Journal of Machine Learning Research 0 (2023) 0-0 Submitted 0/00; Published 0/00

Evaluation of Naïve Null Model Predictor on Diverse Datasets

Tim Chen TCHEN124@JHU.EDU
Department of Computer Science
Johns Hopkins University
Baltimore, MA 21218, USA

Editor: None

Abstract

In this paper, we explore the performance of three variants of the k-Nearest Neighbors (k-NN) algorithm: Standard, Edited, and Condensed, across six diverse datasets. Our primary goal is to understand how each k-NN variant performs in different scenarios, be it classification or regression. Preliminary results highlight nuanced differences in performance between the variants, emphasizing the importance of context when selecting an algorithm variant.

Keywords: Grid Search, k-Nearest Neighbors (k-NN), Edited k-NN, Condensed k-NN, Machine Learning, Distance Metrics, Non-parametric Algorithms, Computational Efficiency

1. Introduction

1.1 Background

The k-Nearest Neighbors (k-NN) algorithm, one of the simplest and most intuitive machine learning algorithms, has seen a broad spectrum of applications ranging from pattern recognition to anomaly detection. Its non-parametric nature makes it particularly appealing for tasks where the data does not conform to a particular shape or distribution. However, in its standard form, k-NN can be computationally expensive and sensitive to noisy data. As a result, several variants of the k-NN algorithm, including Edited and Condensed k-NN, have been proposed to address these challenges.

1.2 Objective

While the standard k-NN is widely used, there is limited comprehensive research comparing its performance with its Edited and Condensed counterparts across a variety of datasets and tasks. This paper seeks to bridge this gap. By evaluating these three k-NN variants across six datasets, we aim to provide insights into their strengths, weaknesses, and ideal use-cases.

1.3 Structure

The remainder of this paper is organized as follows: Section 2 provides a detailed description of our experimental approach and the assumptions made. In Section 3, we present the results of our experiments, with a subsequent discussion of the algorithms' behavior in Section 4. We draw conclusions in Section 5 and offer directions for future research.

2. Methods

The methodology incorporated in this study integrates the preprocessing and evaluation of three variants of the k-Nearest Neighbors (k-NN) algorithm. The objective is to compare their performance to deduce the most effective method in the context of our selected datasets. Below is a detailed breakdown of the process:

2.1 Problem Statement and Hypothesis

The k-Nearest Neighbors (k-NN) is an instance-based algorithm with extensive applications across both classification and regression domains. Our investigation primarily revolves around juxtaposing the performance of the classic k-NN with its two popular derivatives - the Edited k-NN and the Condensed k-NN. Both these derivatives are architected to amplify the efficacy and efficiency of the foundational k-NN method. The primary dataset for this exploration is the Breast Cancer Wisconsin dataset.

Hypothesis Formation:

- The traditional k-NN, while effective, might grapple with the curse of dimensionality, an impediment that could steer it towards overfitting.
- The Edited k-NN is predicted to showcase superior generalization on unseen datasets, given its inherent design to weed out noisy and extraneous data instances.
- We anticipate the Condensed k-NN to substantially contract the dataset volume, simultaneously ensuring the classification performance resonates closely with the conventional k-NN.

2.2 Experimental Strategy and Assumptions

Experimental Paradigm: Our experimental architecture encapsulates three predominant phases:

1. **Preprocessing**: This stage pivots on leveraging the Breast Cancer Wisconsin dataset. After loading the data, we undertake various preprocessing measures. These include the imputation of missing entries and the standardization of specific columns. Once preprocessed, we fractionate the data into training, development (intended for hyperparameter optimization), and testing sets.
2. **Hyperparameter Optimization**: We have harnessed a grid search approach, specifically on the development dataset, to unearth the optimal hyperparameters for the triad of k-NN models. These hyperparameters include the count of neighbors (k), epsilon (tailored for regression tasks), and gamma (employed with the RBF kernel during regression analyses).
3. **Evaluation Phase**: Here, each k-NN variant undergoes training on the designated training dataset. Subsequent evaluations are conducted on both development and

testing datasets. Accuracy stands as our metric of choice for these evaluations. Additionally, to evaluate model consistency, we compute the standard deviation of the performance scores.

