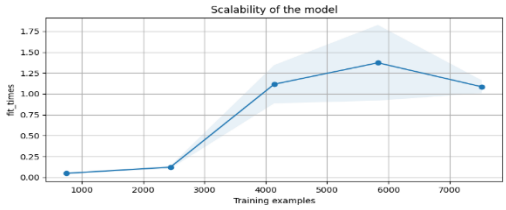
the iterations required for the model to reach convergence. Additional scikit has GridSearchCV to help fine tune hyperparameters.

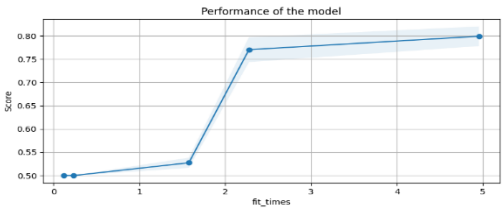


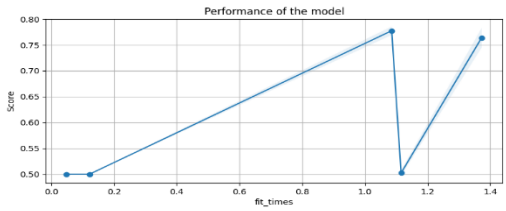

Stroke Learning Curves NeuralNetwork
learning_rate_init:0.001
max_iter:5000
Training score
Cross-validation score
Score
Training examples

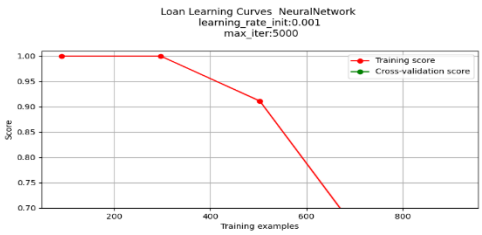

Stroke Learning Curves NeuralNetwork
learning_rate_init:0.005
max_iter:5000
Training score
Cross-validation score
Score
Training examples

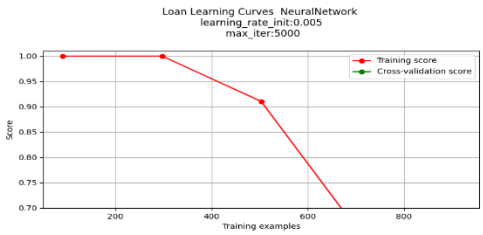

Scalability of the model
fit_times
Training examples

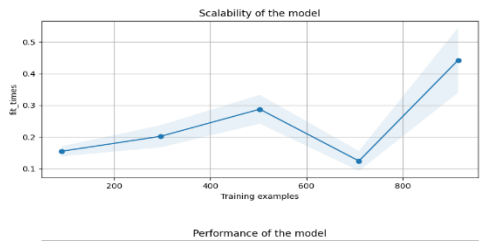

Scalability of the model
fit_times
Training examples

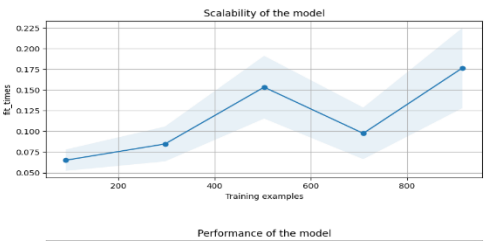

Performance of the model
Score
fit_times

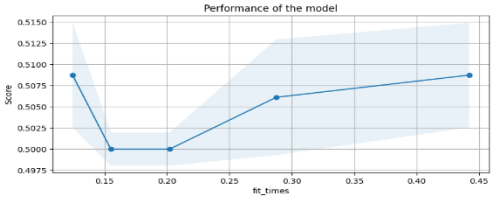

Performance of the model
Score
fit_times

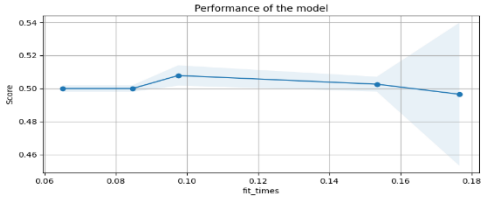

Loan Learning Curves NeuralNetwork
learning_rate_init:0.001
max_iter:5000
Training score
Cross-validation score
Score
Training examples


