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

Pedro Agúndez[*], Bruno Sánchez[*], María del Carmen Ramírez[*], Antonio Lobo[*]

November 3, 2024

Abstract

This study investigates the performance of instance-based learning algorithms, specifically K-Nearest Neighbors (KNN) and Support Vector Machines (SVM), on binary classification tasks using two distinct datasets: a balanced mushroom classification problem and an imbalanced medical dataset for hepatitis prognosis. We evaluate various configurations of these algorithms along with three data reduction techniques: Generalized Condensed Nearest Neighbor (GCNN), Elimination Editing with Nearest-neighbor Threshold (EENTH), and Decremental Reduction Optimization Procedure 3 (DROP3). Through comprehensive statistical analysis using Friedman tests and post-hoc analyses, we demonstrate that while KNN achieves perfect accuracy on the mushroom dataset regardless of configuration, the hepatitis dataset reveals significant differences between weighting methods. Our results indicate that the EENTH reduction method maintains classification performance while significantly reducing storage requirements, whereas GCNN shows performance degradation across both datasets. This research provides insights into the optimal configuration of instance-based learning algorithms and the effectiveness of different reduction techniques for real-world classification problems.

[*]Universitat de Barcelona
`pedro.agundez@estudiantat.upc.edu`
`bruno.sanchez.gomez@estudiantat.upc.edu`
`maria.del.carmen.ramirez@estudiantat.upc.edu`
`antonio.lobo@estudiantat.upc.edu`

1 Introduction

Instance-based learning algorithms remain fundamental tools in machine learning, particularly for classification tasks where local patterns in the data are crucial for decision-making. These algorithms, while conceptually simple, can achieve remarkable performance when properly configured. However, their effectiveness often depends heavily on the choice of hyperparameters and the characteristics of the input data. This study addresses two critical challenges in instance-based learning: optimal algorithm configuration and efficient data storage. We focus on two widely-used algorithms, K-Nearest Neighbors (KNN) and Support Vector Machines (SVM), applying them to binary classification problems with different characteristics - a balanced dataset for mushroom classification and an imbalanced medical dataset for hepatitis prognosis. Additionally, we investigate three data reduction techniques (GCNN, EENTH, and DROP3) to address the storage and computational efficiency challenges inherent in instance-based learning. Our research contributes to the field by providing:

1. A systematic comparison of different distance metrics, weighting methods, and voting policies for KNN.
2. An evaluation of various kernel functions and regularization parameters for SVM
3. An analysis of the effectiveness and trade-offs of different data reduction techniques
4. Statistical validation of the results using rigorous hypothesis testing

2 Methodology

Our study employs a systematic approach to evaluate instance-based learning algorithms and data reduction techniques across two distinct datasets. The methodology is structured to ensure comprehensive evaluation while maintaining statistical rigor.

2.1 Datasets

This study aims to evaluate these algorithms' performance on two datasets with distinct characteristics:

- The Mushroom dataset: A balanced binary classification problem focused on distinguishing edible from poisonous mushrooms based on physical characteristics.
- The Hepatitis dataset: An imbalanced medical dataset aimed at predicting patient survival based on clinical and demographic features.

2.2 Mushroom Dataset

2.2.1 Overview

The mushroom dataset comprises 8,124 samples with 23 categorical features describing various physical characteristics of mushroom species. The classification task involves identifying mushrooms as either edible or poisonous, presenting a balanced binary classification problem with 4,208 edible and 3,916 poisonous samples.

2.2.2 Feature Characteristics

The features can be categorized into three types:

- Ordinal features (3): gill-spacing, ring-number, population.
- Binary features (3): bruises, gill-size, stalk-shape.
- Nominal features (16): Including cap characteristics, colors, and environmental indicators.

2.2.3 Preprocessing Steps

- Missing Values: Only the stalk-root feature contains missing values (30% of samples), which were handled using SimpleImputer with most frequent value strategy.
- Feature Encoding:
  - Ordinal and binary features: Encoded using OrdinalEncoder.
  - Nominal features: Encoded using TargetEncoder to avoid dimension explosion while maintaining meaningful value relationships [5].

2.3 Hepatitis Dataset

2.3.1 Overview

The hepatitis dataset contains 155 samples with both numerical and categorical features, aimed at predicting patient survival. The dataset presents an imbalanced classification problem with 123 survival cases and 32 mortality cases.

2.3.2 Feature Characteristics

The dataset includes:

- Numerical features (6): Age, Bilirubin, Alkaline Phosphate, SGOT, Albumin, Protime.
- Categorical features (13): Including demographic information, symptoms, and clinical indicators.

2.3.3 Preprocessing Steps

- Missing Values:
  - Protime feature removed due to high missing value percentage ($> 40\%$).
  - Remaining missing values handled using median imputation for numerical features and mode imputation for categorical features.
- Feature Scaling:
  - Numerical features: Scaled to [0,1] range using MinMaxScaler.
  - Categorical features: Binary encoded using OrdinalEncoder.

2.4 Feature Distribution Analysis

Initial analysis revealed several features with strong class separation properties:

2.4.1 Mushroom Dataset

Key discriminative features include:

- Odor: Complete class separation for values a,c,m,p,s,y
- Spore print color: Complete separation for values b,o,r,u,y

2.4.2 Hepatitis Dataset

Notable patterns include:

- Sex: All mortality cases were female in the dataset.
- Albumin: Clear threshold effect with mortality cases showing values below 0.4

2.5 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.

2.5.1 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.

2.5.2 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.

2.6 Support Vector Machine (SVM)

The Support Vector Machine (SVM) algorithm was primarily implemented using the existing 'SVC' class from Scikit-Learn [3]. A custom class, 'SVM.py', was created to organize all tools, metrics, and analyses used in this project in a structured way.

Support Vector Machines (SVM) are designed to identify the optimal hyperplane that separates data into distinct classes. SVM can use various kernels for this separation, as it can be linear, polynomial, Gaussian (rbf) or sigmoid. Changing the kernels improve the adaptation of the model different dataset complexities and distributions. Each kernel is specifically tailored to capture unique data patterns, helping to achieve optimal separation across a wide range of data types.
All the four kernels mentioned were implemented in this project, and a brief overview of each is provided below, with the parameters selected for each kernel.

- The **linear** kernel is one of the simplest and most widely used kernel functions, particularly effective for data that is linearly separable data that can be divided by a straight line in two dimensions or a hyperplane in higher dimensions. Its main advantage is computational efficiency. No additional parameters were needed for this kernel.

- The **polynomial** kernel was studied with a degree of 2, as higher values tend to lead to overfitting. A quadratic kernel can represent various forms of geometric boundaries, such as hyperplanes, hyperspheres, hyperellipsoids, hyperparaboloids, and hyperboloids, enabling the learning of a wide range of non-linear models [2].

