University of Coimbra

Pattern Recognition

Default of credit card clients

Author:
António Lima
2011166926

Author:
Pedro Janeiro
2012143629

May 29, 2016

Contents

List of Figures

List of Tables

1 Introduction

Pattern Recognition is a very broad field of studies that has potentially endless applications, be they real or academic. As part of this subject's evaluation, we were poised with the possibility of correctly identifying if a credit card holder would default their payments based only on 30 000 sample card holders. This report contains our findings on the matter, using various techniques to simplify and reduce the amount of data necessary to provide a relatively trustworthy classification.

2 Simulation Characteristics

2.1 Data Normalization

One of the first things we can assess about the provided dataset is that each of its feature dimensions uses different value distributions, making it harder to analyse potentially interesting variables of samples. As such, we first offer the possibility to normalize the data in of each dimension.

2.2 Feature Selection & Reduction

Since the initial dataset boasts some 23 features for each of the 30 000 samples provided for the expected binary classification, it becomes apparent that some, if not most, of those 23 features might be useless.

2.2.1 Feature Selection

As far as selecting features goes, we provide the means to analyse the correlation and covariance between every pair of features to remove redundant variables, analyse the correlation and covariance of each feature with regard to the expected output and, last but not least, perform a Kruskal-Wallis test to assess the p-score of each feature. Any combination of these three feature selection methods can be chosen and, we've found that only 13 to 17 of the features yield any relevant or non-redundant information.

2.2.2 Feature Reduction

Regarding feature reduction, we offer the possibility of performing a Primary Component Analysis (PCA) or Linear Discriminant Analysis (LDA), we even allow for performing one after the other as a purely academic test. For PCA we also provide the possibility of selecting the most relevant primary components based on either a Scree Test or the Kaiser Criteria, both for varying thresholds. The Kaiser Criteria for a threshold of 1 means that only dimensions that represent as much as one full feature before the PCA was applied are relevant for classification; on the other hand, the Scree Test focuses on cumulative representation and for a threshold of 0.9 means that, only the dimensions that (cumulatively summed) account for 90% of the sum of all eigenvalues should be considered.

2.3 Classification

2.3.1 Data Splitting

Since the provided dataset is not uniformly distributed (and probably couldn't be without increasing exponentially in size) we first shuffle the samples randomly, following with a stratifying split. What this means is that we guarantee that for a certain splitting threshold (70% for training and 30% for testing by default) the training set will indeed have X% of each class represented and vice-versa for the testing set. These Thresholds can be customized.

2.3.2 Classifiers

The following classifiers are available in our simulator:

- **Linear Discriminant**: Defines a linear separation plane between every class and the remaining ones
- **Minimum Euclidean Distance**: Defines a centroid from the given train data and calculates the euclidean distance of every test sample to said centroid
- **Minimum Mahalanobis Distance**: Defines a centroid from the given train data and calculates the mahalanobis distance of every test sample to said centroid
- **Decision Tree**: Splits the data according to metric rules for the observed values and creates a tree-like structure
- **Support Vector Machine (SVM)**: Defines a region for each class using a set of support vectors
- **k Nearest Neighbours (kNN)**: Finds the k nearest neighbours from the train set to the test sample and uses the majority rule to classify

Of the above mentioned classifiers, it is noteworthy to mention that only the Euclidean and Mahalanobis Minimum distance classifiers were implemented by us, with the rest coming bundled up in Matlab's Statistics and Machine Learning Toolbox.

Why we chose this set of classifiers is rooted on the fact that they all try to achieve the same goal using very different approaches.

2.3.3 Triple Modular Redundancy

Since this assignment has only two possible classes (default payment or not) we can repeat the classification with two other classifiers and we could, upon majority, determine the predicted output. However, for generalization purposes, if the problem had more possible classes then we wouldn't apply the rule of majority but rather the rounded median value, which is exactly what we did.

2.4 Classifier Performance Analysis

Looking at the confusion matrix, accuracy, precision, recall, prevalence, sensitivity, specificity, F-measure and ROC curves of our the predicted and expected results we can get a good feel of how each specific simulation performs compared with the other ones. We even time each simulation in case a time constraint should be needed.

3 Graphical User Interface

As can be seen in the pre-processing stage, the user can indicate the path to the dataset and choose if it should be normalized after being loaded.

Afterwards, the user can choose if any form of feature selection should be performed on the data and can select any combination of the 3 available methods along with the corresponding thresholds.

After the feature selection stage comes the feature reduction stage which can also be toggled on and off, along with selecting which dimensionality reduction method to apply. We also allow for an additional reduction method to be used in conjunction with PCA and an accompanying threshold for each method.

Last but certainly not least, the classification stage. Here the user can specify which splitting % he wishes for the training set and if should contain stratified examples. Now we are left with deciding if we there should be a triple modular redundancy voter or not, and that is also just a matter of choosing which classifiers will be used.

Simply press the "Run" button and sit back for a couple of seconds.

4 Interesting Simulations

Apart from performing all possible simulations for the contemplated variables in our simulator, we decided to focus on specific subsets of simulations in order to figure out what would be the ideal progression when iteratively building a trustworthy classifier. As such, the following questions posed as the most interesting. Each simulation was repeated 10 times and the presented values are the obtained averages.

4.1 Normalize the dataset?

Starting with a simple dataset of 23 features and 30 000 samples, a training and testing splitting threshold of 70%, and using stratified splitting for the training and testing datasets, we pose the question of whether normalizing the dataset proves beneficial to our classifier.

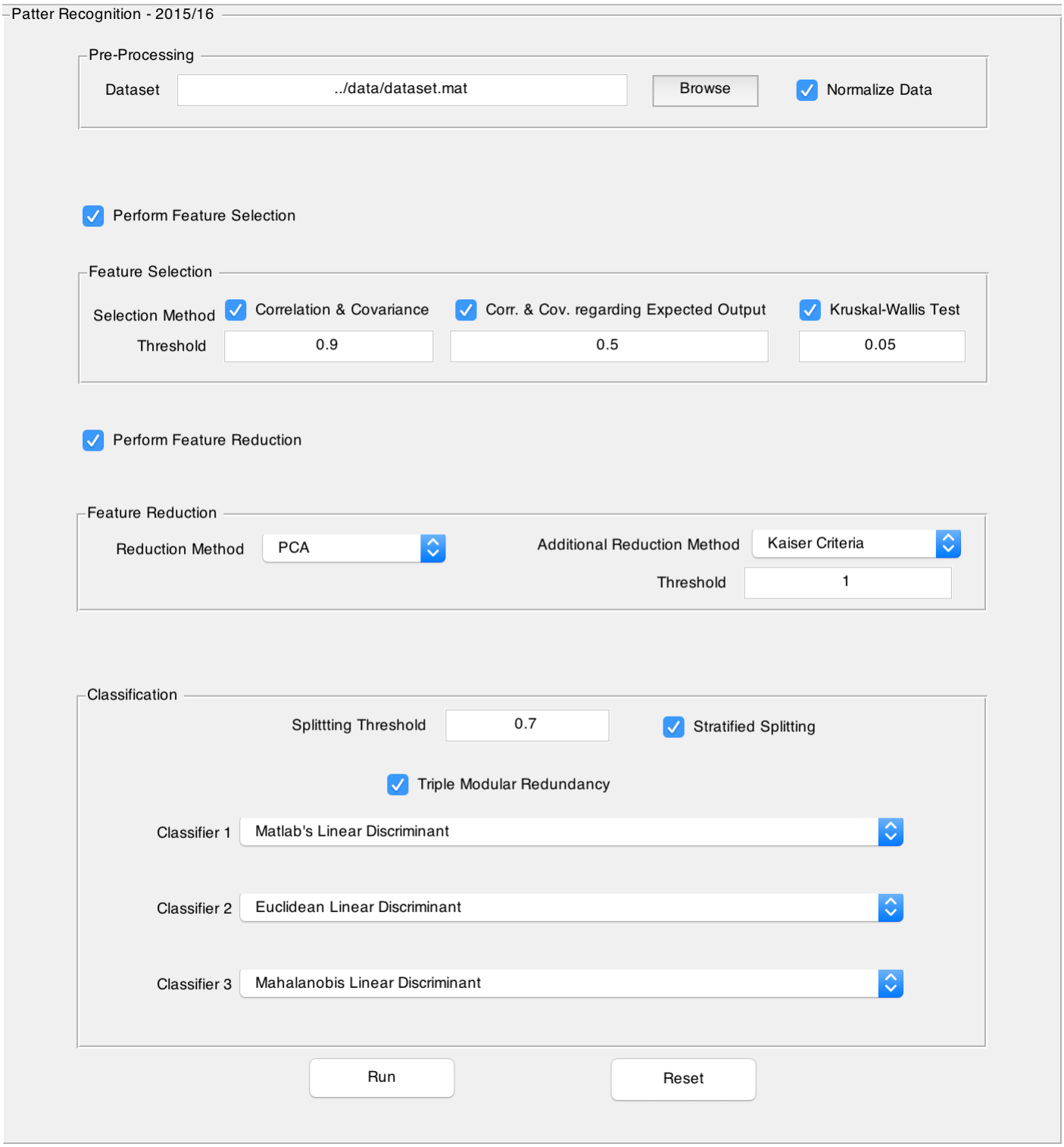

Figure 1: Custom Simulation GUI

Table 1: Pre-Processing Simulation Configurations

#	NORMALIZE	FEATURE_SELECTION	FEATURE_REDUCTION	VOTER	C1_EDC
1	0	0	0	0	1
2	1	0	0	0	1

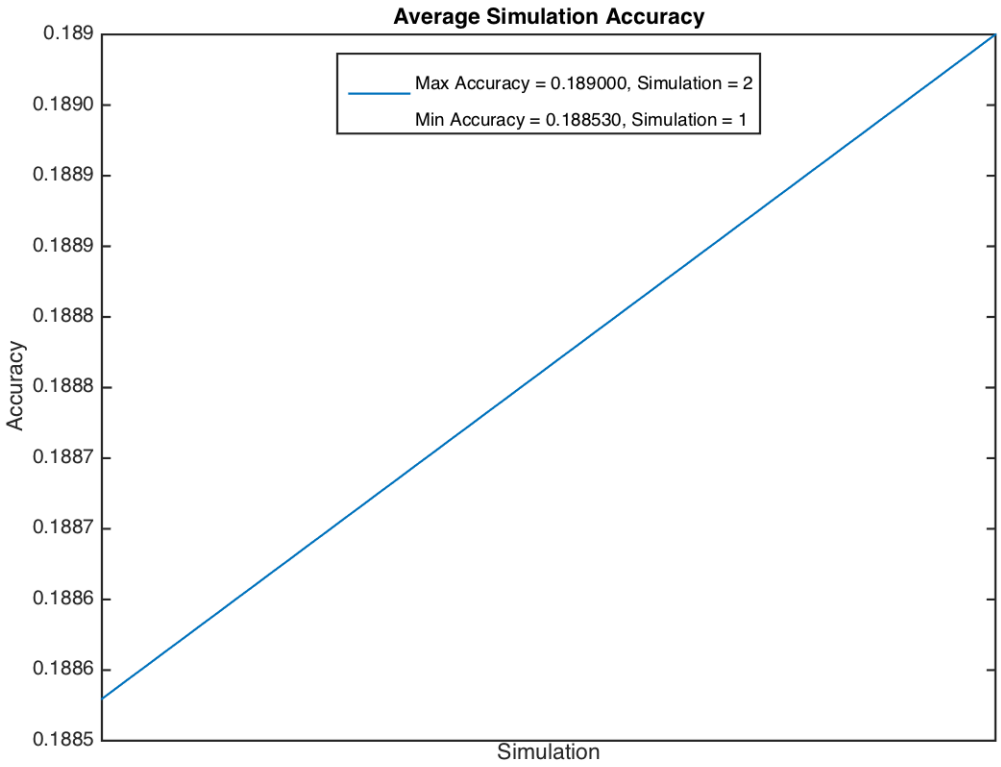

Figure 2: Normalization Simulations: Average Simulation Accuracy

Given this information, we can conclude that normalizing the dataset is beneficial and does not increase the simulation time immensely.

4.2 Which type of feature selection?

Building upon the previous simulation's configurations and assuming that normalizing is beneficial, we now pose the question of which combination of feature selection methods is best.

Using a correlation threshold of 90% between features, a correlation threshold of 25% between the features and the target, and a Kruskal-Wallis test with a p-value threshold of 0.05.