Loan Learning Curves NeuralNetwork
learning_rate_init:0.005
max_iter:5000
Training score
Cross-validation score
Score
Training examples


Scalability of the model
fit_times
Training examples

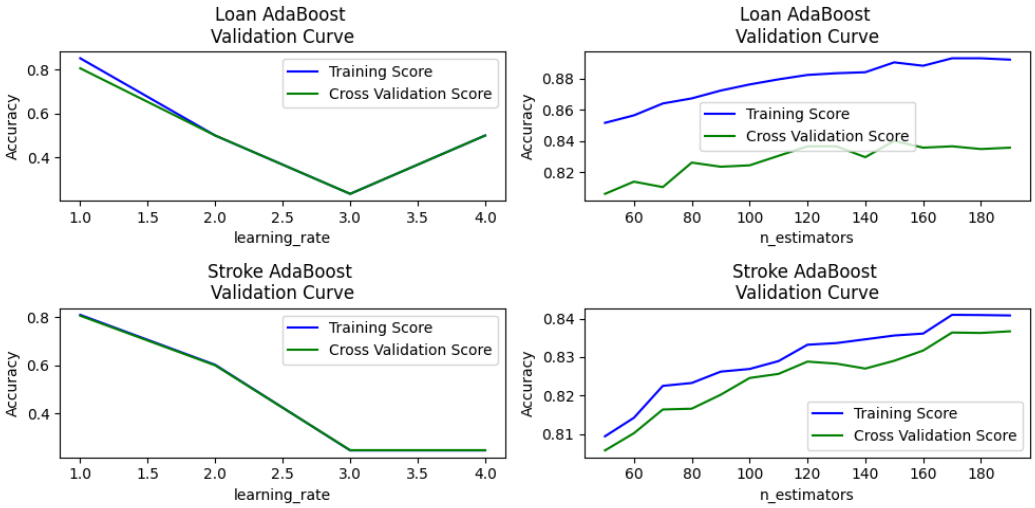

Scalability of the model
fit_times
Training examples

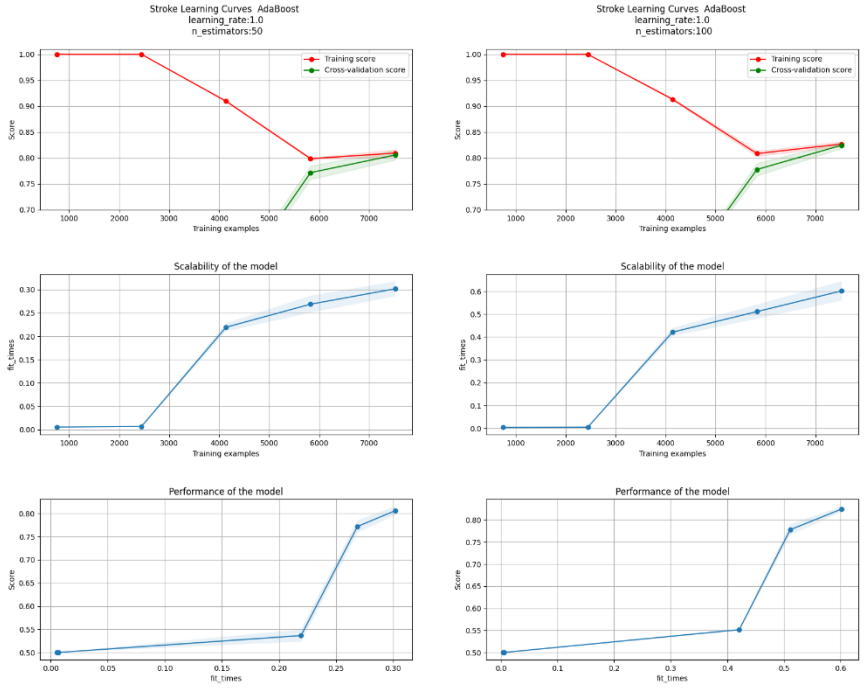

Performance of the model
Score
fit_times

