Programming Assignment 2:
Hyperparameter Tuning for K-Nearest Neighbor Model

Wanyi (Julia) Dai WDAI17@JH.EDU
3400 North Charles St
Baltimore, MD 21218, USA

Abstract

This paper provides a comprehensive overview of Programming Assignment 2 from the course "Introduction to Machine Learning." This project aims to implement the k-nearest neighbor (KNN), edited k-nearest neighbor (Edited KNN), and condensed k-nearest neighbor (Condensed KNN) model for predicting both regression and classification data, ensuring adaptability to future applications. By the project's completion, we will apply kx2 cross-validation on all the datasets with the models, accommodating hyperparameter tunings based on the best parameters (k for KNN; k and epsilon for EKNN and CKNN) to use for each dataset.

Keywords: Programming Assignment 2, k-nearest neighbor model, hyperparameter tuning, kx2 cross-validation

1 Introduction

In the ever-evolving landscape of machine learning, the K-Nearest Neighbors (KNN) algorithm has remained a steadfast and intuitive approach for classification and regression tasks. KNN is a simple yet effective method that relies on proximity – it classifies or predicts based on the majority class of its k-nearest neighbors in the feature space. While traditional KNN serves as a strong foundation, machine learning has witnessed the development of more advanced variants to address specific challenges and enhance performance. This paper explores the intricacies of three distinct approaches: KNN, Edited KNN, and Condensed KNN, exploring their fundamental concepts and applications. Moreover, we introduce the notion of utilizing kx2 cross-validation, a robust technique for fine-tuning hyperparameters, to optimize the performance of these algorithms. Through a comprehensive examination of these methods and the subsequent parameter tuning on datasets with diverse natures, we aim to provide valuable insights into the practical use and optimization of KNN and its variants in various real-world scenarios.

2 Problem Statement

We hypothesize that the application of K-Nearest Neighbors (KNN) and its refined versions, Edited KNN and Condensed KNN, in conjunction with kx2 cross-validation for parameter tuning, will significantly enhance the accuracy classification and regression tasks across various datasets. We anticipate these algorithms exhibiting unique strengths and weaknesses in different datasets. We can harness their full potential through careful parameter optimization to achieve superior predictive performance compared to the naïve null model. Additionally, we expect these techniques to prove particularly beneficial in scenarios characterized by noisy or high-dimensional data, providing valuable insights into their suitability for real-world applications.

We expect regression datasets to yield better results, as KNN works best with numeric attributes. We also expect to have higher k when more data points are in the training set, as more noise will occur with more data. Epsilon for regression data in Edited KNN and Condensed KNN should be higher when the target variable has a higher range because predicting such data with narrow flexibility could filter out the actual valued neighbors.

The categorization of the datasets we will be using for the experiment is as follows:

Regression Datasets: abalone, forest fires, machine

Classification Datasets: breast cancer Wisconsin, car, house votes

3 Pre-processing Steps

For us to experiment, several pre-processing steps exist on the datasets.

3.1 Loading Data

All datasets will be stored as Pandas DataFrames. These datasets are typically stored in .data files, and in most cases, they lack predefined headers. To address this, we will manually generate headers using information from .name files and apply them during the data-loading process. Additionally, the data loading function offers the flexibility to perform log transformations on specific columns, which can benefit various datasets. Decisions regarding header generation and log transformations are made based on each dataset's particular characteristics and requirements. This approach ensures that data is loaded and prepared in a manner that optimizes future performance and analysis based on the unique nature and context of each dataset.

3.2 Handling Missing Values

We first must handle the missing data before giving a dataset to a machine learning algorithm. Most of our datasets are complete, except the breast cancer Wisconsin data. The attribute Bare Nuclei has missing data denoted by “?”. In this case, the missing value is credited with the column mean.

Note: house vote data also has “?”, but it is denoted as a category of neither “yes” nor “no” instead of missing data.

3.3 Handling Categorical Data

Categorical data consists of discrete, distinct categories or labels representing different groups or classes in the dataset. They are often described as strings in the dataset, which could be hard to use in computation. We need to convert the categorical data into numbers while preserving the information. There are two types of categorical data in our datasets.

