Jose Luis Sanchez

Dr. Razvan Andonie

Computational Intelligence and Machine Learning

June 7, 2022

Project 4 (Individual Project)

For my project four assignment, I decided to program using Python by utilizing Spyder, then transitioning to VSCode, to create and observe the training and testing sets from the character representations. The main packages used were NumPy, pandas, sklearn, and mlxtend. Below you will find the visualization of the character matrix representations, as well as the results found by observing the effectiveness of the classifiers.

Character Matrices

The character matrix representations were designed to be more minimalistic to avoid any complications where the classifier would be unable to distinguish characters due to the noise generated. Most of the characters were designed to be simple representations with more white space than dark space.

A representation of several character matrices

Please note that the image above is not actually how the data is formed, as the data was made by utilizing binary values to represent a white pixel (0) and a black pixel (1). As for the noise generation, I have decided to utilize three different noise rates for this experiment: 20, 50, and 85 percent noise. Also, for the sample size, the different sizes that will be used for this experiment will be 100 samples (nine noisy variants for each prototype), 200 samples (19 noisy variants for each prototype), and 300 samples (29 noisy variants for each prototype).

Finding the Optimal Parameters

Utilizing the GridSearchCV function given by sklearn, I was able to find the optimal parameters for each classifier to get the most out of the classifiers at each given combination of sample size and noise rate. Continuing further, I will be referring to this combination as a permutation. Here are the resulting parameters used, but please note that to reduce the amount of permutation and parameter combinations (36 total possible optimal parameters), I will try to reduce the number of possible optimal parameters to two optimal parameters for each classifier, where these optimal

Jose Luis Sanchez

Dr. Razvan Andonie

Computational Intelligence and Machine Learning

June 7, 2022

parameters are for N samples of 10 and 30 and noise rate of 20 percent and 85 percent. This should reduce the number of 36 down to 8 optimal parameters (my code does utilize GridSearchCV to find the optimal parameters for each permutation though).

Classifier	Permutation	Best Parameters
Multilayer Perceptron	Sample: 10/100 Total Noise Rate: 20	'activation': 'tanh' 'hidden_layer_sizes': (25, 15) ‘learning_rate_init’: 0.5 'max_iter': 150 'solver': 'sgd'
Multilayer Perceptron	Sample: 30/300 Total Noise Rate: 85	'activation': 'tanh' 'hidden_layer_sizes': (25) ‘learning_rate_init’: 0.5 'max_iter': 50 'solver': 'sgd'
k-Nearest Neighbor	Sample: 10/100 Total Noise Rate: 20	'n_neighbors': 1 ‘metric’: ‘manhattan’
k-Nearest Neighbor	Sample: 30/300 Total Noise Rate: 85	'n_neighbors': 3 ‘metric’: ‘euclidean’
SVM	Sample: 10/100 Total Noise Rate: 20	'C': 10 'gamma': 0.001 'kernel': 'RBF'
SVM	Sample: 30/300 Total Noise Rate: 85	'C': 10 'gamma': 0.001 'kernel': 'RBF'
Random Forest	Sample: 10/100 Total Noise Rate: 20	'max_depth': 25 'min_samples_leaf': 1 'min_samples_split': 2 'n_estimators': 75
Random Forest	Sample: 30/300 Total Noise Rate: 85	'max_depth': None 'min_samples_leaf': 1 'min_samples_split': 2 'n_estimators': 100

Looking at the best parameters from both extremes, multilayer perceptron parameters had only one noticeable change from the two different permutations used. For sample 10 and noise rate 20 (we will use set10noise20 to shorten that) set, the hidden layer size was two, with nodes 25 and 15. However, for the set30noise85 set, the hidden layer nodes were 25. While I do not

Jose Luis Sanchez

Dr. Razvan Andonie

Computational Intelligence and Machine Learning

June 7, 2022

understand fully why the MLP decided to utilize two hidden layers for a smaller set and only one for the larger set, it could be due to the noise rate found in the noisy variants. It could be possible that with more noise generated, the MLP finds using fewer nodes more optimal to avoid overfitting.

For the kNN classifier, it seems that with smaller sample size, the smaller the k neighbors it needs to find a match. This may be due to the noise rate rather than the sample size, as having low noise means that there are closer data points to separate, so the smaller k value is better as there is less bias to know what to classify. Likewise, a larger k value is better for higher noise rates to create a higher bias to avoid confusing some data points with other non-related points. Also, the two permutations are different in the distance metrics they use. For the smaller sample with less noise, it utilized the Manhattan distance, while for the larger sample with more noise, it utilized the Euclidean distance. While I am not 100 percent sure behind the reasoning, it could be that the Manhattan distance is better for smaller datasets as its better at finding outliers, while Euclidean distance is better for larger datasets as outliers would be less frequent as more points are being used in the classification process.

For the SVM classifier, there was no notable difference between the first and last sets used. From what I could understand, the kernels stayed as RBF due to linear kernels being more ideal for text, and since the data used are binary values in a list, then the RBF is better suited to handle this classification. Likewise, the C and gamma parameters are typically used for RBF kernels, so it makes sense that they did not change so much.

Finally, for the random forest classifier, the biggest differences between the two permutations are the max depth size and the n estimators. My thoughts on their parameters being the difference is that by having a smaller dataset, the less max depth the random forest needs to reach its optimal accuracy. On the other hand, with a larger dataset, the more max depth it needs to reach an optimal accuracy, which is why it becomes unrestricted and does not have a max depth. As for the n estimators, the smaller set does not need that many estimators compared to the larger dataset as there is no need for that many trees when the dataset becomes smaller.

