Introduction to Machine Learning: Work 2
Classification with Lazy Learning and SVM

Pedro Agúndez
Bruno Sánchez
María del Carmen Ramírez
Antonio Lobo

November 1, 2024

Abstract

This is the abstract. Summarize the purpose, methods, and main findings of the study.

1 Introduction

Introduce the problem, background, and significance of the study.

2 Methodology

Describe the datasets and outline each method applied.

2.1 Datasets

Briefly describe Dataset 1 and Dataset 2.

2.2 K-Nearest Neighbors (KNN)

The implementation of the KNN algorithm is encapsulated in the `KNNAlgorithm` class. It allows the selection of different distance metrics, weighting methods, and voting policies to classify test instances using their nearest neighbors. The methodology follows these three main components.

2.2.1 Hyperparameters

1. **k:**
    - The number of Nearest Neighbors to be considered in the algorithm. This can take any integer value. In our study, we have employed values $1, 3, 5$ and 7.

2. **Distance Metrics:**

   - **Euclidean Distance**: Calculates the root of the sum of squared differences between feature values. It is commonly used for continuous data and defined as:

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

   - **Manhattan Distance**: Computes the sum of absolute differences between feature values, suitable for both categorical and continuous data. It is defined as:

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

   - **Clark Distance**: Accounts for proportional differences between feature values, enhancing interpretability for attributes with varying scales. It is computed as:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} \left(\frac{|x_i - y_i|}{x_i + y_i + \epsilon} \right)^2}$$

     where ϵ is a small constant to avoid division by zero.

3. **Weighting Methods:**

   - **Equal Weight**: Assigns equal importance to all features by setting each feature's weight to 1.
   - **Information Gain Weight**: Uses mutual information to assign weights based on each feature's information gain with respect to the class label.
   - **ReliefF Weight**: Computes feature relevance by evaluating differences between feature values of similar and dissimilar instances, adjusted by the specified number of neighbors.

4. **Voting Policies:**

   - **Majority Class**: Assigns the class based on the most common class label among the nearest neighbors.
   - **Inverse Distance Weighted**: Weights each neighbor's vote by the inverse of its distance, giving more influence to closer neighbors. The vote for class y is calculated as:

$$\text{Vote}_y = \sum_{i \in \mathcal{N}_y} \frac{1}{d(q, x_i)}$$

     where $\mathcal{N}_y$ represents neighbors with class y.

- **Shepard's Work**: Similar to the Inverse Distance Weighted policy, except that it applies exponential decay to the distance (instead of the inverse), allowing stronger influence from closer neighbors. The vote for class y is:

$$\text{Vote}_y = \sum_{i \in \mathcal{N}_y} e^{-d(q,x_i)}$$

This structure enables flexible configurations for analyzing the performance of the KNN algorithm across different datasets and hyperparameter values.

2.2.2 Results extraction

To systematically evaluate the KNN model configurations, the following procedure is followed to extract results for each of the 2 data sets, in order to later perform an statistical analysis:

1. **Data Preparation**: Each fold of the dataset is loaded, split into training and testing sets. Features may also be weighted using different weighting methods to analyze their impact on model performance. This is applied as a pre-processing step in order to optimize execution times.

2. **Parameter Configuration**: A comprehensive set of values for the KNN hyperparameters is defined. These combinations reflect various ways to tune the KNN model.

3. **Model Evaluation**: For each fold and parameter combination, the KNN model is trained on the training data and evaluated on the test data. This step yields the following metrics: accuracy, execution time, and F1-score. Together, these measure the model's effectiveness and efficiency.

4. **Results Compilation**: The performance metrics for each parameter combination and fold are recorded in a structured format. These results are saved as a dataset that summarizes the outcomes of all evaluations, forming a basis for analysis.

5. **Statistical Analysis**: After results are compiled across all configurations and folds, statistical analysis is performed to identify the best-performing configurations. This analysis helps determine the most reliable and effective parameter settings for accurate and efficient KNN classification. We will discuss our results in Section 3.

2.3 Support Vector Machine (SVM)

Describe the SVM approach.

2.4 Reduction Methods

In this section, we briefly describe reduction techniques employed for instance-based learning. We use a simple 2D dataset for illustrative purposes.

2.4.1 GCNN

2.4.2 EENTH

This subsection outlines the Elimination Editing with Nearest-neighbor Threshold (EENTH) method [1]. This approach uses a modified k-NN rule, integrating probability-based decisions for instance elimination. The main steps are outlined below.

1. **Probability-based Classification**: For each instance x, calculate the probability $p_i(x)$ of x belonging to each class i based on its k-nearest neighbors. Probabilities are weighted inversely by the distance to each neighbor and normalized:

$$p_i^j = \frac{|\{x_k \in NN_k(x) : y_k = j\}|}{k} \tag{1}$$

$$P_i(x) = \sum_{j=1}^{k} p_i^j \frac{1}{1 + d(x, x^j)} \tag{2}$$

$$p_i(x) = \frac{P_i(x)}{\sum_{j=1}^{M} P_j(x)} \tag{3}$$

2. **Thresholding**: Define a threshold μ to refine classification, we will denote as $p(x)$ the highest probability. Instances near decision boundaries, where $p(x) < \mu$, are identified as candidates for removal.