- The **gaussian** or **rbf** kernel is known to capture the smoothness property of the data [6]. The kernel function is defined as
$$K(\mathbf{x}, \mathbf{x}') = \exp(-\gamma\|\mathbf{x} - \mathbf{x}'\|^2).$$

- The **sigmoid** kernel is particularly effective for complex classification tasks where data point relationships resemble those captured by neural networks. However, selecting optimal parameters for this kernel can be challenging [4]. The kernel function is definded as
$$K(\mathbf{x}, \mathbf{x}') = \tanh(\gamma\langle\mathbf{x}, \mathbf{x}'\rangle).$$

The gamma parameter of the kernel function for the rbf kernel and the sigmoid kernel was set as `gamma='scale'`, so it is automatically set based on the number of features in the dataset. Specifically, it calculates γ as follows:

$$\gamma = \frac{1}{n_{\text{features}} \cdot \text{X.var()}}$$

where n_{features} is the number of features in the training data and X.var() is the variance of the training data.

The optimization problem in soft margin SVMs is formulated with a regularization parameter C, which adjusts tolerance for misclassifications (higher C enforces fewer errors). Several different values for this parameter were studied in order to obtain a better performance of the algorithm.

2.7 Reduction Methods

In this section, we briefly describe reduction techniques used for instance-based learning, illustrating with simple 2D datasets.

2.7.1 GCNN

The Generalized Condensed Nearest Neighbor (GCNN) [1] algorithm extends the traditional Condensed Nearest Neighbor (CNN) method by incorporating adaptive prototype learning. This **undersampling** technique seeks a subset of prototypes $U \subseteq X$, where $X = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ is our dataset, that correctly represents the data. This approach is especially beneficial for unbalanced datasets. The main idea behind CNN is to select prototypes that *absorb* points which they can represent as prototypes. The steps for this method are as follows:

1. **Prototype Selection:** For each class c_j, let $X_j = \{(x, y) : y = c_j\}$ be all points of this class. We select as a prototype x_j^*, the point that is the nearest neighbor of the most other points in X_j. Thus, x_j^* will influence the maximum number of points in any KNN decision. Specifically,

$$x_j^* = \arg\max_{x \in X_j} \left(\sum_{x' \in X_j \setminus \{x\}} \left(x = \text{NN}_{X_j \setminus \{x'\}}(x') \right) \right).$$

   Hence, $U = \{x_j^* : j = 1, \ldots, m\}$ and U_j will denote all prototypes for class c_j.

2. **Absorption:** This step distinguishes GCNN from CNN. Define $\delta_n = \min_{\{(x_i,y_i),(x_j,y_j) \in X : y_i \neq y_j\}} (||x_i - x_j||)$. For each point x_i, let $p = \arg\min_{x \in U_j} (||x - x_i||)$ and $q = \arg\min_{x \in (U \setminus U_j)} (||x - x_i||)$ be the nearest prototypes from the same class and a different class, respectively. We absorb x_i if:

$$||\mathbf{x} - \mathbf{q}|| - ||\mathbf{x} - \mathbf{p}|| > \rho\delta_n, \quad \rho \in [0, 1).$$

   In CNN, $\rho = 0$, meaning x_i is absorbed if its nearest prototype is from its class. The parameter ρ introduces a slack in this decision, illustrated in **Figure** 1.

3. **Prototype Augmentation:** If points remain unabsorbed for class c_j, repeat prototype selection on these unabsorbed points, then return to Step 2. The process stops once all points are absorbed.

In **Figures** 2 and 3, we illustrate the effects of varying ρ from 0 to 1 on a dataset.

2.7.2 EENTH

This subsection outlines the Elimination Editing with Nearest-neighbor Threshold (EENTH) method [7], which uses a modified k-NN rule with probability-based decisions to eliminate instances, particularly useful for noise removal. The main steps are:

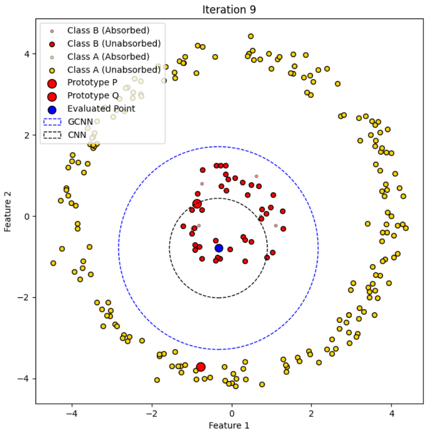
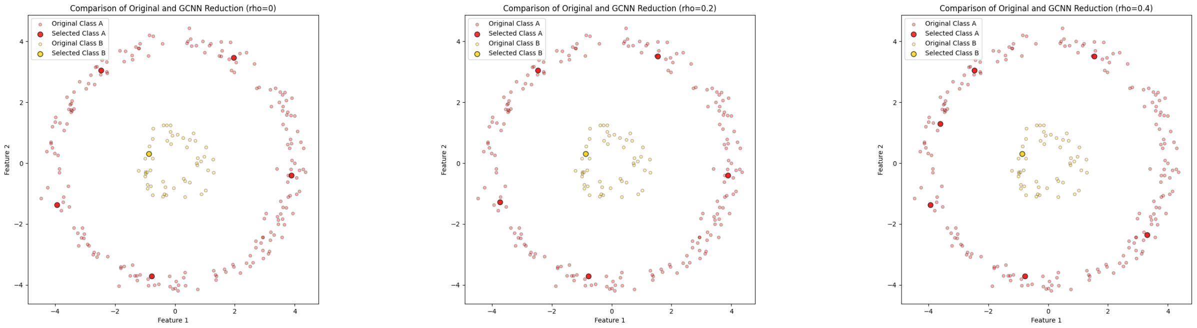


Figure 1: Effect of ρ in GCNN



(a) $\rho = 0$

(b) $\rho = 0.2$

(c) $\rho = 0.4$

Figure 2: GCNN illustration for $\rho = 0, 0.2, 0.4$

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

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

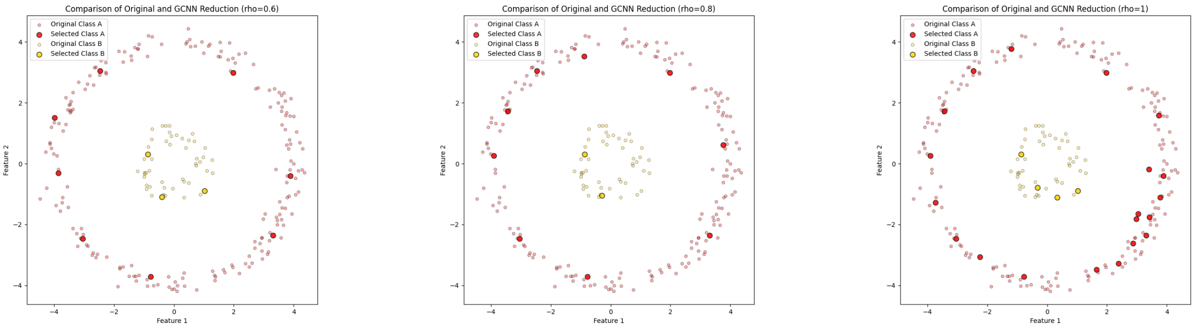
$$P_i(x) = \sum_{j=1}^{k} p_i^j \frac{1}{1 + d(x, x^j)} \tag{2}$$

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

2. **Thresholding**: Define a threshold μ to refine classification by identifying instances near decision boundaries where $p(x) < \mu$ as candidates for removal.