Test Result

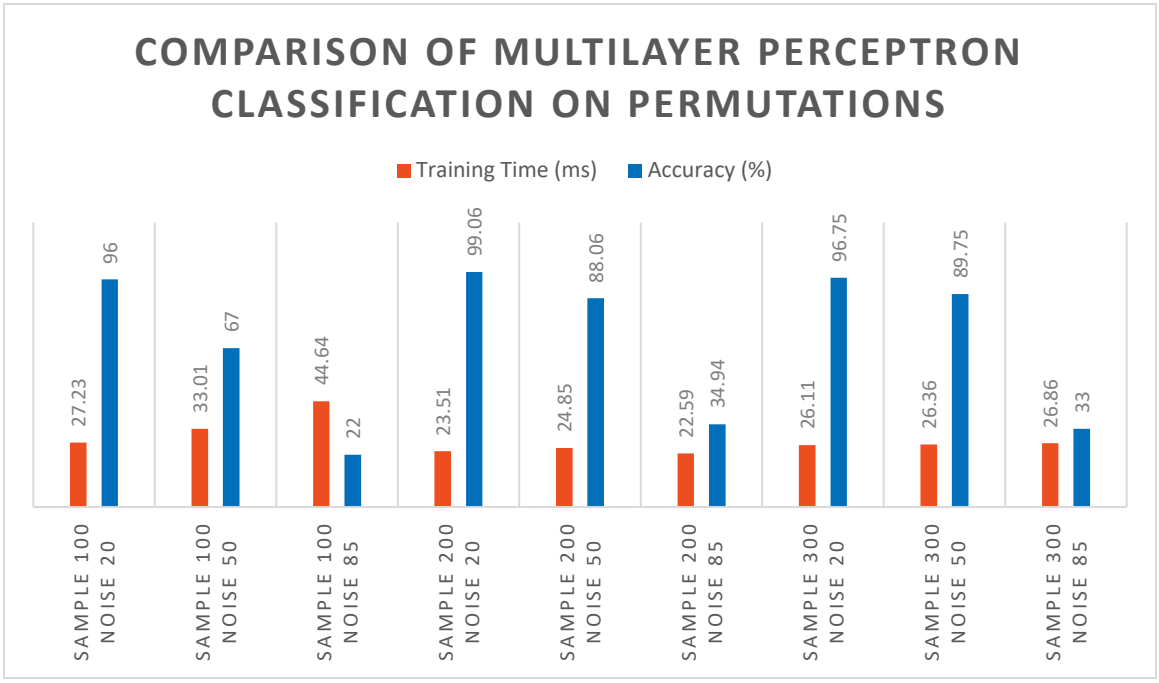
After using the optimal parameters for each combination concerning the values of N samples and noise rate, these are the training times and accuracy scores averaged after 20 repeated tests

