Implementation of Classification using K-Nearest Neighbors (KNN) in Python

Ahmad Farhan AlShammari
Department of Computer and Information Systems
College of Business Studies, PAAET
Kuwait

ABSTRACT
The goal of this research is to develop a classification program using K-Nearest Neighbors (KNN) method in Python. Classification helps to predict the categories of data by comparing the features of test and input data. The distances between the test and input data are measured and sorted to find the (k) nearest neighbors. Then, the predicted category of data is determined by the most common vote among the nearest neighbors.

The basic steps of classification using k-nearest neighbors are explained: preparing observed data, preparing test data, computing distances, sorting distances, computing neighbors, performing majority voting, computing predictions, computing confusion matrix, and computing model accuracy.

The developed program was tested on an experimental dataset. The program successfully performed the basic steps of classification using k-nearest neighbors and provided the required results.

Keywords
Artificial Intelligence, Machine Learning, Classification, K-Nearest Neighbors, KNN, Euclidean Distance, Confusion Matrix, Python, Programming.

1. INTRODUCTION
In recent years, machine learning has played a major role in the development of computer systems. Machine learning (ML) is a branch of Artificial Intelligence (AI) which is focused on the study of computer algorithms to improve the performance and efficiency of computer programs [1-14].

Classification is one of the important applications of machine learning. It is sharing the knowledge between the related fields: machine learning, programming, data science, mathematics, statistics, and numerical methods [15-19].

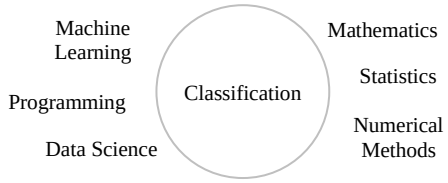

Fig 1: Field of Classification

In this paper, classification is applied using k-nearest neighbors to predict the categories of data by comparing the features of test and input data. Classification has a wide range of applications in different fields such as industry, business, education, marketing, advertising, medicine, public health, agriculture, environment, climate change, etc.

2. LITERATURE REVIEW
The review of literature revealed the major contributions in the field of classification using k-nearest neighbors [20-27].

In general, algorithms in machine learning are divided into three main types: supervised, unsupervised, and reinforcement.

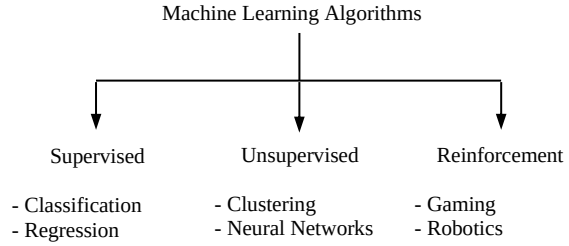

Fig 2: Types of Machine Learning Algorithms

Classification is a supervised learning algorithm, in which the data is labeled and the algorithm can make predictions based on the features of data.

Classification is applied using k-nearest neighbors. It helps to find the nearest neighbors and predict the category of data by selecting the most common vote among the nearest neighbors.

K-nearest neighbors was first developed in 1951 by Evelyn Fix and Joseph Hodges [28]. Then, it was expanded by Thomas Cover and Peter Hart in 1967 [29].

The fundamental concepts of classification using k-nearest neighbors are explained in the following section.

Classification:
Classification is an important algorithm in machine learning. It helps to predict the categories (or classes) of data by comparing the features of test and input data.

The concept of classification is illustrated in the following diagram:

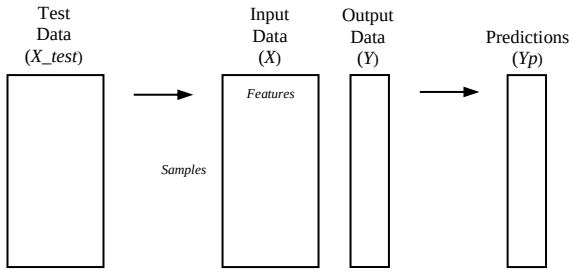

Fig 3: Explanation of Classification

K-Nearest Neighbors:

K-Nearest Neighbors (KNN) is a classification method used to predict the categories of data. The distances between the test and input data are measured and sorted to find the (*k*) nearest neighbors. Then, the majority voting is performed to determine the category of data by selecting the most common vote among the nearest neighbors.