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

The EENTH method balances retaining instances with high confidence while discarding uncertain ones near boundaries. In **Figures** 4 and 5, we illustrate results for μ from 0.15 to 0.85 with 5 neighbors.

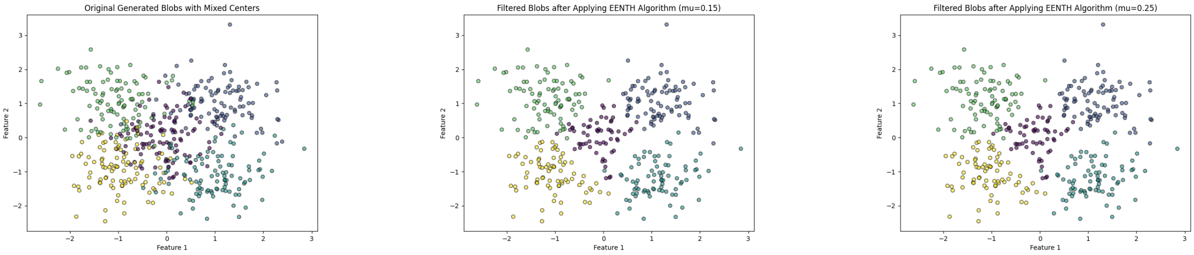


(a) $\rho = 0.6$

(b) $\rho = 0.8$

(c) $\rho = 1$

Figure 3: GCNN illustration $\rho = 0.6, 0.8, 1$



(a) Original

(b) $\mu = 0.2$

(c) $\mu = 0.4$

Figure 4: EENTH method illustration for $\mu = 0.2$ and $\mu = 0.4$.

2.7.3 DROP3

This section covers the main concepts of the third method in the Decremental Reduction Optimization Procedure (DROP) family [9]. We describe the core steps and illustrate the method on D_1 in **Figure** 6.

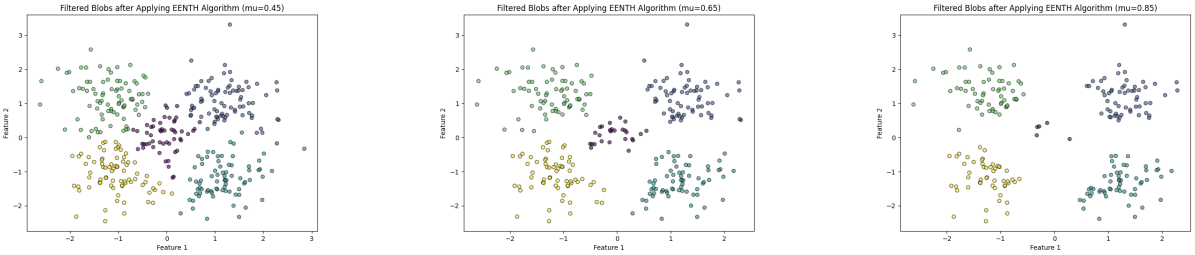
1. **Remove Noise**: First, remove noisy instances using Edited Nearest Neighbor (EEN) [8], where any misclassified instance by its k-nearest neighbors is removed. The outcome is shown in **Figure** 7, with noise removed.
2. **Sort Points**: Next, prioritize removing points farthest from the decision boundary. For each $x_i \in S$ with class y_i, calculate the distance to the nearest point of a different class.
3. **Delete Points**: Starting with points farthest from the boundary, check if any associated points receive more correct classifications without x_i than with it. If so, remove x_i from S.

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

We have evaluated the total 108 different KNN model configurations that can be achieved by assigning the different values available to each of the hyperparameters of the algorithm (k, distance metric, weighting method and voting policy).



(a) $\mu = 0.45$ (b) $\mu = 0.65$ (c) $\mu = 0.85$

Figure 5: EENTH illustration $\mu = 0.45, 0.65, 0.85$

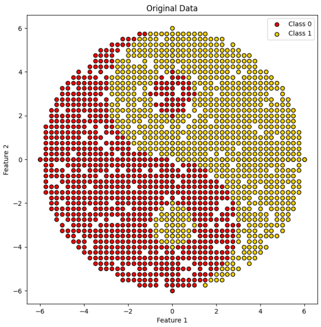


Figure 6: Original Dataset

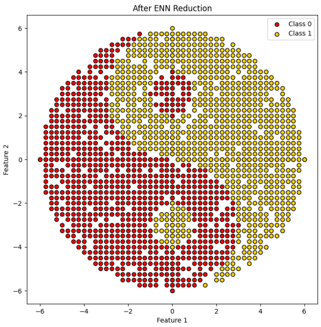


Figure 7: Effect of EEN

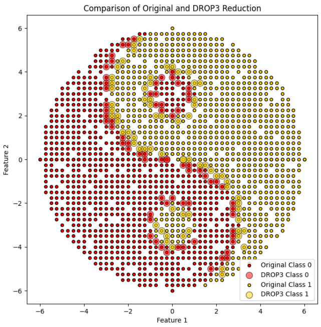


Figure 8: Effect of DROP3

We have trained and tested each of the KNN configurations on the 10 folds of each of the 2 datasets, and stored their achieved accuracy, time and F1 scores in 2 separate CSV files (one per dataset). In the following Subsections, we will discuss some metrics extracted from these results, and we will perform multiple statistical analyses in order to determine whether there are significant differences between the models or not.

3.1.1 Hepatitis

First of all, we summarize the obtained results in a pairplot matrix (Figure 9). Each row and column of the matrix represents a hyperparameter, and the diagonal displays the accuracy histograms for each of the values of the corresponding hyperparameter. Meanwhile, the lower and upper triangles of the matrix display, respectively, the average accuracy and time heatmaps over the folds for each pairwise configurations of the 2 corresponding hyperparameters. With this kind of pairplot we get useful insight into the relationships betweem hyperparameters. We can see a clear trend towards better performance as the value of k goes up; however, this is a preliminary visualization and we cannot extract any real definitive conclusions.

Next, we filter out the top 10 performing models (based on average accuracy over the folds) and we perform a statistical analysis on them to try to find significant differences. After filtering the top 10 models, we apply a Friedman test on their accuracies over the 10 folds, with the null-hypothesis being that there is no significant difference between the models. From the Friedman test we obtain a p-value of **0.9997**, which overwhelmingly supports the null-hypothesis and indicates that there are not any statistically significant differences between the top 10 models.