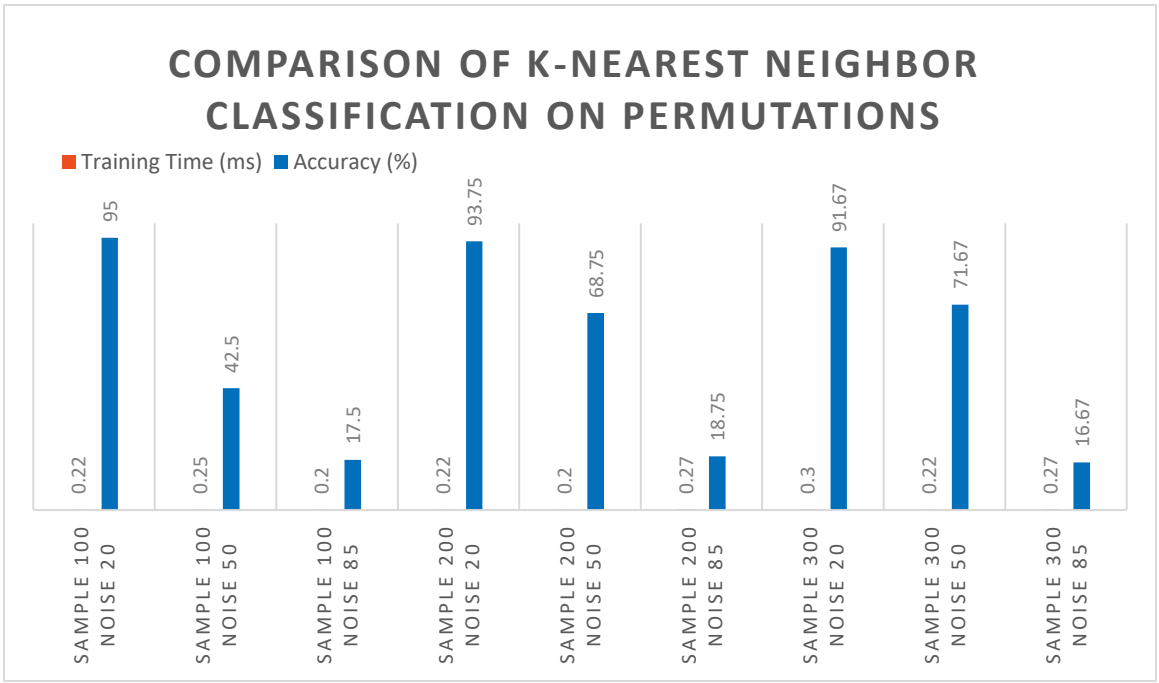
Jose Luis Sanchez

Dr. Razvan Andonie

Computational Intelligence and Machine Learning

June 7, 2022



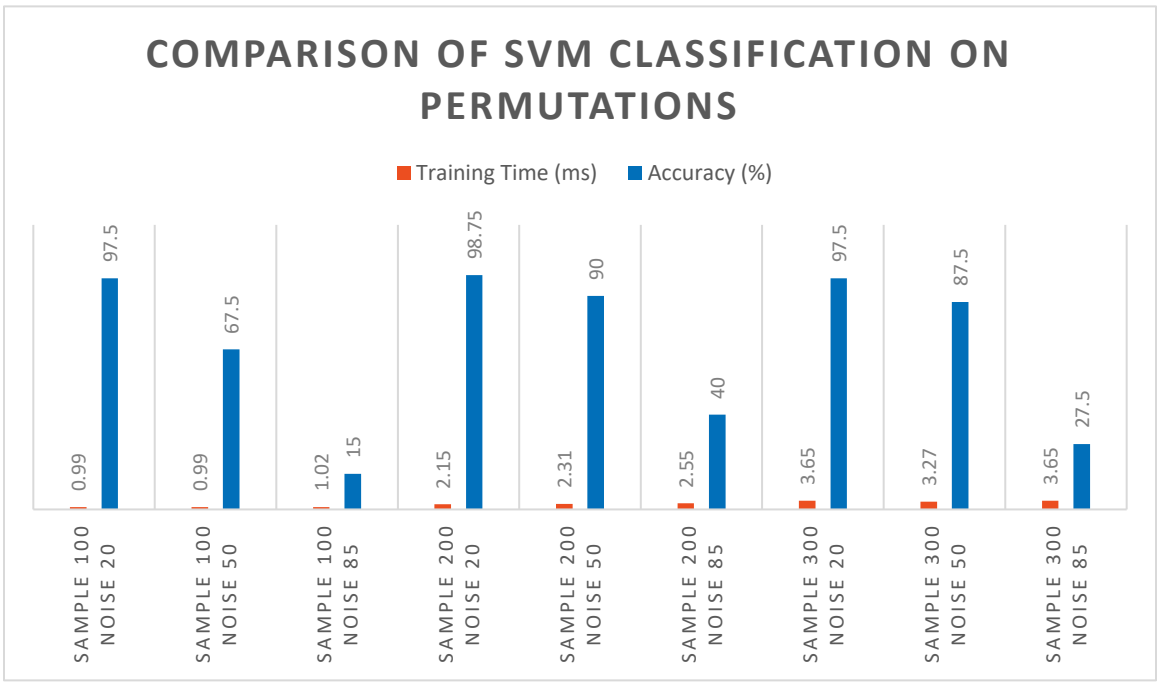


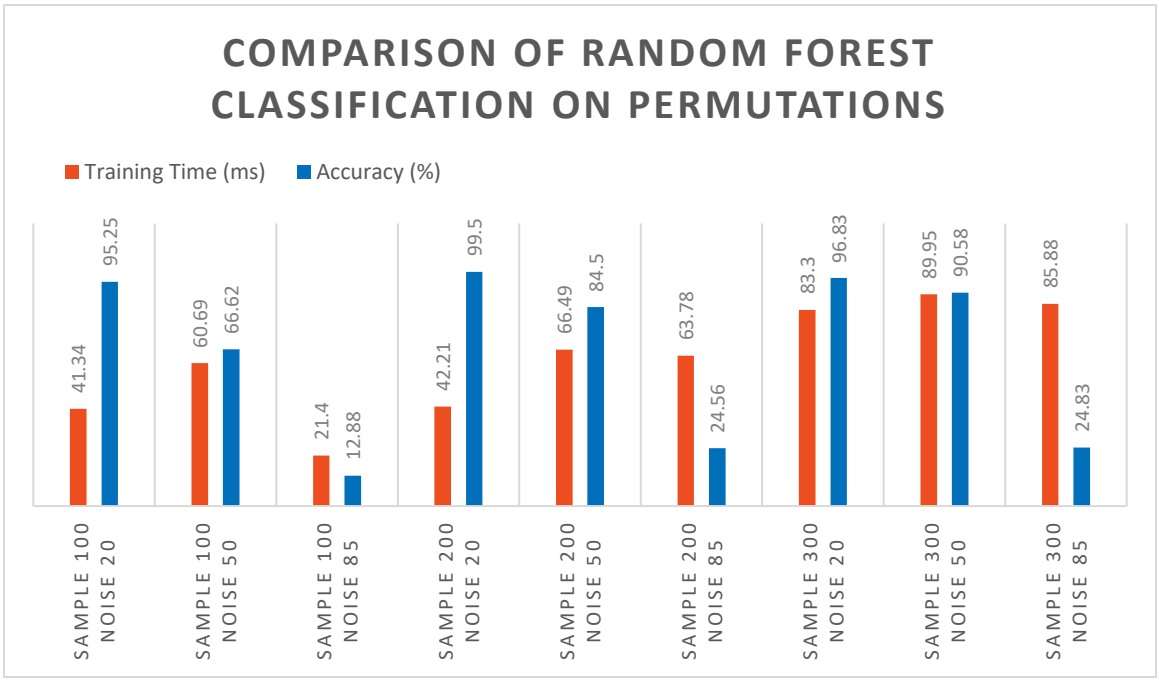
Jose Luis Sanchez

Dr. Razvan Andonie

Computational Intelligence and Machine Learning

June 7, 2022



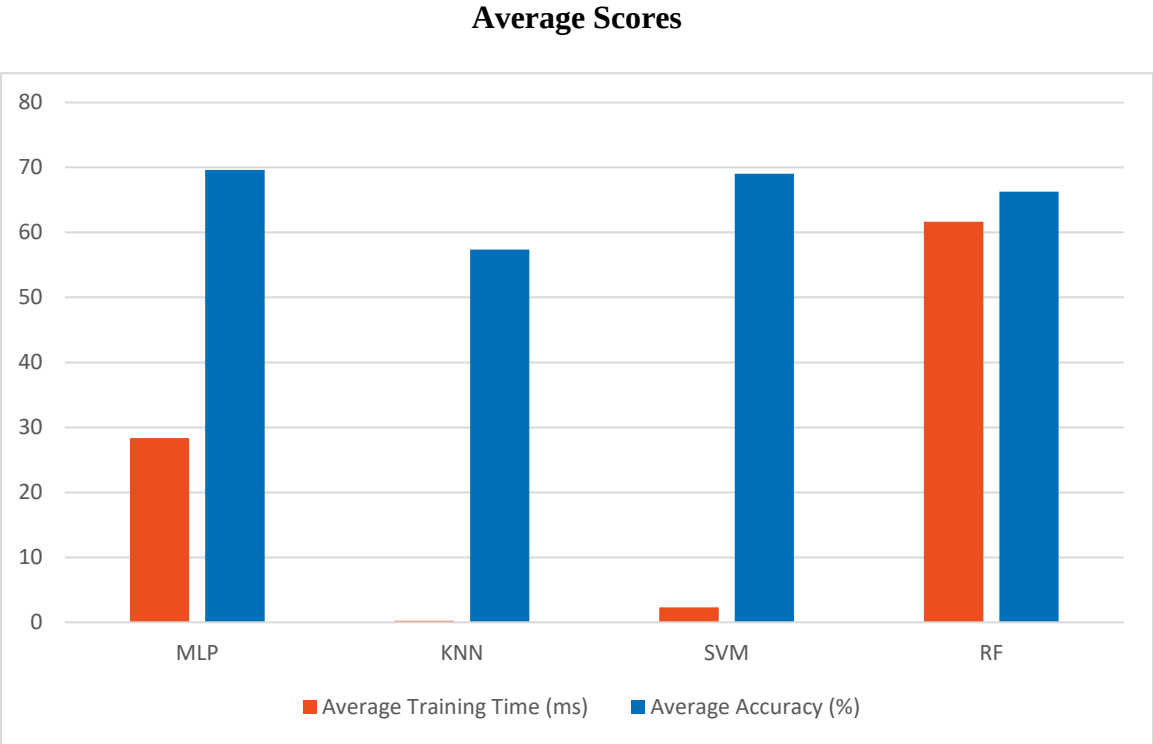


Jose Luis Sanchez

Dr. Razvan Andonie

Computational Intelligence and Machine Learning

June 7, 2022



When observing the average score for both training time and accuracy, some classifiers suffer mostly on training time. Classifiers such as MLP and RF suffered the most in training time, while the fastest classifier was KNN. However, the best overall score was MLP, though SVM is quite close to that. Though when comparing the MLP and SVM classifiers, I can see that SVM is a better option for this dataset as would be SVM, as the accuracy score is quite good, and the training time is quite fast. So for this project, the best classifier would be SVM, with RF being the worst due to training speed.