The concept of k-nearest neighbors is illustrated in the following diagram:

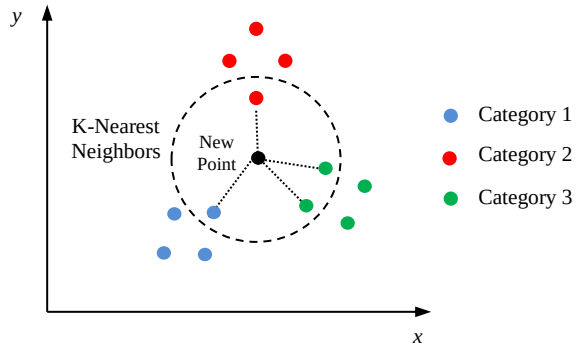

Fig 4: Explanation of K-Nearest Neighbors

Measuring Distance:

There are different methods used to measure the distance between two points. For example, Euclidean distance, Manhattan distance, Minkowski distance, and Chebyshev distance.

The different methods of measuring distance are explained in the following section.

Euclidean Distance:

The Euclidean distance is the direct distance between the two points.

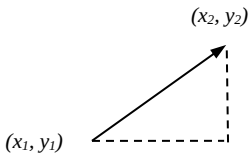

Fig 5: Explanation of Euclidean Distance

It is computed by the following formula:

$$\text{Distance} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \qquad (1)$$

Manhattan Distance:

The Manhattan distance is the sum of the horizontal and vertical distances between the two points.

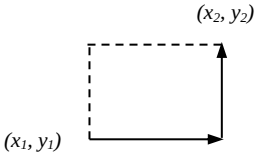

Fig 6: Explanation of Manhattan Distance

It is computed by the following formula:

$$\text{Distance} = |x_1 - x_2| + |y_1 - y_2| \qquad (2)$$

Minkowski Distance:

The Minkowski distance is a generalization of both the Euclidean and Manhattan distances.

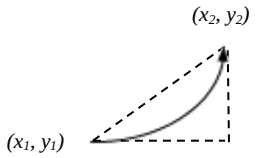

Fig 7: Explanation of Minkowski Distance

It is computed by the following formula:

$$\text{Distance} = \left(|x_1 - x_2|^p + |y_1 - y_2|^p \right)^{\frac{1}{p}} \qquad (3)$$

Chebyshev Distance:

The Chebyshev distance is the maximum of horizontal and vertical distances between the two points.

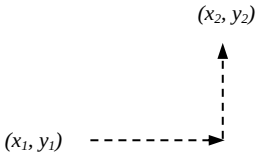

Fig 8: Explanation of Chebyshev Distance

It is computed by the following formula:

$$\text{Distance} = \max\left(|x_1 - x_2| , |y_1 - y_2| \right) \qquad (4)$$

Note: In this research, the Euclidean distance is applied.

The steps of k-nearest neighbors' method are explained in the following algorithm:

Algorithm 1: K-Nearest Neighbors Method

```
# Observed Data
X = [ ... ]
Y = [ ... ]
# Test Data
X_test = [ ... ]
# Predictions
predictions = []
for point in X_test do
    for i = 1 to n do
        distances  = compute_distance(point, X [i])
        distances = sorted(distances)
        neighbors = compute_neighbors(distances, k, Y)
        max_vote  = majority_voting(neighbors)
        prediction = max_vote
    end for
    predictions.add(prediction)
end for
```

Confusion Matrix:

Confusion matrix is a statistical tool used to summarize the results of the classification model. It is represented as shown in the following diagram:

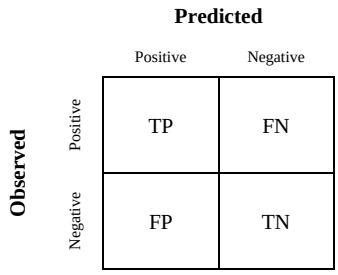

Fig 9: Representation of Confusion Matrix

Where: TP: is the number of true positives.
TN: is the number of true negatives.
FP: is the number of false positives.
FN: is the number of false negatives.