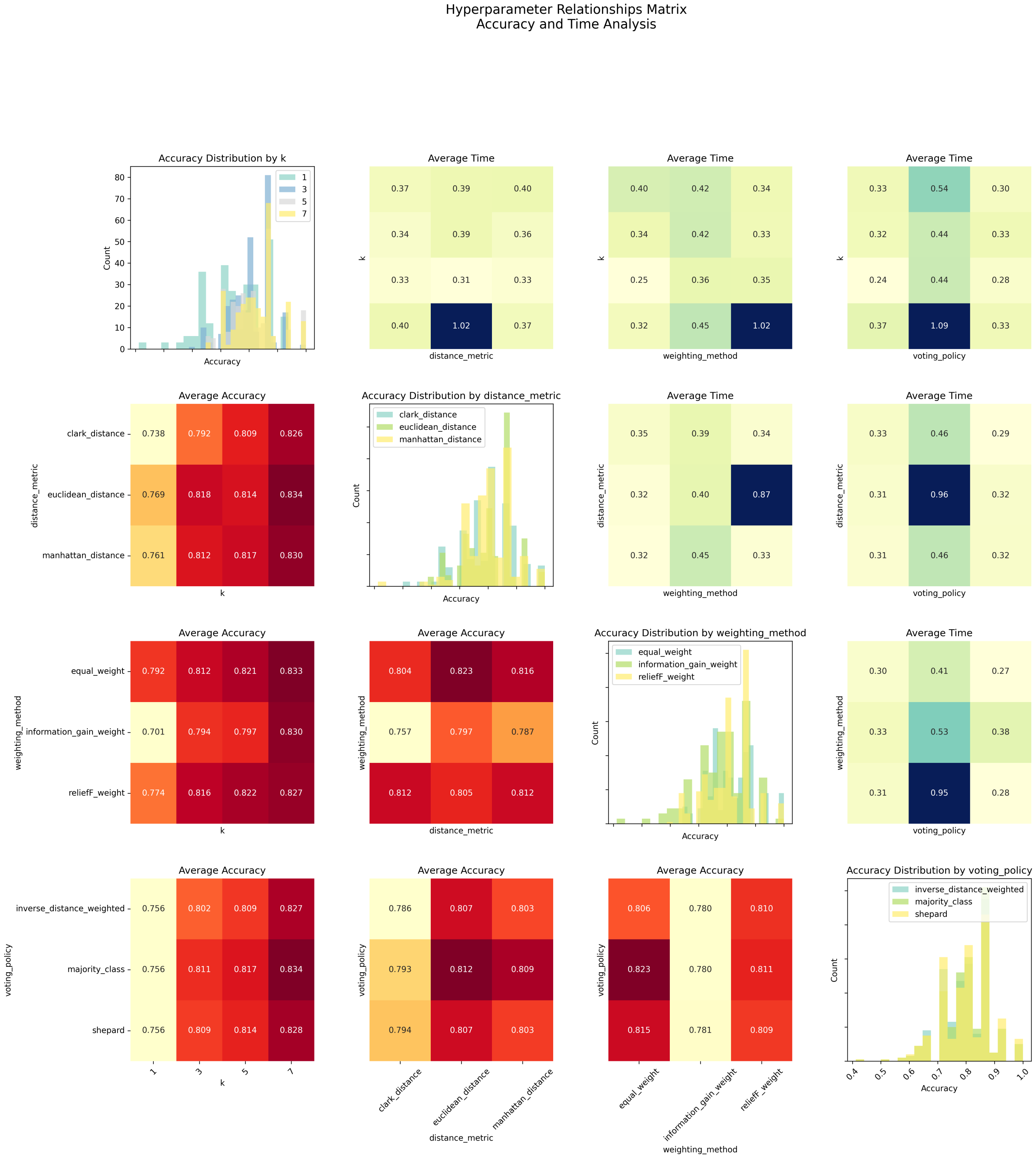


Figure 9: Hepatitis pairplot matrix summary

Since we could not find a model that stands out over the rest with the standard statistical analysis over individual models, we now aim to at least find some indication as to which hyperparameter values are more prominent in succesful models. In order to do this, we performed 4 separate statistical tests by grouping the models according to their configuration of each of the hyperparameters. In other words, we applied a Friedman test to each of the hyperparameters to try to find significant differences between each of their possible values. The obtained p-values for each of the 4 Friedman tests can be seen in Table 1.

k	Distance metric	Weighting Method	Voting Policy
0.2071	0.2765	0.1211	nan

Table 1: Friedman test p-values per hyperparameter.

If we set a level of significance of $\alpha = 0.15$ (which is already a relatively high value), we can only find significant differences between the different weighting methods. Note that the "nan" value in the test over the voting policy means that the ranks across the 4 different voting policies are identical, and therefore the Friedman test cannot be applied. This supports the null-hypothesis of not finding significant differences between the voting policies. We can see the results in Table 2.

Finally, we perform a post-hoc test on the weighting methods in order to further study the significant differences that we have found. In this case, the most appropriate post-hoc test to perform is the Bonferroni test with a control, since there is a clear "standard" choice of weighting method, which is Equal Weights, and we want to find if the other 2 weighting methods have any advantage over it.

	p-value	Difference in accuracy (%)
Information Gain	0.2138	-7.7961%
ReliefF	0.1358	-3.3461%

Table 2: Results of the Bonferroni post-hoc test

Considering the calculated p-values, we can conclude that, with a significance level of $\alpha = 0.15$, we find statistically significant differences between the ReliefF weighting method and the control (Equal Weights), and we do not have strong enough evidence to support the same claim for the Information Gain method. Looking at the average difference in accuracy percentage, we see that the ReliefF weighting method usually performs worse than Equal Weights on the Hepatitis dataset, and therefore we can conclude that it is siginificantly worse to utilize it instead of the standard Equal Weights when classifying samples of this dataset.

Before going to the next section, it is important to note that, since we did not find significant differences between the top performing KNN models, we can assume that all of the top models have statistically the same performance on the Hepatitis dataset. Hence, for simplicity's sake, we will take the one with the highest average accuracy rate (0.8519) in order to perform later tests of comparison between models, and between reduction methods. This model has a k value of 7, the Manhattan Distance, the Equal Weight method, and the Majority Class voting.

3.1.2 Mushroom

For the Mushroom dataset we will follow essentially the same steps as we did for the Hepatitis dataset, starting with visualizing the results in a pairplot matrix (Figure 10). Interestingly, in this case we observe the opposite trend on the values of k as we did for the Hepatitis dataset, which indicates better perfomance of lower values of k this time. We can also see a tendency towards slightly better accuracy when using the Clark distance; however, we also observe extreme values of the computation time for said distance metric.

As we did before, we proceed with a Friedman test to try to find significant differences among the top performing models. This time we get a p-value of **nan**, which indicates that there is absolutely no difference in the ranks between the top 10 models. When observing the results, we find that all of the 10 models have an average accuracy of 1.0, which means that they all perfectly classify all of the test samples across the 10 folds of the Mushroom dataset. The only conclusion that we can take from this result is that the Mushroom dataset is not complex enough to show any difference in performance between these model configurations.

