Exploring the Underlying Performance Profiles of Nearest Neighbor Learning Algorithms

Braeden Hunt BRAEDEN.HUNT@STUDENT.MONTANA.EDU
Department of Computer Science
Montana State University
Bozeman, MT 59715, USA

Tyler Koon TYLER.KOON@STUDENT.MONTANA.EDU
Department of Computer Science
Montana State University
Bozeman, MT 59715, USA

Editor: Braeden Hunt and Tyler Koon

Abstract

In this study, we consider the differences in performance between the k-Nearest Neighbor, k-Means Clustering, and Edited k-Nearest Neighbor learning algorithms with respect to different contexts. This is achieved through a six independent experiments which measure the average performance of the three underlying algorithms using stratified ten-fold cross validation to produce a set of evaluation metrics. Each experiment is based on one of six unique data sets which contains either classification or regression data. For each experiment, the averages of these metrics were compared across all three algorithms with the intent of identifying how each model performs with respect to the underlying context of the data. The results from these experiments offered interesting insight into the relationship between these algorithms with respect to the underlying data.

Keywords: k-Nearest Neighbor, k-NN, k-Means Cluster, Edited k-Nearest Neighbor, Edited k-NN, Cross Validation, Regression, Classification

1. Introduction

Choosing a learning algorithm is among the most critical steps when applying machine learning to a given problem. Using the right model for the task can provide more accurate predictions and result in more efficient use of computational resources. Generally, deciding on an algorithm requires extensive exploration in the context of the problem domain. This is often an arduous and time consuming task that can delay progress for the underlying project. Consequently, it is understandable that the machine learning community expends a great deal of effort to better understand the "performance envelope" of these learning algorithms—that is, proactively exploring the relative strengths and weaknesses of these models, so as to offer more extensive resources when performing model selection. Our paper contributes to this goal by considering the relative performance of three different Nearest-Neighbor algorithms: k-Nearest Neighbor, Edited k-Nearest Neighbor, and k-Means clustering.

Our study considers the question of how the performance of these three models differs with respect to variations in the underlying training data. To address this question, we considered model performance in the context of six independent and widely varied data sets divided into three regression sets and three classification sets. We defined model *performance* as a composite of the 0/1 Loss metric, F_1 score, and Squared Mean Error, which were computed for each experiment using a stratified cross-validation technique. We predicted that the Edited k-Nearest Neighbor algorithm would offer better average performance in both the classification and regression contexts compared to models trained using the k-Means or k-Nearest Neighbor algorithms, because it strikes a balance of removing noise while not smoothing the data as much as k-Means. Our offered interesting results, indicating that Edited k-Nearest Neighbor models may be less performant in the regression setting than initially predicted while, suggesting a more complex relationship between the performance of these three learning algorithms across contexts.

In the following sections, we describe the details of our study as well as discuss the results and implications of our findings, starting with a comprehensive overview of our experimental design in Section 2. This is accompanied by a presentation of our experimental results in Section 3, the implications and limitations of which are explored in Section 4. We conclude in Section 5 with a summary of our study and a brief discussion of further research that could be performed on the subject.

2. Experimental Design

In this section, we describe our approach to addressing the underlying research question. This includes an exploration of our chosen data sets (Section 2.1), a discussion of our program and its components (Section 2.2), a comprehensive overview of our evaluation process (Section 2.3), a review of the three learning algorithms under consideration (Section 2.4), and finally a detailed walk through of our experimental design (Section 2.5).

2.1 Data Sets

Our study used six independent data sets sourced from the UCI Machine Learning repository to explore our hypothesis. Of these data sets, three contained classification data and three contained regression data. The three classification data sets were the Wisconsin Breast Cancer Database (January 8, 1991), Glass Identification Database, and Small Soybean Database. The three chosen regression data sets for this study were Abalone, Computer Hardware, and Forest Fires. Each of these data sets offered a unique combination of features, sample sizes, and response values, improving the breadth of our study by affording a more diverse set of experiments. The diversity of these six data sets offered an appropriate platform to evaluate our hypothesis with respect to many different contexts in both a classification and regression setting. Accompanying this variability in the composition of these data was a number of issues that needed to be addressed through data wrangling and preprocessing. This included the need to handle categorical features of varying complexity, account for missing data, drop irrelevant features, and process unconventional data such as cyclical features. The details of these challenges and our solutions are discussed at length in Section 2.5.