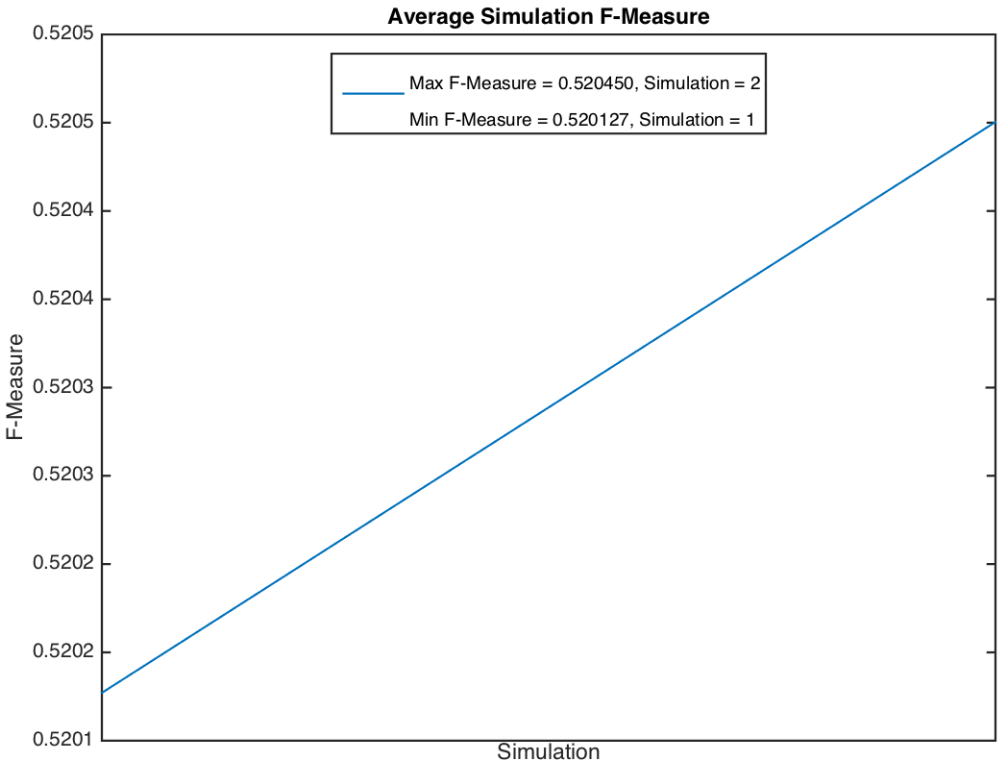

Figure 3: Normalization Simulations: Average Simulation F-Measure

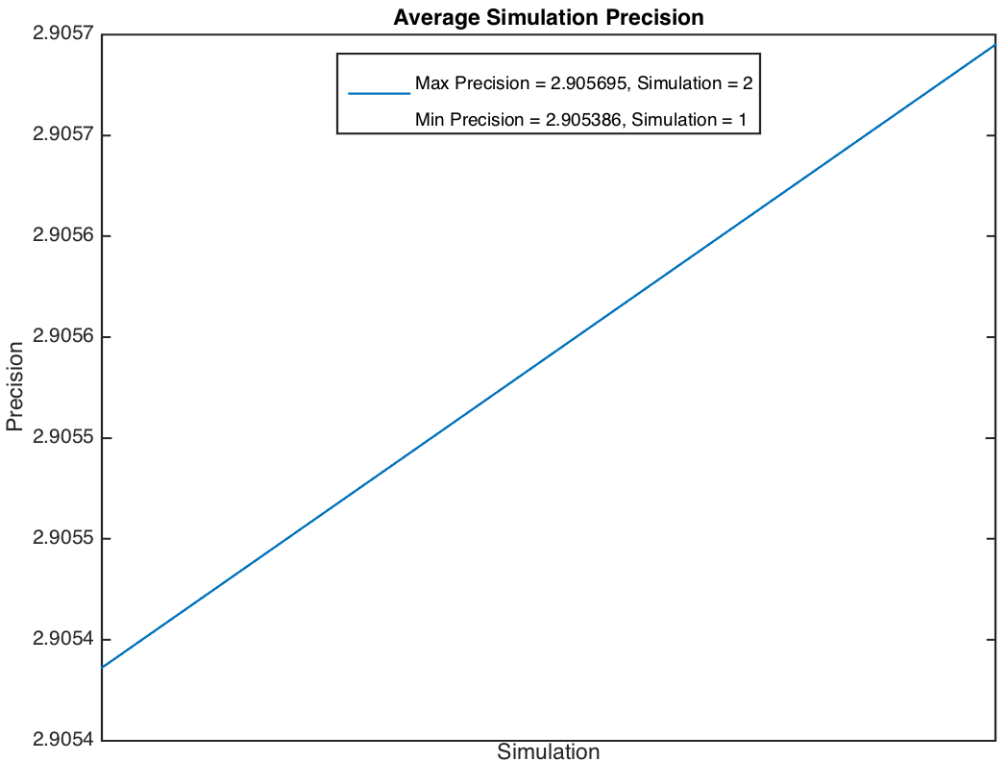

Figure 4: Normalization Simulations: Average Simulation Precision

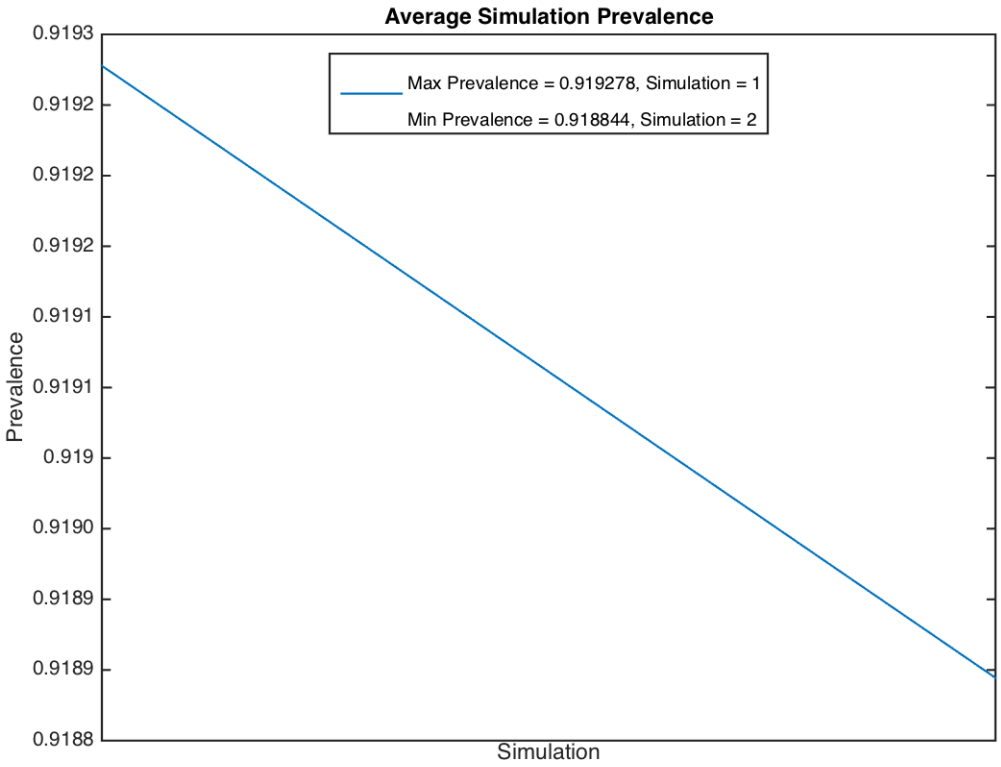

Figure 5: Normalization Simulations: Average Simulation Prevalence

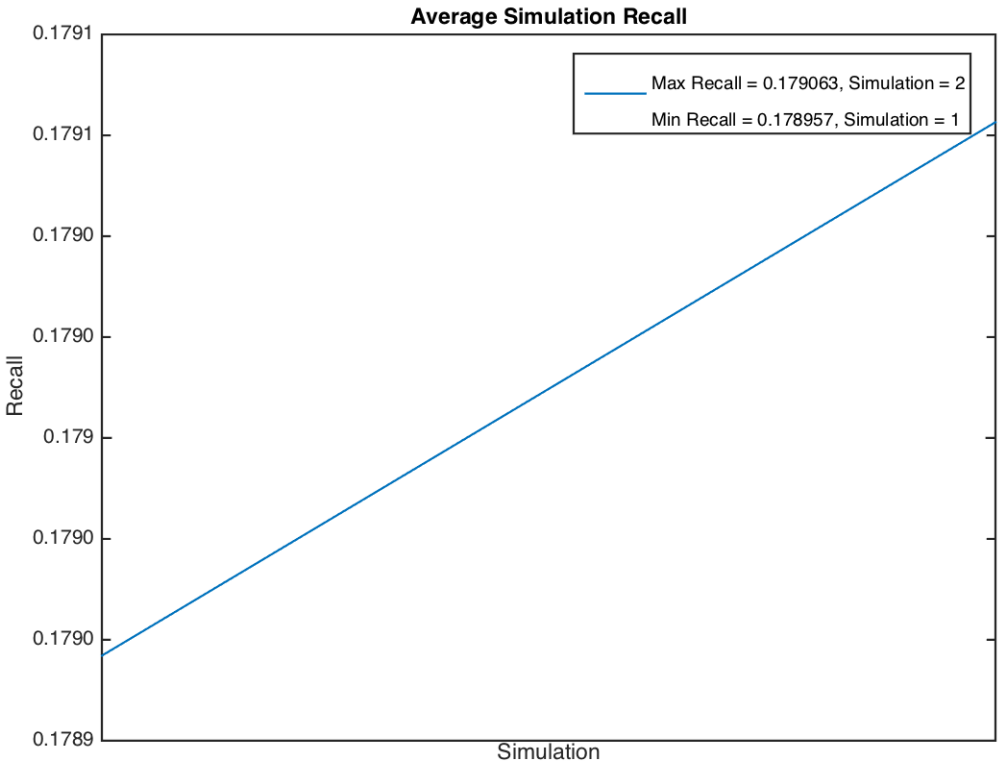

Figure 6: Normalization Simulations: Average Simulation Recall

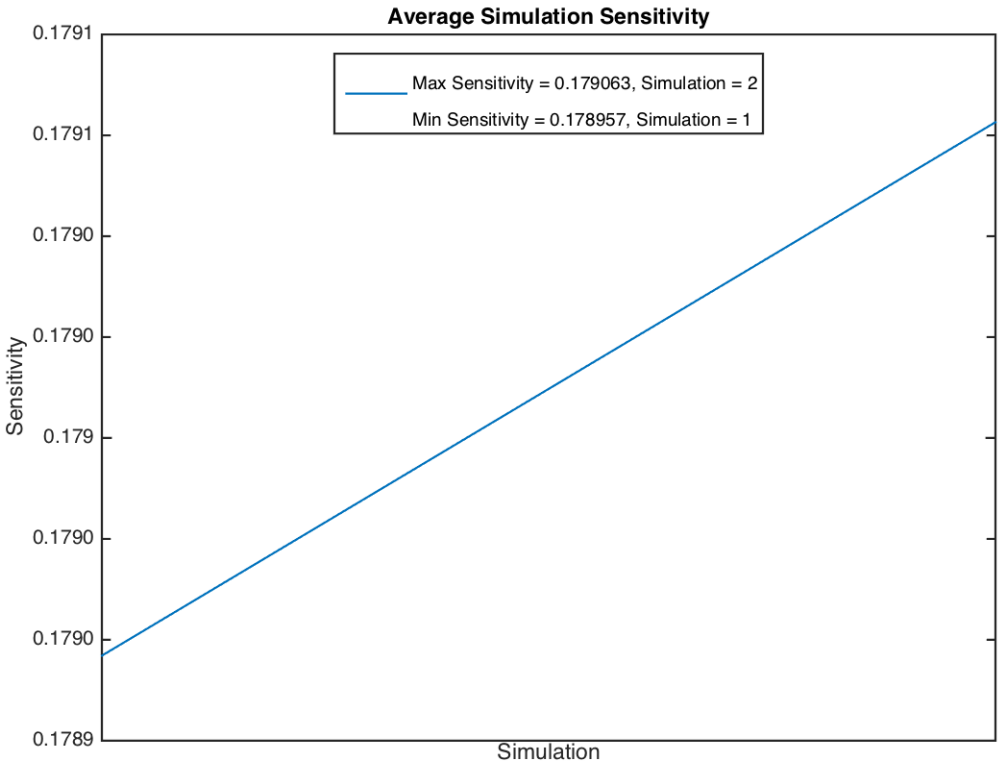

Figure 7: Normalization Simulations: Average Simulation Sensitivity

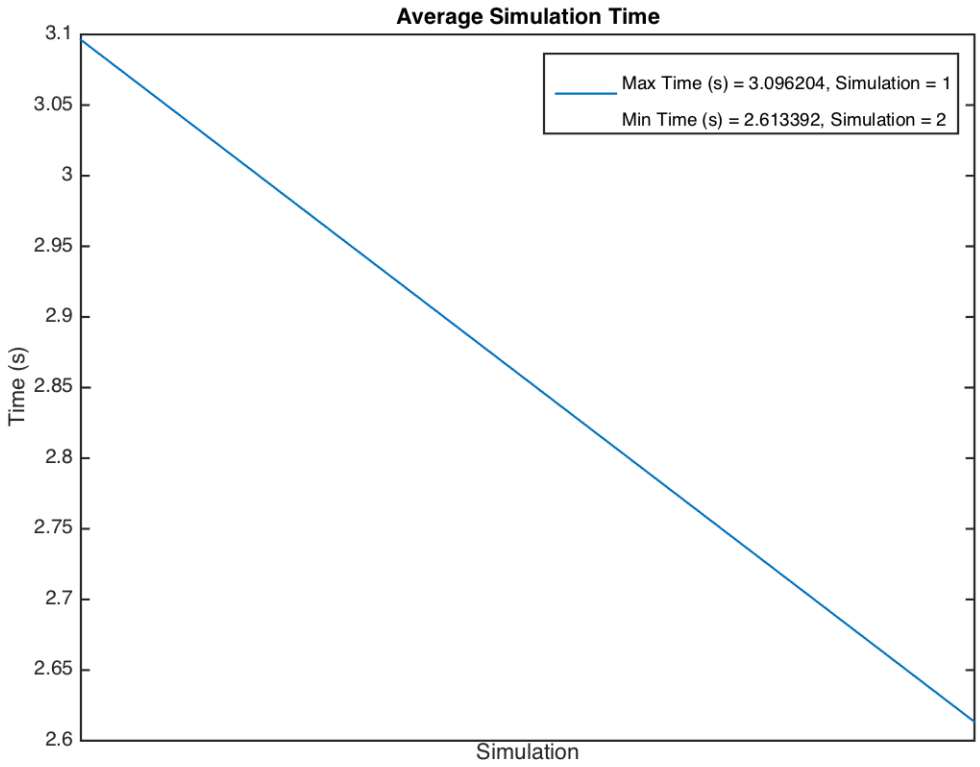

Figure 8: Normalization Simulations: Average Simulation Time

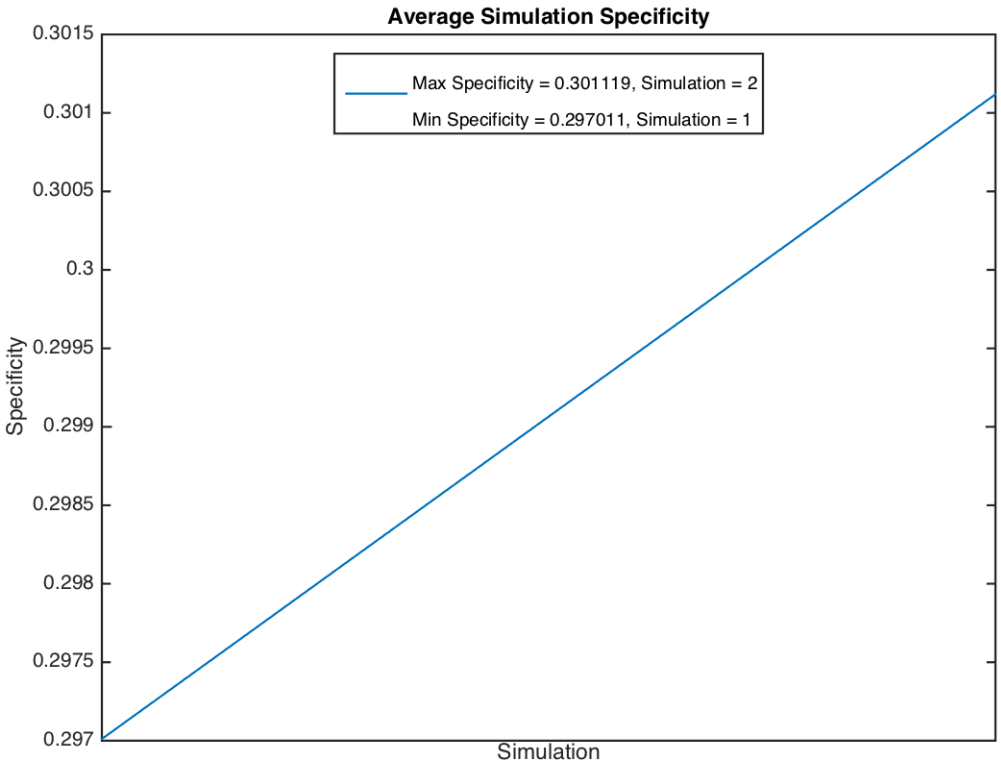

Figure 9: Normalization Simulations: Average Simulation specificity

Table 2: Feature Reduction Simulation Configurations

#	FEATURE_SELECTION	X_COR_COV	XY_COR_COV	KRUSKAL_WALLIS
1	1	0	0	0
2	1	0	0	1
3	1	0	1	0
4	1	0	1	1
5	1	1	0	0
6	1	1	0	1
7	1	1	1	0
8	1	1	1	1