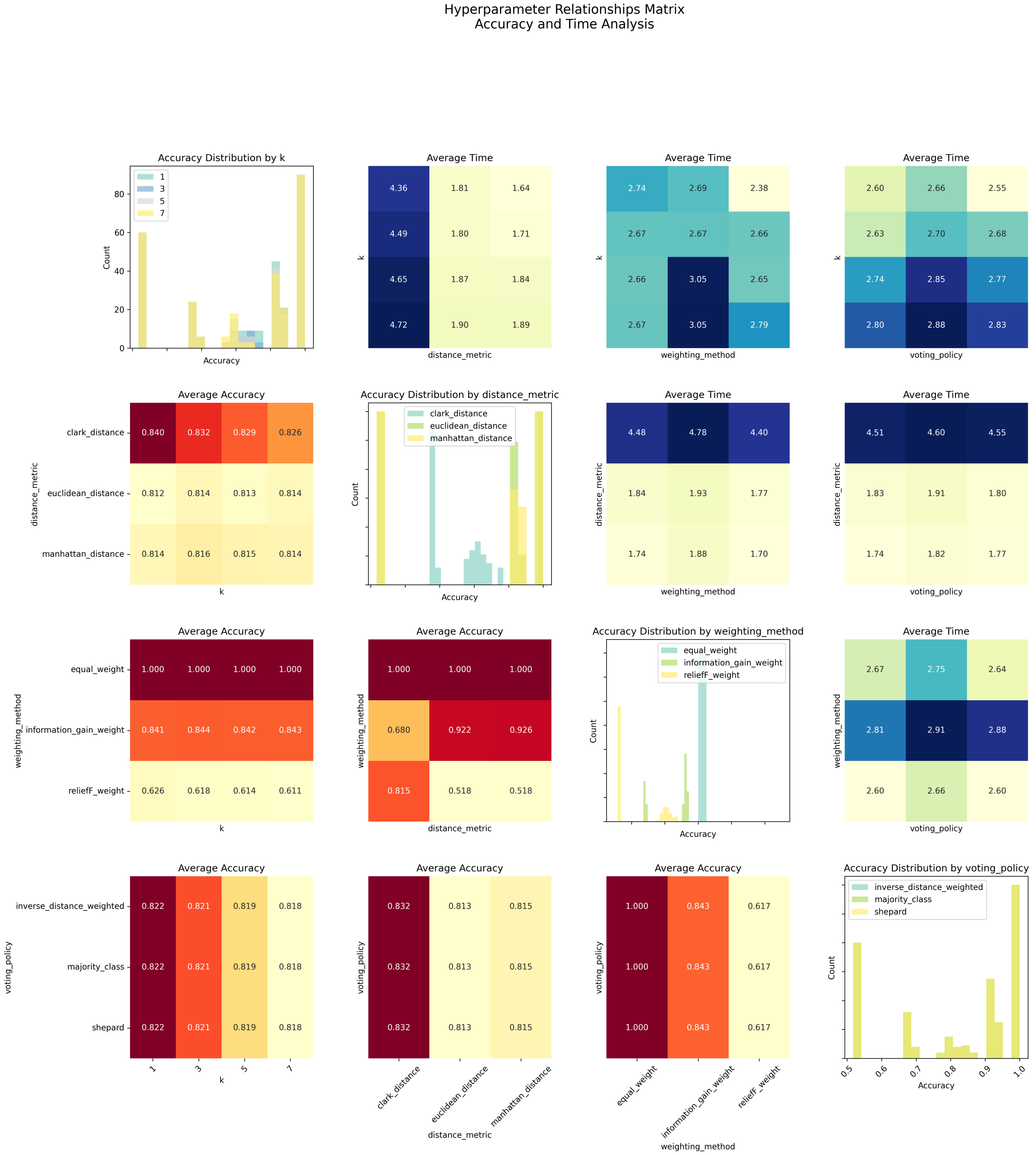


Figure 10: Mushroom pairplot matrix summary

Next, we perform a statistical test on the hyperparameters in order to find significant differences between their possible values. The results can be found in Table 3.

k	Distance metric	Weighting Method	Voting Policy
nan	nan	0.0000	nan

Table 3: Friedman test p-values per hyperparameter.

We see that the Friedman test cannot find any performance differences whatsoever between the different values of k, distance metric and voting policy. On the other hand, it is interesting to find that the p-value is **0.0000** for the weighting method, which essentially *guarantees* (due to numerical approximation of the p-value) significant differences between the various weighting methods.

After applying the Bonferroni post-hoc test on the weighting methods, we get the results displayed on Table 4.

	p-value	**Difference in accuracy (%)**
Information Gain	0.0039	-8.0502%
ReliefF	0.0039	-48.2029%

Table 4: Results of the Bonferroni post-hoc test

With these results, we can state that there is very strong evidence supporting that there is indeed significant differences between the Equal Weights method and the other 2; since even with a significance level of $\alpha = 0.01$ (which is more stringent than the $\alpha = 0.15$ that we have for this test) we would reject the null-hypothesis. Looking at the average difference in accuracy percentage, we can see that both weighting methods show poorer average performance than the Equal Weights, and therefore it is detrimental to use them over the "standard" weighting method.

Similarly as with Hepatitis dataset, we could not find significant differences between the top performing models upon the Mushroom dataset. Therfore, for future comparisons we could utilize any of the top 10 models (all of them have accuracy rate of 1), so we choose one that seems to have the simplest configuration of hyperparameters: a k value of 1, the Euclidean Distance, the Equal Weight method, and Majority Class voting.

3.2 SVM Results

A total of 16 different configurations of the SVM algorithm were studied, combining the four kernels mentioned in Section 2.6 (linear, RBF, polynomial, and sigmoid) with four different values for the parameter C. After training and testing each configuration on the datasets, an initial preselection of the best models based on accuracy was conducted. A Friedman test was then performed to determine whether there were statistically significant differences in accuracy between the models. For each dataset (hepatitis and mushroom), the model with the highest accuracy was selected.

3.2.1 Hepatitis

For this dataset, the five models with the highest mean accuracies were selected from the initial 16. The mean accuracies and hyperparameters of these models are presented in Figure 11. A more in-depth analysis of these five models can be found in Table 5.

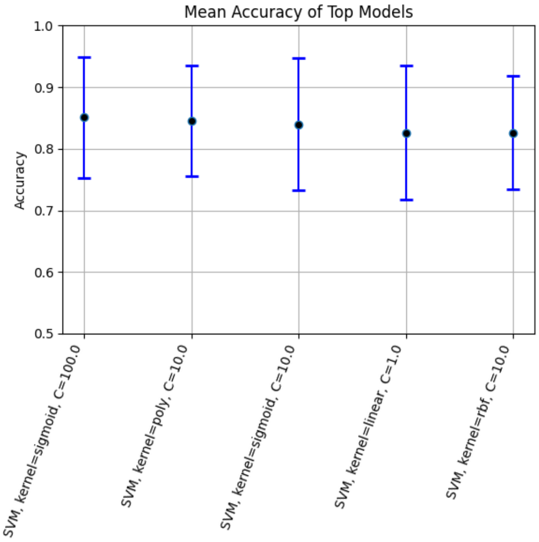


Figure 11: Mean accuracy for the five models of SVM with higher mean accuracies.

Model	Accuracy	F1 Score	Time (s)
SVM, kernel=sigmoid, C=100.0	0.851 ± 0.010	0.91 ± 0.06	0.0006 ± 0.0005
SVM, kernel=poly, C=10.0	0.84 ± 0.09	0.91 ± 0.05	0.0008 ± 0.0005
SVM, kernel=sigmoid, C=10.0	0.84 ± 0.11	0.90 ± 0.07	0.0005 ± 0.0007
SVM, kernel=linear, C=1.0	0.82 ± 0.11	0.89 ± 0.07	0.0011 ± 0.0006
SVM, kernel=rbf, C=10.0	0.82 ± 0.09	0.89 ± 0.06	0.0011 ± 0.0006

Table 5: Performance metrics for the best five SVM models with different kernels and regularization parameters.