Algorithmic Assumptions:

- The algorithm is architected with the supposition that categorical attributes in our dataset are already one-hot encoded.
- For numerical attributes, the distance measure utilized is the Euclidean distance. However, for the categorical counterparts, we have employed the Value Difference Metric (VDM).
- In the paradigm of the Edited k-NN, instances that are misclassified are earmarked for removal. On the other hand, for the Condensed k-NN, misclassified instances are incorporated into the condensed set.
- We have operated under the premise that all attributes (or features) in the dataset hold equal significance. Consequently, no feature selection or extraction methodologies were implemented.

2.3 Implementation Details

Development Framework: The entire experiment was architected in Python 3.8, leveraging foundational libraries for data manipulation and machine learning.

- **Library Dependencies**:
  - `pandas (v1.2.4)`: Deployed for efficient data loading, manipulation, and operations on data structures.
  - `numpy (v1.20.2)`: Employed for mathematical operations and efficient array operations.

2.3.1 Custom k-NN Implementation

Our k-NN model is defined within the `KNN` class, encapsulating functionalities for the standard k-NN along with its edited and condensed variants.

- **Radial Basis Function (RBF) Kernel**: The `rbf_kernel` function computes the similarity between two instances employing the RBF kernel, accounting for a parameter gamma.
- **Value Difference Metric (VDM)**: The `compute_vdm` and `vdm_distance` functions manage categorical distances. VDM considers the relative frequencies of categorical values per class, computing distances that reflect differences in these distributions. The calculations are cached for efficiency.
- **Nearest Neighbors Calculation**: In `k_nearest_neighbors`, the closest k training instances to a given test instance are identified. It considers both numerical and categorical attributes.

- **Edited and Condensed Variants**: Functions `edited_k_nearest_neighbors` and `condensed_k_nearest_neighbors` respectively implement the edited and condensed variants of k-NN. The edited variant iteratively removes misclassified training instances, while the condensed variant establishes a subset of the training data, including instances that are wrongly classified.

- **Model Training and Prediction**: The `fit` method trains the model, potentially editing or condensing the training set, based on the chosen method. The `predict` method, subsequently, uses the trained model to predict the labels of test instances.

The implementation ensures a modular approach, enabling easy switching between k-NN variants and providing a customizable interface for different hyper-parameters.

3. Results

Experiments were conducted on six datasets using three variants of the k-NN algorithm: Standard, Edited, and Condensed. The summarized performance metrics for each dataset, both on cross-validation and the test set, are provided in Table 3.

Dataset	**Variant**	**Task Type**	**Avg Perf. (5-fold CV)**	**Test Set Perf.**
Breast Cancer	Standard	Classification	ACC: 97%	ACC: 98%
	Edited	Classification	ACC: 97%	ACC: 98%
	Condensed	Classification	ACC: 97%	ACC: 96%
Car Evaluation	Standard	Classification	ACC: 89%	ACC: 92%
	Edited	Classification	ACC: 81%	ACC: 83%
	Condensed	Classification	ACC: 89%	ACC: 90%
Congressional Vote	Standard	Classification	ACC: 94%	ACC: 93%
	Edited	Classification	ACC: 93%	ACC: 94%
	Condensed	Classification	ACC: 93%	ACC: 94%
Abalone	Standard	Regression	ACC: 46%	ACC: 43%
	Edited	Regression	ACC: 48%	ACC: 47%
	Condensed	Regression	ACC: 45%	ACC: 41%
Computer Hardware	Standard	Regression	ACC: 89%	ACC: 88%
	Edited	Regression	ACC: 76%	ACC: 79%
	Condensed	Regression	ACC: 81%	ACC: 83%
Forest Fires	Standard	Regression	ACC: 51%	ACC: 45%
	Edited	Regression	ACC: 66%	ACC: 62%
	Condensed	Regression	ACC: 52%	ACC: 50%

Table 1: Performance of k-NN Variants on Various Datasets

The table showcases the performance metrics obtained using the different k-NN variants on various datasets. Classification tasks are evaluated based on accuracy (ACC), while regression tasks utilize the Mean Squared Error (ACC) metric. The results indicate nuanced differences in performance between the k-NN variants across datasets.

