

Introduction to Machine Learning: Work 2

Classification with Lazy Learning and SVM

Pedro Agúndez
Bruno Sánchez
María del Carmen Ramírez
Antonio Lobo

November 2, 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:

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

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

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

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 $T \subseteq D_1$.

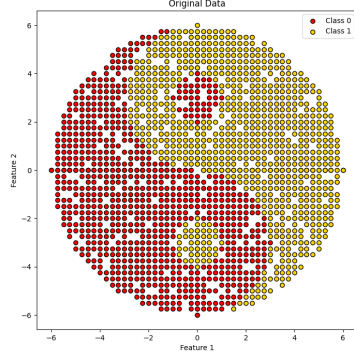


Figure 1: Original Dataset

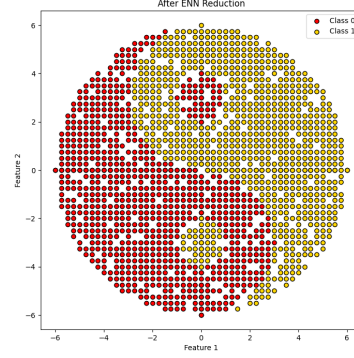


Figure 2: Effect of EEN

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

We have evaluated the total 108 different KNN model configurations that can be achieved by assigning the different values available to each of the hyper-

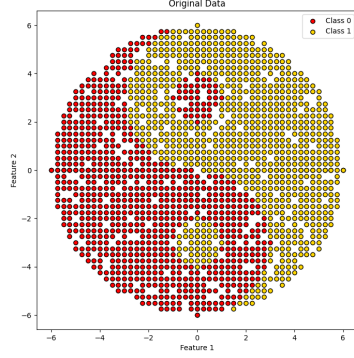


Figure 3: Original Dataset

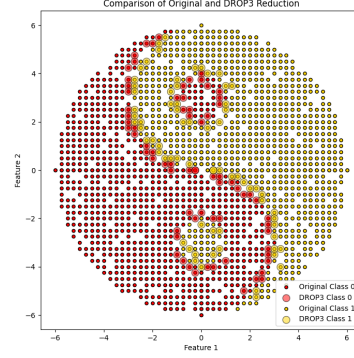


Figure 4: Effect of DROP3

parameters of the algorithm (k , distance metric, weighting method and voting policy).

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 5). 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 between 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.

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 successful 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 significantly 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 6). Interestingly, in this case we observe the opposite trend on the values of k as we did for the Hepatitis dataset, which indicates better performance 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.

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. Therefore, 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

Describe the SVM results.

3.3 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.

3.3.1 Hepatitis

Before starting the statistical analysis, let us first look at the results, summarized in Figure 7, 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.

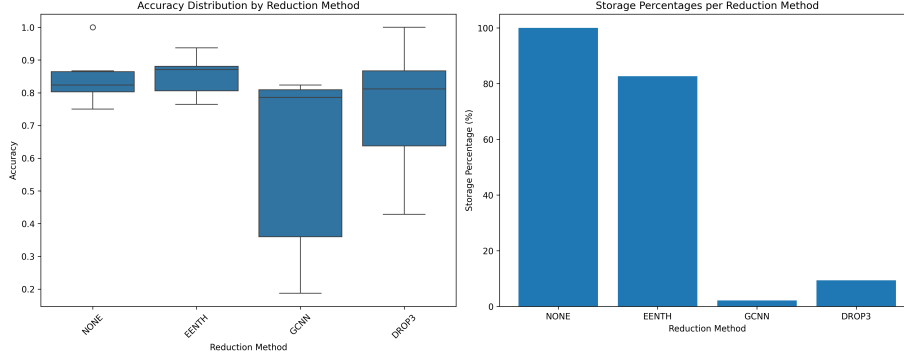


Figure 7: Hepatitis accuracy and storage comparison

We observe that the GCNN reduction method is the most inconsistent of them, with a very wide accuracy distribution, followed by the DROP3 method, which is more consistent but not as much as NONE and EENTH. Interestingly, the more inconsistent 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 **0.0132**; 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 5.

	p-value	Difference in accuracy (%)
EENTH	1.000	2.2288%
GCNN	0.084	-26.3502%
DROP3	0.414	-9.4072%

Table 5: 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 **0.084**, 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 **0.414** 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.

3.3.2 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 8).

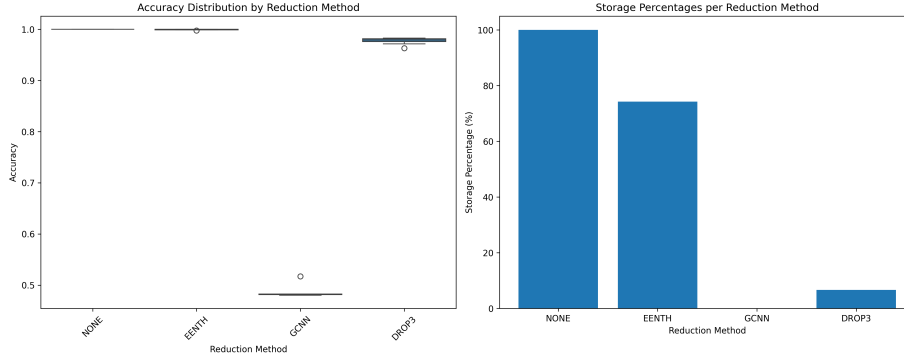


Figure 8: Mushroom accuracy and storage comparison

Although the plots are 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 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 achieve the highest storage reduction percentage while keeping a reasonable performance.

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 6.

	p-value	Difference in accuracy (%)
EENTH	0.3264	-0.0493%
GCNN	0.0059	-51.4531%
DROP3	0.0059	-2.2644%

Table 6: 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.

