Vrije Universiteit Amsterdam

Faculty of Science

Machine Learning Project Group 66

Machine Learning

Authors:

Dat Le

Noah Cladera

Beata Jahangiri

Jordi Plótnikov

Perry Boer

March 2023

1 Abstract

This study investigates the use of Machine Learning algorithms for the classification of food items based on their nutritional values, with the objective of accurately predicting food classes, evaluating model effectiveness, and understanding diverse results from different models. Using a labeled data set of food items and nutritional values, we implemented and evaluated the K-Nearest Neighbours algorithm in Python, employing data pre-processing, feature selection, different distance metrics [1], hyper-parameter tuning using grid search and validation techniques such as training-test split and .

This study focuses on the accuracy of the model as its primary evaluation metric. Results demonstrated varying levels of effectiveness across different models, emphasizing the importance of understanding the underlying assumptions and mechanisms of each ML algorithm. This work contributes to the development of a robust food classification system with potential applications in dietary management, health monitoring, and personalized nutrition planning.

2 Introduction

The increasing availability of nutritional data [2] offers new opportunities for the development of effective food classification systems, which can facilitate dietary management, health monitoring, and personalized nutrition planning. In this report, we focus on the application of machine learning algorithms, specifically K-Nearest Neighbors to classify food items based on their nutritional values. Initially, we undertake a comprehensive analysis, visualization, and pre-processing of the data set, aiming to enhance its utility and comprehensiveness for the subsequent implementation of the selected machine learning algorithms. This stage involves the exploration of the data structure, identification of correlations and patterns, and treatment of missing or inconsistent values.

Following the data preparation, we proceed with the implementation of KNN and algorithm, emphasizing their respective strengths and suitability for the task at hand. To maximize the performance of each model, we systematically tune their hyper-parameters through methods such as grid search and cross-validation. The process ensures the selection of the most appropriate parameters for each algorithm, resulting in more accurate and reliable predictions. Subsequently, we validate the models using appropriate performance such as accuracy.

In summary, this report presents a systematic procedure to develop a robust food classification system using KNN and algorithm. By adhering to a rigorous methodology that encompasses data analysis, visualization, pre-processing, model implementation, hyper-parameter tuning, and validation, we aim to provide valuable insights into the effectiveness and potential applications of these machine learning techniques in the realm of nutritional science.

3 Methodoloy

3.1 Data collection and Pre-Processing

The data collection and pre-processing are critical steps as they determine the quality of the methods used, which are brought to light in another section, for the intended task. The pre-processing steps involves cleaning the data, aspects such as handling missing values, adding labels and feature engineering were used. The task of the report is classifying food items based on their nutritional values. In this study, we collected a dataset of roughly 7000 instances of food items, each with their respective nutritional values, such as carbohydrates, proteins, sugars, water content, fibers, and kcal. The dataset was obtained from a publicly available online database.

