Journal of Machine Learning Research 1 (2020) 1-7 Submitted 9/27

Nearest Neighbor Classifiers and Regressors

Jieyi Chen

September 27, 2020

1. Introduction

This project applies various nearest neighbor methods on six datasets from the UCI Machine Learning Repository [1]. Specifically, we apply the k-nearest neighbor algorithm (k-NN), edited k-nearest neighbor (e-NN) and condensed k-nearest neighbor (c-NN) on both classification and regression datasets [2]. These nonparametric algorithms have no assumption on the parameters of the population. The main concept of the algorithm is to make prediction of an instance using its closest neighbors' information. The difference between nearest neighbor classification and regression model is that the classification algorithm uses a plurality vote to determine the class while the regression algorithm applies a Gaussian radial basis function kernel to make predictions.

The accuracy of a classifier can be improved by reducing the size of the training dataset because the model's performance can be influenced by the noise. The edited and condensed k-nearest neighbor models are two modified nearest neighbor methods that can help solving this problem. The edited k-nearest neighbor leverages the idea of k-NN and edit the training set to improve the algorithm. The model can either focuses on instances that have bad performance or the ones that perform well. In addition, the editing process can be incremental or in batches. The algorithm keeps removing the instances until the classification errors on the validation set starts to increase or no further points can be removed. For this project, we use incremental editing process to remove observations from a training set that are misclassified using the rest of the instances. The condensed k-nearest neighbor algorithm, a model that reduces the size of the training set by only including observations that add value to the prediction. It incrementally adds instance to a set Z where the instance added are misclassified using the existing datapoints in Z. Instead of using the training set, the algorithm feeds the set Z to make predictions. We expect the condensed k-nearest neighbor has the lowest classification and mean square error, followed by the k-NN and edited k-nearest neighbor. In this project, we use three classification and regression datasets to test this hypothesis.

The rest of the paper is organized as follows. Section 2 describes the algorithms that are implemented. Section 3 discusses the experimental approach. Section 4 presents the results of the experiments. Section 5 discusses the results. Section 6 concludes.

2. Algorithm Implementation:

2.1 Five-fold Stratified Cross-Validation:

Five-Fold Stratified Cross Validation is an out-of-sample method that examines the algorithm's performance in an independent dataset [3]. Stratification makes sure that each fold of the dataset is a good representation of the population. First, it calculates the number of observations in each class of the dataset. Then, it evenly divides the datasets into five subsets while maintaining the same number of instances in each class across five partitions. Then, it iteratively uses four subsets for training and the remaining subset for testing. Stratification applies to the classification dataset only. We simply shuffle the dataset and evenly divide it into five subsets for the regression datasets.

2.2 K-Nearest Neighbor:

K-Nearest Neighbor is a supervised learning algorithm developed by Thomas Cover, which makes classification or regression prediction using the information from the data instance's closest neighbors.

Model:

The k-NN algorithm makes prediction using the steps below:
For each instance in the test set,
1. Calculate its distance with all the observations in the training set
2. Find the k nearest points in the training set by ranking the distance from smallest to largest. The smaller the distance is, the more likely the observations are neighbors of the instance
3. The classification prediction of the instance is the class label that gets the plurality vote from the training set. It predicts the instance using the Gaussian (radial basis function) kernel for regression problems
4. Examine the performance of the k-NN classifier with the classification error rate and the regressor with the mean square error

Euclidean Distance Formula:

- Calculates the distance between the numeric attributes

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)}$$

Date Distance Formula:

- Calculates the distance between two dates

$$d\ (d1, d2) = \min(\text{abs}(d1\text{-}d2), \min(d1, d2) + 12 - \max(d1, d2))$$

Value Distance Metric Formula:

- Calculate the distance between the categorical attributes

Let f(x) be a feature of example X.