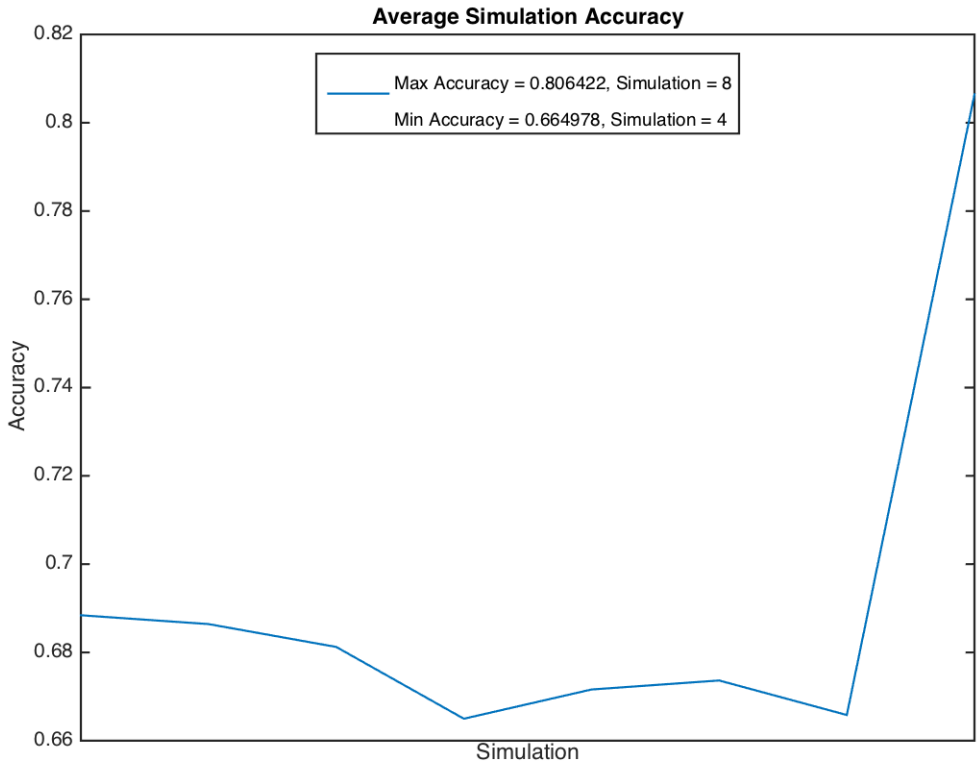

Figure 10: Feature Selection Simulations: Average Simulation Accuracy

Given this information, we conclude that the more features we discard the better, and for that we should employ as many feature selection techniques as we have available.

4.3 Which type of feature reduction?

In addition to the previous results, we now prepare to test weather we should perform a PCA, LDA, or even both, whilst also using a Kaiser Criteria with a threshold of 1, a Scree Test with a threshold of 90% or no additional reduction at all.