2.2 Program Design

For this study, we developed a robust and scalable set of software tools to carry out our experiments. Our program was broken down into three categories: Algorithms, Evaluation Tools, and Utilities. Algorithms contains our implementations for k-Nearest Neighbor (KNN), Edited k-Nearest Neighbor (EditedKNN, EKNN), and k-Means Clustering (K-Means). Our use of Object Oriented Programming allowed us to make the rest of the program handle these models generically.

Our k-Nearest Neighbor class implements searching neighbors and predicting values, taking in generic model parameters. Our implementations of the other two algorithms utilize these implementations, with the primary difference being that they implement algorithms to prune/reduce the data being searched by our KNN class.

The Evaluation component contains our tools for evaluating model performance. These tools consist of a *CrossValidation* class and a *EvaluationMeasure* class. The *CrossValidation* class implements k-fold cross validation to compute the mean performance of a model. This includes methods for folding a data set, holding out tuning data, and collating the prediction results across all folds. Additionally, our implementation of the cross-validation algorithm offers stratified folds for both classification and regression data, offering training folds and holdout sets that represent the whole data set. The *EvaluationMeasure* class implements the performance metrics defined in Section 2.3. Each metric was defined as a class method, offering easy expandability and tunability for our experiments.

The Utilities component contains the *Preprocess*, *Utilities*, and *TuningUtility* classes. The *Preprocess* class exposes a number of functions to facilitate wrangling data, including reading in raw data, dropping missing data and irrelevant features, and saving and loading the processed data sets. Our *Utilities* class has tools for handling min-max normalization, one hot encoding, and cyclical features. Lastly, our *TuningUtility* class provided a generic framework for tuning each hyper parameter. Together, these three utility classes offered generic tools to facilitate all the experiments.

2.3 Evaluation Methodology

To compare the performance of models, we used three performance metrics. For classification data, we used both 0/1 Loss and F_β Score for $\beta = 1$. We chose $\beta = 1$ as it equally weights the the precision and recall metrics. For regression problems, we used the Mean Squared Error of the model. This metric measures the average of the squared errors. This offers a weighted approach to comparing error, providing polynomially larger addons for predictions further from the actual value.

These metrics focus on the general model performance while being simple and effective, allowing us to easily compare the cross validation results among all three models to draw conclusions with respect to the underlying context.

2.4 Algorithms

For this study, we considered three algorithms based on k-Nearest Neighbor: k-Nearest Neighbor, Edited k-Nearest Neighbor, and k-Means Clustering. These algorithms use distance functions to find similar instances in the training data to base predictions off of. We

relied on two particular distance functions. For continuous features, we used Euclidean distance. For discrete-valued features, we used One Hot Encoding.

2.5 Experiments

To answer our research question, we developed six experiments that compare the relative performance of our three learning algorithms. Each experiment implemented these algorithms for one of the data sets, allowing us to compare relative performance different contexts. Each experiment had three primary stages: Preprocessing, Tuning, and Evaluation.

The first stage involved loading the data set into memory, and preparing it. We read the data from disk, dropped irrelevant features, removed samples that were missing data, and handled cyclical features (such data and time data). Our method for handling cyclical data involved encoding each level as an object that handles modular subtraction.

The second stage in our experimental design involved tuning of the three learning algorithms. We evaluated the performance of each set of hyper parameters by using the 0-1 Loss metric. We decided to use this specific metric for its computational simplicity and speed. Lower values of k were used to break ties, which improved our overall speed. For classification problems, we only needed to tune k. To do this, we pulled out and stratified 10% of the data for tuning. The remaining data was then grouped into 10 stratified folds for testing the model. For each iteration of the cross-validation, we remove the fold from the training data and evaluated a selection of k values against the held-out tuning data. On the first fold, we evaluated $k \in [1, \sqrt{n}]$ where n is the size of the training data, and stored the best values for k". During the next iteration, we only looked at k values between the smallest and the largest "best k", plus/minus five. This allowed us to significantly reduced the size of the search.

Regression data sets offered more hyper parameters to tune. In addition to the k value for each individual fold, we had to tune the σ value used in the Gaussian Kernel, and ϵ value for Edited k-Nearest Neighbors. To accomplish this, we defined a range of ϵ and σ values with corresponding step sizes. We used this along with the k tuning algorithm to perform a near three dimensional grid search. We defined our ranges for these values based on some quick estimations using larger ranges on two-fold cross validation, greatly increasing our tuning speed. For each fold in each algorithm, we recorded the best k, σ, and ϵ (ϵ was only recorded for Edited KNN).