$$f \in \{V_1, \dots, V_k\}$$

1) Calculate the number of instances with attribute V_i in the dataset

$$C_i = \#\ \{f(x) = V_i\}$$

2) Calculate the number of instances with attribute V_i and has a class c

$$C_{i,a} = \#\{f(x) = V_i \cap \text{class} = a\}$$

3) Construct feature difference matrices

$$\delta(v_i, v_j) = \sum_{a=1}^{\#\ \text{classes}} \left| \frac{C_{i,a}}{C_i} - \frac{C_{j,a}}{C_i} \right|^p$$

$$D(x,y) = \left(\sum_{i=1}^{d} \delta(x_i, y_j)\right)^{\frac{1}{p}}$$

Performance Evaluation Metric:

Classification:

$$\text{classification error } rate = \frac{\#\ of\ instances\ that\ are\ misclassified}{\#\ of\ instances\ in\ the\ test\ set}$$

Regression:

$$Mean\ Sqaure\ Error\ (MSE) = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

2.3 Edited Nearest Neighbor

This algorithm uses incremental editing process to remove observations from the training set that are misclassified using the rest of the instances. 10% of the dataset is selected for tuning and 90% for training and testing. For the editing process, we use k = 1 and performs the steps below.

Model / Editing Process:

1. Select an instance from the training set, classify the example using the rest of the dataset. If the classification prediction is incorrect, remove the instance from the training set, apply the k-NN algorithm with the reduced training set on the validation set and calculate the classification error
2. Repeat the step above until the classification error on the validation set starts to increase or no further points can be removed

2.4 Condensed Nearest Neighbor

This algorithm is designed to select a smallest subset (Z) from the training set that aims to reduce the algorithm's running time and space complexity [4]. It applies the 1-NN algorithm to each instance in the subset (Z) and the data point in the training set and continuously add the data point that are misclassified to Z.

Model:

1. Start with an empty set Z
2. The initial point added to Z is considered as misclassified
3. For each instance in the training set, find the nearest point in the set Z, if the classes are matched, we go to the next instance. If their classes are mismatched, add the instance to Z and remove it from the training set
4. Input Z as a training set into the k-NN classification algorithm

3. Experimental Approach:

Datasets:

The datasets are from UCI Machine learning Repository [1]. Table 1 summarizes the properties of the six datasets.

Dataset Property Summary Table						
	Glass Dataset	**Segmentation Dataset**	**Vote Dataset**	**Abalone Dataset**	**Computer Hardware Dataset**	**Forest Fires Dataset**
# Attributes	9	18	16	8	7	12
# Examples	214	210	435	4,177	209	517
Type of dataset	Classification	Classification	Classification	Regression	Regression	Regression

Table 1 - Dataset Property Summary Table

Data Preprocessing Steps:

Glass Dataset:
* Remove the id number column because it is unique to each observation

Segmentation Dataset:
* Assign numerical values to the class
* Remove 'REGION-PIXEL-COUNT' column because they are the same

Vote Dataset:
* No changes needed

Abalone Dataset:
* Assign numerical values to the sex category

Computer Hardware Dataset:
* Remove the 'Model Name' column because it is unique to each observation
* Normalize the attributes so that they are between 0 and 1

Forest Fires Dataset:
* Assign numerical values to month and day

Hyperparameter Tuning Process:

In order to obtain the low classification or mean squared error, we want to find the ideal k, the hyperparameter for all the nearest neighbor algorithms. We compare the algorithms' performance using k from 1 to 9. In addition, for e-NN and cNN on the regression problems, we tune the ε, the error threshold for determining whether the prediction is correct and σ, the bandwidth for the Gaussian kernel. Tables in the result section summarizes the tuning outputs with different algorithms, datasets and hyperparameters.