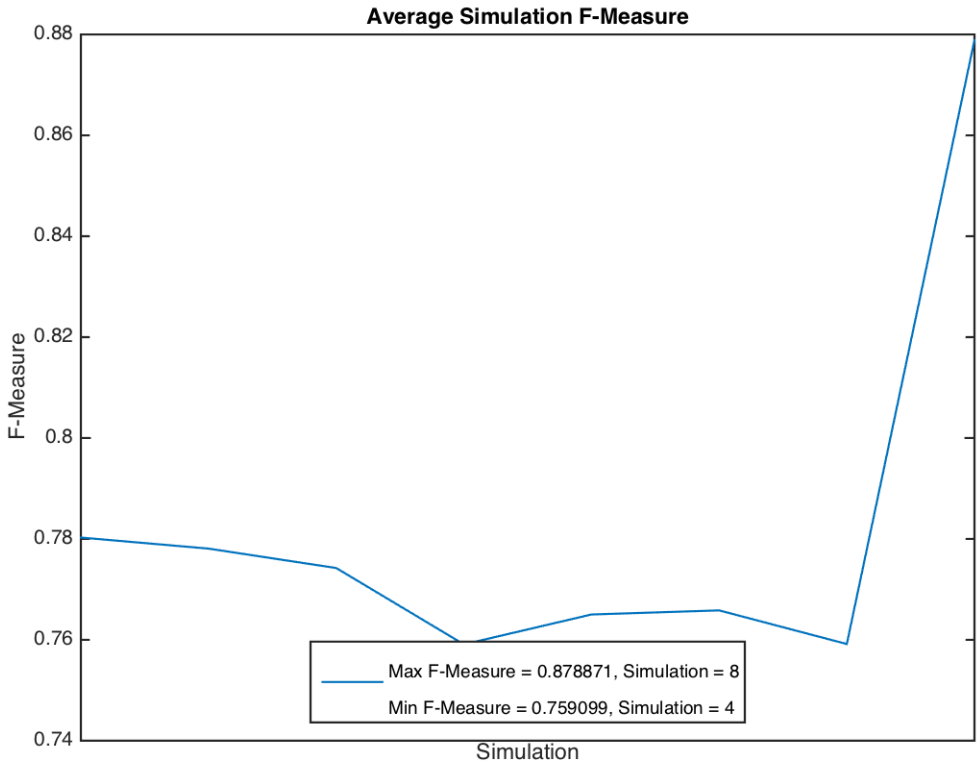

Figure 11: Feature Selection Simulations: Average Simulation F-Measure

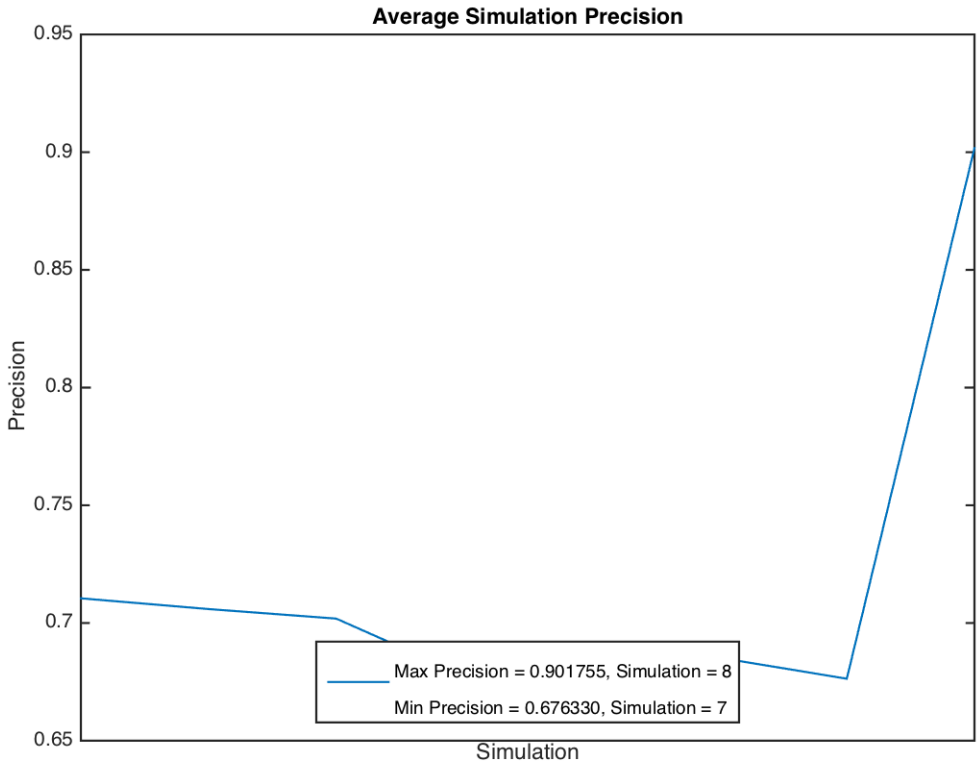

Figure 12: Feature Selection Simulations: Average Simulation Precision

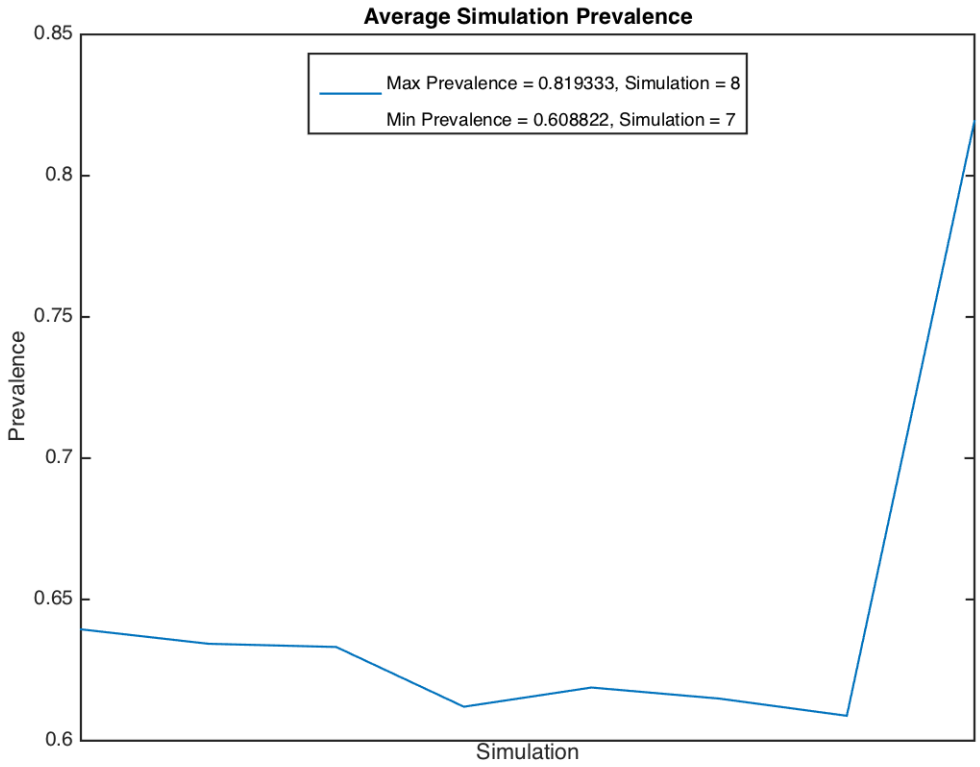

Figure 13: Feature Selection Simulations: Average Simulation Prevalence

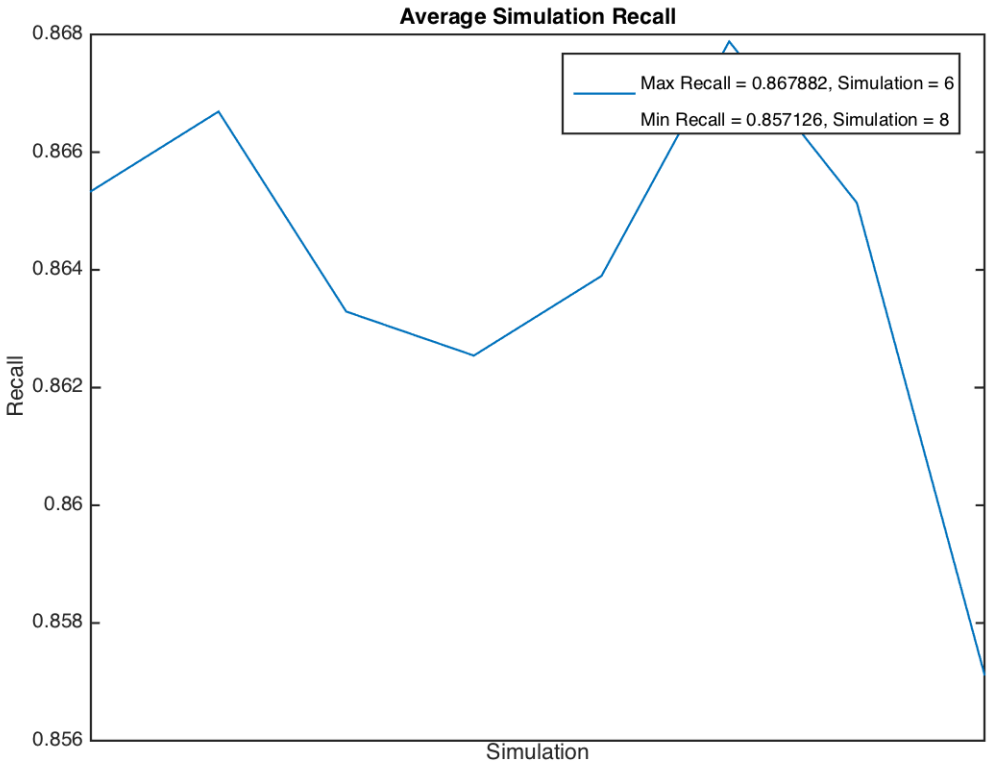

Figure 14: Feature Selection Simulations: Average Simulation Recall

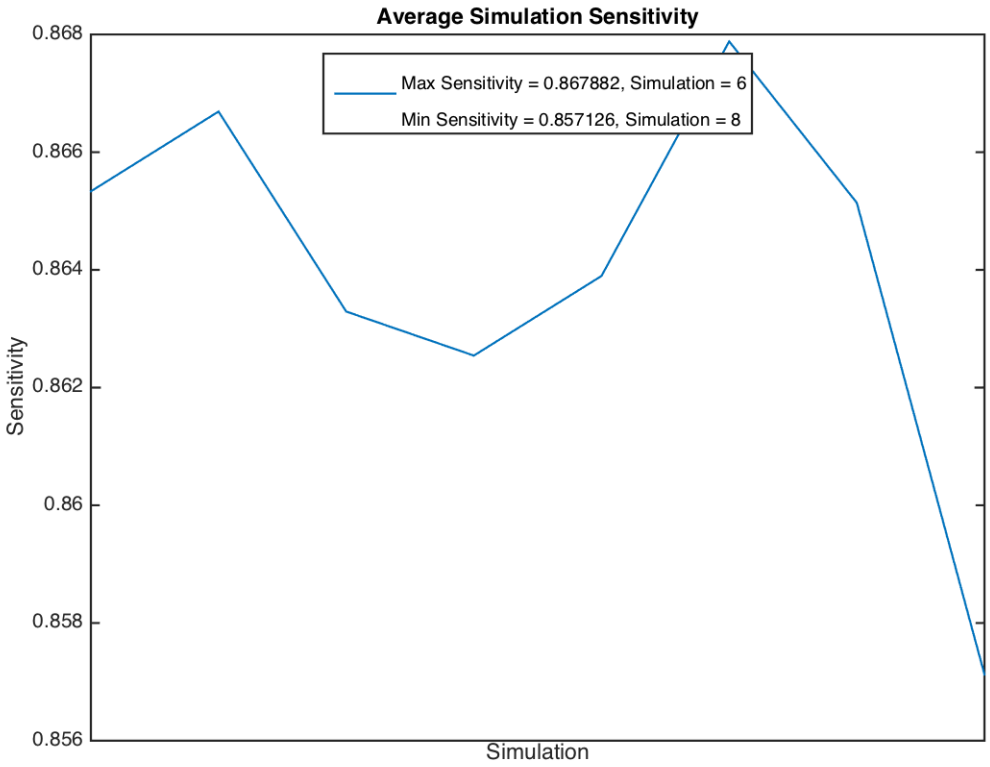

Figure 15: Feature Selection Simulations: Average Simulation Sensitivity

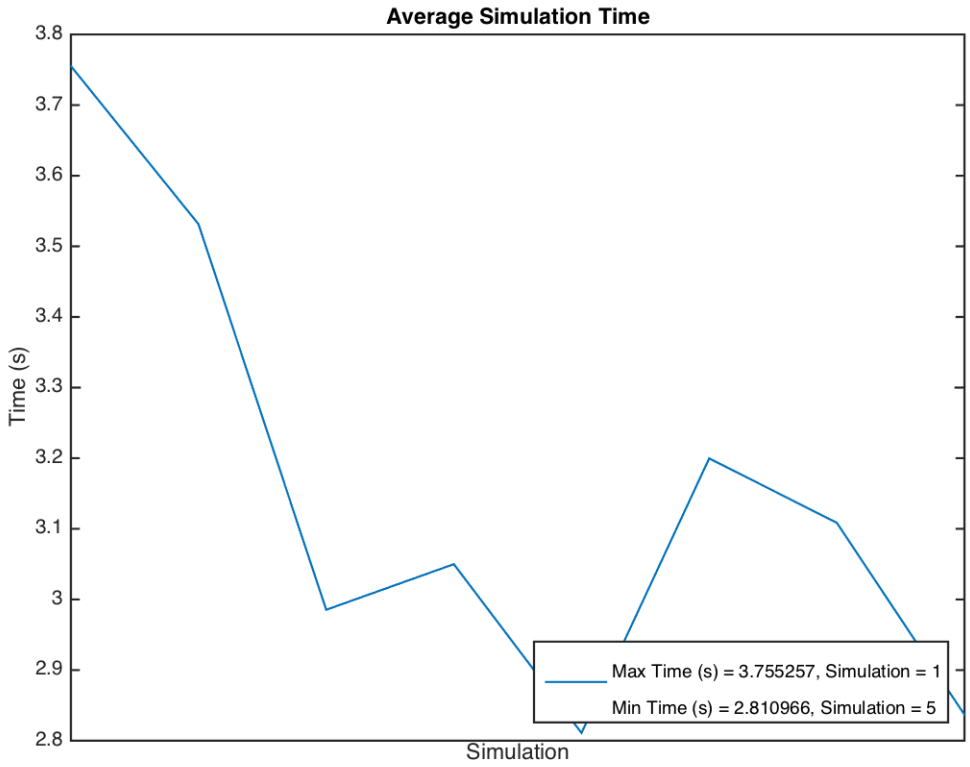

Figure 16: Feature Selection Simulations: Average Simulation Time

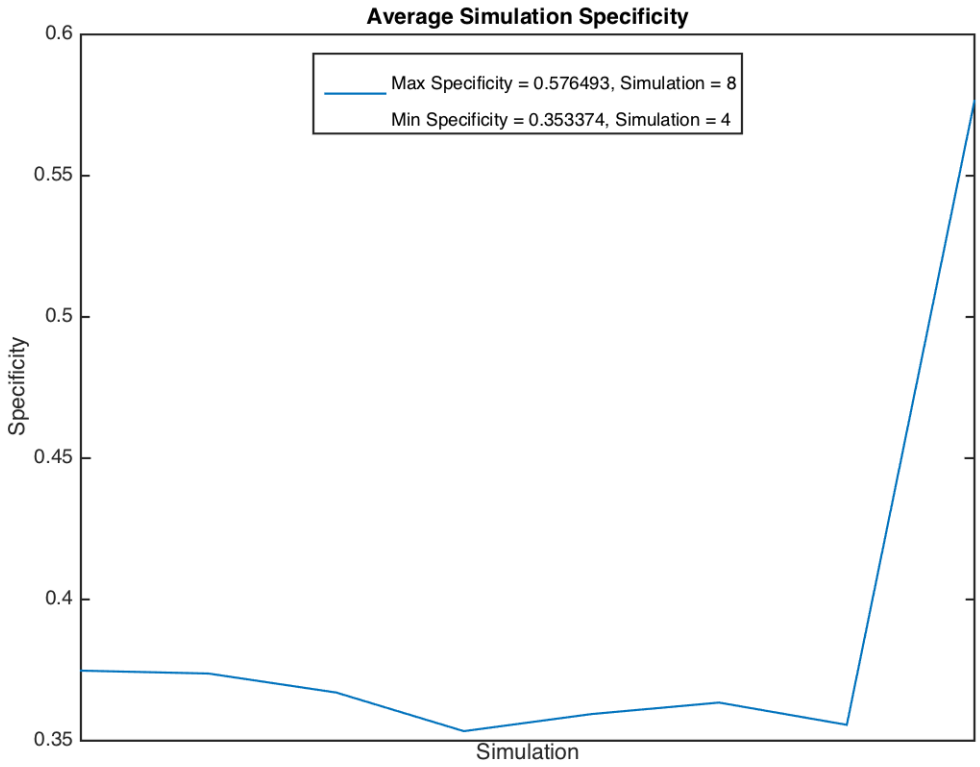

Figure 17: Feature Selection Simulations: Average Simulation specificity

Table 3: Feature Reduction Simulation Configurations

#	FEATURE_REDUCTION	PCA	LDA	PCA_LDA	LDA_PCA	KAISER_CRITERIA	SCREE_TEST
1	1	0	0	0	1	0	1
2	1	1	0	0	0	0	0
3	1	0	0	0	0	0	0
4	1	1	0	0	0	0	0
5	1	0	0	0	0	0	1
6	1	0	0	0	0	1	0
7	1	0	0	0	0	0	0
8	1	0	0	0	1	1	0
9	1	0	0	0	1	0	0
10	1	0	0	1	0	0	1
11	1	0	0	1	0	1	0
12	1	0	0	1	0	0	0
13	1	0	1	0	0	0	0
14	1	1	0	0	0	0	1
15	1	1	0	0	0	1	0

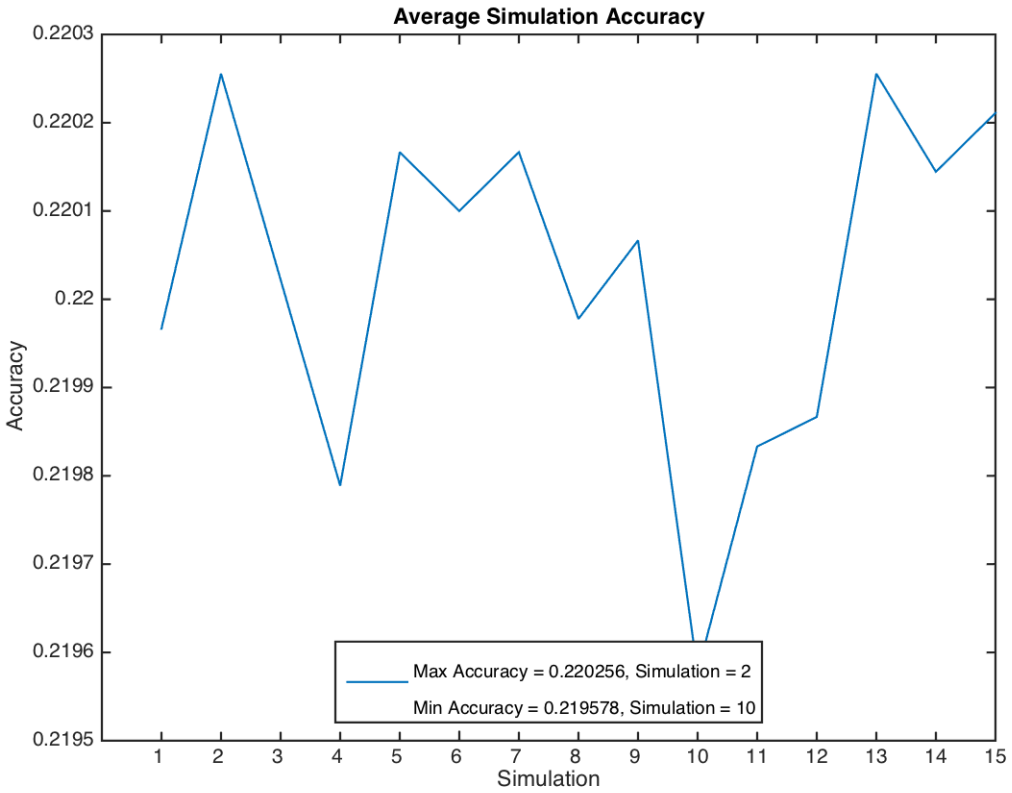

Figure 18: Feature Reduction Simulations: Average Simulation Accuracy

Given this information, we conclude continue to conclude that the more features we discard the better, and for that we should employ a PCA in conjunction with a Scree test.

4.4 Which classifier?

The final but perhaps most important question we pose, using what we have learnt from previous simulations, is which is the best classifier regarding the metrics we have already proposed before.