Model Accuracy:

The accuracy of the classification model is defined as the number of correct predictions divided by the total number of predictions. It is computed by the following formula:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \qquad (5)$$

Classification System:

The classification system is explained in the following outline:

Input: Observed data (*X*, *Y*).
Output: Predicted data (*Yp*).
Processing: The observed and test data are prepared for processing. First, the distances between the test and input data are measured using the Euclidean distance. Then, the distances are sorted in ascending order to find the (*k*) nearest neighbors. After that, the predicted category of data is determined by selecting the most common vote among the nearest neighbors.

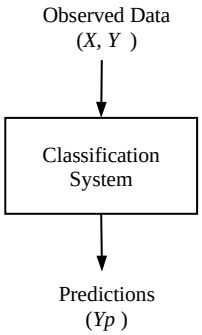

Fig 10: Diagram of Classification System

Python:

Python [30] is a general high-level programming language. It is simple, easy to learn, and powerful. It is the most popular programming language for the development of machine learning applications.

Python provides additional libraries for different purposes such as Numpy [31], Pandas [32], Matplotlib [33], NLTK [34], SciPy [35], and SK Learn [36].

In this research, the standard functions of Python are used without any additional library.

3. RESEARCH METHODOLOGY

The basic steps of classification using k-nearest neighbors are: (1) preparing observed data, (2) preparing test data, (3) computing distances, (4) sorting distances, (5) computing neighbors, (6) performing majority voting, (7) computing predictions, (8) computing confusion matrix, and (9) computing model accuracy.

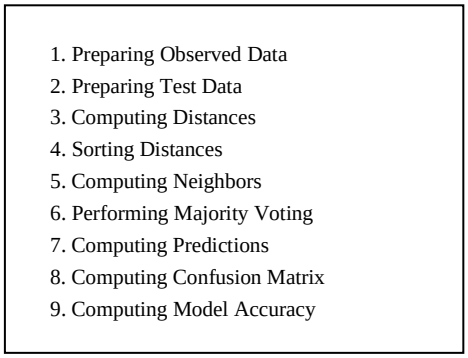

Fig 11: Steps of Classification

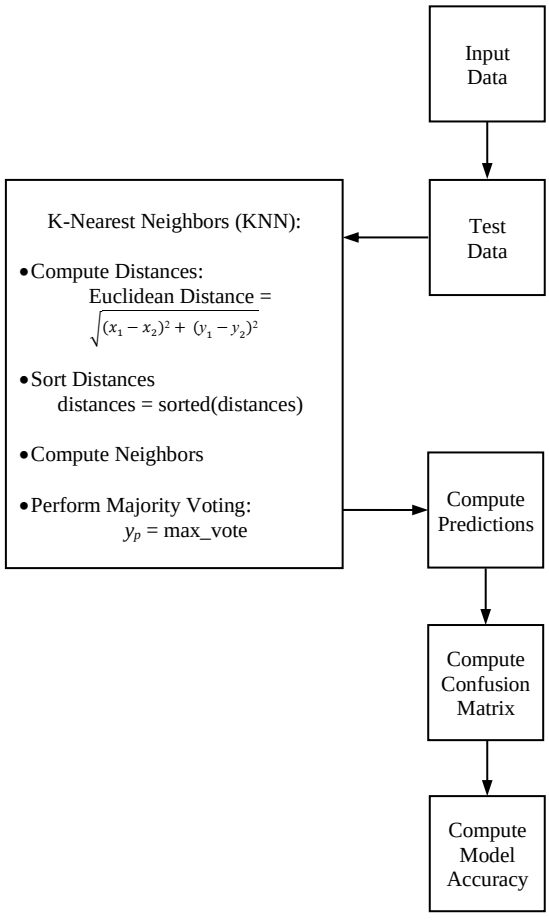

Fig 12: Flowchart of Classification

The steps of classification using k-nearest neighbors are explained in the following section.