Five-Fold Stratified Cross Validation:

Classification:
1) Use groupby function to calculate the proportion of each class in the dataset

2) Calculate the size of each fold by dividing the number of instances in each class by five since we are doing a five-fold cross validation
3) Randomly shuffle the dataset and create a dictionary to store the data in each fold using the calculated fold size
4) Iteratively exclude one-fold each time and combine the remaining subsets as a training set

Regression:
1) Randomly shuffle the dataset
2) Calculate the fold size by dividing the total number of observations in the dataset by five
3) Create a dictionary to store the data in each fold
4) Iteratively exclude one-fold each time and combine the remaining subsets as a training set

4. Result

Nearest Neighbor Classifier Tuning Table (Classification Error)									
	Glass Dataset			**Segmentation Dataset**			**Vote Dataset**		
K	**k-NN**	**Edited Nearest Neighbor**	**Condensed Nearest Neighbor**	**k-NN**	**Edited Nearest Neighbor**	**Condensed Nearest Neighbor**	**k-NN**	**Edited Nearest Neighbor**	**Condensed Nearest Neighbor**
1	0.348	0.435	0.217	0.238	0.286	0.381	0.068	0.046	0.06
2	0.391	0.391	0.261	0.476	0.381	0.571	0.091	0.083	0.042
3	0.435	0.348	0.304	0.476	0.286	0.619	0.023	0.043	0.053
4	0.478	0.391	0.304	0.524	0.286	0.571	0.023	0.013	0.023
5	0.435	0.348	0.435	0.571	0.524	0.667	0.023	0.013	0.013
6	0.435	0.391	0.391	0.571	0.381	0.619	0.023	0.033	0.013
7	0.435	0.391	0.435	0.571	0.429	0.667	0.023	0.013	0.013
8	0.348	0.435	0.391	0.524	0.429	0.619	0.023	0.023	0.013
9	0.348	0.391	0.304	0.524	0.476	0.667	0.000	0.010	0.000

Table 2 - Nearest Neighbor Classifier Tuning Table

Nearest Neighbor Classifier – Optimal Hyperparameters								
Glass Dataset			**Segmentation Dataset**			**Vote Dataset**		
k-NN	**e-NN**	**c-NN**	**k-NN**	**e-NN**	**c-NN**	**k-NN**	**e-NN**	**c-NN**
K = 1	K = 1	K = 1	K = 1	K = 1	K = 1	K = 9	K = 8	K = 4

Table 3 - Nearest Neighbor Classifier – Optimal Hyperparameters

Nearest Neighbor Classifier Performance Summary (Classification Error)								
Glass Dataset			**Segmentation Dataset**			**Vote Dataset**		
k-NN	**Edited Nearest Neighbor**	**Condensed Nearest Neighbor**	**k-NN**	**Edited Nearest Neighbor**	**Condensed Nearest Neighbor**	**k-NN**	**Edited Nearest Neighbor**	**Condensed Nearest Neighbor**
0.321	0.244	0.185	0.286	0.238	0.129	0.067	0.05	0.033

Table 3 - Nearest Neighbor Classifier Performance Summary

k- Nearest Neighbor (k-NN) Regressor Tuning Table						
	Abalone Dataset		Computer Hardware Dataset		Forest Fires Dataset	
K	**σ**	**k-NN**	**σ**	**k-NN**	**σ**	**k-NN**
1	0.001	9.508	1	1,275	1	17,999
1	0.01	9.625	10	1,275	10	17,684
1	0.1	9.684	20	1,275	20	17,999
2	0.001	6.892	1	1,224	1	3,000
2	0.01	5.966	10	1,195	10	2,986
2	0.1	5.867	20	1,194	20	2,984
3	0.001	6.816	1	1,813	1	1,071
3	0.01	5.635	10	1,706	10	1,213
3	0.1	5.428	20	1,700	20	1,222
4	0.001	6.664	1	1,356	1	687
4	0.01	5.418	10	1,323	10	764
4	0.1	5.184	20	1,322	20	770
5	0.001	6.209	1	1,033	1	571
5	0.01	5.159	10	1,025	10	619
5	0.1	4.925	20	1,024	20	624
6	0.001	6.098	1	1,188	1	754
6	0.01	4.879	10	1,162	10	713
6	0.1	4.739	20	1,160	20	713
7	0.001	5.945	1	1,513	1	991
7	0.01	4.765	10	1,465	10	730
7	0.1	4.653	20	1,462	20	721
8	0.001	6.254	1	1,495	1	878
8	0.01	4.808	10	1,460	10	714
8	0.1	4.63	20	1,458	20	705
9	0.001	5.983	1	1,698	1	428
9	0.01	4.674	10	1,653	10	428
9	0.1	4.582	20	1,650	20	429