The final stage of our experimental design was responsible for evaluating the performance of our three learning algorithms. This was achieved using the same stratified 10-fold cross-validation from the previous step. For each fold in the validation, we applied a min-max normalization technique to bound the feature values between 0 and 1. We performed this normalization step for each individual fold to ensure that the folds remained independent. We used the same hyper parameters for *each* fold that returned from the tuning process. The prediction results for each fold in the cross validation were recorded and later used to compute the 1-0 Loss, F_1 Score, and Mean Square Error evaluation metrics described in Section 2.3. This cross-validation process was repeated for each of the three learning algorithms, offering a quantitative measure of the mean performance for each algorithm.

3. Results

This section provides additional context for the underlying experiments, and reports the results obtained in the study.

3.1 Breast Cancer Identification Results

The Wisconsin Breast Cancer Database (January 8, 1991) has instances of breast tumors. Each instance has nine continuous features, ranging from 1-10. These nine features were used to determine the distance between instances. The tenth feature is a binary representation of the class—whether or not the tumor is malignant or benign. For this data set, we considered malignant to be "positive" and benign as "negative". There are a total of 699 instances in the data set: 241 malignant, and 458 benign. There are 16 instances missing a single attribute. There are no instances missing two or more attributes. All instances missing attributes are dropped during the preprocessing step. For the F_β score, we used *Malignant* as the positive class. The results from the experiment are as follows:

KNN				
Fold	k	Tuning 0/1 Loss	Testing 0/1 Loss	Testing F_β
1	1	0.971	0.935	0.909
2	2	0.971	0.968	0.955
3	3	0.986	0.967	0.955
4	1	0.971	0.984	0.976
5	2	0.986	0.967	0.955
6	2	0.971	0.983	0.977
7	5	0.971	1.000	1.000
8	2	0.986	0.935	0.905
9	2	0.971	1.000	1.000
10	2	0.986	0.952	0.930
Avg.	2.2	0.977	0.969	0.956

EKNN				
Fold	k	Tuning 0/1 Loss	Testing 0/1 Loss	Testing F_β
1	1	0.957	0.967	0.952
2	2	0.957	1.000	1.000
3	2	0.986	0.984	0.977
4	1	0.957	0.967	0.952
5	2	0.971	0.968	0.955
6	2	0.957	0.952	0.930
7	2	0.957	0.952	0.930
8	2	0.971	0.984	0.976
9	2	0.957	0.952	0.933
10	3	0.971	0.967	0.955
Avg.	1.9	0.964	0.969	0.956

K-Means				
Fold	k	Tuning 0/1 Loss	Testing 0/1 Loss	Testing F_β
1	4	0.971	0.951	0.930
2	4	0.971	0.984	0.976
3	1	0.971	0.983	0.977
4	1	0.971	0.952	0.930
5	4	0.971	0.935	0.909
6	4	0.971	1.000	1.000
7	6	0.971	0.952	0.930
8	3	0.971	0.967	0.955
9	1	0.957	0.952	0.930
10	3	0.986	0.968	0.952
Avg.	3.1	0.971	0.964	0.949

Table 1: Experimental results for the Breast Cancer Identification data set

3.2 Soybean Identification Results

The Small Soybean Database describes 47 instances of soybeans using 35 features. The soybeans are classified into four groups: D1 (10 instances), D2 (10 instances), D3 (10 instances), and D4 (17 instances). D1 is considered the positive class for F_β, and there are no missing attributes in the data. The results from the experiment are as follows:

KNN				
Fold	k	Tuning 0/1 Loss	Testing 0/1 Loss	Testing F_β
1	3	1.0	1.0	1.0
2	3	1.0	1.0	1.0
3	3	1.0	1.0	1.0
4	3	1.0	1.0	1.0
5	3	1.0	1.0	1.0
6	1	1.0	1.0	1.0
7	3	1.0	1.0	1.0
8	3	1.0	1.0	1.0
9	3	1.0	1.0	1.0
10	3	1.0	1.0	1.0
Avg.	2.8	1.0	1.0	1.0

EKNN				
Fold	k	Tuning 0/1 Loss	Testing 0/1 Loss	Testing F_β
1	3	1.0	1.0	1.0
2	1	1.0	1.0	1.0
3	3	1.0	1.0	1.0
4	3	1.0	1.0	1.0
5	3	1.0	1.0	1.0
6	1	1.0	1.0	1.0
7	3	1.0	1.0	1.0
8	3	1.0	1.0	1.0
9	3	1.0	1.0	1.0
10	3	1.0	1.0	1.0
Avg.	2.6	1.0	1.0	1.0