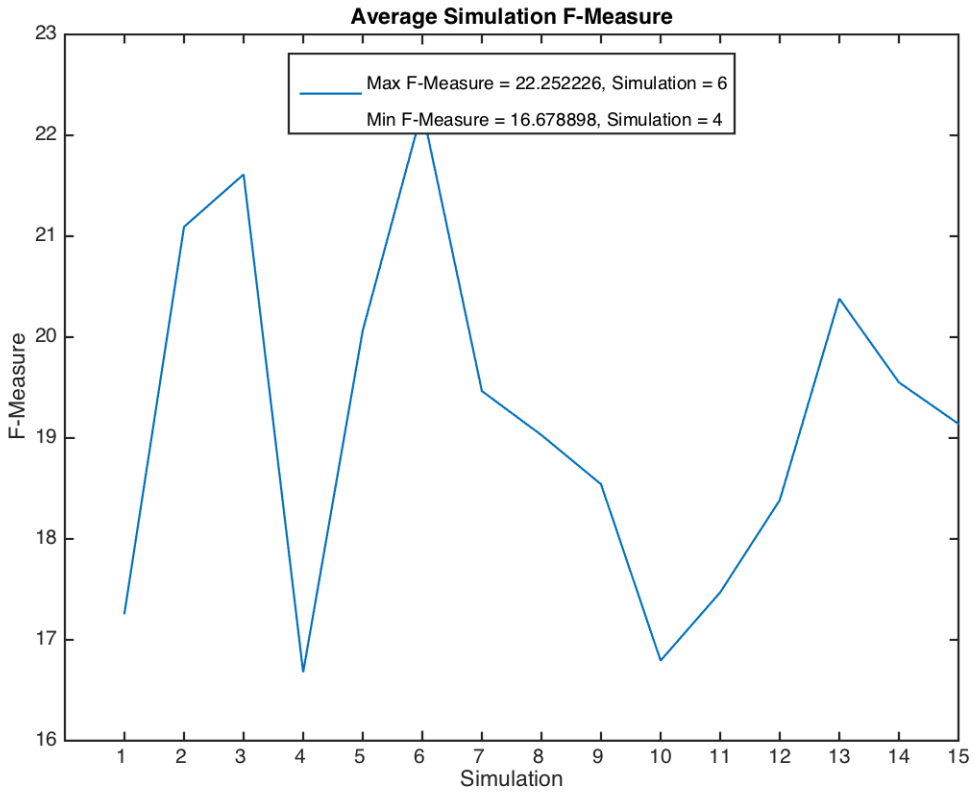

Figure 19: Feature Reduction Simulations: Average Simulation F-Measure

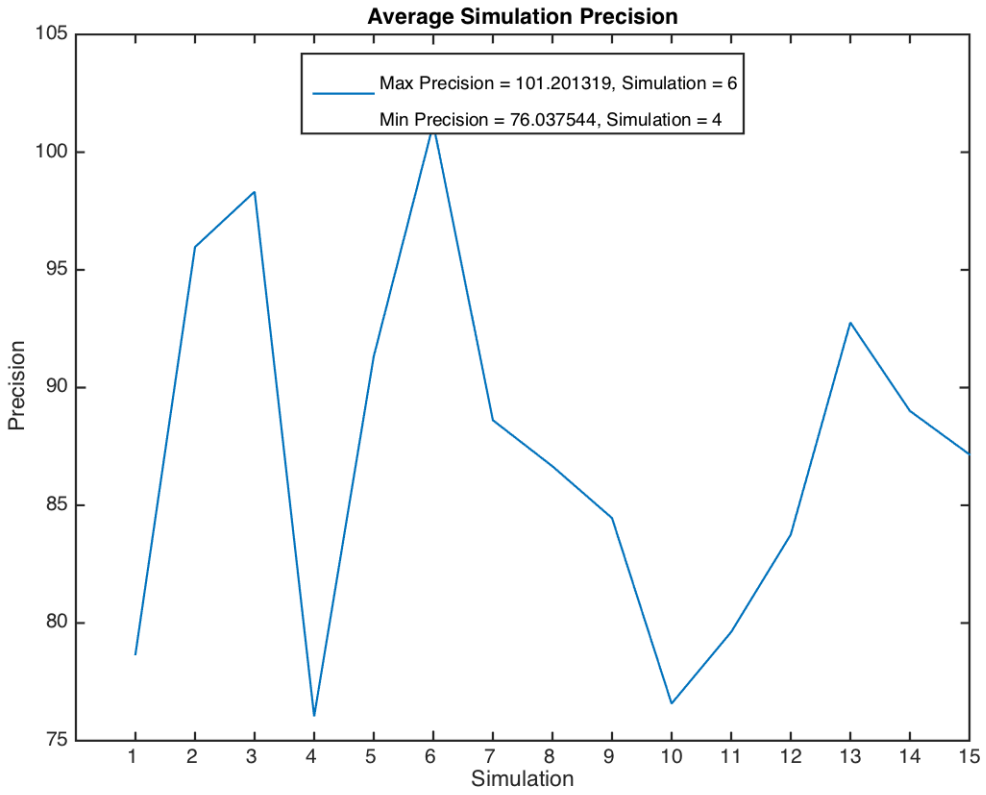

Figure 20: Feature Reduction Simulations: Average Simulation Precision

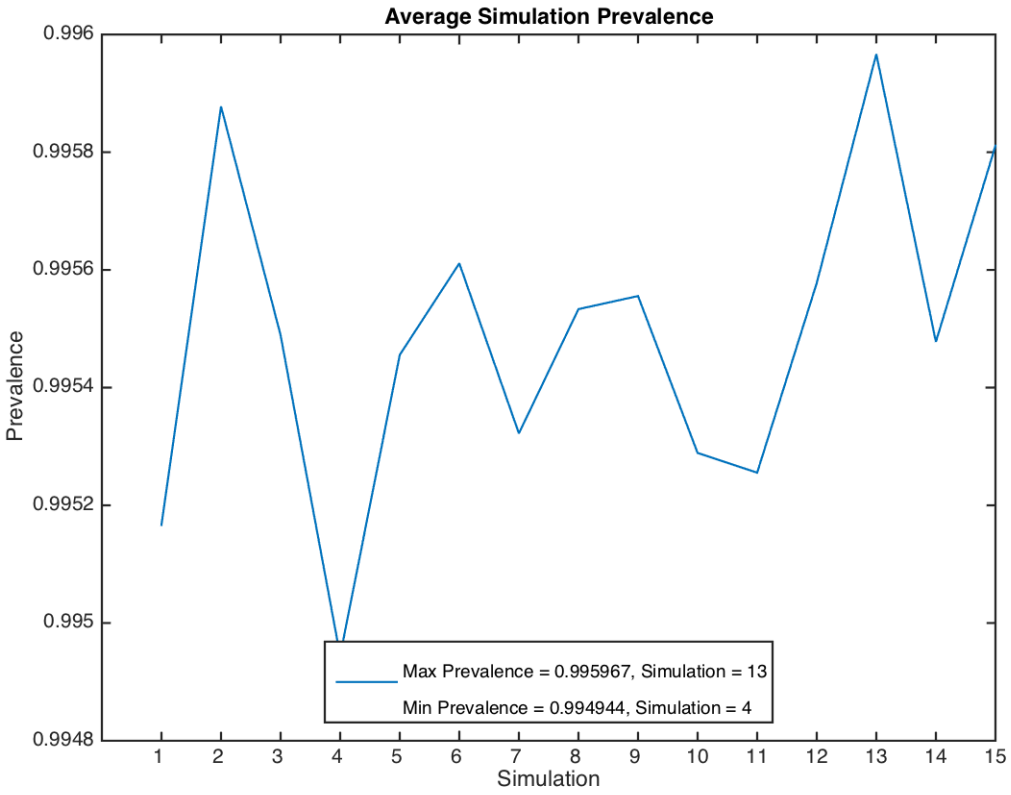

Figure 21: Feature Reduction Simulations: Average Simulation Prevalence

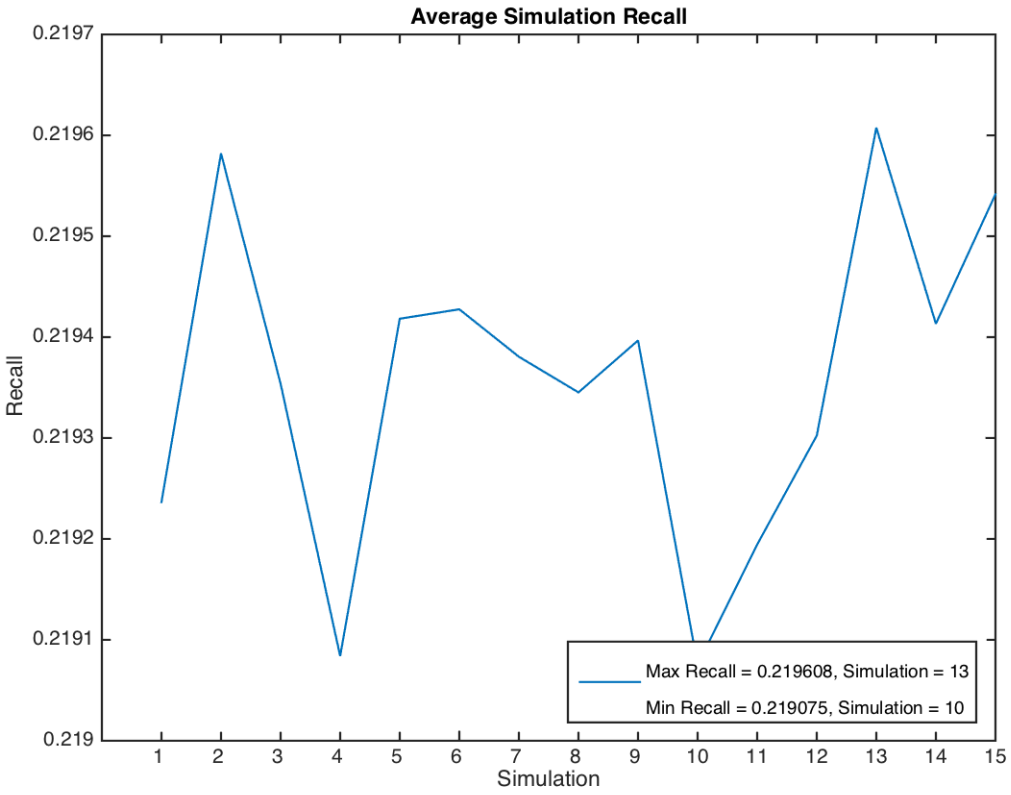

Figure 22: Feature Reduction Simulations: Average Simulation Recall

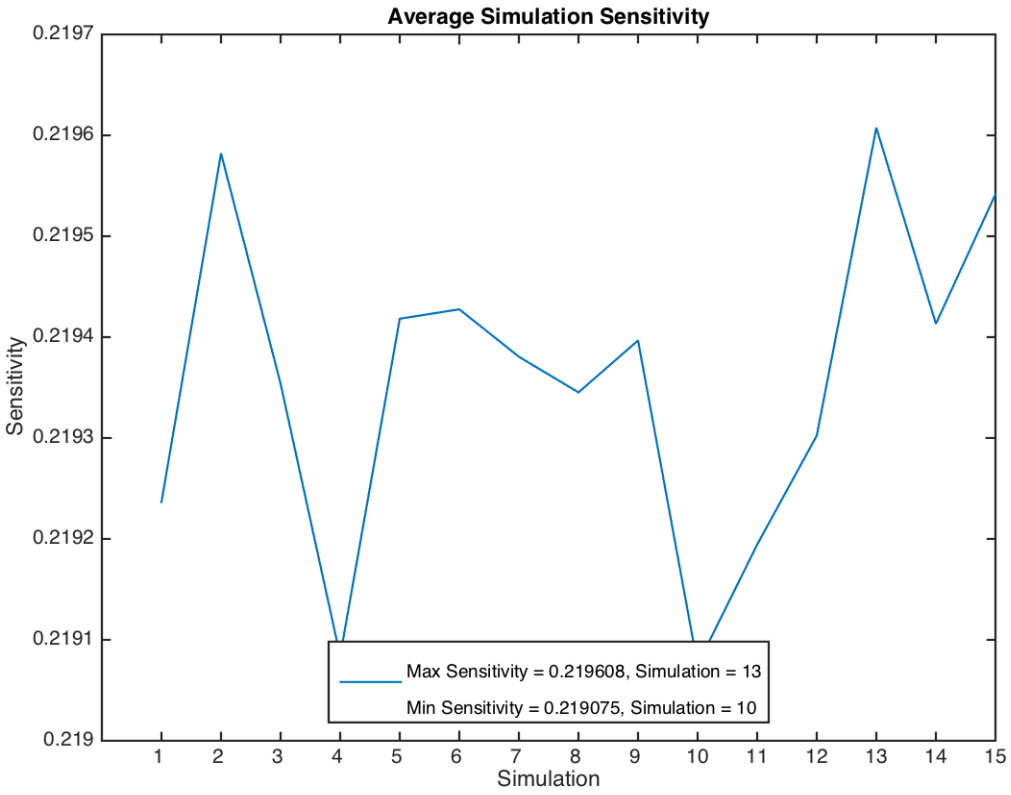

Figure 23: Feature Reduction Simulations: Average Simulation Sensitivity

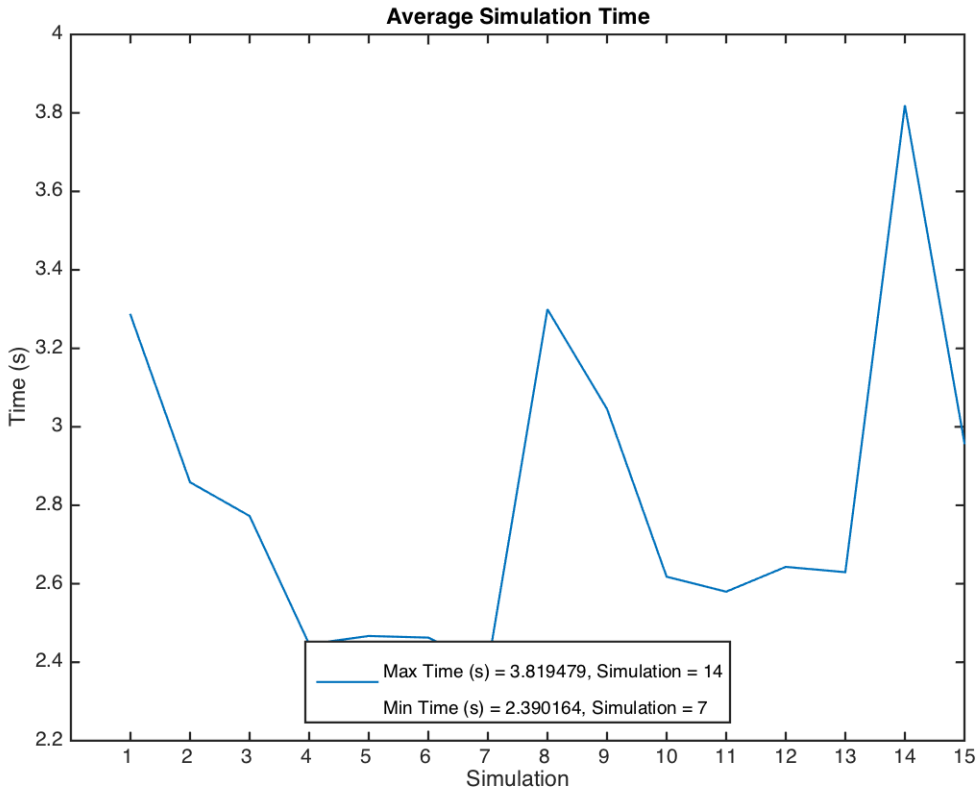

Figure 24: Feature Reduction Simulations: Average Simulation Time

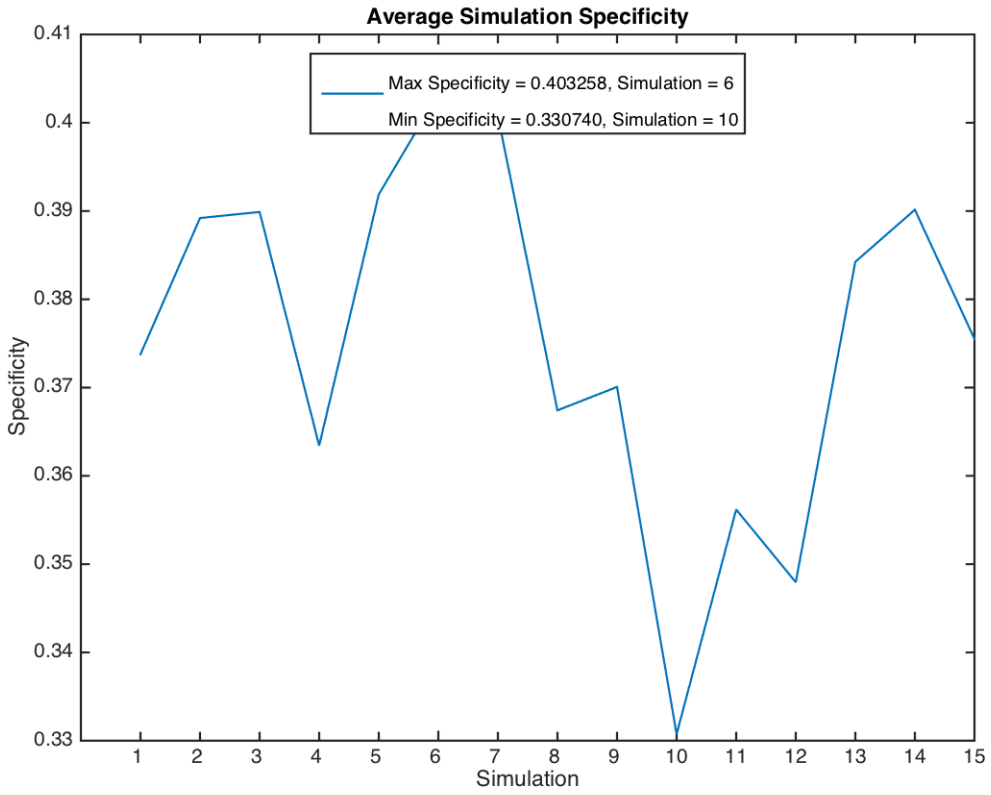

Figure 25: Feature Reduction Simulations: Average Simulation specificity

Table 4: Classifier Simulation Configurations

#	VOTER	C1_LDC	C1_EDC	C1_MDC	C1_DT	C1_SVM	C1_KNN	C2_LDC	C2_EDC	C2_MDC	C2_DT	C2_SVM	C2_KNN	C3_LDC	C3_EDC	C3_MDC	C3_DT	C3_SVM	C3_KNN
1	0	0	0	0	0	0	1												
2	1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0
3	1	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1
4	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
5	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0
6	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1
7	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0
8	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0
9	1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1
10	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
11	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0
12	0	0	0	0	0	1	0												
13	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1
14	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0
15	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0
16	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1
17	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0
18	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0
19	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0
20	0	0	0	0	1	0	0												
21	0	0	0	1	0	0	0												
22	0	0	1	0	0	0	0												
23	0	1	0	0	0	0	0												
24	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1
25	1	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1
26	1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