Table 5 - kNN Regressor Tuning Table

Edited Nearest Neighbor (e-NN) Regressor Tuning Table						
	Abalone Dataset		Computer Hardware Dataset		Forest Fires Dataset	
K	**ε**	**e-NN**	**ε**	**e-NN**	**ε**	**e-NN**
1	1	10.89	5	3,065	1	12,919
1	3	7.73	10	3,063	3	12,919
1	5	7.34	20	2,983	5	12,919
2	1	6.47	5	3,496	1	2,656
2	3	6.14	10	3,591	3	2,656
2	5	6.17	20	3,496	5	2,656
3	1	5.65	5	2,949	1	1,174
3	3	5.59	10	2,949	3	1,174
3	5	5.59	20	2,949	5	1,174
4	1	4.98	5	2,279	1	798
4	3	4.94	10	2,279	3	798
4	5	4.94	20	2,279	5	798
5	1	4.90	5	2,187	1	1,412
5	3	4.90	10	2,187	3	1,412
5	5	4.90	20	2,187	5	1,412
6	1	4.98	5	2,349	1	1,142
6	3	4.98	10	2,349	3	1,142
6	5	4.98	20	2,349	5	1,142
7	1	4.87	5	2,090	1	1,152
7	3	4.87	10	2,090	3	1,152
7	5	4.87	20	2,090	5	1,152
8	1	4.83	5	2,007	1	1,433
8	3	4.83	10	2,007	3	1,433
8	5	4.83	20	2,007	5	1,433
9	1	4.82	5	2,189	1	1,255
9	3	4.82	10	2,189	3	1,255
9	5	4.82	20	2,189	5	1,255

Table 6 - Edited Nearest Neighbor (e-NN) Regressor Tuning Table

Condensed Nearest Neighbor (c-NN) Regressor Tuning Table						
	Abalone Dataset		Computer Hardware Dataset		Forest Fires Dataset	
K	ε	c-NN	ε	c-NN	ε	c-NN
1	1	10.08	5	3,011	1	1,388
1	3	7.18	10	3,177	3	1,388
1	5	6.73	20	3,083	5	1,388
2	1	5.81	5	1,691	1	512
2	3	5.51	10	1,646	3	512
2	5	5.48	20	1,646	5	512
3	1	5.72	5	1,881	1	2,038
3	3	5.68	10	1,881	3	2,038
3	5	5.65	20	1,881	5	2,038
4	1	5.23	5	2,822	1	1,671
4	3	5.22	10	2,822	3	1,671
4	5	5.19	20	2,822	5	1,671
5	1	5.02	5	3,275	1	1,572
5	3	5.01	10	3,275	3	1,572
5	5	5.01	20	3,275	5	1,572
6	1	4.96	5	3,518	1	1,559
6	3	4.95	10	3,518	3	1,559
6	5	4.95	20	3,518	5	1,559
7	1	4.87	5	3,984	1	1,246
7	3	4.87	10	3,984	3	1,246
7	5	4.87	20	3,984	5	1,246
8	1	4.91	5	4,258	1	1,034
8	3	4.91	10	4,258	3	1,034
8	5	4.91	20	4,258	5	1,034
9	1	4.83	5	4,479	1	959
9	3	4.83	10	4,479	3	959
9	5	4.83	20	4,479	5	959