K-Means				
Fold	k	Tuning 0/1 Loss	Testing 0/1 Loss	Testing F_β
1	1	1.0	1.0	1.0
2	1	1.0	1.0	1.0
3	1	1.0	1.0	1.0
4	1	1.0	1.0	1.0
5	1	1.0	1.0	1.0
6	1	1.0	1.0	1.0
7	1	1.0	1.0	1.0
8	1	1.0	1.0	1.0
9	1	1.0	1.0	1.0
10	1	1.0	1.0	1.0
Avg.	1.0	1.0	1.0	1.0

Table 2: Experimental results for the Soybean data set

3.3 Glass Identification Results

The Glass Identification Database describes 214 instances of glass fragments. Each instance has nine continuous features one one class attribute. None of the instances had missing attributes. The glass has seven classes labeled 1-7. The F_β score uses class '1' as the positive class. The results from the experiment are as follows:

KNN				
Fold	k	Tuning 0/1 Loss	Testing 0/1 Loss	Testing F_β
1	1	0.826	0.667	0.833
2	1	0.826	0.750	0.909
3	1	0.826	0.688	0.857
4	1	0.826	0.818	0.727
5	1	0.826	0.556	0.600
6	1	0.783	0.895	0.800
7	2	0.870	0.600	0.500
8	1	0.826	0.700	0.667
9	3	0.826	0.619	0.700
10	1	0.826	0.619	0.500
Avg.	1.3	0.826	0.691	0.709

EKNN				
Fold	k	Tuning 0/1 Loss	Testing 0/1 Loss	Testing F_β
1	1	0.652	0.812	1.000
2	1	0.87	0.842	0.800
3	1	0.913	0.650	0.706
4	1	0.739	0.444	0.667
5	1	0.870	0.500	0.556
6	1	0.826	0.688	0.800
7	1	0.696	0.611	0.667
8	1	0.870	0.667	0.667
9	1	0.826	0.667	0.700
10	1	0.739	0.800	0.714
Avg.	1.0	0.800	0.668	0.728

K-Means				
Fold	k	Tuning 0/1 Loss	Testing 0/1 Loss	Testing F_β
1	1	0.696	0.812	1.000
2	1	0.783	0.667	0.667
3	1	0.783	0.688	0.750
4	1	0.826	0.500	0.667
5	1	0.783	0.737	0.714
6	1	0.783	0.600	0.533
7	2	0.826	0.727	0.714
8	1	0.739	0.571	0.571
9	1	0.826	0.650	0.571
10	1	0.783	0.524	0.556
Avg.	1.1	0.783	0.648	0.674

Table 3: Experimental results for the Glass Identification data set

3.4 Forest Fires

The Forest Fires data set describes 517 instances, none of which contain missing feature values. There are a total of 13 attributes, two of which are cyclical. The rest are continuous values that represent the conditions of the recorded fire. We attempt to calculate the burned area of the forest from these values. Note that the data set is highly skewed towards the burned area being zero. The results from the experiment are as follows:

KNN				
Fold	k	σ	Tuning MSE	Testing MSE
1	1	1	14.756	24610.331
2	2	10	355.587	858.039
3	1	1	3.122	2340.314
4	1	1	38.010	2118.529
5	1	5	23.291	12272.23
6	1	5	3.305	1899.609
7	2	2	229.562	4298.4
8	1	1	17.921	1146.472
9	1	1	27.221	3609.704
10	1	9	79.253	441.839
Avg.	1.2	3.6	79.203	5359.547

EKNN						
Fold	k	σ	ϵ	Tuning MSE	Testing MSE
1	6	3	15	245.052	32999.928
2	2	3	15	244.985	25897.847
3	2	3	15	244.985	9790.302
4	2	3	15	245.248	2571.481
5	7	3	15	912.968	4192.345
6	1	3	15	246.891	25727.687
7	2	3	15	244.965	40428.953
8	2	3	15	244.985	17493.041
9	2	3	15	244.985	2343.35
10	3	3	15	428.847	27103.989
Avg.	2.9	3.0	15.0	330.391	18854.892

K-Means				
Fold	k	σ	Tuning MSE	Testing MSE
1	2	0.005	16.254	15271.871
2	1	0.005	13.139	640.858
3	1	0.005	12.137	543.146
4	1	0.005	1.890	1915.529
5	1	0.005	353.794	2730.321
6	1	0.005	0.126	138.789
7	1	0.005	0.051	358.153
8	1	0.005	11.753	1925.598
9	1	0.005	10.539	1617.479
10	1	0.005	4.338	31459.963
Avg.	1.1	0.005	42.402	5660.171