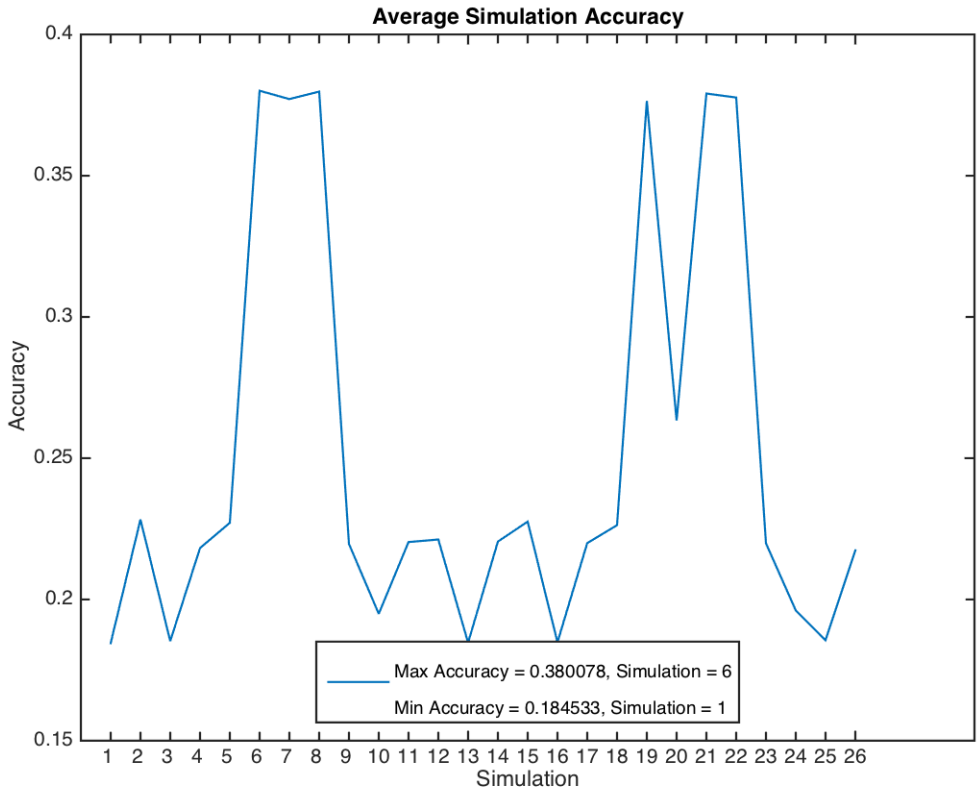

Figure 26: Classifier Simulations: Average Simulation Accuracy

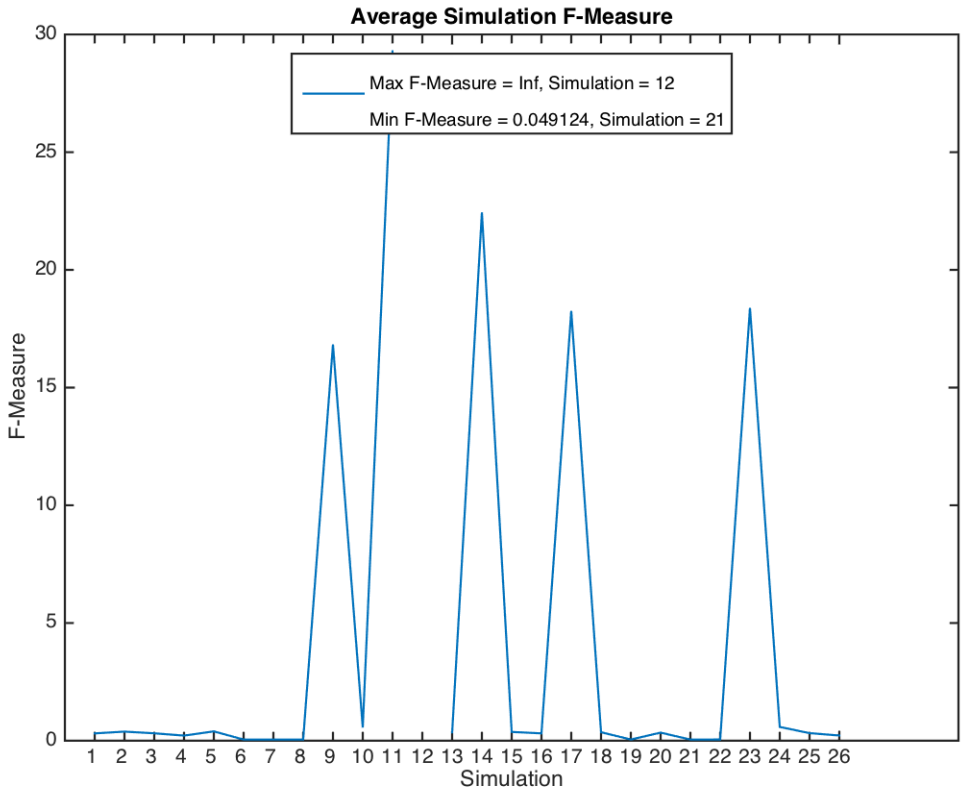

Figure 27: Classifier Simulations: Average Simulation F-Measure

5 All Simulations

Although we performed our very own "interesting" simulations, the true power of mass simulations is that they can effectively try out nearly every interesting simulation. So we did just that, we ran plenty of simulations, permuting almost all of the simulation parameters in the GUI in order to get a better understanding of how to build a good classifier for the task at hands. Unfortunately, due to the ludicrous combinatorial explosion, we could not recreate every simulation, but we do present the ones we did see through until the end

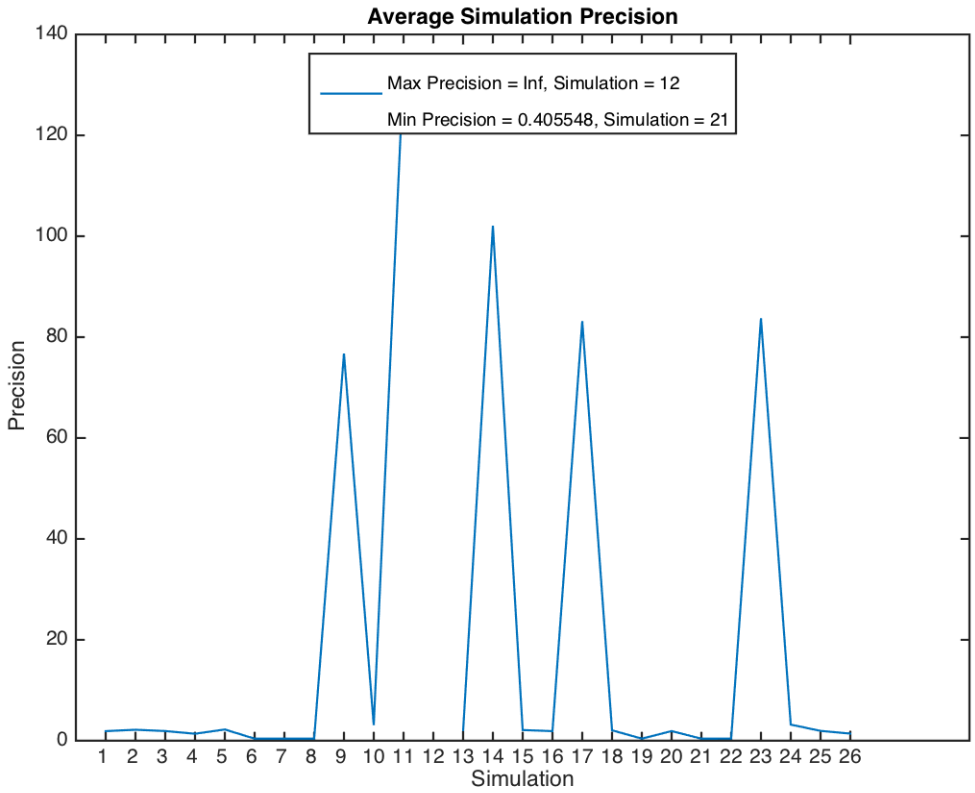

Figure 28: Classifier Simulations: Average Simulation Precision

6 Final Remarks

6.1 Conclusions

Regarding the comparison of classifier performance, it is not uncommon for a high ranking classifier in terms of accuracy to be worse than another in specificity or sensitivity, as such, selecting the best classifier is rather problematic and relative, if not entirely impossible. We did however, place more emphasis on accuracy than the other metrics in our conclusions.

Still on the topic of accuracy, it is noteworthy that although we use a variety of classifiers, they all have very similar accuracies overall. We assume this is because, although using very distinct methods to separate the classes they still have some things in common:

- the dataset is identical for all of them, even if shuffled randomly for each simulation
- they have only two classes to separate. If this was multi-dimensional classification problem the initial choices within each algorithm would undoubtedly influence further choices down the line
- all of the proposed classifiers are examples of supervised learning algorithms, meaning that they have prior knowledge of the expected outcome

Regarding the problem at hands, we can safely state that the unfortunately low number of defaulted credit card holders in comparison with the number of regular clients is the largest drawback. Even when using stratified data splitting for training and testing, the negative class has

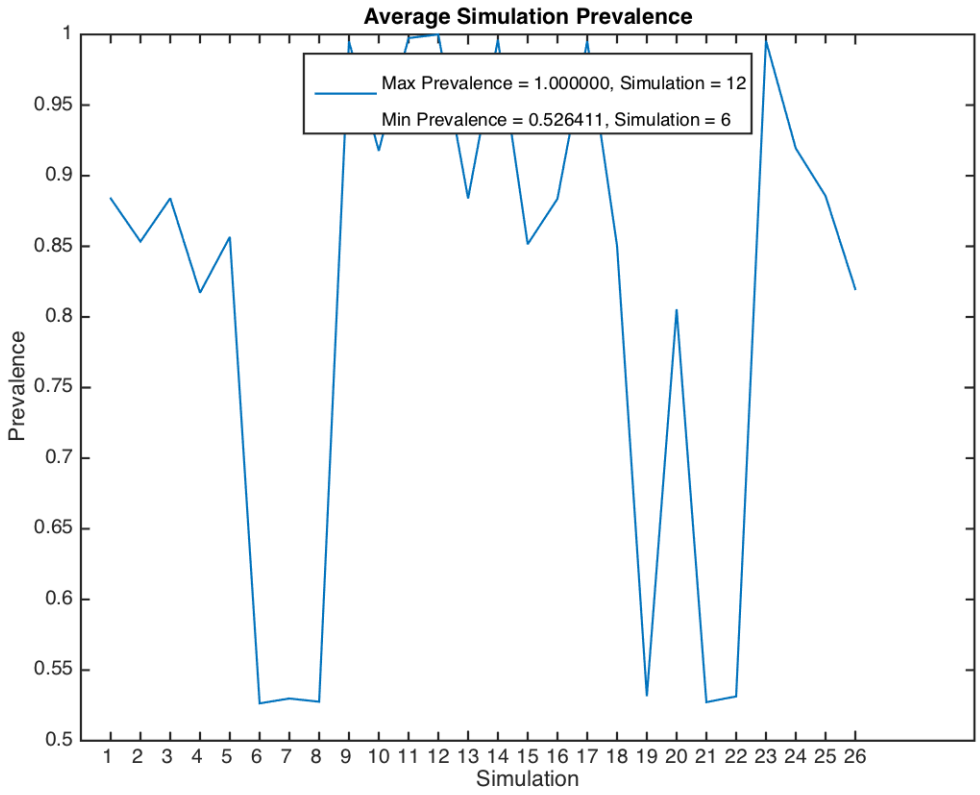

Figure 29: Classifier Simulations: Average Simulation Prevalence

too few and spread out examples to create a truly efficient classifier. This is not to say that we did not achieve classifiers that, unless they are proven overfitted, provided very good classification accuracies but rather that we feel there could be room for improvement.

6.2 Future Work

We would like to improve our simulation set, but alas, time was of the essence. Another point we would like to improve upon would be the inclusion of other non-supervised learning algorithms. Finally, the use of a more balanced dataset for academic purposes could be a nice experiment, in order to see how the data trully influences the classifier.