Wrapper Feature Selection Method

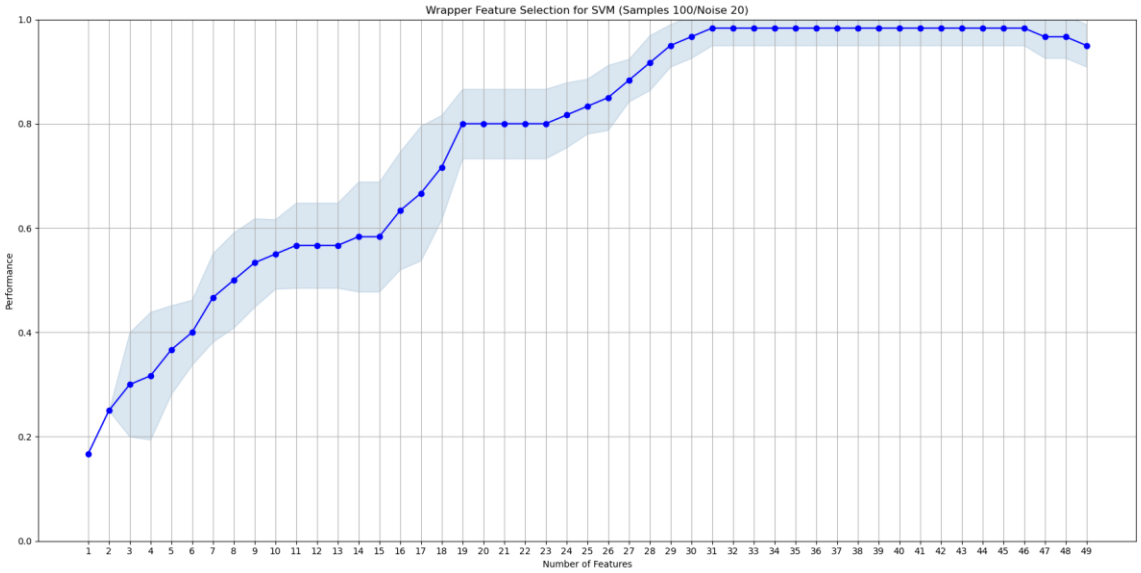
The wrapper feature selection method I used for this project was the Sequential Forward Selector (SFS) provided by mlxtend, another machine learning package, like sklearn, for Python. Since SVM was the best classifier in this experiment, I will be using it for the SFS. After running the SFS with SVM on all nine combinations, I was able to find the best combinations of features that are utilized the most for this dataset.

Jose Luis Sanchez

Dr. Razvan Andonie

Computational Intelligence and Machine Learning

June 7, 2022



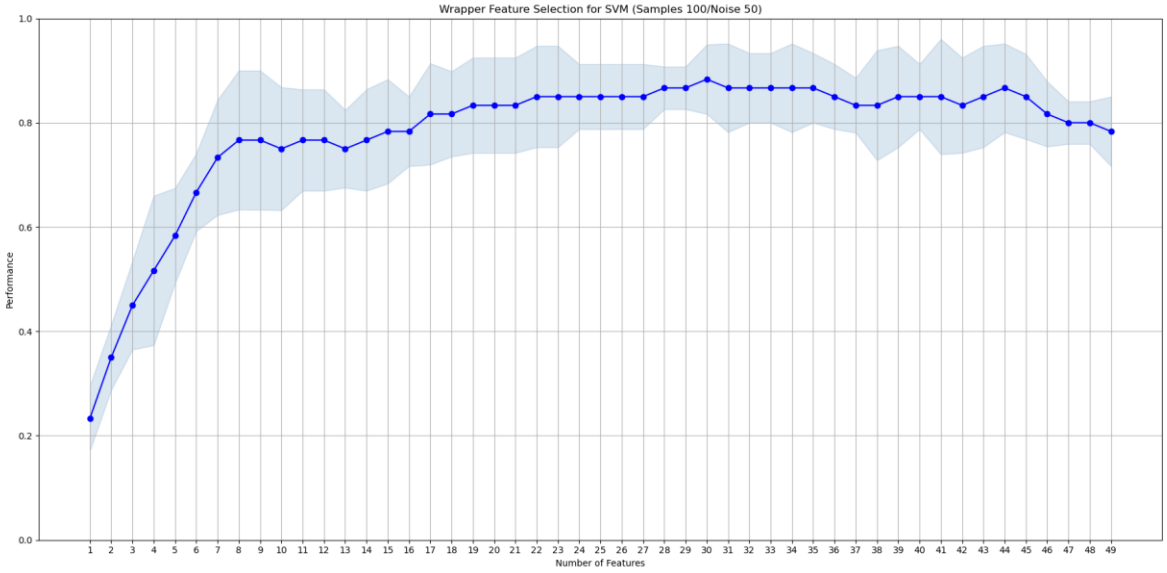
Best combination for sample size 100 and noise rate 20 (Accuracy: 98.3%): ('bit1', 'bit2', 'bit3', 'bit4', 'bit5', 'bit6', 'bit7', 'bit8', 'bit9', 'bit11', 'bit13', 'bit15', 'bit16', 'bit21', 'bit22', 'bit23', 'bit25', 'bit26', 'bit27', 'bit28', 'bit29', 'bit32', 'bit35', 'bit36', 'bit39', 'bit42', 'bit43', 'bit44', 'bit45', 'bit46', 'bit47')

