Introduction to Machine Learning: Work 3. Clustering and Visualization.

Andreu Garcies Ramon[1], Maria Guasch Torres[1], Natalia Muñoz Moruno[1] and Helena Sánchez Ulloa[1]

[1] *Master in Artificial Intelligence, Universitat Politècnica de Catalunya UPC, Barcelona, Catalonia.*

Abstract—Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Keywords—

I. INTRODUCTION

Variable	Type	Description	Missing values
A1	Categorical	Gender {b, a}	Yes
A2	Continuous	Age in years	Yes
A3	Continuous	Debt ratio	No
A4	Categorical	Marital status {u, y, l, t}	Yes
A5	Categorical	Bank Customer {g, p, gg}	Yes
A6	Categorical	Education level {c, d, cc, i, j, k, m, r, q, w, x, e, aa, ff}	Yes
A7	Categorical	Ethnicity {v, h, bb, j, n, z, dd, ff, o}	Yes
A8	Continuous	Years Employed	No
A9	Categorical	Prior Default {t, f}	No
A10	Categorical	Employed {t, f}	No
A11	Continuous	Credit Score	No
A12	Categorical	Drivers License {t, f}	No
A13	Categorical	Citizen {g, p, s}	No
A14	Continuous	Income	Yes
A15	Continuous	Zip Code	No

Table 1: Credit Approval dataset feature description.

Class	Num. of instances	Percentage (%)
+	307	44,49
-	383	55,51

Table 2: Output of Credit Approval dataset

II. Datasets

a. Credit Approval

The first dataset chosen is Credit Approval, from now on denoted as *credit-a*, from the UCI repository [1]. This dataset contains 690 instances and 15 features and is used for classification. Specifically, it contains data from credit card applications, combining continuous and categorical attributes. It also contains missing values.

However, the names and values of the attributes have been encrypted to protect confidentiality. For this reason, the official source [1] does not indicate what each attribute consists of. However, based on [2] and inference of the values, it is hypothesized that each feature corresponds to the descriptions listed in Table 1.

Finally, as can be seen in Table 2, it is a binary classification problem with classes representing whether the credit has been approved or not, and the classes are fairly balanced, with 55.51% of the samples representing class '-' and 44.49% of the samples representing class '+' [1].

b. Pen-Based Recognition of Handwritten Digits

For the second dataset, Pen-Based Recognition of Handwritten Digits [3], from now on denoted as *pen-based*, was chosen. This dataset consists of a database of digits from 44 different writers, with a total of 10,992 instances. To capture how people write numbers, a digitizing tablet (WACOM PL-100V) was used, with a sampling frequency of 100 ms and a resolution of 500×500 pixels. The data captured by the tablet were the x and y coordinates and the pressure level of the pen; however, the latter was ignored in the dataset.

After capturing the points, the coordinates were normalized, making the digits comparable. To do this, translation was eliminated by centering, and the scale was eliminated so that all digits had the same relative size.

One of the problems encountered was that the digits had a different number of captured points. However, in order to use the data in classifiers, all digits must have the same vector size. To achieve this, spatial resampling was used. This method uses linear interpolation to obtain a sequence of equally spaced points in arc lenght, with the digits represented in a 2T-dimensional vector. After testing several values for T, they found that $T = 8$ presented the best balance between accuracy and complexity. Thus, this dataset contains 16 features.

As can be seen in Table 3, there are 10 different classes, representing the digits 0 to 9, which are fairly balanced.

c. TAO-Grid

The TAO-Grid dataset, from now on denoted as *grid*, was used for statistical analysis purposes and consists of a two-dimensional geometric pattern inspired by the Yin-Yang symbol. Each instance corresponds to a point on the plane (x, y coordinates) and it is assigned a class label (`black` or `white`). The dataset consists of 1888 instances and 2 features.

Class	Num. of instances	Percentage(%)
0	1143	10,39
1	1143	10,39
2	1144	10,41
3	1055	9,59
4	1144	10,41
5	1055	9,59
6	1056	9,61
7	1142	10,39
8	1055	9,59
9	1055	9,59

Table 3: Output of Pen-based dataset

d. Vowel

This dataset was used for statistical analysis and consists of 990 instances and 12 features. One feature indicates whether the instance belongs to the training or test set, another identifies the speaker (15 in total), other the gender and the other remaining correspond to real values [4]. The labels represent 11 classes of English vowels, coded as "h+d" with a central vowel.

e. Preprocessing

For data preprocessing, the training and test files were first merged. This step was necessary to apply the same preprocessing to all folds, which requires the entire dataset to compute global statistics, such as feature means. To do this, the function `merge_fold` was implemented. Additionally, this function also removes instances with more than three missing values, as these were considered to contain excessive missing data for relaible imputation. After this initial step, the *credit-a* dataset contained 684 examples while *pen-based* remained the same as it had no missing values.