The Friedman test was performed, yielding a p-value of 0.4292, leading us to accept the null hypothesis that there are no significant differences between the five models studied. The model with hyperparameters kernel=sigmoid and C=100.0, as shown in Table 5, exhibited higher mean accuracy, a higher mean F1 score, and lower mean computation time. Therefore, this model will be considered for further analysis on the reduced datasets.

3.2.2 Mushroom

For this dataset, the ten models with the highest mean accuracies were selected from an initial set of sixteen. Although a larger number of models was considered in the preliminary analysis, no significant differences in accuracy were observed, with over five models achieving 100% accuracy. Consequently, the Friedman test could not be applied, as there was insufficient variance in the accuracies of the ten selected models. This lack of variability in performance metrics rendered the Friedman test ineffective for distinguishing among these top models. The values obtained for the studied metrics are shown in Table 6

Model	Accuracy	F1 Score	Time (s)
SVM, kernel=linear, C=1.0	1.0000 ± 0.0000	1.0000 ± 0.0000	0.0031 ± 0.0005
SVM, kernel=linear, C=10.0	1.0000 ± 0.0000	1.0000 ± 0.0000	0.0038 ± 0.0011
SVM, kernel=rbf, C=100.0	1.0000 ± 0.0000	1.0000 ± 0.0000	0.0077 ± 0.0013
SVM, kernel=linear, C=100.0	1.0000 ± 0.0000	1.0000 ± 0.0000	0.0034 ± 0.0006
SVM, kernel=poly, C=10.0	1.0000 ± 0.0000	1.0000 ± 0.0000	0.0023 ± 0.0007
SVM, kernel=poly, C=100.0	1.0000 ± 0.0000	1.0000 ± 0.0000	0.0021 ± 0.0005
SVM, kernel=rbf, C=10.0	1.0000 ± 0.0000	1.0000 ± 0.0000	0.0087 ± 0.0022
SVM, kernel=rbf, C=1.0	0.9994 ± 0.0010	0.9994 ± 0.0011	0.0245 ± 0.0025
SVM, kernel=poly, C=1.0	0.9978 ± 0.0015	0.9977 ± 0.0016	0.0070 ± 0.0018
SVM, kernel=linear, C=0.1	0.992 ± 0.004	0.992 ± 0.004	0.0060 ± 0.0017

Table 6: Performance metrics for various SVM models with different kernels and regularization parameters (C).

3.3 Comparation between KNN and SVM algorithms

Finally, to better understand the performance of the two algorithms studied (KNN and SVM), they were compared for each of the two datasets. Specifically, the hyperparameters that yielded the best model, according with what was discussed in the previous subsections, were selected.

3.3.1 Hepatitis

The two compared models were SVM with hyperparameters: kernel='sigmoid' and C=100 with KNN with hyperparameters: k=7, manhattan distance, equal weight and majority class. The mean metrics valued can be consulted in Table 7

A deeper analysis was carried out to determine if there were significant differences between each of the metrics; for this, the Wilcoxon Signed-Rank Test was performed on each metric.

Model	Accuracy	Time (s)	F1 Score
SVM	0.85 ± 0.10	0.0010 ± 0.0000	0.91 ± 0.06
KNN	0.85 ± 0.06	0.004 ± 0.001	0.84 ± 0.08

Table 7: Performance metrics for SVM and KNN models.

For the accuracy metric, a p-value of 1 was obtained, indicating no significant difference between the two algorithms. Regarding time performance, although the mean time for SVM was considerably lower than for KNN, the test could not be performed due to insufficient variance in the time metric values for each algorithm. Finally, when analyzing the F1 score, the Wilcoxon test showed a p-value of 0.0209, leading to the conclusion that SVM performed significantly better than KNN.

3.3.2 Mushroom

The two compared models were SVM with hyperparameters: kernel='linear' and C=1 with KNN with hyperparameters: k=1, euclidian distance, equal weight and majority class. The mean metrics valued can be consulted in Table 8.

Model	Accuracy	Time (s)	F1 Score
SVM	1.0000 ± 0.0000	0.004 ± 0.002	1.0000 ± 0.0000
KNN	1.0000 ± 0.0000	2.0 ± 0.8	1.0000 ± 0.0000

Table 8: Performance metrics for the best SVM and KNN models on the Mushroom dataset.

As with the previous dataset, a deeper analysis was carried out to determine if there were significant differences between each of the metrics using the Wilcoxon Signed-Rank Test on each metric.

For the accuracy and F1 score metrics, the test was not performed due to lack of variance. However, time performance showed a significant difference between the two methods, as anticipated from the mean values shown in Table 8. The p-value obtained was 0.002, leading to the conclusion that SVM performed significantly better than KNN.

3.4 KNN Reduction Results

In this section we will discuss the results obtained when training and testing KNN models on reduced versions of the folds from our 2 original datasets, in order to perform statistical analyses on the reduction methods and extract conclusions about their different impact on classifier performance.

We perform this experiment separately for the Hepatitis and Mushroom datasets, and for each of them we employ the KNN model configuration that we concluded to be the best for that dataset, in Section 3.1. We first train and test the corresponding KNN models over the original folds of the datasets, and then we repeat the process over the reduced folds for each of the 3 reduction methods: GCNN, EENTH, and DROP3. We store this evaluation data in a CSV for each of the datasets, for its later evaluation and statistical analysis.

3.4.1 Hepatitis

Before starting the statistical analysis, let us first look at the results, summarized in Figure 12, where we can see the accuracy distributions per reduction method (in boxplots) and the storage percentage comparison, against the "no-reduction method" (i.e. `NONE`). The storage percentage refers to the percentage of samples each of the reduction methods keeps from the original folds.

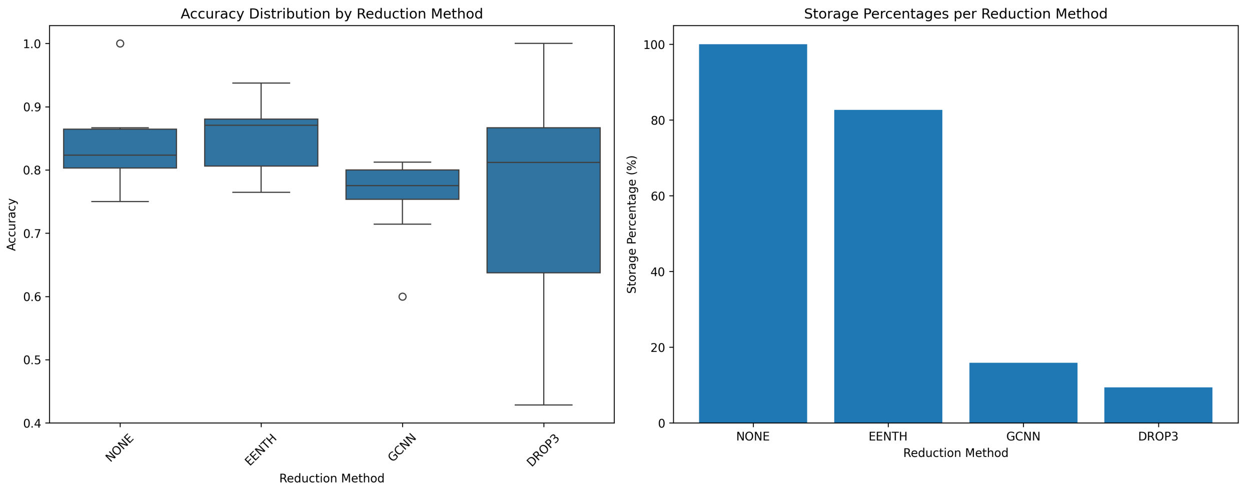


Figure 12: Hepatitis accuracy and storage comparison

We observe that the DROP3 reduction method is the most inconsistent of them, with a very wide accuracy distribution, followed by the GCNN method, which is more consistent but not as accurate as NONE and EENTH. Interestingly, these 2 methods (GCNN and DROP3) are also the most radical ones in terms of reduction of samples, which is clearly shown in the Storage Percentage graph. We can start to see a trend where the more extreme the reduction percentage, the more inconsistent the models performance; which is an intuitive deduction, since we lose information when reducing samples.