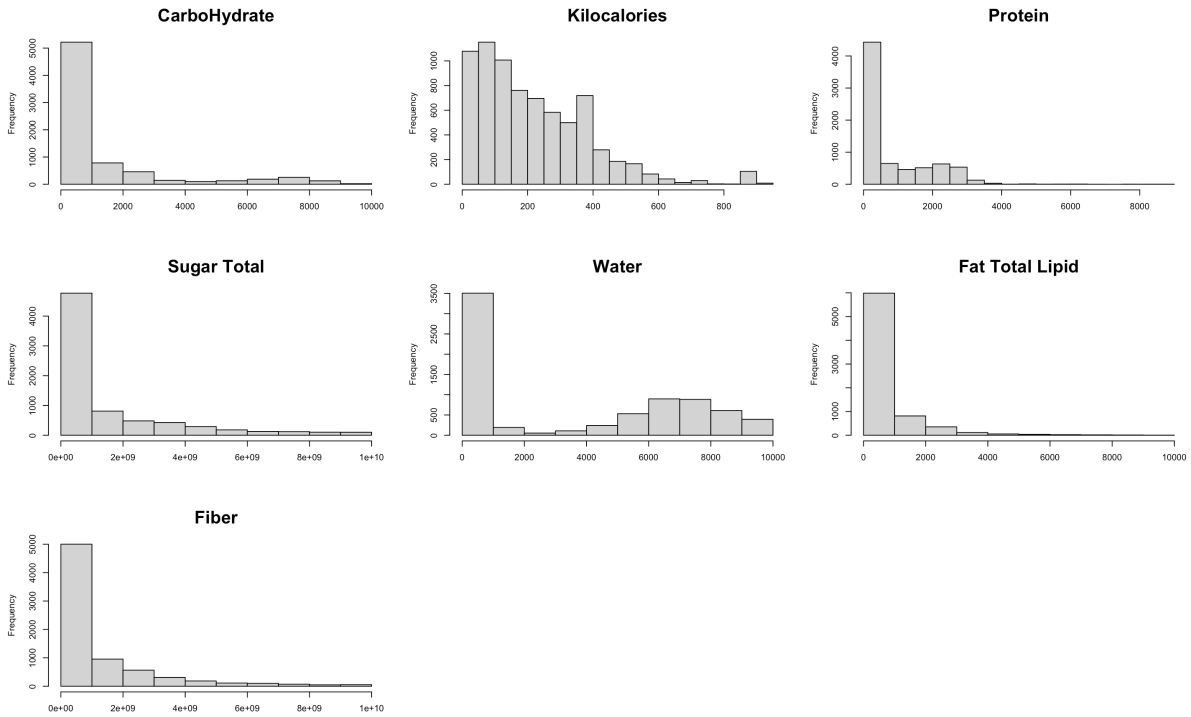

Figure 1: Histograms for important features

Before the dataset could be used for training machine learning models, we performed multiple pre-processing steps on the dataset. These steps include handling missing values, filtering non-relevant instances, handling instances with no category, feature selection [3] and normalization. Firstly, we removed instances with missing values for any of the nutritional attributes. Secondly, some instances in the dataset were products from fast food chains, such as McDonald's and KFC, which did not have clear nutritional information. We either removed these instances or assigned them a label to indicate that they were not relevant to our task. Thirdly, we handled instances with no category by either removing them or assigning them a label indicating that they could not be classified. Feature selection was performed to select a subset of the nutritional attributes that were most relevant for our classification task. This was achieved by analyzing the correlations between different attributes and selecting the ones that had the highest predictive power for the food class labels. Finally, we normalized the data to ensure that all nutritional attributes had a similar scale and range of values.