1. Preparing Observed Data:

The observed data (*X*, *Y*) is obtained from the original source and converted into lists in the following form:

```
X = [[x0,0, x0,1, x0,2, ..., x0,n-1],
     [x1,0, x1,1, x1,2, ..., x1,n-1],
      ...
     [xm-1,0, xm-1,1, xm-1,2, ..., xm-1,n-1]]
Y = [y0, y1, y2, ..., ym-1]
```

2. Preparing Test Data:

The test data is obtained from the original source and converted into lists in the following form:

```
X_test = [[x0,0, x0,1, x0,2, ..., x0,n-1],
          [x1,0, x1,1, x1,2, ..., x1,n-1],
           ...
          [xt-1,0, xt-1,1, xt-1,2, ..., xt-1,n-1]]
Y_test = [y0, y1, y2, ..., yt-1]
```

3. Computing Distances:

The distances are computed between each point in the test data (*X_test*) and the points in the input data (*X*). It is done by the following code:

```
def compute_distances(X, point):
    distances = []
    for i in range(len(X)):
        distance = euclidean(X[i], point)
        distances.append([distance, i])
    return distances
```

The Euclidean distance between two points is computed by the following code:

```
def euclidean(x1, x2):
    sum = 0
    for i in range(len(x1)):
        sum += (x1[i] - x2[i])**2
    return math.sqrt(sum)
```

4. Sorting Distances:

The distances are sorted in ascending order by the distance value using the standard function sorted(). It is done by the following code:

```
distances = sorted(distances)
```

5. Computing Neighbors:

The neighbors are computed by selecting the first (*k*) items in the distances list. It is done by the following code:

```
def compute_neighbors(distances, k, Y):
    neighbors = []
    for i in range(k):
        index = distances[i][1]
        neighbors.append(Y[index])
    return neighbors
```

6. Performing Majority Voting:

The majority voting is performed to determine the category of data by selecting the most common vote among the nearest neighbors. It is done by the following code:

```
def majority_voting(neighbors):
    votes = []
    for cat in categories:
        votes.append([neighbors.count(cat),
                      cat])
    sorted_votes = sorted(votes, reverse=True)
    max_vote = sorted_votes[0][1]
    return max_vote
```

7. Computing Predictions:

The predictions (*Yp*) are computed for each point in the test data (*X*_test). It is done by the following code:

```
Yp = []
for i in range(m):
    distances = compute_distances(X, X_test[i])
    distances = sorted(distances)
    neighbors = compute_neighbors(distances,
                                  k, Y)
    max_vote = majority_voting(neighbors)
    Yp.append(max_vote)
```

8. Computing Confusion Matrix:

The confusion matrix of the classification model is computed by the following code:

```
ncat = len(categories)
CM = zeros(ncat, ncat)
for t in range(len(Y_test)):
    for i in range(ncat):
        for j in range(ncat):
            if (Y_test[t]==i and Yp[t]==j):
                CM[i][j] += 1
```

9. Computing Model Accuracy:

The accuracy of the classification model is computed by the following code:

```
def compute_accuracy(Y_test, Yp):
    sum = 0
    for i in range(len(Y_test)):
        if (Y_test[i] == Yp[i]):
            sum += 1
    return sum/len(Y_test)
```

4. RESULTS AND DISCUSSION

The developed program was tested on an experimental dataset from Kaggle [37]. The program performed the basic steps of classification using k-nearest neighbors and provided the required results. The program output is explained in the following section.