We now proceed with the statistical analysis to study whether there are significant differences between the impact of each reduction method on the performance of the KNN classifier. For this, we apply a Friedman test on the results, grouping by reduction method. The p-value that we obtain from this test is **0.0093**; hence, if we establish a level of significance of $\alpha = 0.15$, we can reject the null-hypothesis and state that there are indeed statistically significant differences between the performances of the classifier when applying the different reduction methods.

The next step is performing a post-hoc test to determine among which of the reduction methods there are significant differences. There is an obvious "control" reduction method, which is no-reduction, so we apply a Bonferroni test to study the differences of each of the other reduction methods when compared to the original training sets. The results are displayed on Table 9.

	p-value	**Difference in accuracy (%)**
EENTH	1.0000	2.2288%
GCNN	0.0527	-9.3174%
DROP3	0.4140	-9.4072%

Table 9: Results of the Bonferroni post-hoc test

From these results we can extract multiple conclusions. For one, we see that on average the EENTH reduction method seems to have slightly better accuracy than the control; however, the p-value of **1.000** indicates that this difference could be due to pure chance, and it is not meaningful at all. On the other hand, the GCNN obtains a p-value of **0.0527**, which means that we can state with our level of significance of $\alpha = 0.15$ that there are statistically significant differences with the control; and the negative difference in accuracy percentage indicates that these differences are against using GCNN over the control. Lastly, DROP3 shows worse performance than no-reduction, but the p-value of **0.4140** indicates that this decline in classification performance is not too significant, and could be due to randomness or noise in the data.

The final conclusion of these results is that, for the Hepatitis dataset, the GCNN reduction method causes significantly worse classification performance than not reducing the data, while the other 2 reduction methods show no meaningful difference.

3.4.2 Mushroom

In this section we will follow for the Mushroom dataset the same procedure as we did in the previous section for the Hepatitis dataset; starting with a plot to summarize the results (Figure 13).

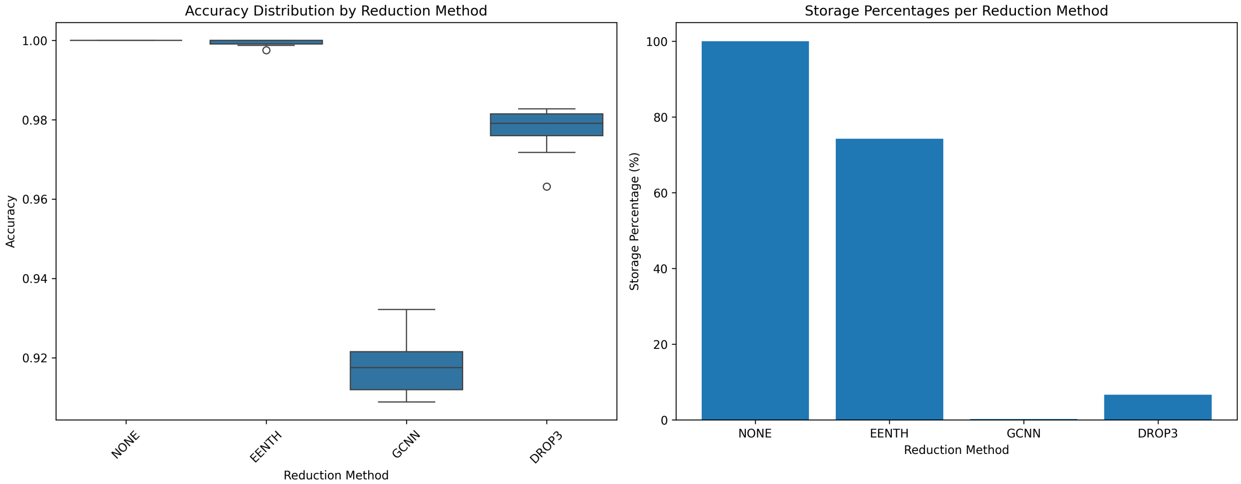


Figure 13: Mushroom accuracy and storage comparison

Although the plots seem more radicalized than for the Hepatitis dataset (due to extremely high performance of the classifier), they still follow a similar trend as for the Hepatitis dataset, which supports the idea that as the reduction gets more intense, the KNN performance declines. This time, however, we can observe an extreme difference (relatively, since all achieve accuracy over 0.9) between the GCNN and the other 3 methods, which might indicate that it is especially not suitable for this specific dataset. The DROP3 seems to keep a reasonable performance while achieving a high storage reduction percentage.

After applying the Friedman test on the results we obtain a p-value of **0.000**, which strongly suggests we can reject the null-hypothesis with any level of significance. Therefore, we deduce that there is indeed significant differences in classification performance on the Mushroom dataset with the different reduction methods.

Again, we perform a Bonferroni post-hoc test to further study these differences, comparing against the control (no-reduction). The results are shown in Table 10.

	p-value	**Difference in accuracy (%)**
EENTH	0.3264	-0.0493%
GCNN	0.0059	-8.2349%
DROP3	0.0059	-2.2644%

Table 10: Results of the Bonferroni post-hoc test

Considering the level of significance of $\alpha = 0.15$ that we have employed for the KNN statistical analyses, and looking at the differences in accuracy percentages, we can conclude from these results that the GCNN and the DROP3 reduction methods cause significantly poorer classification performance than no-reduction, while EENTH does not show statistically meaningful differences.

3.5 SVM Reduction Results

The best model for each dataset was trained and tested on the reduced dataset obtained after applying the methods described in Section 2.7. Subsequently, an additional statistical analysis was carried out to determine which reduction method provided better performance.

3.5.1 Hepatitis

The SVM model with hyperparameters ‘kernel=sigmoid‘ and ‘C=100.0‘ was trained and tested on the reduced data samples obtained for each reduction method (DROP3, EENTH, and GCNN). The mean values of the evaluated statistical metrics are presented in Table 11. Again, the Friedman test was conducted, yielding a p-value of 0.0057 for accuracy and 0.0210 for the F1-score. Due to these low values, the null hypothesis was rejected, and a post-hoc test was implemented. The Nemenyi test was chosen to identify specific differences between methods. Significant differences were found between EENTH and GCNN, and also between GCNN and NONE (understanding NONE as the original dataset without any reduction method), as it can be seen in the Figure 14

Reduction Method	**Accuracy**	**F1 Score**	**Time (s)**
DROP3	0.79 ± 0.11	0.86 ± 0.07	0.0009 ± 0.0006
EENTH	0.85 ± 0.09	0.91 ± 0.06	0.0011 ± 0.0007
GCNN	0.64 ± 0.13	0.73 ± 0.12	0.0014 ± 0.0007
NONE	0.85 ± 0.10	0.91 ± 0.06	0.0011 ± 0.0004

Table 11: Performance metrics for different reduction methods for the SVM algorithm with kernel=sigmoid and C=100.