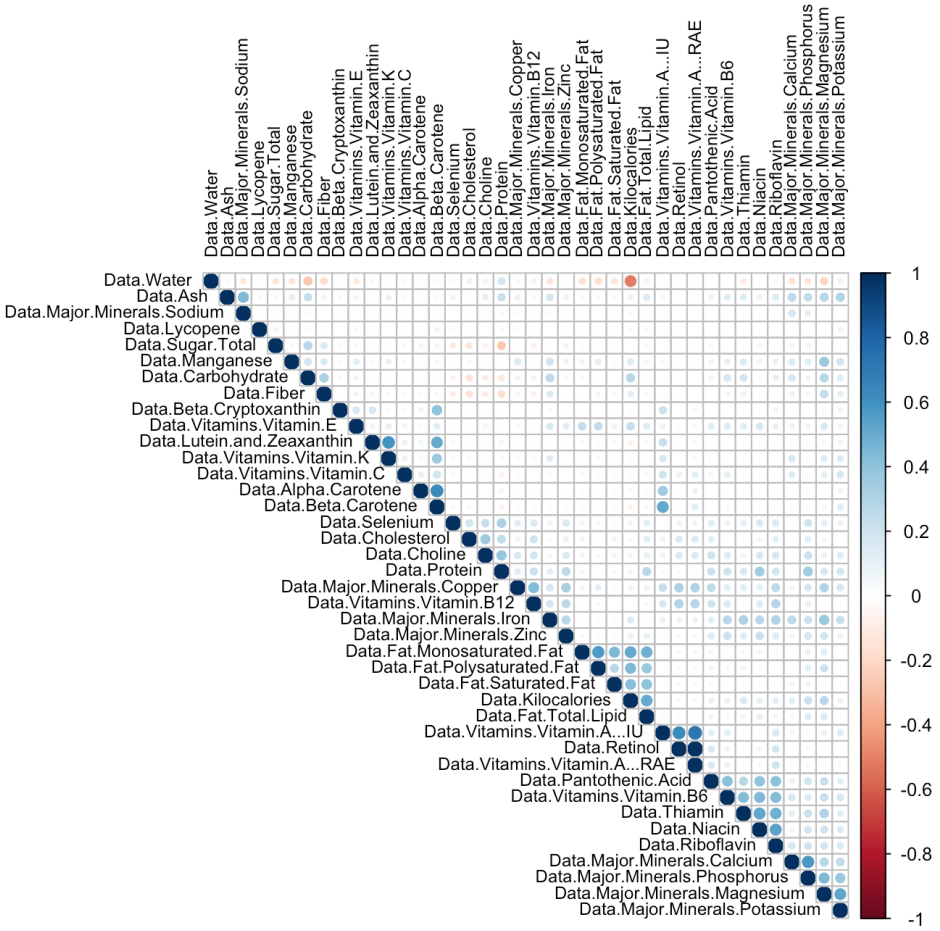

Figure 2: Correlation plot for all features used for feature engineering

The resulting preprocessed dataset is divided into training and testing sets for the KNN model. The KNN model is implemented and evaluated using the preprocessed data, where feature selection and validation techniques are used, as well as hyperparameter tuning. In the next section we will describe the machine learning model K nearest neighbor and the evaluations methods used for performance.

3.2 k-Nearest Neighbour

The k-Nearest Neighbors (KNN) algorithm is a popular machine learning algorithm used for classification and regression. In our study, we used the KNN algorithm for the classification of food items based on their nutritional values. It is a relatively simple algorithm that works by finding similarity between the new data and labelled data in the training set. Or, in other words, finding the K nearest neighbors of unknown data points based on the distance in the feature space, where the feature space is defined as a domain

where the features of the dataset are placed. The dimensionality of the feature space is defined by the number of features the dataset has and the possible range or values each feature could have. The preprocessed dataset is divided into training and testing sets, respectively in a ratio of 80 percent training and 20 percent testing. The KNN has an important hyperparameter that affects the performance of the model. In this case the numerical choice of K represent the number of neighbours to consider for a new label. An optimal value of K control the bias and variance of the model the best [4]. Hence, a structural method of finding the optimal value of K is of great important, because it is crucial for achieving a good accuracy and thus performance of the algorithm.

One insight on the K nearest neighbor algorithm is that it is a computationally efficient and flexible algorithm. That means the algorithm is capable of handling large datasets with high dimensionality without using of a significant amount of memory. Furthermore, K nearest neighbor is a non-parametric algorithm which indicates that the algorithm does not make assumptions about the distribution of the data. Or in other words the algorithm could learn from the given data without information on the distribution or parameters.

To classify a new instance, in this case a food item, the KNN algorithm computes the distance between an arbitrary x, where x belongs to a dataset of

$$X = [x_1, x_2, x_n]$$

and instances in the model space and selects the K closest instances and labels them accordingly the different food classes. The different distance metrics used are the Euclidean distance, Manhattan distance [5] and Chebyshev distance which are defined in the following subsections.

3.3 Experiment

To investigate the effectiveness of Machine Learning models for classification of food products, we implemented the kNN algorithm from scratch using different software sources to better understand the mathematics, mechanics and calculations behind the algorithm.

For the structure of the experiment, we split our dataset into a training set and a testing set with 80% of the dataset reserved for training and the remaining 20% of the dataset for testing. To ensure repeatability, we assigned random state values to our test runs. In our testing, we used the values of 10, 24, 42, 57, 87 as seeds for the random states. As for tuning the k-value, we performed a grid search to find the optimal k-value

[6] in terms of accuracy. Furthermore, we investigated the impact of different distance metrics [7] on the accuracy of the model. For the distance metrics, we chose the Euclidean distance, the Manhattan distance and the Chebyshev distance.