Jose Luis Sanchez

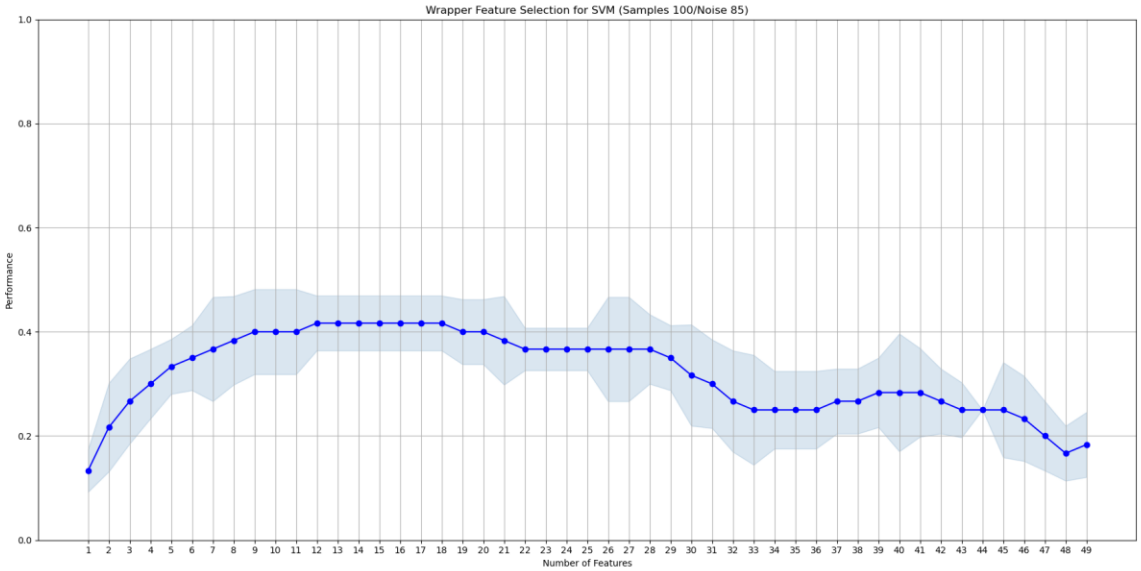
Dr. Razvan Andonie

Computational Intelligence and Machine Learning

June 7, 2022



Best combination for sample size 100 and noise rate 50 (Accuracy: 88.3%): ('bit1', 'bit2', 'bit3', 'bit4', 'bit5', 'bit6', 'bit7', 'bit8', 'bit9', 'bit11', 'bit12', 'bit15', 'bit17', 'bit18', 'bit19', 'bit22', 'bit23', 'bit25', 'bit26', 'bit27', 'bit29', 'bit33', 'bit37', 'bit42', 'bit43', 'bit44', 'bit45', 'bit47', 'bit48', 'bit49')



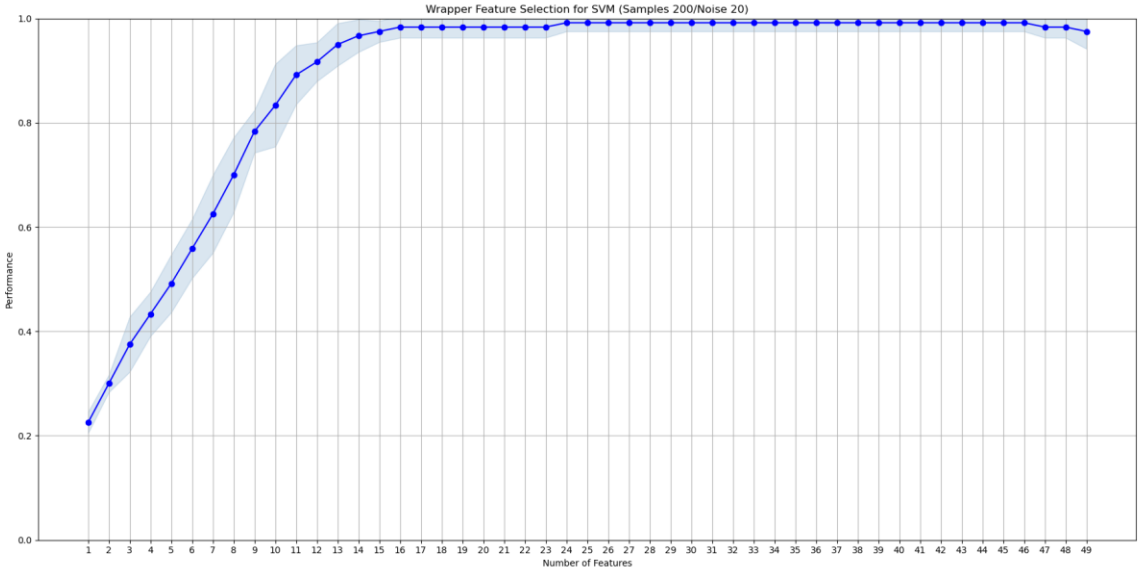
Best combination for sample size 100 and noise rate 85 (Accuracy: 41.7%): ('bit1', 'bit5', 'bit6', 'bit8', 'bit15', 'bit22', 'bit23', 'bit24', 'bit26', 'bit35', 'bit41', 'bit47')