Next, the remaining missing values were imputed. On the one hand, in the case of categorical data, two methods were implemented: KNNImputer and imputation using the most frequent value. In the first method, imputation is performed using the k-NN, so that the average value of these is used to impute the missing value. It should be noted that two instances are defined as similar if the features that are not missing present similar values [5] . This method was implemented using the `KNNImputer` [5] class from scikit-learn[1]. The second method simply replaces the missing value with the most frequent value in the column. In this case, the SimpleImputer function [6] from scikit-learn was used, with the `most_frequent` strategy.

On the other hand, for the imputation of numerical data, the scikit-learn's `SimpleImputer` function [6] was again used. This method allows data to be imputed based on the mean, median, most frequent value, and a constant value.

For the normalization of numerical data, scikit-learn's `MinMaxScaler` [7] was used. This transformation scales the data to a given range, in our case [0,1]. The transition is given by Equation 1.

$$X_{\text{scaled}} = X_{\text{std}} \cdot (\max - \min) + \min. \tag{1}$$

The categorical variables were coded using One Hot encoding, transforming each categorical feature into multiple binary columns, representing the different classes. To do this, the `OneHotEncoder` [8] function from scikit-learn was used. It should be noted that for features that were already binary initially, `LabelEncoder` [9], also from scikit-learn, was used instead, converting the classes to 0 and 1.

Finally, the preprocessed data was separated again into training and test for all folds.

Among the different preprocessing methods implemented, `KNNImputer` was selected for imputing categorical data, as it better preserves the structure of the data by studying the correlation between instances. For the imputation of numerical data, the median was chosen, as it is a robust measure of central tendency and is less affected by outliers.

For the *credit-a* dataset, given that it had missing values, imputation and normalization were performed. Figure 1 shows the resulting dataset, reduced to two dimensions using t-SNE[2]. In the case of *pen-based*, as there were no missing values, only its attributes were normalized. Figure 2 shows the two-dimensional representation of the preprocessed data.

[1]https://scikit-learn.org/stable/

[2]t-Distributed Stochastic Neighbor Embedding (t-SNE) is a nonlinear dimensionality-reduction algorithm used to visualize high-dimensional data in two or three dimensions [10].

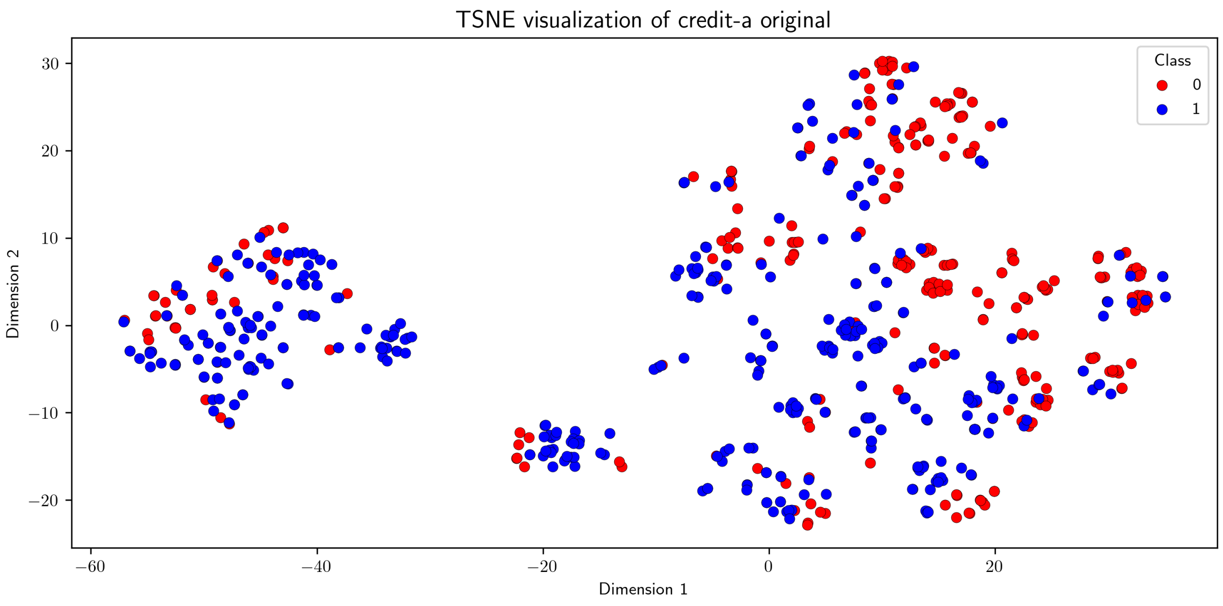

Figure 1: Two-dimensional visualization of the *credit-a* dataset.