The Euclidean distance measures the distance between two points in a Euclidean space. It is the length of the shortest line segment that connects the two points. It is calculated by taking the square root of the sum of the squared differences between each corresponding coordinates. The mathematical formula for the Euclidean distance in n-dimensional space is:

$$d(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + ... + (x_n - y_n)^2}$$

The Manhattan distance is measured as the distance between two points measured along the axis at right angles. It is calculated by adding the absolute differences between each corresponding coordinates along each dimension. The mathematical formula for the Manhattan distance in n-dimensional space is:

$$d(x, y) = |x_1 - y_1| + |x_2 - y_2| + ... + |x_n - y_n|$$

The Chebyshev distance measures the distance between two points based on the maximum difference between their coordinates along any dimension. It is calculated by taking the absolute difference between the coordinates of two points along each dimension, and selecting the maximum of the distances as the distance between the two points. The mathematical formula for the Chebyshev distance in n-dimensional space is:

$$d(x, y) = max(|x_1 - y_1|, |x_2 - y_2|, ..., |x_n - y_n|)$$

4 Results and Validation

4.1 Results

For the experiment, we ran a grid search over 3 different distance metrics to find the optimal k-value that produced the highest accuracy score. As shown in table 1, it is observable that the Manhattan distance metric produced the highest accuracy scores over all k-values from 1 to 10, with k = 1 being the optimal k-value with an accuracy score of 58.05%.

Random state = 10			
k-value	Manhattan	Euclidean	Chebyshev
1	0.5805	0.5134	0.4657
2	0.5343	0.4762	0.4374
3	0.5306	0.4694	0.4344
4	0.5164	0.4598	0.4225
5	0.5276	0.4672	0.4218
6	0.5179	0.4523	0.418
7	0.5186	0.4508	0.4121
8	0.5134	0.4493	0.4151
9	0.5104	0.4419	0.4069
10	0.4978	0.4396	0.4009

Table 1: Accuracy scores for different distance metrics, random state = 10

4.2 Validation

To ensure that our results were valid, we ran the same experiment with 4 other random states with seed values of 24, 42, 57 and 87 to validate our findings. The full table of results can be found in the appendix. The values in table 2 show the accuracy scores of the k-value k = 1 for each distance metric for random states 10, 24, 42, 57, 87. To validate our findings, we will perform a significance test between each corresponding 5 values of each random state with each other to see if there is a significant difference between them.

Random state	10	24	42	57	87
Manhattan	0.5805	0.5805	0.5820	0.5745	0.5864
Euclidean	0.5134	0.5075	0.5201	0.5171	0.5186
Chebyshev	0.4657	0.4531	0.4665	0.4739	0.4672

Table 2: Accuracy scores for each distance metric and random state, k = 1

4.2.1 Analysis of Variance

To do this, the ANOVA or Analysis of Variance [8] test can be performed to test whether there are significant differences between the 5 values. This test can be performed by calculating the mean accuracy score for each of the 5 random states. Then we calculate the total sum of squares (SS) by summing the squared deviations of each group mean

from the overall mean, weighted by the sample size of each group. After that we calculate the between-group sum of squares (SSB) by summing the squared deviations of each group mean from the overall mean, weighted by the sample size of each group. Next, we calculate the within-group sum of squares (SSW) by summing the squared deviations of each individual score from its group mean, weighted by the sample size of each group. Then we calculate the degrees of freedom (df) for each of the three sum of squares values:

$$df(SS) = N - 1, df(SSB) = k - 1, df(SSW) = N - k$$

- with N is the total sample size and k is the number of groups. After that we calculate the F statistic by dividing the between-group sum of squares by the within-group sum of squares, and multiplying by the appropriate degrees of freedom.

$$F = (SSB/df(SSB))/(SSW/df(SSW))$$

Lastly we determine the p-value associated with the F statistic by comparing it to the appropriate F distribution with degrees of freedom df(SSB) and df(SSW) and compare the resulting p-value with our chosen level of significance of p = 0.05