1. **Handling Ordinal Data:** Ordinal data is data with order, such as [low, med, high]. We could match this data to an integer series to replace the string type. Modifications are made for the following datasets and columns.
   a. Car: replacing the following columns with integer series {1, 2, 3, …}.
      i. Buying: [low, med, high, vhigh]
      ii. Maint: [low, med, high, vhigh]
      iii. Doors: [2, 3, 4, 5more]
      iv. Persons: [2, 4, more]
      v. Lug_boot: [small, med, big]
      vi. Safety: [low, med, high]
      vii. Class: [unacc, acc, good, vgood]
   b. Forest Fire: replacing the following columns with integer series {1, 2, 3, …}. Note that these data are month and day, which have cyclic features. Since this is regression data, we do not want to use the one-hot encoding, so the distance calculation normalization could be an issue. We will treat the following columns as ordinal data and take special care when calculating the distance with their cyclic nature.
      i. Month: [jan, feb, mar, apr, may, jun, jul, aug, sep, oct, nov, dec]
      ii. Day: [mon, tue, wed, thu, fri, sat, sun]
2. **Handling Nominal Data:** Nominal data is categorical data without order. We apply one-hot encoding to cast this data type into higher dimensional binary space. Modifications are made for the following datasets and columns.
   a. Abalone: sex

        i. This is the only time we do the calculation directly using one-hot encoding.
    b. House votes: all columns
    c. Machine: vendor, model
        i. As Prof. Sheppard suggested, we will remove these two columns as they will add a very high binary space dimension to a regression dataset.

4 Experimental Approach

There are generally two steps in our experiment. First, build the KNN models we will be using on our datasets. Then, apply the kx2 cross-validation to tune the hyperparameters.

4.1 KNN, Edited KNN, and Condensed KNN models

Our experiments will use KNN, Edited KNN, and Condensed KNN models. The Edited and Condensed KNN are built based on the traditional KNN. The prediction algorithm of the KNN model is as follows:

1. Compute the distance between the data point in the test set and all data points in the training set using the chosen distance metric.
2. Sort the computed distances in ascending order so that the nearest neighbors come first in the list.
3. Choose the k data points (neighbors) with the smallest distances to the test data point. These k neighbors are the most similar data points to the one being predicted.
4. Predict the value:
    a. **Classification: plurality vote.** Count the occurrences of each class among the k-nearest neighbors and assign the class with the highest count as the predicted class for the test data point.
    b. **Regression: Gaussian (radial basis function) Kernel**: Calculate the weights using the kernel $K(x,xq) = exp(-\gamma||x-x_q||_2)$, where $||\cdot||_2$ indicates the L2 norm (i.e. Euclidean distance in this case). Apply the normalized weights to the targets of the nearest neighbors and sum them to calculate the final prediction.

The Edited KNN adds the editing process to form the edited training set. The editing process can be summarized as follows (*Quoting from Project 2 description*):

1. Consider each data point individually.
2. For each data point, use its nearest neighbor to make a prediction.
3. If the prediction is (your choice: correct or incorrect, but don't do both), mark the data point for deletion.
4. Stop editing once performance on the held-out 20% starts to degrade.

(*End of Quote*)

The Condensed KNN adds the condensed process to form the condensed training set. The condensing process can be summarized as follows (*Quoting from Project 2 description*):

1. Add the first data point from the training set into the condensed set.
2. Add the first data point from the training set into the condensed set.
3. Add the first data point from the training set into the condensed set.
4. Add the first data point from the training set into the condensed set.
5. Add the first data point from the training set into the condensed set.

(*End of Quote*)

Assumptions made for the algorithm:

1. The bandwidth is tuned to be $\gamma = 1/\sigma$.

2. For distance calculation, some care must be taken to handle categorical (i.e., discrete) attributes. We are using the value difference metric in this case. Specifically, let $f(x)$ be a feature of data instance x, where $f(x) \in \{v_1,...,v_k\}$. Let $C_i = \#\{f(x) = v_i\}$ and $C_{i,a} = \#\{f(x) = v_i \wedge class = a\}$. We then define the "distance" between feature value v_i and feature value v_j as

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

   Then, the distances between two examples, x and y, is

   $$D_{\text{cat}}(\mathbf{x}, \mathbf{y}) = \left(\sum_{k=1}^{d} \delta(\mathbf{x}_k, \mathbf{y}_k) \right)^{1/p} .$$

3. For the regression datasets in the Edited and Condensed KNN models, a "correct" prediction is determined by whether the prediction falls within some ε of the ground truth value. The tuning of the possible values of ε is considered by examining the range of target values in the training set.