Jose Luis Sanchez

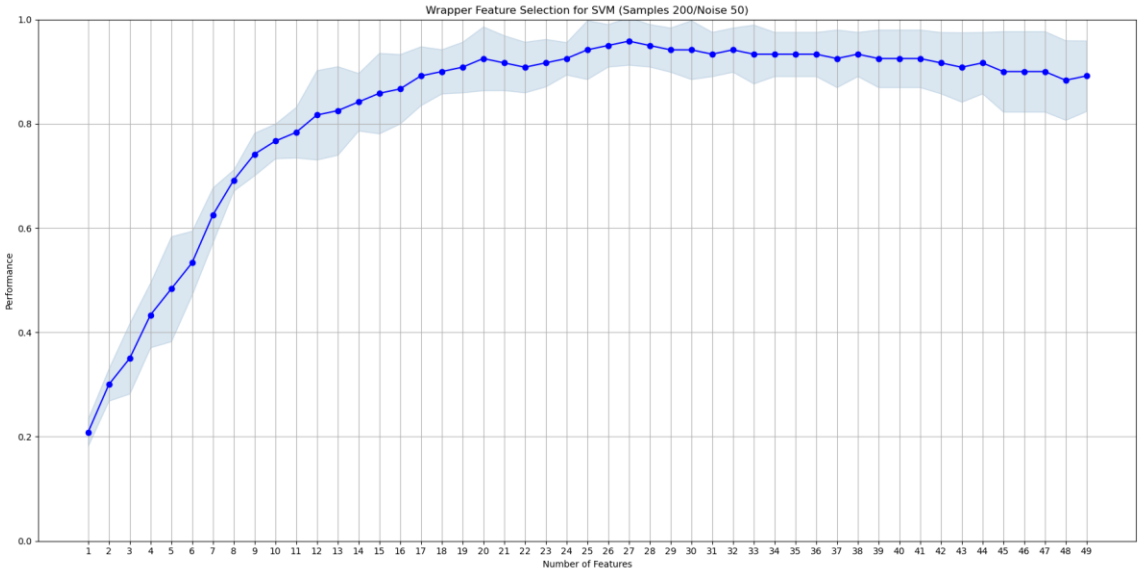
Dr. Razvan Andonie

Computational Intelligence and Machine Learning

June 7, 2022



Best combination for sample size 200 and noise rate 20 (Accuracy: 99.2%): ('bit1', 'bit2', 'bit3', 'bit4', 'bit5', 'bit6', 'bit7', 'bit8', 'bit10', 'bit11', 'bit14', 'bit18', 'bit20', 'bit21', 'bit22', 'bit24', 'bit25', 'bit27', 'bit28', 'bit30', 'bit31', 'bit35', 'bit47', 'bit49')



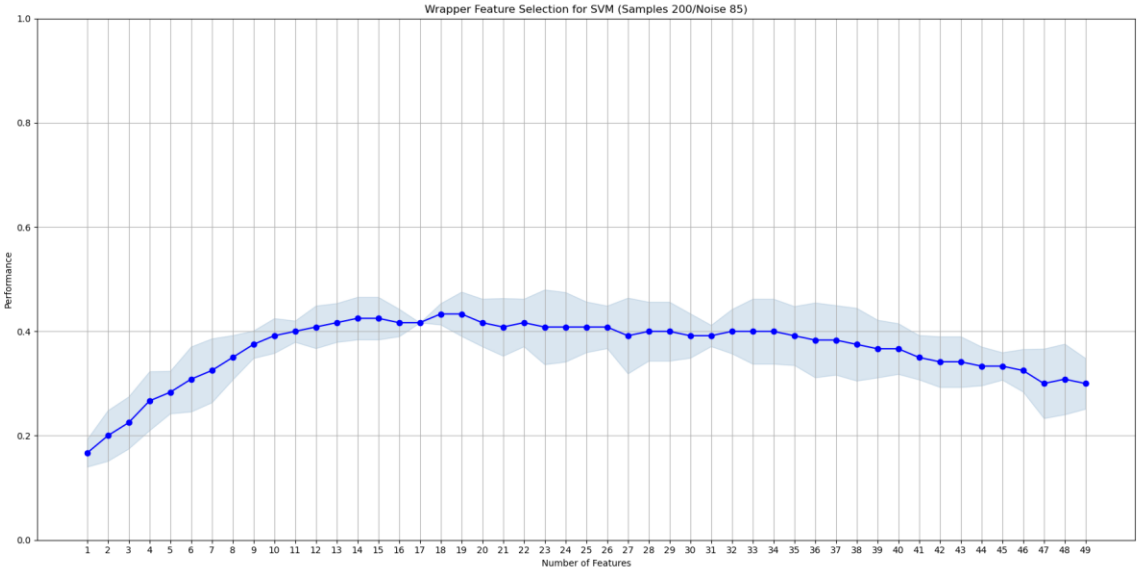
Best combination for sample size 200 and noise rate 50 (Accuracy: 95.8%): ('bit3', 'bit4', 'bit5', 'bit8', 'bit10', 'bit13', 'bit14', 'bit15', 'bit16', 'bit19', 'bit20', 'bit21', 'bit23', 'bit24', 'bit25', 'bit27', 'bit28', 'bit29', 'bit30', 'bit33', 'bit39', 'bit42', 'bit43', 'bit45', 'bit47', 'bit48', 'bit49')