4. Discussion and Observations

Upon comprehensive evaluation of the k-NN variants, it becomes evident that the performance predominantly hinges on the dataset's intrinsic characteristics. The results lead us to infer the following:

1. **Performance across Datasets**: The foundational k-NN showcased commendable performance ubiquitously across most datasets, a trend prominently noticeable in the Breast Cancer and Car Evaluation datasets. The Edited and Condensed k-NN derivatives manifested similar trends, albeit with nuanced variations in specific metrics.

2. **Influence of k**: In datasets delineated by stark decision boundaries, such as Breast Cancer and Congressional Vote, a diminutive 'k' value typically led to superior outcomes. Contrastingly, intricate datasets like Forest Fires exhibited a preference for augmented 'k' values.

3. **Comparison of Edited and Condensed Variants**: In certain datasets, particularly Abalone, the Edited k-NN overshadowed the Condensed version in terms of performance. This indicates that in specific scenarios, eliminating noisy or superfluous instances might hold a greater advantage over dataset condensation.

4. **Methodological Significance**: The chosen k-NN variant (be it standard, edited, or condensed) profoundly impacted the outcome. For instance, while the Breast Cancer dataset resonated well with the standard method, the Congressional Vote dataset exhibited a proclivity for the condensed variant.

5. Conclusion

The k-NN algorithm, enriched with its derivatives, emerges as a potent tool to tackle diverse classification and regression challenges. The decision to employ standard, edited, or condensed k-NN should be meticulously crafted, contingent on the dataset's unique attributes. An unwavering commitment to consistent evaluation and hyperparameter fine-tuning remains pivotal in harnessing the maximum potential of these algorithms.

As a prospective direction for future endeavors, it might be insightful to delve into alternative distance metrics, ingeniously engineer feature selection strategies, and investigate other k-NN variants. Such pursuits have the potential to further optimize model performance across a spectrum of datasets.

Appendix A: Code Structure and Implementation Details

The project is methodically structured to efficiently apply the k-NN algorithm and its variants on diverse datasets. The modular architecture ensures scalability, flexibility, and easy debugging, making it adaptable to varying dataset sizes and characteristics.

- The `rbf_kernel` function applies the Radial Basis Function kernel, mainly used for regression tasks.

- The `compute_vdm` and `vdm_distance` methods are responsible for calculating the Value Difference Metric, enabling the k-NN algorithm to work with categorical attributes.
- The `k_nearest_neighbors` method is the core k-NN function, which finds the k-nearest instances for a given test instance and predicts its label or value.
- The `edited_k_nearest_neighbors` and `condensed_k_nearest_neighbors` methods are implementations of the Edited and Condensed k-NN variants, respectively.
- The `_get_categorical` method identifies one-hot encoded columns.
- The `fit` method is used to train the model, wherein the chosen k-NN variant is applied on the training data.
- The `predict` method facilitates predictions on new data after the model is trained.

Initialization and Execution Guide

To utilize the k-NN model and its variants for your dataset, follow the below steps:

1. **Initialization:** Instantiate the `KNN` class by providing the necessary hyperparameters. The primary hyperparameter is 'k' – the number of neighbors. Additional parameters include:

   - `task` (default="classification"): Choose between "classification" and "regression".
   - `gamma`: Used for regression tasks with the RBF kernel.
   - `epsilon`: Threshold for the edited k-NN in regression tasks.
   - `method` (default="standard"): Select the k-NN variant – "standard", "edited", or "condensed".

   For example:

   ```
   model = KNN(k=3, method="condensed", task="classification")
   ```

2. **Training:** Train the model using the `fit` method. Provide the training data (features) and labels (or target values) as arguments.

   ```
   model.fit(X_train, y_train)
   ```

3. **Prediction:** After training, you can make predictions on new data using the `predict` method.

   ```
   predictions = model.predict(X_test)
   ```

Ensure your dataset is preprocessed adequately, particularly in handling missing values and converting categorical features to a suitable format, before using it with the model.