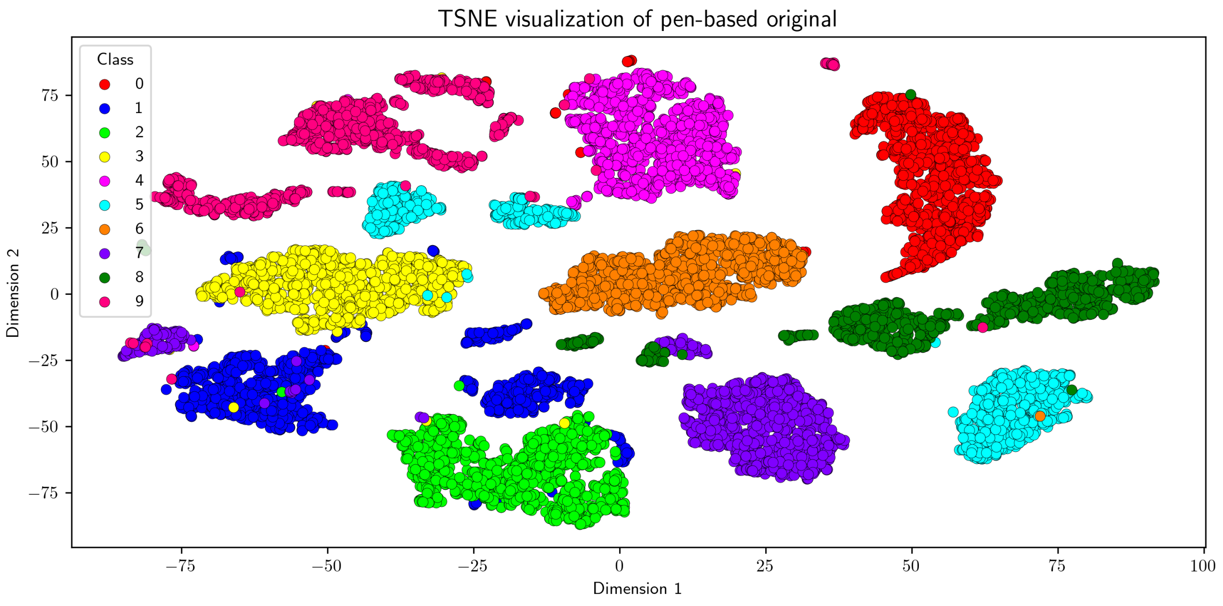

Figure 2: Two-dimensional visualization of the *pen-based* dataset.

III. Methods

IV. Results

The code for the project is available at: `https://github.com/mariagt02/IML_Labs_2025_MAI`.

All the times reported in this section were obtained using a computer running Ubuntu on an AMD Ryzen 9 7900X 12-core processor. Therefore, the particular results may vary when executed on different hardware or software environments. Nevertheless, the relative performance differences between models should remain consistent.

Model (top 5)	**Average Rank**
`euc_mp_3_ar`	22.75
`euc_bc_5_ar`	23.04
`euc_bc_7_dd`	23.19
`euc_bc_5_dd`	23.86
`euc_bc_5_dc`	23.99
Friedman Test Summary	
Statistic	339.26
p-value	1.27×10^{-36}
Reject H_0 at $\alpha = 0.1$?	**Yes**

Table 4: Results of the Friedman and Nemenyi statistical tests for different k-IBL hyperparameter configuration, in terms of accuracy. Average ranks correspond to the Nemenyi post-hoc analysis ($\alpha = 0.1$).

V. Discussion

VI. Conclusions

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

References

[1] UCI Machine Learning Repository. *Credit Approval Data Set*. Accessed: 2025-11-01. n.d. URL: `https://archive.ics.uci.edu/dataset/27/credit+approval`.

[2] R. Kuhn. *Analysis of Credit Approval Data*. Accessed: 2025-11-01. n.d. URL: `https://rstudio-pubs-static.s3.amazonaws.com/73039_9946de135c0a49daa7a0a9eda4a67a72.html`.

[3] UCI Machine Learning Repository. *Pen-Based Recognition of Handwritten Digits Data Set*. Accessed: 2025-11-01. n.d. URL: `https://archive.ics.uci.edu/dataset/81/pen+based+recognition+of+handwritten+digits`.

[4] OpenML. *Vowel Dataset*. `https://www.openml.org/search?type=data&sort=runs&id=307&status=active`. Accessed: 2025-11-02. n.d.

[5] Scikit-learn. *KNNImputer*. Accessed: 2025-11-01. n.d. URL: `https://scikit-learn.org/stable/modules/generated/sklearn.impute.KNNImputer.html`.

[6] Scikit-learn. *SimpleImputer*. Accessed: 2025-11-01. n.d. URL: `https://scikit-learn.org/stable/modules/generated/sklearn.impute.SimpleImputer.html`.

[7] Scikit-learn. *MinMaxScaler*. http://scikit-learn.org/sklearn.preprocessing.MinMaxScaler.html. Accessed: 2025-11-02. n.d.

[8] Scikit-learn. *OneHotEncoder*. Accessed: 2025-11-01. n.d. URL: `https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html`.

[9] Scikit-learn. *LabelEncoder*. Accessed: 2025-11-01. n.d. URL: `https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html`.

[10] Laurens van der Maaten and Geoffrey Hinton. "Visualizing data using t-SNE". In: *Journal of machine learning research* 9.Nov (2008), pp. 2579–2605.