Jose Luis Sanchez

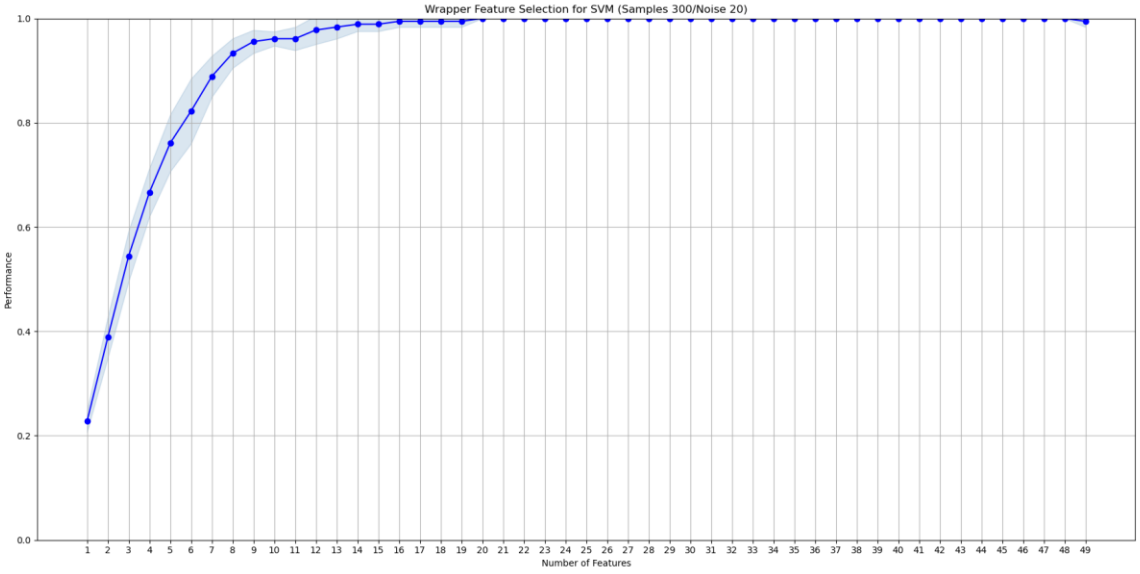
Dr. Razvan Andonie

Computational Intelligence and Machine Learning

June 7, 2022



Best combination for sample size 200 and noise rate 85 (Accuracy: 43.3%): ('bit1', 'bit2', 'bit3', 'bit4', 'bit6', 'bit7', 'bit8', 'bit15', 'bit22', 'bit23', 'bit24', 'bit26', 'bit32', 'bit36', 'bit41', 'bit43', 'bit44', 'bit46', 'bit49')



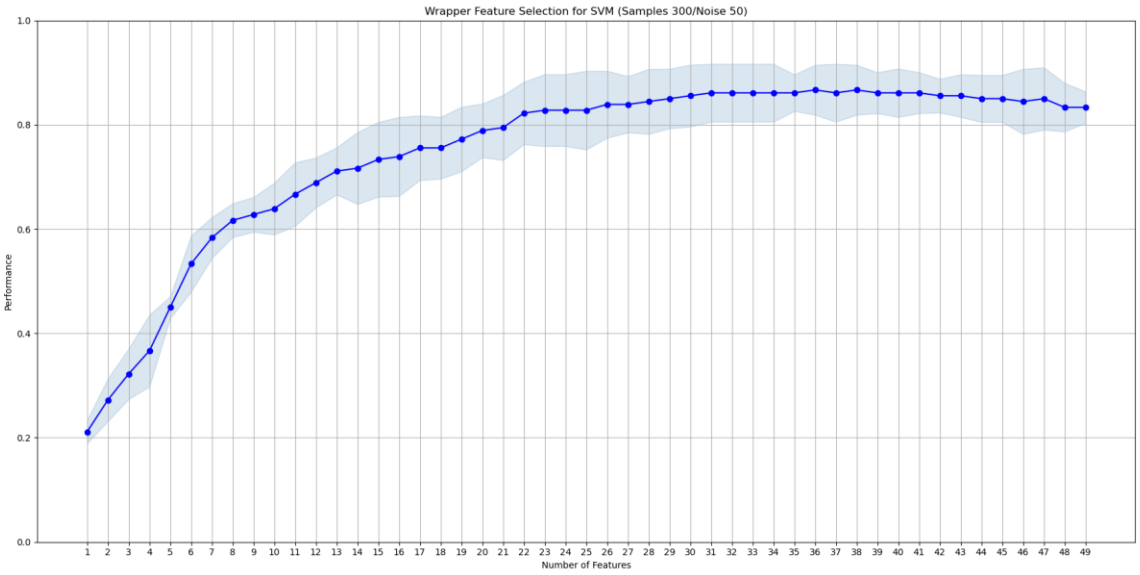
Best combination for sample size 300 and noise rate 20 (Accuracy: 100%): ('bit1', 'bit2', 'bit3', 'bit4', 'bit5', 'bit6', 'bit8', 'bit12', 'bit14', 'bit17', 'bit23', 'bit24', 'bit25', 'bit28', 'bit29', 'bit36', 'bit42', 'bit45', 'bit48', 'bit49')