4.2 Kx2 Cross Validation

We will use kx2 cross-validation with k = 5 for hyperparameter tuning on each dataset's KNN models. The 5x2 cross-validation includes the following steps *(Quoting from the Project 1 description)*:

1. Divide data into two parts: 80% will be used for training, and 20% will be used for tuning, pruning, or other types of "validation." Stratify so that the class distributions are the same in the partitions if the dataset type is classification.
2. Do the following five times:
   a. Divide the 80% into two equally sized partitions.
   b. Construct the KNN model of our choice using a candidate set of parameters. Each model will be trained with one of the halves and tested on the held-out 20%.
3. Average the results of these ten experiments for each parameter setting and pick the parameter settings with the best performance.
4. Do the following five times:
   a. Divide the 80% again into two equally sized partitions (stratified).
   b. Train a model using the tuned hyperparameters on the first half and test on the second half.
   c. Train a model using the tuned hyperparameters in the second half and test in the first.
5. Average the results of these ten experiments and report the average and the chosen hyperparameter setting.

(*End of Quote*)

Assumptions made for the algorithm:

1. The passed-in data and target are NumPy arrays with the features separated by columns.
2. The mean square error denotes the regression performance.
3. The accuracy denotes the classification performance.
4. To determine our choice of parameters, the regression datasets select the parameters with minimum mean square error; the classification datasets select the parameters with the maximum accuracy.

5 Results

To compare the performance and best hyperparameters chosen for each model, tables will be utilized to summarize our results from running the 5x2 cross-validation. We will not record the steps of the 5x2 cross-validation but compare the results directly. The attributes of our result tables are as follows:

1. Dataset Type: whether the dataset is regression or classification, we will combine datasets of the same type for easy comparison.
2. Dataset: name of the dataset.
3. Number of Rows: number of rows in the dataset.
4. Tuning Values: the values passed in for parameter tuning.
5. Best Hyperparameters: the best hyperparameter (combination if there is more than one tuning value) will be used to calculate the average performance.
6. Performance: the average performance of the ten folds using the best hyperparameter. Note we have mean square error (MSE) for regression data and accuracy (A) for classification data.

5.1 KNN Model

The tuning value for the KNN model is k. We set the same tunning values for all our datasets for comparison purposes. The k values are chosen to be odd numbers to avoid ties and reduce overfitting.

Dataset Type	Dataset	# of Rows	Tuning Values	Best Parameters	Performance
Regression	Abalone	4177	k = {1, 3, 5, 7, 9, 11, 13}	k = 13	MSE = 1.5431
	Forest Fire	517		k = 7	MSE = 19.5725
	Machine	209		k = 3	MSE = 34.4541
Classification	Breast Cancer	699		k = 1	A = 0.9484
	Car	1728		k = 9	A = 0.9079
	House Votes	435		k = 5	A = 0.9144

Table 1. Result table of KNN model using 5x2 cross-validation.

5.2 Edited KNN Model

We added the range of the target data, which would help us determine the possible epsilons we want to use for tuning. Note that epsilon is only tuned for regression datasets. Since the variation in the range of the regression targets are relatively large, we will be using epsilon = { 0.001, 0.01, 0.1, 1} for all of our regression datasets, for comparison purposes.