3. **Elimination**: If an instance x does not match the class with the highest probability, or if its highest class probability falls below μ, it is removed from the dataset, resulting in an edited set $S \subseteq X$.

The EENTH method thus provides a balance between retaining instances with high classification confidence and discarding uncertain instances near decision boundaries.

2.4.3 DROP3

In this subsection, we describe the basic concepts of the third method in the Decremental Reduction Optimization Procedure (DROP) family, as presented in Section 3 of Wilson et al. [3]. Although we will not delve into every detail, we describe the main ideas of the algorithm and illustrate them on D_1. See **Figure** 1.

1. **Remove noise**: The first step is to remove noisy instances using Edited Nearest Neighbor (EEN) [2], where any instance misclassified by its k-nearest neighbors is removed. The outcome of applying this technique is shown in **Figure** 2, where noise has been removed. We denote the reduced dataset as $T \subseteq D_1$.

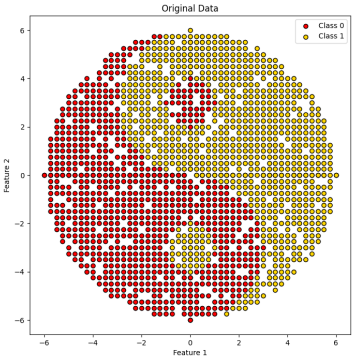

Figure 1: Original Dataset

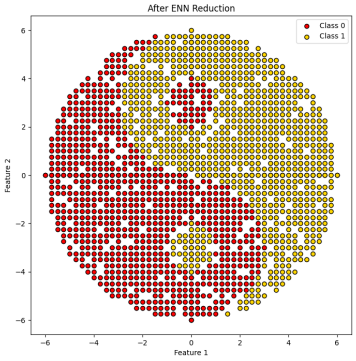

Figure 2: Effect of EEN

2. **Sort points**: The next step is to prioritize removing points that are farthest from the decision boundary. For each point $x_i \in S$ with class y_i, we compute the distance to the nearest point with a different class, denoted as $x_j \in D$ such that $y_j \neq y_i$ and $\nexists x_k : |x_k - x_i| < |x_j - x_i| \wedge y_i \neq y_k$.

3. **Delete points**: Let $S = T$. Starting with the points farthest from the boundary, we check if any associated points (points that have x_i as a neighbor) a_j receive more votes for their correct class with x_i as a neighbor (denoted as with) or if they would be classified correctly if x_i were removed (denoted as without). If without $>$ with, we remove x_j from S, resulting in $S' = S \setminus \{x_j\}$.

4. **Selecting neighbors**: A key distinction between DROP1 and DROP2 is that DROP1 removes points that are removed from the dataset from the list of associates while DROP2 doesn't.

3 Results

Present the findings and the conclusions of the statistical analysis. For each technique, we will separately discuss the results obtained with each of the 2 datasets. It is important to note that in the study of each dataset we have considered each of the 10 folds as separate datasets in order to perform the statistical analysis. This consideration is sub-optimal, but necessary due to the time and hardware constraints.

3.1 KNN Results

Discuss the outcomes for KNN.

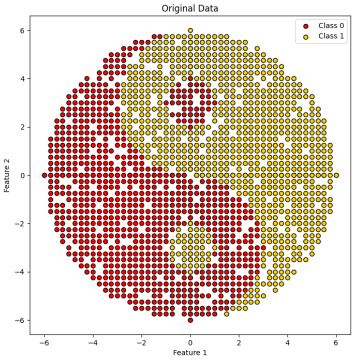

Figure 3: Original Dataset

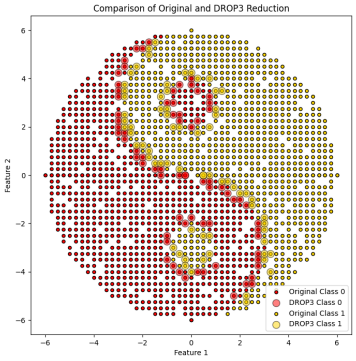

Figure 4: Effect of DROP3

3.2 SVM Results

Describe the SVM results.

3.3 KNN Reduction Results

Discuss the outcomes for KNN with reduction methods.

3.4 SVM Reduction Results

Describe the SVM results with reduction methods.

4 Discussion

Interpret the results, relate to previous work, and discuss implications.

5 Conclusion

Conclude the report.

References

[1] Fernando Vázquez, Josep Sánchez, and Filiberto Pla. A stochastic approach to wilson's editing algorithm. pages 35–42, 01 2005.

[2] D. L. Wilson. Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man, and Cybernetics*, 2(3):408–421, 1972.

[3] Dennis R. Wilson and Tony R. Martinez. Reduction techniques for instance-based learning algorithms. *Machine Learning*, 38(3):257–286, 2000.