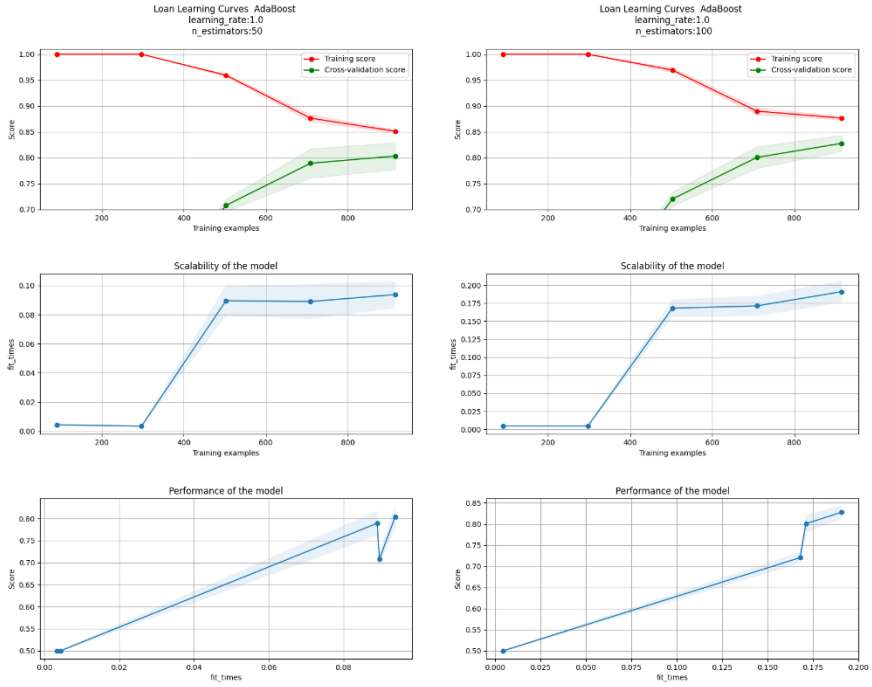

Performance of the model
Score
fit_times

Boosting

For the boosting algorithm I decided to use an AdaBoost model in scikit-learn known as AdaBoostClassifier. The AdaBoostClassifier implements the algorithm known as AdaBoost-SAMME (AdaBoostClassifier, n.d.) The learning_rate parameter reached it optimal selection at a weight of two for each dataset.