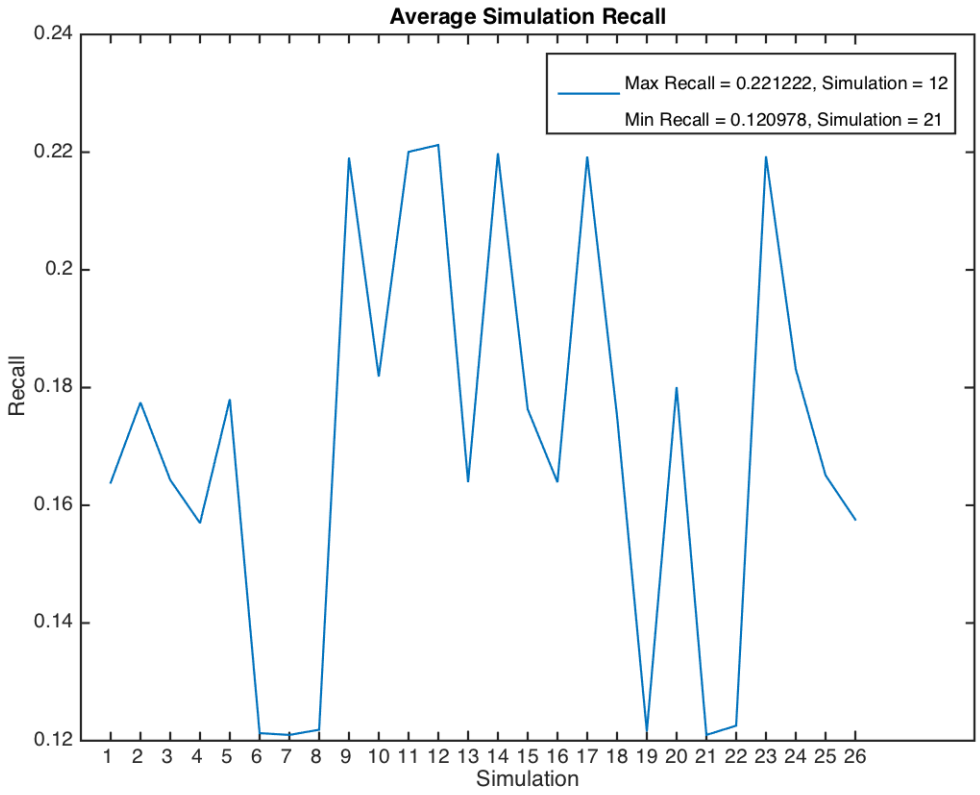

Figure 30: Classifier Simulations: Average Simulation Recall

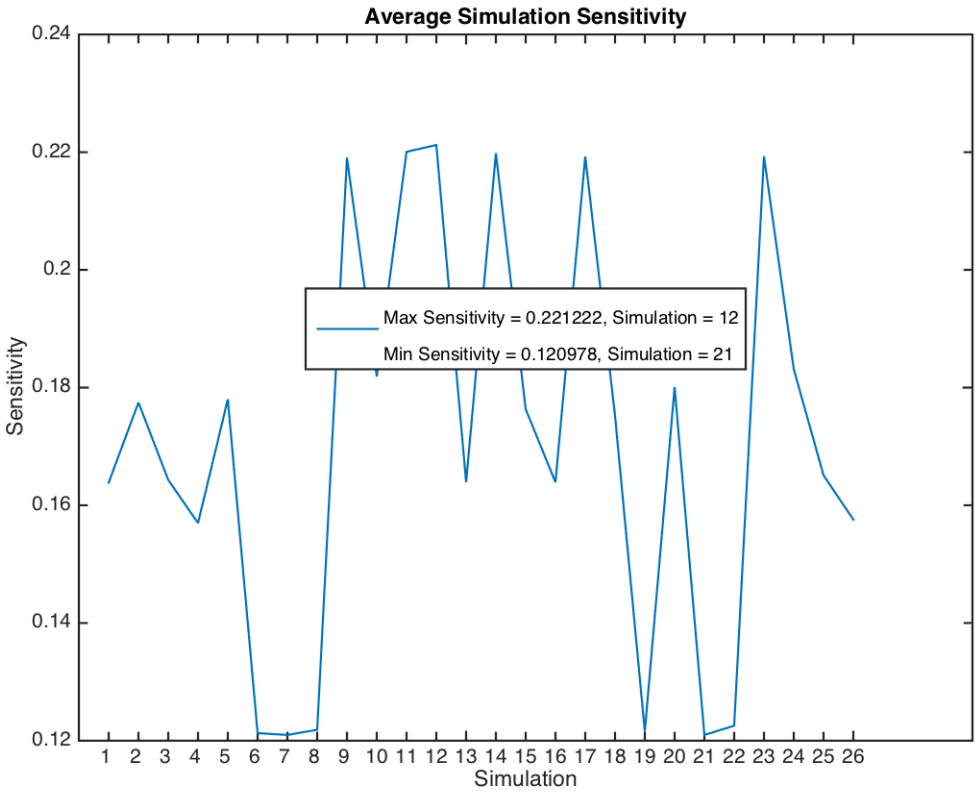

Figure 31: Classifier Simulations: Average Simulation Sensitivity

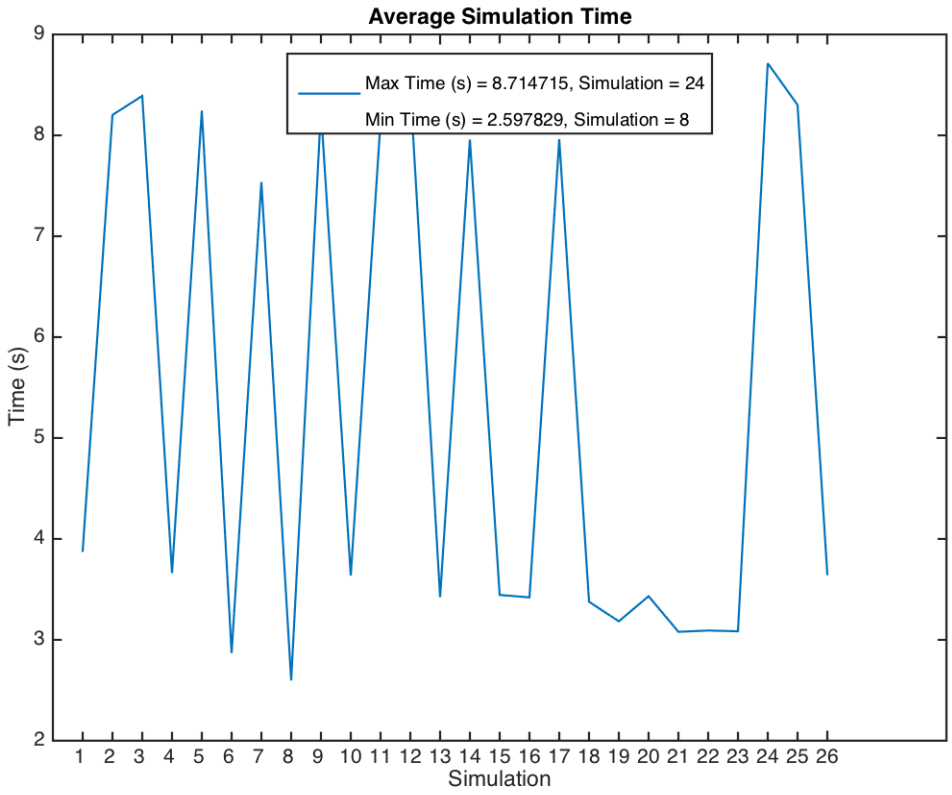

Figure 32: Classifier Simulations: Average Simulation Time

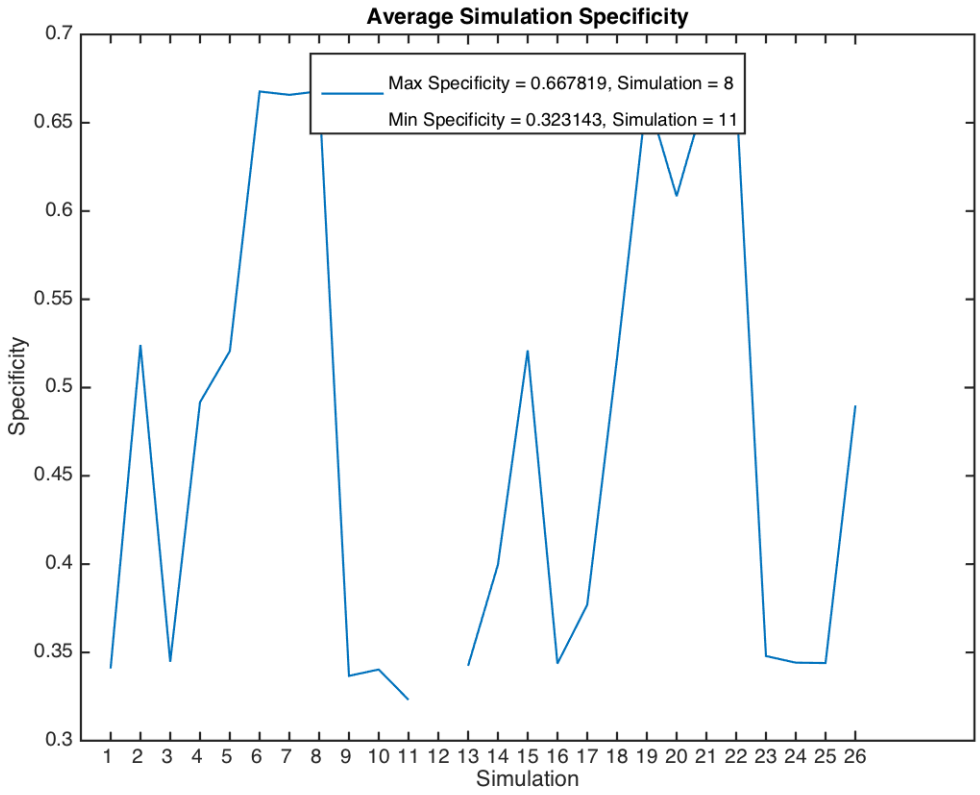

Figure 33: Classifier Simulations: Average Simulation specificity

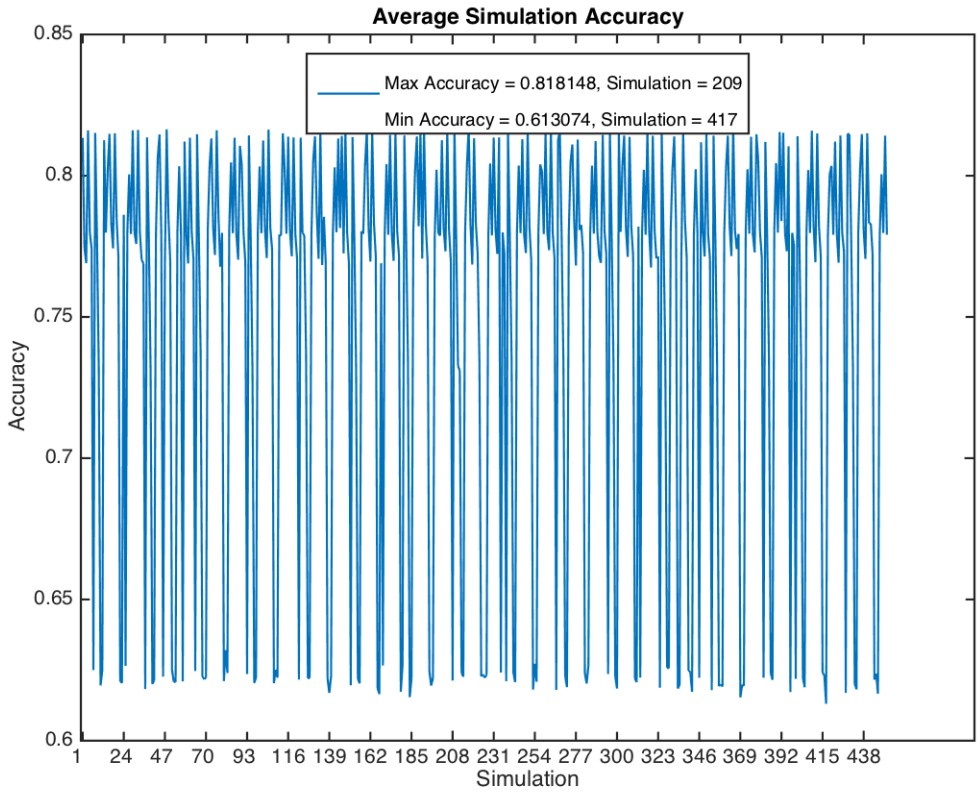

Figure 34: All Simulations: Average Simulation Accuracy

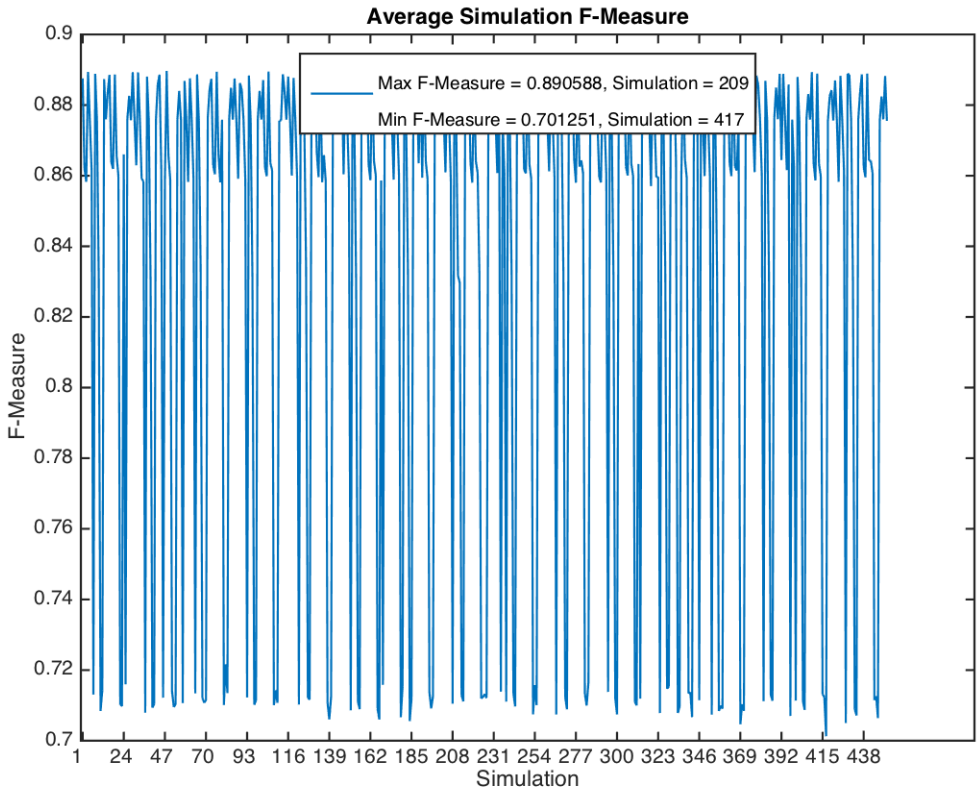

Figure 35: All Simulations: Average Simulation F-Measure