4.2.2 Validation Manhattan distance results

Null hypothesis: There is no significant difference in the mean value of the 5 values. Alternative hypothesis: There is a significant difference in the mean value of at least two of the values. Calculate the mean accuracy score:

$$mean = (0.5745 + 0.5864 + 0.5805 + 0.5805 + 0.5820)/5 = 0.5808$$

Calculate the total sum of squares:

$$SS = (0.5745 - 0.5808)^2 + (0.5864 - 0.5808)^2 + (0.5805 - 0.5808)^2 +$$

$$(0.5805 - 0.5808)^2 + (0.5820 - 0.5808)^2 = 0.000112$$

Calculate the between-group sum of squares (SSB):

$$SSB = ((0.5745 - 0.5808)^2 + (0.5864 - 0.5808)^2 + (0.5805 - 0.5808)^2 +$$

$$(0.5805 - 0.5808)^2 + (0.5820 - 0.5808)^2)/4 = 0.000019$$

Calculate the within-group sum of squares (SSW):

$$SSW = (0.5745 - 0.5805)^2 + (0.5864 - 0.5805)^2 + (0.5805 - 0.5805)^2 +$$

$$(0.5805 - 0.5805)^2 + (0.5820 - 0.5805)^2 = 0.000091$$

Calculate the degrees of freedom (df):

$$df(SS) = N - 1 = 4, df(SSB) = k - 1 = 4 - 1 = 3, df(SSW) = N - k = 5 - 4 = 1$$

Calculate the F statistic:

$$F = (SSB/df(SSB))/(SSW/df(SSW)) = (0.000019/3)/(0.000091/1) = 0.070$$

Converting the F statistic F = 0.070 with a DF-numerator of 3 and DF-denominator of 1, we get a p-value of p = 0.972, which is significantly larger than our chosen level of significance of p = 0.05, thus we fail to reject the null hypothesis and conclude that there are no significant difference in the mean value of the 5 values.

4.2.3 Validation Euclidean distance results

Null hypothesis: There is no significant difference in the mean value of the 5 values. Alternative hypothesis: There is a significant difference in the mean value of at least two of the values. Calculate the mean accuracy score:

$$mean = (0.5201 + 0.5075 + 0.5134 + 0.5186 + 0.5171)/5 = 0.5153$$

Calculate the total sum of squares:

$$SS = (0.5201 - 0.5153)^2 + (0.5075 - 0.5153)^2 + (0.5134 - 0.5153)^2 +$$

$$((0.5186 - 0.5153)^2 + (0.5171 - 0.5153)^2 = 0.000350$$

Calculate the between-group sum of squares (SSB):

$$SSB = ((0.5201 - 0.5153)^2 + (0.5075 - 0.5153)^2 + (0.5134 - 0.5153)^2 +$$

$$(0.5186 - 0.5153)^2 + (0.5171 - 0.5153)^2)/4 = 0.000039$$

Calculate the within-group sum of squares (SSW):

$$SSW = (0.5201 - 0.5182)^2 + (0.5075 - 0.5078)^2 + (0.5134 - 0.5128)^2 +$$

$$(0.5186 - 0.5179)^2 + (0.5171 - 0.5179)^2 = 0.000312$$

Calculate the degrees of freedom (df):

$$df(SS) = N - 1 = 4, df(SSB) = k - 1 = 4 - 1 = 3, anddf(SSW) = N - k = 5 - 4 = 1$$

Calculate the F statistic:

$$F = (SSB/df(SSB))/(SSW/df(SSW)) = 0.000039/0.000312 = 0.125$$

Converting the F statistic F = 0.125 with a DF-numerator of 3 and DF-denominator of 1, we get a p-value of p = 0.934, which is significantly larger than our chosen level of significance of p = 0.05, thus we fail to reject the null hypothesis and conclude that there are no significant difference in the mean value of the 5 values.