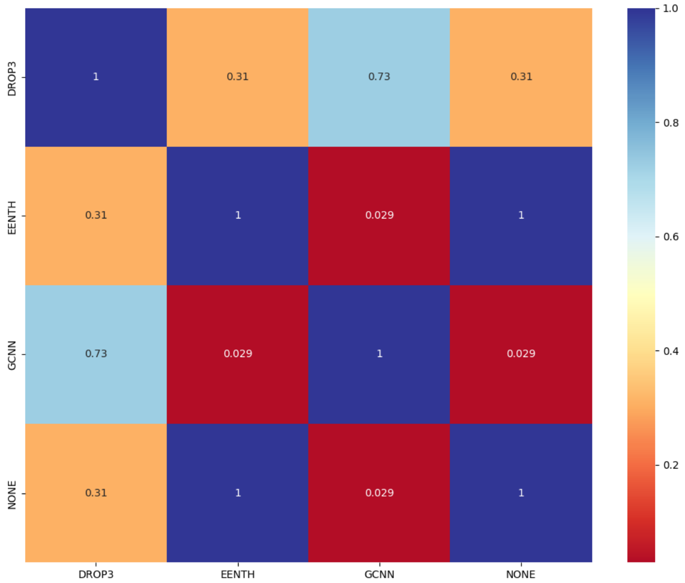


Figure 14: Normalized measure of the difference between the mean ranks of the methods. Values closer to 0 indicate stronger evidence of a difference between methods, while values closer to 1 indicate greater similarity.

3.5.2 Mushroom

To study the reduction methods, an SVM model with hyperparameters kernel=linear and C=1, which achieved 100% accuracy, was trained and tested on the reduced data samples obtained from each reduction method (DROP3, EENTH, and GCNN). The mean values of the evaluated statistical metrics are presented in Table 12. These values were very similar, and the Friedman test could not be applied due to insufficient variance between the metrics obtained with the different reduction methods.

4 Conclusion

This study evaluated the performance of KNN, SVM, and various training reduction techniques on multiple datasets. Our analysis aimed to identify optimal configurations for each algorithm and assess the impact of

Method	Accuracy	F1 Score	Time (s)
DROP3	0.985 ± 0.004	0.9844 ± 0.004	0.0020 ± 0.0008
EENTH	0.9985 ± 0.0014	0.9985 ± 0.0015	0.0023 ± 0.0005
GCNN	0.974 ± 0.017	0.97 ± 0.02	0.0015 ± 0.0008
NONE	1.0000 ± 0.0000	1.0000 ± 0.0000	0.004 ± 0.002

Table 12: Mean performance metrics by method, showing accuracy, F1 score, and processing time.

specific parameters on classification accuracy and computational efficiency.

First, we performed an in-depth evaluation on the different configurations of the KNN model. Our analysis found no evidence that the choice of k significantly impacts accuracy, although certain specific trends are observed for each dataset. Similarly, no significant differences were found among different distance metrics or voting policies. Notably, however, it is consistently observed that the weighting method does have a significant impact on KNN accuracy, with a favorable trend toward equal weighting.

For SVM, we observed that different kernel functions produced varying results depending on the dataset. In the hepatitis dataset, the sigmoid kernel with $C = 100.0$ achieved the highest accuracy and F1 score, making it the optimal choice for that dataset. In contrast, for the mushroom dataset, multiple configurations achieved perfect accuracy, indicating limited complexity in the data.

Despite the limitations in the statistical conclusions of the comparison between the best SVM and KNN methods for each dataset, primarily due to limited variation in the metrics used, the results adequately capture the nature of each method. In the hepatitis dataset, it was concluded with precision that the *f1 score* was higher for SVM compared to KNN. Given the medical nature of this data, achieving a higher *f1 score* is crucial, as it indicates that the model has both high precision and recall, correctly identifying positive cases without generating excessive false alarms. In the mushroom dataset, however, no significant differences were observed: both SVM and KNN achieved 100% in accuracy and *f1 score* across all tests. The most relevant difference between these two algorithms was in execution time, as shown by the Wilcoxon test. This result aligns with expectations, as KNN is a *lazy learning* method, while SVM is *eager learning.*

We have also compared the performance of the KNN classifier when training it on reduced datasets instead of the original training sets. From this study we consistently found significant differences across the various reduction methods. GCNN consistently resulted in poorer classification performance compared to the original datasets. DROP3 also appeared to reduce performance, though less conclusively than GCNN. In contrast, EENTH showed no significant difference from the original datasets, suggesting it may be advantageous, by achieving statistically similar results with fewer samples.

When evaluating the performance of the SVM classifier over the reduced datasets, we obtained similar results as with the KNN. With the Hepatitis dataset, the GCNN and DROP3 caused worse performance while EENTH showed no meaningful difference with the original training set. On the Mushroom dataset, there was not enough variance between the metrics to apply the statistical tests; nevertheless, the mean values of the metrics show very similar performance for all the reduction methods, which leads us to conclude again that there is no significant differences between the reduction mehtods.

Feature selection, though often beneficial for high-dimensional data, was not applied in this study as it was out of scope. However, it could potentially yield better results by reducing complexity and enhancing interpretability without significantly altering the training set size. Based on our findings, feature selection may be preferable over certain reduction methods for balancing data simplification and accuracy preservation, though further study would be needed to confirm this hypothesis.

In summary, our results indicate that feature weighting on KNN, specific SVM configurations, and certain training reduction techniques may significantly influence model performance but that each dataset requires its own refining for getting the best specific results. This study provides a framework for selecting machine learning methods and configurations based on dataset characteristics, supporting the development of effective and computationally efficient models.

References

[1] Fu Chang, Chin-Chin Lin, and Chi-Jen Lu. Adaptive prototype learning algorithms: Theoretical and experimental studies. Journal of Machine Learning Research, 7:2125–2148, 2006.

[2] I. Dagher. Quadratic kernel-free non-linear support vector machine. Journal of Global Optimization, 41:15–30, 2008.

[3] Scikit learn Developers. Support vector machines. `https://scikit-learn.org/stable/modules/svm.html`, 2024. Accessed: 2024-10-23.

[4] Hsuan-Tien Lin and Chih-Jen Lin. A study on sigmoid kernels for svm and the training of non-psd kernels by smo-type methods. Neural Computation, 06 2003.

[5] Daniele Micci-Barreca. A preprocessing scheme for high-cardinality categorical attributes in classification and prediction problems. SIGKDD Explor. Newsl., 3(1):27–32, July 2001.

[6] I. Roman, R. Santana, A. Mendiburu, et al. In-depth analysis of svm kernel learning and its components. Neural Computing and Applications, 33:6575–6594, 2021.

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

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

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