3.4 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.

Hyperparameter Relationships Matrix
Accuracy and Time Analysis

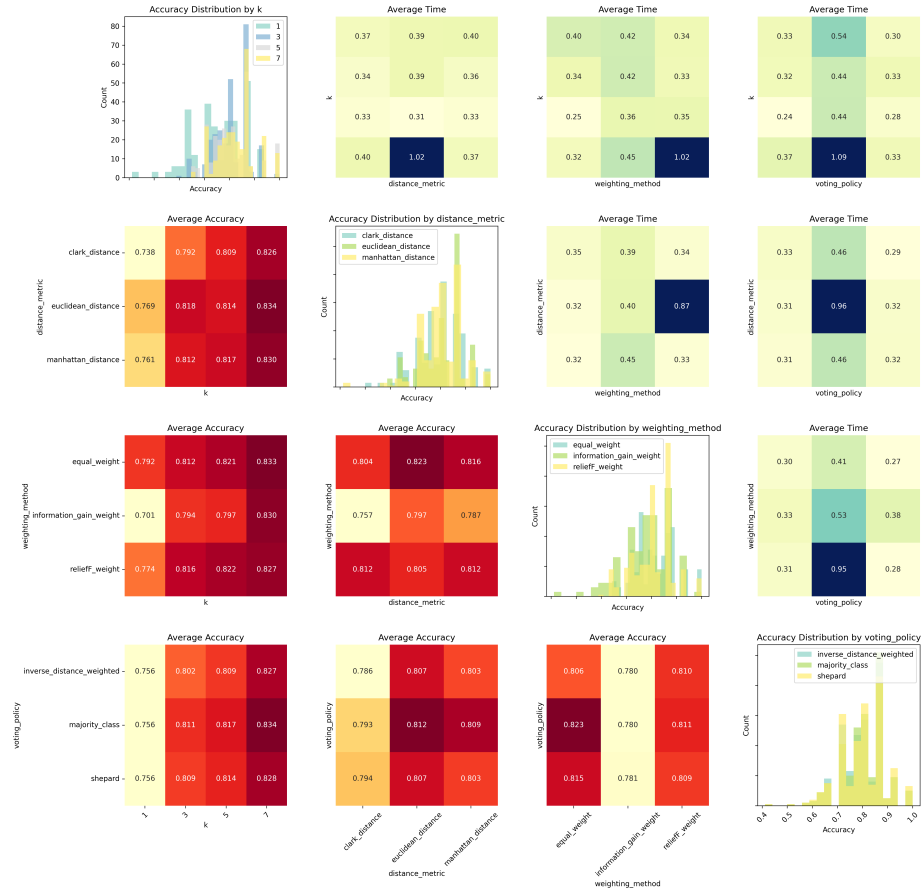


Figure 5: Hepatitis pairplot matrix summary

Hyperparameter Relationships Matrix
Accuracy and Time Analysis

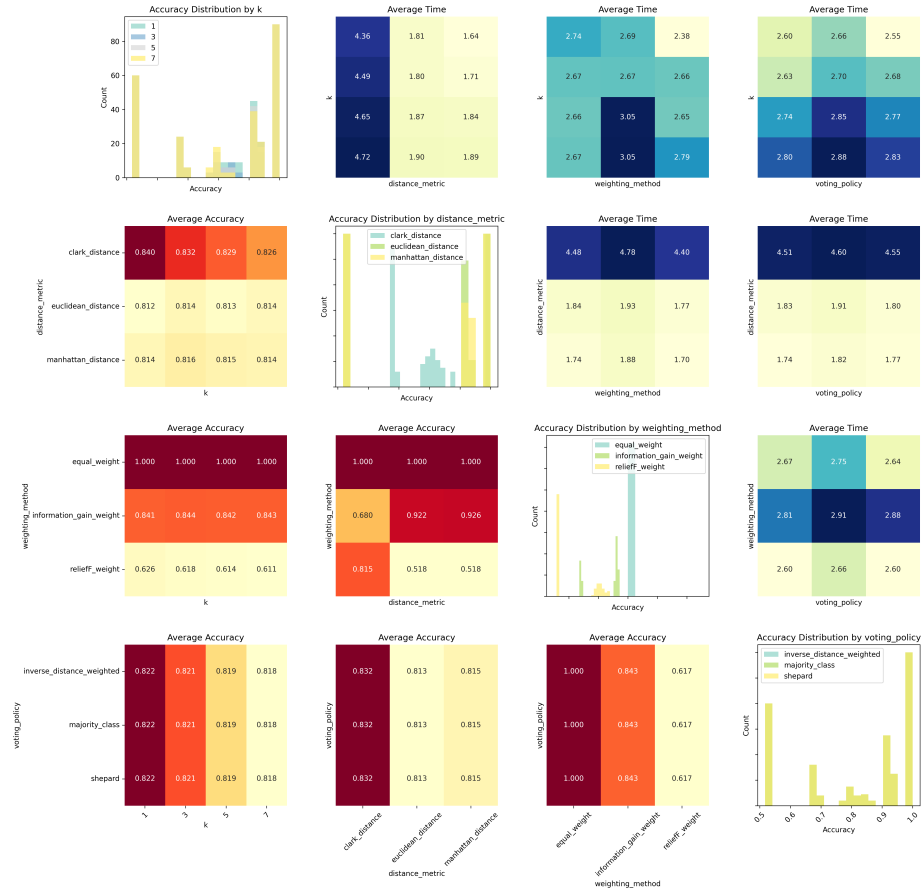


Figure 6: Mushroom pairplot matrix summary