Jose Luis Sanchez

Dr. Razvan Andonie

Computational Intelligence and Machine Learning

June 7, 2022



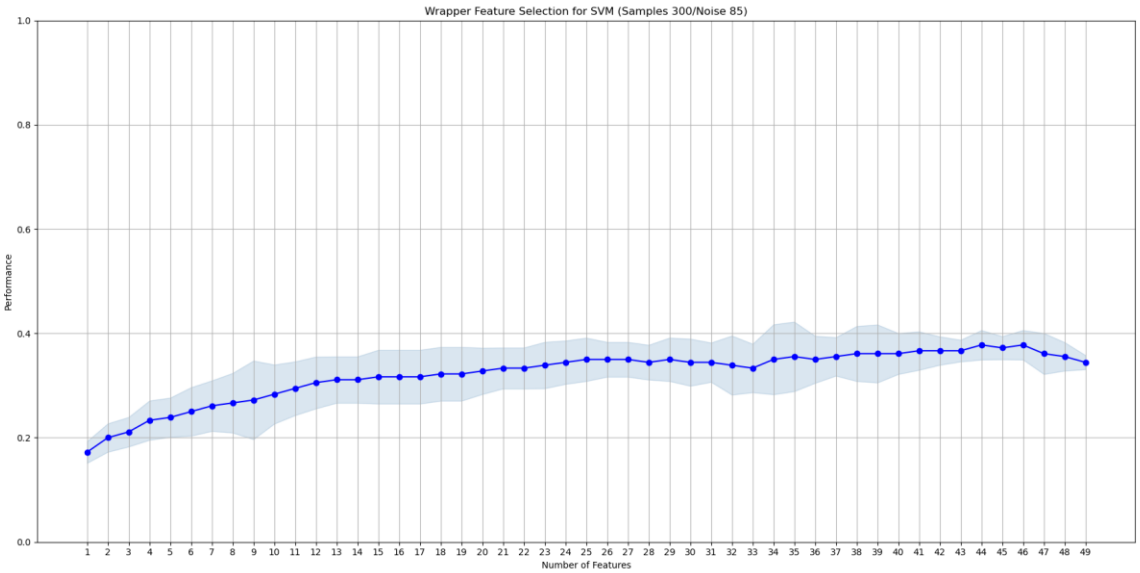
Best combination for sample size 300 and noise rate 50 (Accuracy: 86.7%): ('bit1', 'bit2', 'bit3', 'bit4', 'bit5', 'bit6', 'bit9', 'bit11', 'bit12', 'bit14', 'bit15', 'bit17', 'bit18', 'bit19', 'bit20', 'bit23', 'bit24', 'bit25', 'bit27', 'bit28', 'bit29', 'bit33', 'bit34', 'bit35', 'bit36', 'bit37', 'bit39', 'bit40', 'bit42', 'bit43', 'bit44', 'bit45', 'bit46', 'bit47', 'bit48', 'bit49')

Jose Luis Sanchez

Dr. Razvan Andonie

Computational Intelligence and Machine Learning

June 7, 2022



Best combination for sample size 300 and noise rate 85 (Accuracy: 0.378): ('bit1', 'bit2', 'bit3', 'bit4', 'bit5', 'bit6', 'bit7', 'bit8', 'bit9', 'bit10', 'bit11', 'bit12', 'bit13', 'bit14', 'bit15', 'bit16', 'bit17', 'bit18', 'bit20', 'bit21', 'bit22', 'bit23', 'bit24', 'bit25', 'bit26', 'bit27', 'bit28', 'bit29', 'bit30', 'bit31', 'bit32', 'bit35', 'bit36', 'bit37', 'bit38', 'bit40', 'bit41', 'bit42', 'bit43', 'bit44', 'bit45', 'bit46', 'bit48', 'bit49')

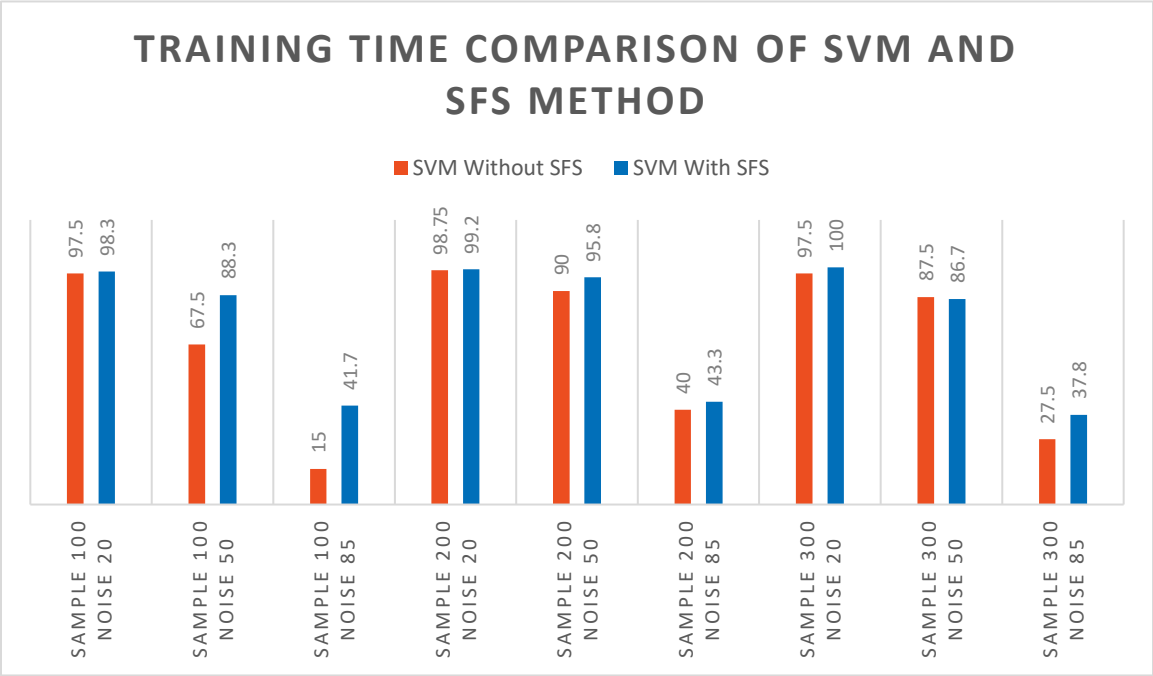
Jose Luis Sanchez

Dr. Razvan Andonie

Computational Intelligence and Machine Learning

June 7, 2022

When looking at the different results from SFS, SVM shines best when the dataset is larger and has less noise, though when more noise is added, its accuracy doesn't become that awful compared to other classifiers like KNN.



Also, it seems like the best features selected for each combination are varied, having no clear distinction between sample size and noise rate. There are some datasets where there aren't many features that need to score a high accuracy score (such as the sample size 300 and noise rate 20 dataset), but there are also datasets where they don't have many features as well but have a low accuracy score (such as the sample size 200 and noise rate 85 dataset). Likewise, some datasets require a lot of features to score high (sample size 100 and noise rate 20), while other datasets that require a lot of features end up scoring lower (sample size 300 and noise rate 85). While the results seem random, I do believe that using the SFS and other wrapper feature selection methods increases the accuracy quite considerably, seeing as the accuracy score is a lot better compared to its accuracy score without the wrapper method.