Table 7 - Condensed Nearest Neighbor (c-NN) Regressor Tuning Table

Nearest Neighbor Regressor– Optimal Hyperparameters								
Abalone Dataset			Computer Hardware Dataset			Forest Fires Dataset		
k-NN	e-NN	c-NN	k-NN	e-NN	c-NN	k-NN	e-NN	c-NN
K = 9	K = 9	K = 9	K = 5	K = 8	K = 2	K = 9	K = 4	K = 2
$\sigma = 0.1$	$\varepsilon = 0.1$	$\varepsilon = 1$	$\sigma = 20$	$\varepsilon = 5$	$\varepsilon = 10$	$\sigma = 1$	$\varepsilon = 1$	$\varepsilon = 1$

Table 8 - Nearest Neighbor Regressor – Optimal Hyperparameters

Nearest Neighbor Regressor Performance Summary								
Abalone Dataset			Computer Hardware Dataset			Forest Fires Dataset		
k-NN	Edited Nearest Neighbor	Condensed Nearest Neighbor	k-NN	Edited Nearest Neighbor	Condensed Nearest Neighbor	k-NN	Edited Nearest Neighbor	Condensed Nearest Neighbor
5.173	5.065	5.154	7,008	9,177	6,582	1,451	3,876	4,960

Table 9 - Nearest Neighbor Regressor Performance Summary

Nearest Neighbor Method Performance Ranking				
Ranking	**Classification Dataset**	**Regression Dataset**		
		Abalone Dataset	**Computer Hardware Dataset**	**Forest Fires Dataset**
1	Condensed Nearest Neighbor	Edited Nearest Neighbor	Condensed Nearest Neighbor	k-NN
2	Edited Nearest Neighbor	Condensed Nearest Neighbor	k-NN	Edited Nearest Neighbor
3	k-NN	k-NN	Edited Nearest Neighbor	Condensed Nearest Neighbor

Table 10 - Nearest Neighbor Method Performance Ranking

5. Discussion:

Table 2, 5, 6 and 7 show the tuning outputs for the k-NN, e-NN and c-NN. They show the classification and mean square errors when using different hyperparameters on these algorithms. Table 3 and 8 summarize the optimal hyperparameters for each algorithm. Table 4 and 9 show the performance for all the nearest neighbor classifiers and regressors. Table 10 ranks the algorithms on the classification and regression datasets based on their performance. As we can see, condensed nearest neighbor performs the best on the classification dataset, followed by edited nearest neighbor and k-NN. However, their performance on regression dataset varies. For the Abalone Dataset, the edited nearest neighbor has the lowest mean square error, followed by the condensed nearest neighbor and k-NN. The condensed Nearest Neighbor performs the best on the Computer Hardware dataset, followed by k-NN and edited nearest neighbor. For the Forest Fires dataset, k-NN performs the best, followed by edited nearest neighbor and condensed nearest neighbor. The various performances on the regression datasets might have to do with the hyperparameters and further investigations can be done.

6. Conclusions:

This project helps us gain a better understanding of various nearest neighbor algorithms and their performance on different types of datasets. In addition, we have learned various ways to calculate the distance between categorical, numerical and date variables. This paper can help analysts to decide which algorithm to implement when they encounter classification or regression problems. Although the running time for the condensed nearest neighbor is longer than k-NN, it provides a higher accuracy in the classification dataset. Some future work on how to employ computational improvements in efficiency for the condensed nearest neighbor can be done. It would also be interesting to apply nearest neighbor algorithms and linear regression models on more regression datasets to summarize conditions where one algorithm can outperform another.

7. Reference:

[1] Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[2] Altman, Naomi S. (1992). "An introduction to kernel and nearest-neighbor nonparametric regression" (PDF). The American Statistician. 46 (3): 175–185. doi:10.1080/00031305.1992.10475879. hdl:1813/31637

[3] Stone, M (1977). "An Asymptotic Equivalence of Choice of Model by Cross-Validation and Akaike's Criterion". Journal of the Royal Statistical Society: Series B (Methodological). 39 (1): 44–47. JSTOR 2984877.}

[4] P. Hart, "The condensed nearest neighbor rule (Corresp.)," in IEEE Transactions on Information Theory, vol. 14, no. 3, pp. 515-516, May 1968, doi: 10.1109/TIT.1968.1054155.