4.2.4 Validation Chebyshev distance results

Null hypothesis: There is no significant difference in the mean value of the 5 values. Alternative hypothesis: There is a significant difference in the mean value of at least two of the values. Calculate the mean accuracy score:

$$mean = (0.4739 + 0.4672 + 0.4657 + 0.4531 + 0.4665)/5 = 0.4652$$

Calculate the total sum of squares:

$$SS = (0.4739 - 0.4652)^2 + (0.4672 - 0.4652)^2 + (0.4657 - 0.4652)^2 +$$

$$(0.4531 - 0.4652)^2 + (0.4665 - 0.4652)^2 = 0.000228$$

Calculate the between-group sum of squares (SSB):

$$SSB = ((0.4739 - 0.4652)^2 + (0.4672 - 0.4652)^2 + (0.4657 - 0.4652)^2 +$$

$$(0.4531 - 0.4652)^2 + (0.4665 - 0.4652)^2)/4 = 0.000057$$

Calculate the within-group sum of squares (SSW):

$$SSW = (0.4739 - 0.4652)^2 + (0.4672 - 0.4652)^2 + (0.4657 - 0.4652)^2 +$$

$$(0.4531 - 0.4652)^2 + (0.4665 - 0.4652)^2 = 0.000228$$

Calculate the degrees of freedom (df):

$$df(SS) = N - 1 = 4, df(SSB) = k - 1 = 4 - 1 = 3, df(SSW) = N - k = 5 - 4 = 1$$

Calculate the F statistic:

$$F = (SSB/df(SSB))/(SSW/df(SSW)) = (0.000057/3)/(0.000228/1) = 0.2632$$

Converting the F statistic F = 0.070 with a DF-numerator of 3 and DF-denominator of 1, we get a p-value of p = 0.854, which is significantly larger than our chosen level of

significance of p = 0.05, thus we fail to reject the null hypothesis and conclude that there are no significant difference in the mean value of the 5 values.

With our significance test on all 3 distance metrics, we can conclude that there is no difference in the mean value of each value of the 5 random states, therefore, we can conclude that the results obtained from the experiment are valid.

5 Discussion

This report outlines a systematic approach for developing a food classification system using the k-Nearest Neighbors algorithm. We collected data, pre-processed it, implemented the model, tuned hyper-parameters, and validated our results. Our pre-processing steps included handling missing values, filtering non-relevant instances, feature selection, and normalization. We used the preprocessed dataset to train and test the KNN model.

The KNN algorithm was efficient and flexible, handling large datasets with high dimensionality. The optimal value of k had a significant impact on the model's performance, and hyper-parameter tuning was necessary for good accuracy.

Our experiment showed promising potential for machine learning in nutritional science, particularly in food classification. Further research and development can lead to improved tools for nutritional analysis [9] and better health outcomes.

To improve the accuracy in the future, some changes can be made in this study. In the data pre-processing phase, the data can be visualised to find frequent occurring nutritional data. Consequently, weighted features [10] can be implemented to give certain features with a high occurrence more weight than features that do not occur often. This allows the model to know which features are more important.

References

[1] A. Singh, A. Yadav, and A. Rana, "K-means with three different distance metrics," *International Journal of Computer Applications*, vol. 67, no. 10, 2013.

[2] J. Deeks, M.-F. Verreault, and W. Cheung, "Canadian nutrient file (cnf): Update on canadian food composition activities," *Journal of Food Composition and Analysis*, vol. 64, pp. 43–47, 2017, The 39th National Nutrient Databank Conference: The Future of Food and Nutrient Databases: Invention, Innovation, and Inspira-

tion, ISSN: 0889-1575. DOI: https://doi.org/10.1016/j.jfca.2017.04.009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S088915751730100X.

[3] S. Li, E. J. Harner, and D. A. Adjeroh, "Random knn feature selection-a fast and stable alternative to random forests," *BMC bioinformatics*, vol. 12, pp. 1–11, 2011.

[4] S. Zhang, X. Li, M. Zong, X. Zhu, and R. Wang, "Efficient knn classification with different numbers of nearest neighbors," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 5, pp. 1774–1785, 2018. DOI: 10.1109/TNNLS.2017.2673241.

[5] X. Gao and G. Li, "A knn model based on manhattan distance to identify the snare proteins," *Ieee Access*, vol. 8, pp. 112 922–112 931, 2020.