Dataset Type	Dataset	# of Rows	Range	Tuning Values	Best Parameters	Performance

Regression	Abalone	4177	28	k = {1, 3, 5, 7, 9, 11, 13}
				epsilon = { 0.001, 0.01, 0.1, 1}
					k = 13
					epsilon= 0.001
						MSE = 1.5408

	Forest Fire	517	1090.84		k = 3 epsilon= 0.001	MSE = 15.0499

	Machine	209	1144		k = 5 epsilon= 0.001	MSE = 40.456
Classification	Breast Cancer	699	NA Classification	k = {1, 3, 5, 7, 9, 11, 13}	k = 3	A = 0.9706
	Car	1728			k = 5	A = 0.9104
	House Votes	435			k = 7	A = 0.9144

Table 2. Result table of Edited KNN model using 5x2 cross-validation.

5.3 Condensed KNN model

The Condensed KNN model has the same table structure as the Edited.

Dataset Type	**Dataset**	**# of Rows**	**Range**	**Tuning Values**	**Best Parameters**	**Performance**

Regression	Abalone	4177	28	k = {1, 3, 5, 7, 9, 11, 13}
				epsilon = { 0.001, 0.01, 0.1, 1}
					k = 11
					epsilon = 0.001
						MSE = 1.5812

	Forest Fire	517	1090.84		k = 7 epsilon = 0.01	MSE = 24.1614
	Machine	209	1144		k = 1 epsilon = 1	MSE = 43.4075
Classification	Breast Cancer	699	NA Classification	k = {1, 3, 5, 7, 9, 11, 13}	k = 1	A = 0.7584
	Car	1728			k = 1	A = 0.2125
	House Votes	435			k = 1	A = 0.7529

Table 3. Result table of Condensed KNN model using 5x2 cross-validation.

5.4 Best Performance Summarization for Each Dataset

In the above results, we could see different data have different performances among the models. To ease comparison, the summarization below is the best model (KNN, Edited KNN, or Condensed KNN), the best parameters in that model, which results in the best performance, and the best performance among all the 5x2 cross-validation.

Dataset Type	**Dataset**	**Best Model**	**Best Parameters**	**Performance**

Regression	Abalone	Edited KNN	k = 13
			epsilon = 0.001
				MSE = 1.5408

	Forest Fire	Edited KNN	k = 3
			epsilon = 0.001
				MSE = 15.0499

	Machine	KNN	k = 3	MSE = 34.4541
Classification	Breast Cancer	Edited KNN	k = 3	A = 0.9705
	Car	Edited KNN	k = 5	A = 0.9104
	House Votes	KNN or Edited KNN	k = 7	A = 0.9144

Table 4. Best performance summarization of best model, best params, and best performance.

6 Discussion of the Behavior

We could discuss two aspects here: longer runtime in Edited and Condensed KNN models and the performance of each model on each dataset. We will elaborate on each of them in the following sections as supports to our conclusion.

6.1 Runtime

Edited KNN and Condensed KNN are variants of the traditional K-Nearest Neighbors (KNN) algorithm that aims to improve its efficiency by reducing the number of data points used in the classification or regression process. While these variants can offer benefits in reducing computational complexity and potentially improving model generalization, we observed that they are taking more time to run compared to the standard KNN algorithm, especially in abalone data, our largest dataset containing 4177 rows. Although I did not compute the exact timing, using KNN on abalone data takes a few minutes, while Edited KNN takes about half an hour. Some reasons for this behavior could be:

1. Data Reduction Process: Edited KNN and Condensed KNN involve an initial data reduction step where they analyze the training dataset to select a subset of relevant data points. This process can be computationally intensive, especially when dealing with large datasets. Edited KNN, for example, identifies and removes misclassified or noisy data points, which may require multiple passes through the data and doing prediction on each of the training data point.
2. Complexity of the Data: In cases where the dataset contains complex relationships or noisy data, the data reduction step can be more time-consuming. Edited KNN and Condensed KNN need to carefully analyze each data point to decide whether it should be kept or removed, and this process becomes more intricate when the data is not well-behaved.
3. Hyperparameter Tuning: Condensed and Edited KNN models' tuning process is longer in the regression dataset, with epsilon also needing to be tuned. For example, if we have 5 k's or settings for KNN, we have 5 k's and 5 epsilons, which results in 20 settings we need to check.