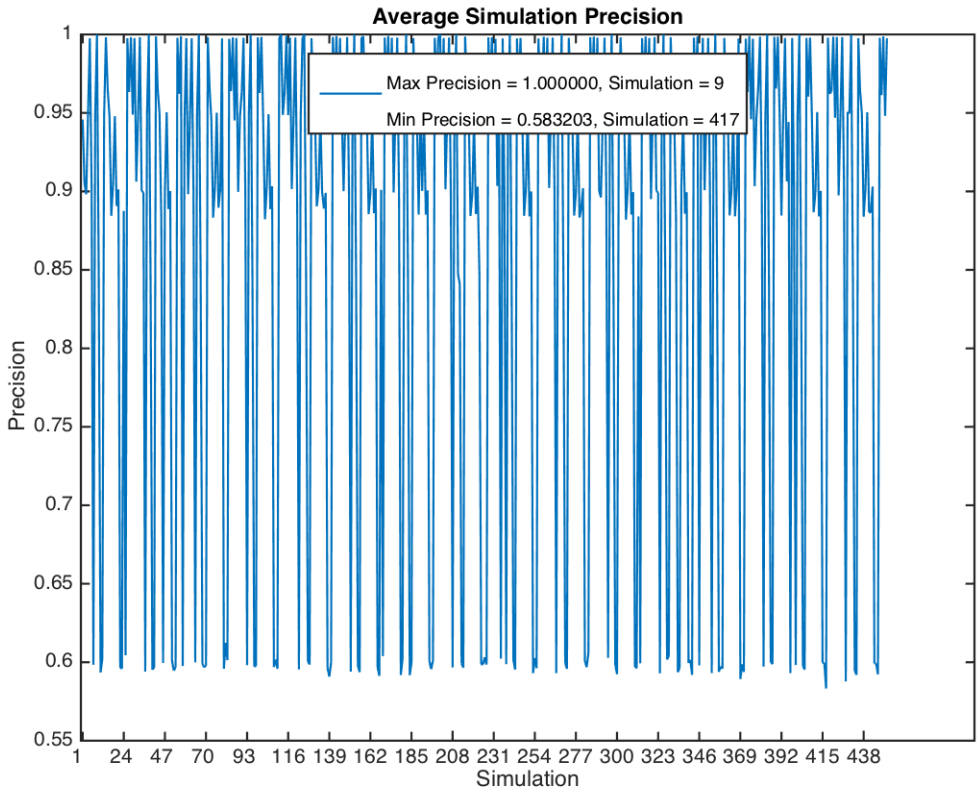

Figure 36: All Simulations: Average Simulation Precision

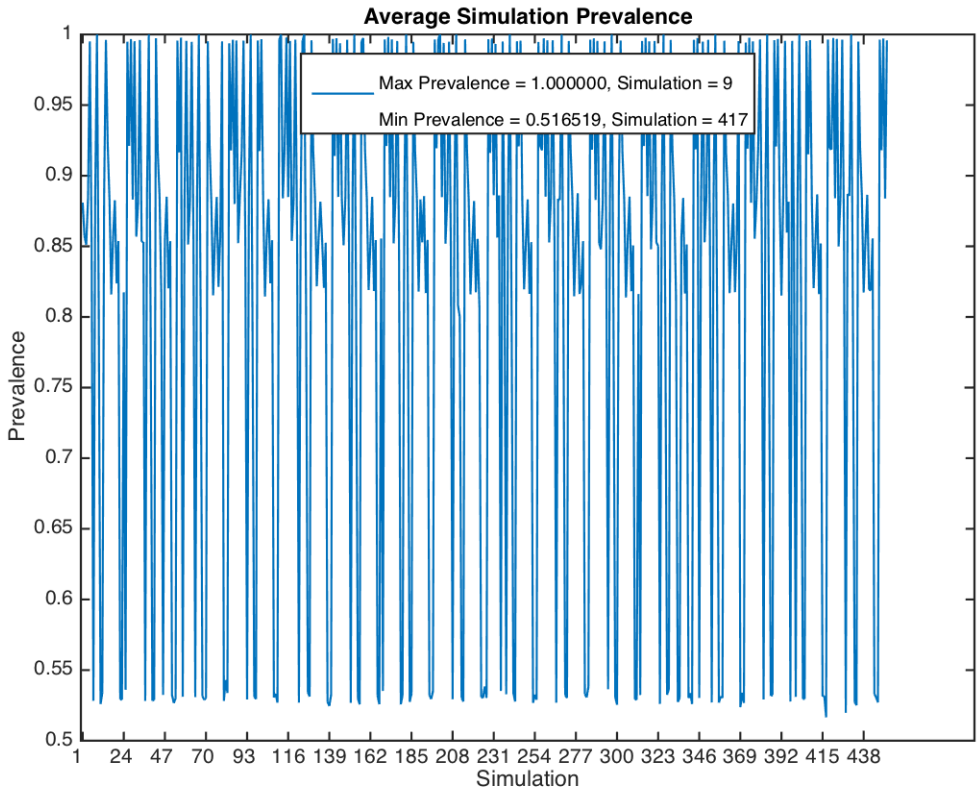

Figure 37: All Simulations: Average Simulation Prevalence

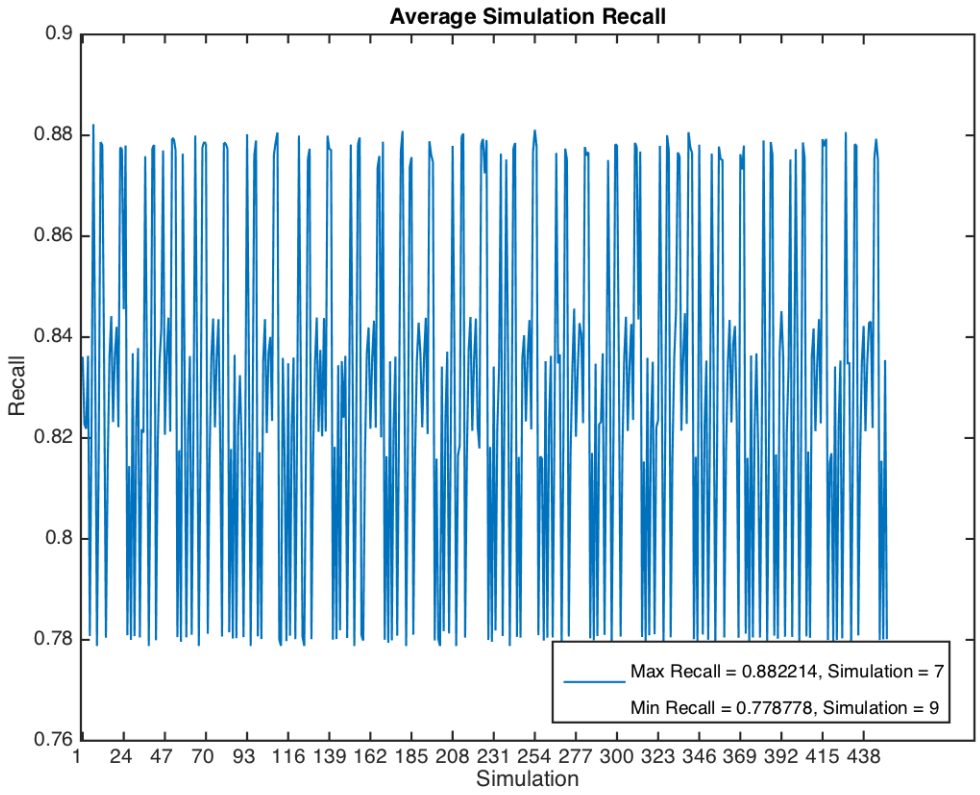

Figure 38: All Simulations: Average Simulation Recall

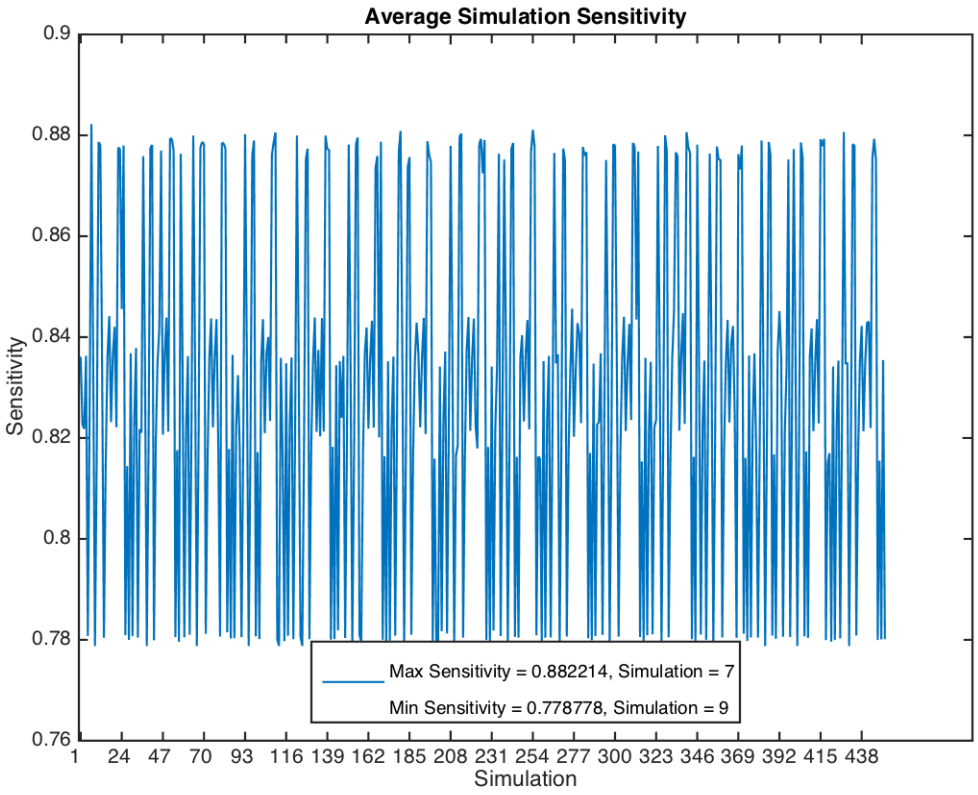

Figure 39: All Simulations: Average Simulation Sensitivity

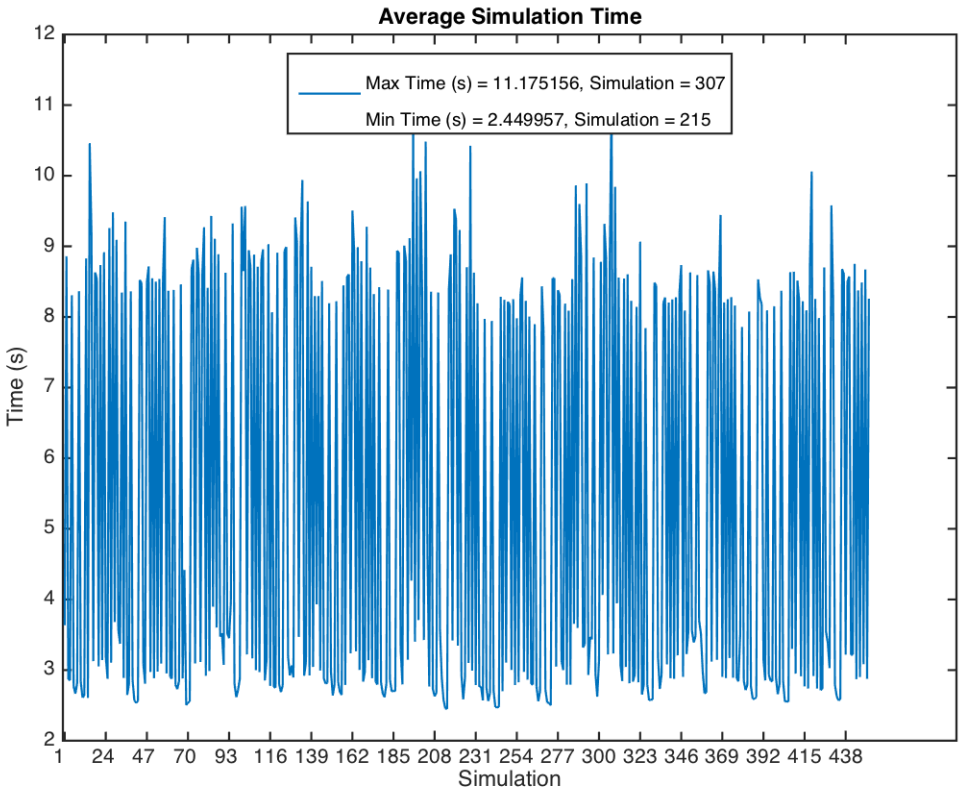

Figure 40: All Simulations: Average Simulation Time

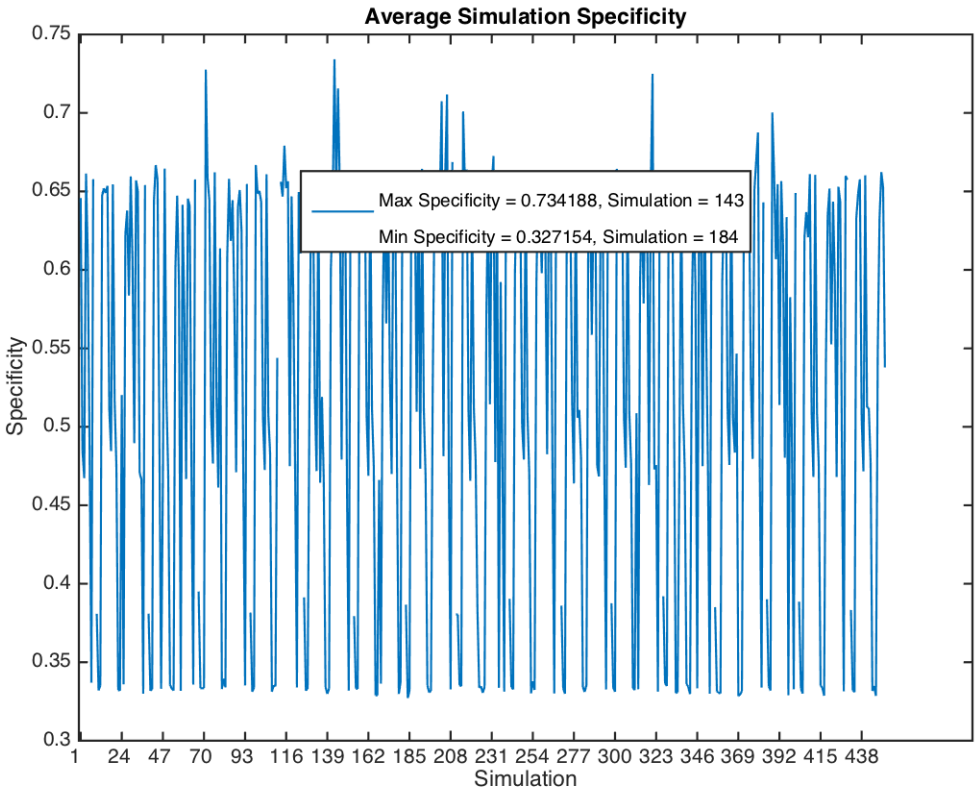

Figure 41: All Simulations: Average Simulation specificity