[6] R. Ghawi and J. Pfeffer, "Efficient hyperparameter tuning with grid search for text categorization using knn approach with bm25 similarity," *Open Computer Science*, vol. 9, no. 1, pp. 160–180, 2019.

[7] K. Chomboon, P. Chujai, P. Teerarassamee, K. Kerdprasop, and N. Kerdprasop, "An empirical study of distance metrics for k-nearest neighbor algorithm," in *Proceedings of the 3rd international conference on industrial application engineering*, vol. 2, 2015.

[8] H. Scheffe, *The analysis of variance*. John Wiley & Sons, 1999, vol. 72.

[9] D. Kirk, E. Kok, M. Tufano, B. Tekinerdogan, E. J. M. Feskens, and G. Camps, "Machine Learning in Nutrition Research," *Advances in Nutrition*, vol. 13, no. 6, pp. 2573–2589, Sep. 2022, ISSN: 2161-8313. DOI: 10.1093/advances/nmac103. eprint: https://academic.oup.com/advances/article-pdf/13/6/2573/48352551/nmac103.pdf. [Online]. Available: https://doi.org/10.1093/advances/nmac103.

[10] J. Huang, Y. Wei, J. Yi, and M. Liu, "An improved knn based on class contribution and feature weighting," in *2018 10th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA)*, 2018, pp. 313–316. DOI: 10.1109/ICMTMA.2018.00083.

6 Appendix

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

# Read the data from the CSV file
df = pd.read_csv('C:/VU - Machine Learning/MLProject66/food_cleaned_v1.csv', delimiter=';')

# Extract the features and labels from the DataFrame
X = df.iloc[:, 1:].values

# Create a label encoder and fit/transform the 'y' array
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(df.iloc[:, 0].values)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Define the range of k values to consider
k_values = range(1, 21)

# Initialize an array to store the accuracy, precision, and F1-score scores for each k value
accuracy_scores = np.zeros(len(k_values))


# Loop over each k value and compute its accuracy score
for i, k in enumerate(k_values):
    # Initialize an array to store the predicted labels for the test data
    y_pred = np.zeros(y_test.shape[0])

    # Loop through each test sample and predict its label
    for j in range(X_test.shape[0]):
        # Compute the Euclidean distance between the test sample and each training sample
        #dists = np.sqrt(np.sum((X_train - X_test[j])**2, axis=1))

        # Compute the Manhattan distance between the test sample and each training sample
        #dists = np.sum(np.abs(X_train - X_test[j]), axis=1)

        # Comput the Chebyshev distance between the test sample and each training sample
        dists = np.max(np.abs(X_train - X_test[j]), axis=1)

        # Find the indices of the k closest training samples
        idx = np.argsort(dists)[:k]

        # Get the corresponding labels for the k closest training samples
        closest_y = y_train[idx]

        # Assign the predicted label for the test sample as the mode (most common value) of the corresponding labels
        y_pred[j] = np.argmax(np.bincount(closest_y))

    accuracy_scores[i] = np.mean(y_pred == y_test)
    print(f"{accuracy_scores[i]:.4f}")