Table 4: Experimental results for the Forest Fires data set

3.5 Computer Hardware

The Computer Hardware data set describes 209 instances, none of which are missing feature values. There are a total of 10 attributes, two of which were dropped due to being identifiers. The rest are continuous values that describe computer hardware. We attempt to calculate the Estimated Relative Performance (ERP) from these values. Note that for k-Means, we used 20 as the number of clusters for each fold, as EKNN reduced the data sets down to 20 for each of the folds. The results from the experiment are as follows:

KNN				
Fold	k	σ	Tuning MSE	Testing MSE
1	6	0.001	46.6	207
2	5	0.001	32.2	994
3	1	0.001	21.0	347
4	5	0.001	18.2	396
5	1	0.001	43.9	158
6	5	0.001	31.7	319
7	7	0.001	52.5	617
8	6	0.001	36.6	196
9	6	0.001	28.8	460
10	6	0.005	35.5	3154
Avg.	4.8	0.0014	34.7	685

EKNN						
Fold	k	σ	ϵ	Tuning MSE	Testing MSE
1	4	0.005	1	976	2196
2	4	0.005	1	976	3046
3	4	0.005	1	976	16184
4	4	0.005	1	976	431
5	4	0.005	1	976	973
6	4	0.005	1	961	21623
7	4	0.005	1	976	3550
8	1	0.005	1	98	1063
9	4	0.005	1	976	3481
10	2	0.015	1	1160	3233
Avg.	3.5	0.006	1	905	5358

K-Means				
Fold	k	σ	Tuning MSE	Testing MSE
1	2	0.005	68.3	287
2	5	0.005	36.9	652
3	1	0.012	21.0	347
4	5	0.005	18.8	260
5	1	0.015	43.9	158
6	5	0.005	34.2	343
7	5	0.005	63.5	713
8	6	0.005	39.3	251
9	6	0.005	35.7	568
10	6	0.005	35.5	3154
Avg.	4.2	0.005	39.7	673

Table 5: Experimental results for the Computer Hardware data set

3.6 Abalone

The Abalone data set describes 4177 instances, none of which are missing feature values. There are a total of 9 attributes. One is a nominal value ('sex') that we implemented one-hot-encoding for. The rest are continuous values about the size and weight of the abalones. We attempt to calculate the abalone rings (which corresponds to their age) from these values. Note that for k-Means, we used 100 as the number of clusters for each fold, as Edited k-Nearest Neighbor reduced the data sets down to 100 for each of the folds. Due to the very large size of this data set, we were not able to process and tune our hyper

parameters for the entirety of the data set. We elected to reduce the data set down to a quarter of its original size. We used our stratification process to pull out this subset to ensure that we still have all the classes in their original proportions. The results from the experiment are as follows:

KNN				
Fold	k	σ	Tuning MSE	Testing MSE
1	1	1	1.082	2.99
2	1	1	0.809	2.8
3	1	1	1.127	2.8
4	1	1	0.873	3.684
5	1	1	0.655	2.267
6	1	1	0.727	3.933
7	1	1	0.882	2.944
8	1	1	0.891	2.533
9	1	1	0.691	4.289
10	1	1	0.718	3.889
Avg.	1	1	0.845	3.213

EKNN						
Fold	k	σ	ϵ	Tuning MSE	Testing MSE
1	20	0.005	0.5	2.522	2.105
2	25	0.005	0.5	2.519	2.189
3	25	0.005	0.5	2.481	2.062
4	25	0.005	0.5	2.516	2.114
5	25	0.005	0.5	2.519	1.650
6	25	0.005	0.5	2.519	2.225
7	25	0.005	0.5	2.527	2.249
8	25	0.005	0.5	2.519	2.322
9	25	0.005	0.5	2.519	1.951
10	20	0.005	0.5	2.390	2.802
Avg.	24	0.005	0.5	2.503	2.167

K-Means				
Fold	k	σ	Tuning MSE	Testing MSE
1	1	0.005	1.082	2.990
2	1	0.005	0.809	2.800
3	1	0.005	1.127	2.800
4	1	0.005	0.873	3.684
5	1	0.005	0.655	2.267
6	1	0.005	0.727	3.933
7	1	0.005	0.882	2.944
8	1	0.005	0.891	2.533
9	1	0.005	0.691	4.289
10	1	0.005	0.718	3.889
Avg.	1	0.005	0.845	3.213

Table 6: Experimental results for the Abalone data set

4. Exploration of Results

We now consider the results of these experiments and any conclusions we can draw relative to our three algorithms. Additionally, we consider limitations of the study and propose future research that could be done in this area of study.