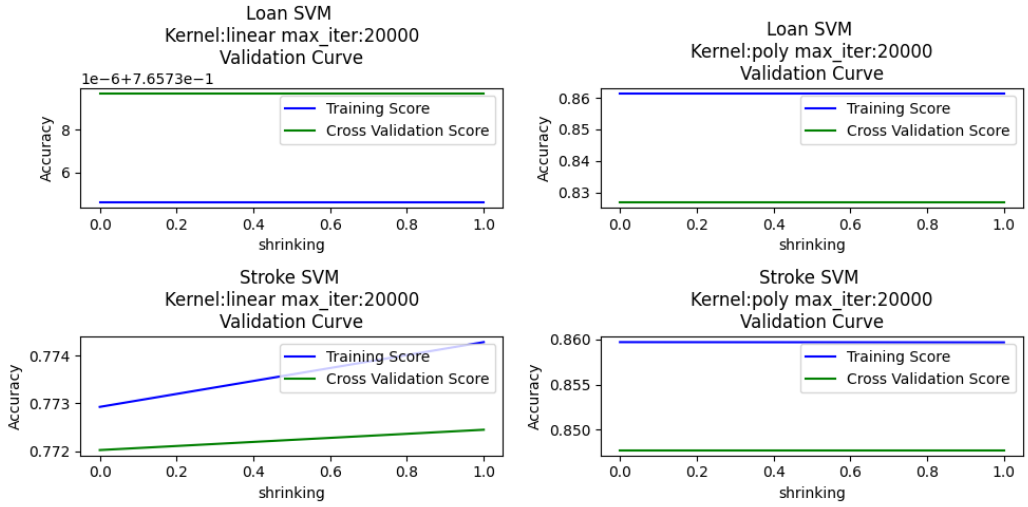


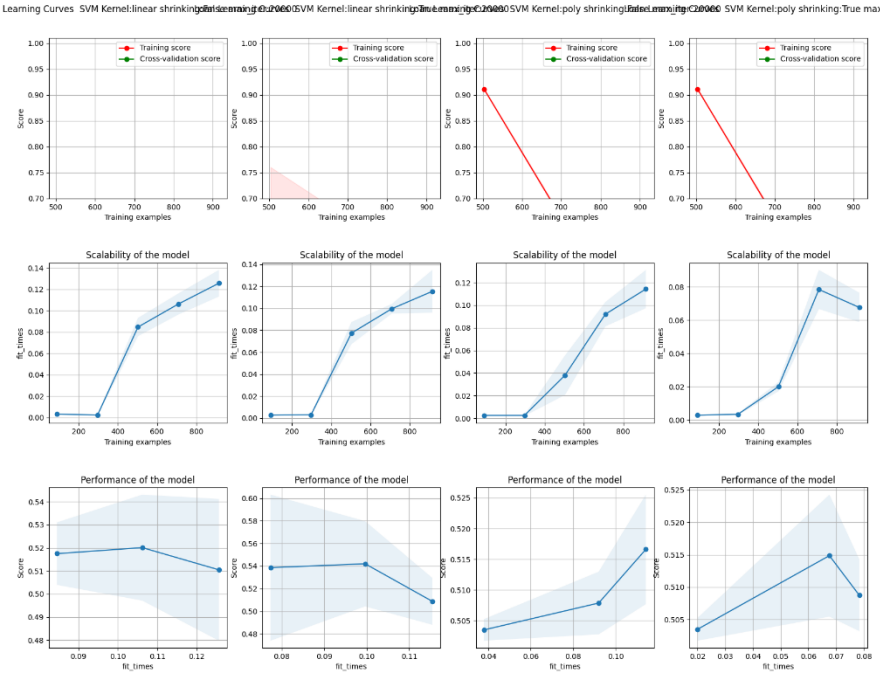


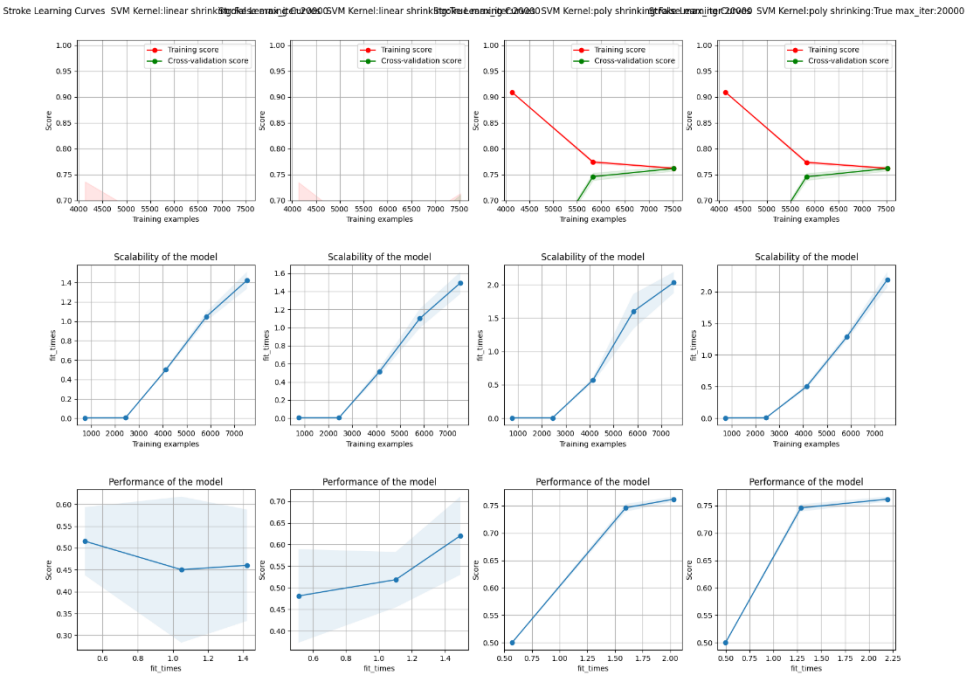


Support Vector Machines

For my SVM implementation I used a C-Support Vector Classification, with a linear and polynomial kernel functions. Both kernel functions resulted in an unfitted model, and I suspect the reason is due to not adjusting the weights of the features.