# Find the k value with the highest accuracy score
best_k = k_values[np.argmax(accuracy_scores)]
print(f"\nBest k value: {best_k}, accuracy={accuracy_scores.max():.4f}")
```

Figure 3: Code for k-Nearest Neighbour algorithm with accuracy test

Table 3: Accuracy scores for different distance metrics, random state = 10

	Random State = 10		
	Manhattan	Euclidean	Chebyshev
1	0.5805	0.5134	0.4657
2	0.5343	0.4762	0.4374
3	0.5306	0.4694	0.4344
4	0.5164	0.4598	0.4225
5	0.5276	0.4672	0.4218
6	0.5179	0.4523	0.418
7	0.5186	0.4508	0.4121
8	0.5134	0.4493	0.4151
9	0.5104	0.4419	0.4069
10	0.4978	0.4396	0.4009
11	0.4918	0.4359	0.3994
12	0.4873	0.4314	0.3957
13	0.4814	0.4314	0.3927
14	0.4784	0.4262	0.3845
15	0.4739	0.4218	0.3793
16	0.4747	0.4143	0.38
17	0.468	0.4128	0.3778
18	0.4635	0.4113	0.3703
19	0.4627	0.4121	0.3681
20	0.4583	0.4031	0.3636

Table 4: Accuracy scores for different distance metrics, random state = 24

	Random State = 24		
	Manhattan	Euclidean	Chebyshev
1	0.5805	0.5075	0.4531
2	0.5179	0.4665	0.4151
3	0.5365	0.4799	0.4352
4	0.5253	0.4739	0.4322
5	0.5268	0.4724	0.4329
6	0.5231	0.4665	0.4262
7	0.5216	0.4672	0.4203
8	0.5082	0.4598	0.4143
9	0.5007	0.4523	0.4136
10	0.494	0.4471	0.4061
11	0.4866	0.4426	0.4024
12	0.4836	0.4374	0.3905
13	0.4851	0.4307	0.3838
14	0.4799	0.4307	0.386
15	0.4784	0.424	0.3838
16	0.468	0.418	0.3763
17	0.4657	0.4151	0.3711
18	0.4642	0.4151	0.3711
19	0.465	0.4098	0.3674
20	0.4613	0.4113	0.3651

Table 5: Accuracy scores for different distance metrics, random state = 42

	Random State = 42		
	Manhattan	Euclidean	Chebyshev
1	0.582	0.5201	0.4665
2	0.532	0.4829	0.4352
3	0.5395	0.4844	0.4352
4	0.538	0.4829	0.4277
5	0.5283	0.4694	0.4314
6	0.5209	0.462	0.421
7	0.5149	0.4516	0.4173
8	0.5149	0.4486	0.4151
9	0.5022	0.4426	0.4098
10	0.5067	0.4411	0.4046
11	0.503	0.4314	0.392
12	0.4955	0.4344	0.3964
13	0.4925	0.4322	0.3905
14	0.4888	0.4314	0.3845
15	0.4888	0.4255	0.3875
16	0.4806	0.4218	0.3823
17	0.4844	0.421	0.3778
18	0.4791	0.4188	0.3718
19	0.4754	0.4203	0.3726
20	0.4732	0.4173	0.3681

Table 6: Accuracy scores for different distance metrics, random state = 57

	Random State = 57		
	Manhattan	Euclidean	Chebyshev
1	0.5745	0.5171	0.4739
2	0.5335	0.4732	0.4247
3	0.5402	0.4762	0.4374
4	0.5373	0.4694	0.424
5	0.5253	0.4575	0.4158
6	0.5149	0.456	0.4128
7	0.5164	0.4575	0.4173
8	0.5089	0.4523	0.4121
9	0.503	0.4463	0.4046
10	0.4925	0.4382	0.3972
11	0.4978	0.4382	0.3927
12	0.4955	0.4314	0.3934
13	0.4911	0.4255	0.3934
14	0.4836	0.427	0.3882
15	0.4769	0.4218	0.3882
16	0.4724	0.4188	0.3823
17	0.4702	0.4195	0.3726
18	0.4724	0.4128	0.3741
19	0.4687	0.4098	0.3659
20	0.4642	0.4083	0.3644

Table 7: Accuracy scores for different distance metrics, random state = 87

	Random State = 87		
	Manhattan	Euclidean	Chebyshev
1	0.5864	0.5186	0.4672
2	0.5261	0.465	0.427
3	0.5313	0.4769	0.4352
4	0.5231	0.4739	0.424
5	0.5209	0.4635	0.4136
6	0.5164	0.4486	0.4098
7	0.5112	0.456	0.4113
8	0.5119	0.4441	0.4001
9	0.5075	0.4463	0.4001
10	0.5037	0.4478	0.4031
11	0.503	0.4426	0.3942
12	0.5022	0.4434	0.3927
13	0.4963	0.4463	0.392
14	0.4903	0.4374	0.3882
15	0.4829	0.4359	0.383
16	0.4769	0.4329	0.3793
17	0.4732	0.4285	0.383
18	0.468	0.4262	0.383
19	0.465	0.421	0.3763
20	0.462	0.4203	0.3748