4.1 Discussion

Initially, we predicted that models trained using the Edited k-Nearest Neighbor algorithm would perform better in both the classification and regression setting compared to models trained using k-Nearest Neighbor and k-Means. This hypothesis was based on Edited k-Nearest Neighbor's ability to remove noisy data from the training process, which we believed would generally improve the resulting model. Results from the Breast Cancer, Soybean, and Glass Identification experiments suggest that this prediction is accurate in the classification setting. Our results showed the Edited k-Nearest Neighbor algorithm reported better 0-1 loss and F_1 scores than that of the k-Means algorithm across all three classification data sets. Surprisingly, it appears that the k-Nearest Neighbor algorithm offers comparable performance in these contexts, reporting 0-1 loss and F_1 scores similar to those of k-Nearest Neighbor. We contributed this similarity in scores to these data sets containing a minimal amount of noise, effectively making Edited k-Nearest Neighbor identical to k-Nearest Neighbor.

Our results for the Forest Fires, Computer Hardware, and Abalone experiments proved to be more interesting, indicating that our original prediction does not hold up in the regression setting. In these experiments, the k-Nearest Neighbor and k-Means Clustering algorithms tended to provide significantly smaller mean square error compared to the Edited k-Nearest Neighbor algorithm. The Abalone experiment deviated from these results,

however this experiment was subject to alterations which may have tarnished the integrity of the results. We believe the lower performance from the Edited k-Nearest Neighbor algorithm to have been caused by the sparse training data produced from the editing process. The effects of this sparse data would have profound effects on the average *weighted* distance produced by the Gaussian kernel function, especially for data containing outliers. This effect is amplified when there are few training examples on the upper end of the regression values, as mean squared errors produce out-sized error results. Our models performed well when predicting values in the single and double digits, but not so well with values with three or four digits, and these (proportional) differences in the upper values dominate the resulting mean squared error metric.

4.2 Limitations

Though the results of this study offer interesting insight into the relative performance between nearest-neighbor based learning algorithms, it should be noted that there were a number of limitations which warrant additional exploration of the underlying research question. In particular, the range of hyper parameters considered in the tuning processes were limited, resulting in less-than-optimal tuned parameters. This was largely the result of time constraints imposed on the project, requiring us to expedite the inherently time intensive tuning process. The effects of this limitation are likely related to the reportedly large mean standard error in many of our regression experiments. Further experimentation that considers larger range of hyper parameters should be performed so as to offer more accurate insight into the performance of the underlying algorithms.

An additional limitation of our study was the small number of experiments performed. Using only three data sets each for the regression and classification settings offers limited generalization of our results. More research should be performed that considers additional contexts so as to provide a more holistic and accurate representation of the underlying performance and behavior of these algorithms. Similarly, additional exploration of using distance functions beyond just Euclidean and One-Hot Encoding within these contexts would contribute to a more robust comparison of these algorithms performance.

5. Summary

Access to information on the performance and behavior of learning algorithms in diverse contexts offers resources to help streamline the often lengthy and complex model selection process for applied machine learning projects. Our research contributes to the expansion of these resources by exploring behavior of the k-Nearest Neighbor, Edited k-Nearest Neighbor, and k-Means learning algorithms in a variety of situations. In particular, we considered changes in relative model performance between these three algorithms in the context of six independent data sets describing both the classification and regression setting. For each of these data sets we developed an experiment which independently tuned and trained the three learning algorithms on the underlying data. These experiments implemented stratified ten-fold cross validation to produce average 1-0 Loss, F_1 Scores, and Mean Square Error metrics for each model, offering a robust and standardized means of comparing relative performance. The results from these experiments indicated that the k-Nearest Neighbor and Edited k-Nearest Neighbor learning algorithms tend to outperform the k-Means Clustering

algorithm in the classification context. Conversely, the k-Means Clustering and k-Nearest Neighbor appeared to offer better performance in regression settings. These results contradicted our initial prediction in which we believed the Edited k-Nearest Neighbor algorithm would offer the most robust performance between contexts. Though these results appear to offer a conclusion to our original research question, there were many limitations with our experiments which warrant additional research on the subject.

References

We used the course materials provided to us by Dr. Sheppard.