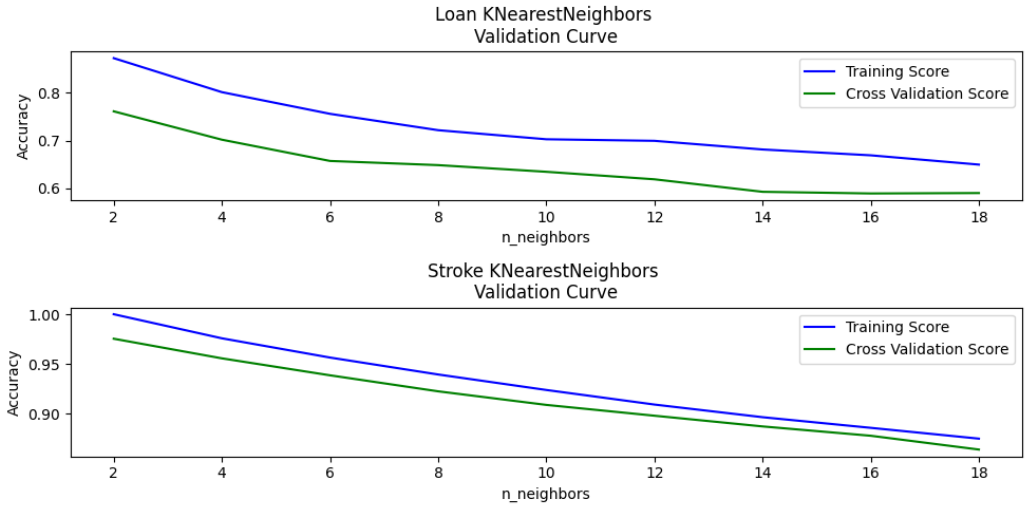


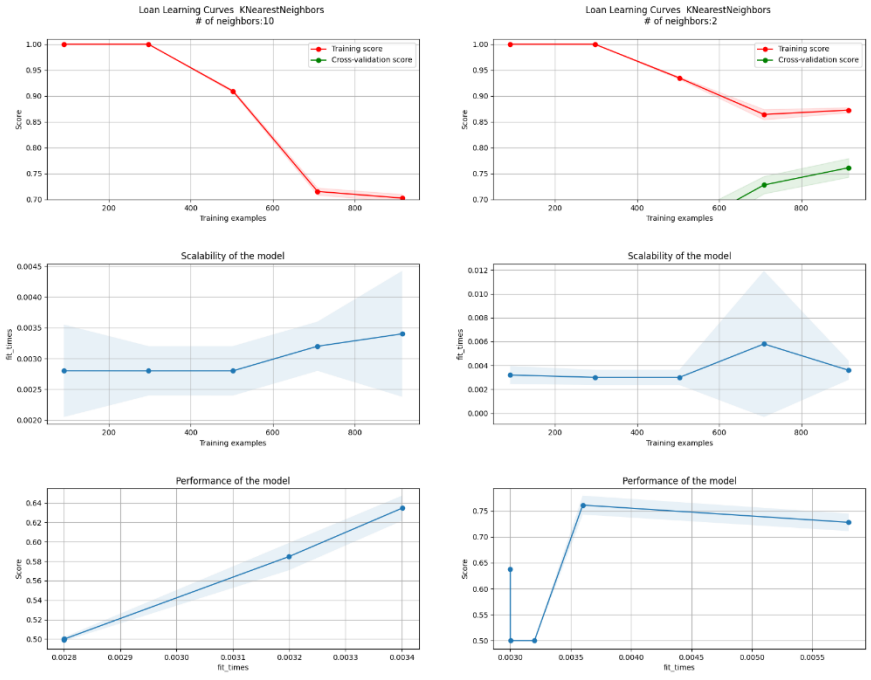

Loan Learning Curves SVM Kernel:linear shrinking:True max_iter:20000
SVM Kernel:linear shrinking:True max_iter:20000
SVM Kernel:poly shrinking:True max_iter:20000
SVM Kernel:poly shrinking:True max_iter:20000
Training score
Cross-validation score
Score
Training examples
Scalability of the model
fit_times
Performance of the model

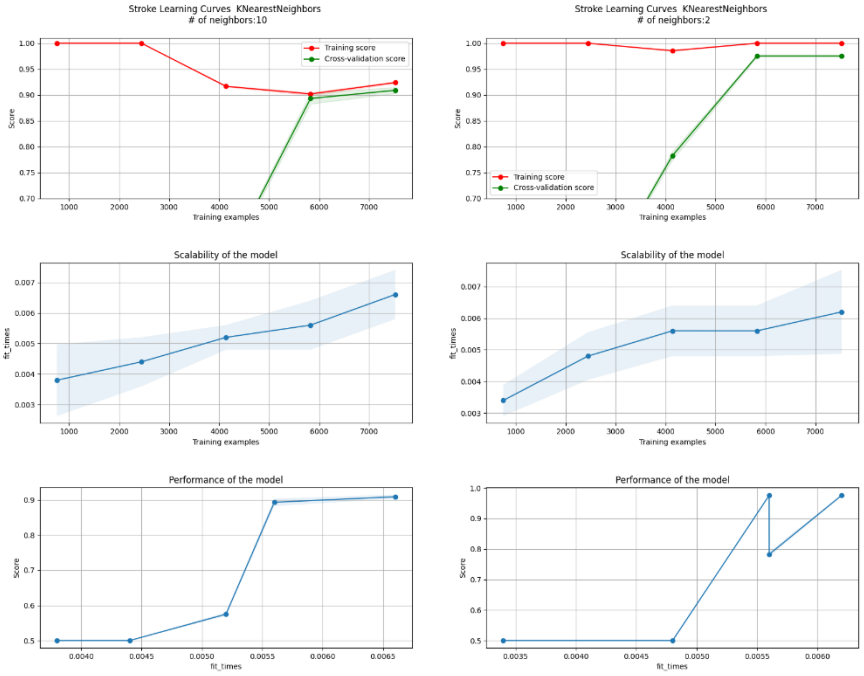

Stroke Learning Curves SVM Kernel:linear shrinking:True max_iter:20000
SVM Kernel:linear shrinking:True max_iter:20000
SVM Kernel:poly shrinking:True max_iter:20000
SVM Kernel:poly shrinking:True max_iter:20000
Training score
Cross-validation score
Score
Training examples
Scalability of the model
fit_times
Performance of the model

KNearesrNeighbors

The k-nearest neighbors validation shows a good fitting model, but this is a false positive due to the oversampling we performed on both datasets.







4 SUMMARY

Each of the five learning algorithms can provide an answer but not necessarily the answer we are looking for. Each requires regression testing to valid the selected hyperparameters and human observations to determine the output is the correct answer.

5 REFERENCES

AdaBoostClassifier. (n.d.). Retrieved from Scikit-learn: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html

Scikit-learn. (n.d.). Retrieved from Scikit-learn: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html#sklearn.neural_network.MLPClassifier