It's important to note that the choice between KNN and its variants, Edited KNN and Condensed KNN, should be made based on the specific characteristics of the dataset and the goals of our machine learning task. In some scenarios, the computational overhead of the data reduction step in Edited KNN and Condensed KNN might be justified by improved model performance, improved prediction complexity, or reduced memory usage. However, in situations where computational efficiency is a primary concern, and the dataset is not too large or noisy, the standard KNN algorithm may be more time-effective. Ultimately, the choice depends on the specific application's trade-offs.

6.2 KNN, Edited KNN, and Condensed KNN Performance in Different Datasets

By reviewing the result of the KNN model, we could identify the datasets with more rows, abalone (4177) and car (1728), that need relatively more neighbors to maximize the performance. However, we could not conclude that there is a definite correlation between the best k value and the row size since the breast cancer data has 699 with the best performance at k=1, while the machine data has 209 rows with the best performance at k=3. As stated in the hypothesis, when there are more data points, it is more likely to have noises, and more neighbors are needed for the correct prediction. However, it is possible that datasets with more data points have less noise. Hence, the choice of the best k value in KNN depends on the complexity of the data, the distribution of the data points, and the specific problem being addressed. Larger datasets may exhibit different characteristics and may require different values of k compared to smaller datasets. This behavior highlights the importance of performing hyperparameter tuning to determine the most suitable k value for each specific scenario, considering the dataset's unique properties.

The results of Edited and Condensed KNN models show that the k value could vary when using different algorithms, although these are variants of the traditional KNN. The incorporation of the epsilon could be a factor contributing to the change. As we assumed when conducting the experiment, we expect the epsilon to correlate with the range of the target data, which is also not valid in our results. Although we have such an ordered correlation in the Condensed KNN results (range 28, best epsilon 0.001; range 1090, best epsilon 0.01, range 1144, epsilon 1), in the Edited KNN results, we have epsilon=0.001 for all three datasets. However, it is possible that our choice of possible epsilons is too large. Hence, we will need more experiments to determine whether it shows any definite correlations. More experiments might need to be done to find the best parameter.

Comparing the best results among the datasets, Edited KNN consistently performed strongly in classification tasks, outperforming KNN and Condensed KNN in most cases. The Condensed KNN model performs especially poorly in classification datasets, with the car dataset only having 21.25% accuracy. This indicates that Condensed KNN might not be the best algorithm to use for classification data, especially the car data. In regression tasks, traditional KNN was often the top performer, while Edited KNN and Condensed KNN achieved competitive but slightly worse results. However, the performance for regression could be caused by the tuning of epsilon is not the universal best. It could be future research, specified on tuning the epsilon to get the best potential for using Condensed or Edited KNN in regression data.

These findings highlight the importance of experimenting with different variants of KNN and tuning hyperparameters based on the dataset's characteristics and the task's nature to achieve the best predictive performance. The behavior observed in these results underscores that no single algorithm is universally superior, and the choice should be made considering the problem's context and requirements.

7 Conclusions

In this comprehensive exploration of K-Nearest Neighbors (KNN) and its refined variants, Edited KNN, and Condensed KNN, we have delved into the intricacies of these algorithms across regression and classification tasks. Through a series of experiments and 5x2 cross-validation, we aimed to shed light on these techniques' behavior and performance characteristics. The traditional KNN algorithm often exhibited competitive performance for regression tasks. Edited KNN consistently demonstrated robust performance in classification tasks, surpassing KNN and Condensed KNN in most cases. The choice of the best k value varied across different datasets, and there was no clear linear correlation between dataset size and the optimal k.

Our results highlight the importance of hyperparameter tuning in achieving optimal model performance. Choosing k and other hyperparameters can significantly impact the accuracy and efficiency of KNN-based models, and fine-tuning is crucial for harnessing their full potential.

In summary, the behavior of KNN, Edited KNN, and Condensed KNN is influenced by the nature of the dataset and the specific task at hand. While traditional KNN can perform admirably, Edited KNN emerges as a robust choice for classification tasks due to its ability to reduce noise and improve model generalization. However, the efficacy of these algorithms is not solely determined by dataset size, as other factors such as data complexity, class distribution, and noise play pivotal roles. As we move on to future applications, the practice as analysis in the paper could provide us insights on making choices and optimizations on KNN-based models across a wide array of real-world scenarios.