Observed Data:

The observed data (*X*, *Y*) is printed as shown in the following view:

```
        X                                Y
-----------------------------------------
0:      5.4     3       4.5     1.5     1
1:      4.6     3.2     1.4     0.2     0
2:      6.7     2.5     5.8     1.8     2
3:      4.9     3       1.4     0.2     0
4:      5       2.3     3.3     1       1
5:      6.7     3.3     5.7     2.5     2
6:      7.2     3.2     6       1.8     2
7:      5.8     2.6     4       1.2     1
8:      6.7     3.1     4.7     1.5     1
9:      5.1     3.8     1.6     0.2     0
...
```

Test Data:

The test data (*X_test*, *Y_test*) is printed as shown in the following view:

```
        X_test                                  Y_test
-----------------------------------------------
0:      6.4     2.8     5.6     2.1     2
1:      5.7     3.8     1.7     0.3     0
2:      7.4     2.8     6.1     1.9     2
3:      7.6     3       6.6     2.1     2
4:      7.3     2.9     6.3     1.8     2
5:      6       2.9     4.5     1.5     1
6:      6       2.7     5.1     1.6     1
7:      5.8     4       1.2     0.2     0
8:      5.4     3.9     1.7     0.4     0
9:      6.3     2.8     5.1     1.5     2
...
```

Predictions:

The predictions (*Yp*) are printed as shown in the following view:

```
        Y_test  Yp
-----------------------
0:      2       2
1:      0       0
2:      2       2
3:      2       2
4:      2       2
5:      1       1
6:      1       2
7:      0       0
8:      0       0
9:      2       2
...
```

Confusion Matrix:

The confusion matrix of the classification model is printed as shown in the following view:

```
Confusion Matrix:
-------------------------
0:      16      0       0
1:      0       4       1
2:      0       0       9
```

The confusion matrix is plotted using the matplotlib library. It is plotted as shown in the following chart:

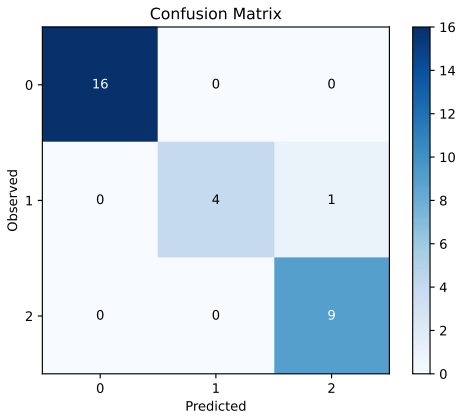

Fig 13: Confusion Matrix Plot

The plot shows that the predicted data (*Yp*) is strongly related to the test data (*Y_test*).

Model Accuracy:

The accuracy of the classification model is printed as shown in the following view:

```
Accuracy = 0.9666666666666667
```

The model accuracy is very high (about 97%) which indicates that the predicted data strongly fits with the observed data.

Testing Model Accuracy:

The model accuracy is tested for different (*k*) values (from 1 to 20). The result is plotted as shown in the following chart:

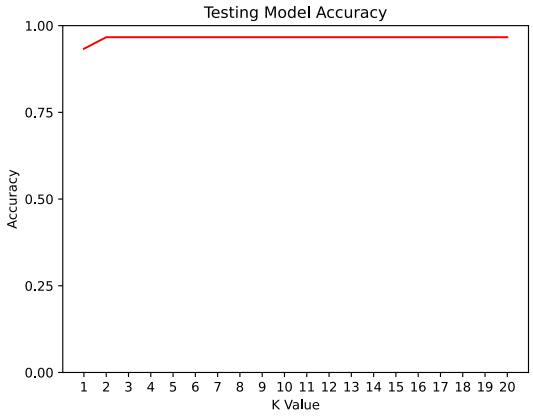

Fig 14: Model Accuracy vs K Value

The plot shows that the model accuracy is very high (about 97%) for different (*k*) values which indicates that the classification model is very accurate.

In summary, the program output shows that the program has successfully performed the basic steps of classification using k-nearest neighbors and provided the required results.

5. CONCLUSION

Machine learning is playing a major role in the development of computer systems. It helps to improve the performance and efficiency of computer programs.

Classification is one of the important applications in machine learning. It is applied using the k-nearest neighbors' method. The distances between the test and input data are measured and sorted to find the (*k*) nearest neighbors. Then, the majority voting is performed to determine the category of data by selecting the most common vote among the nearest neighbors.

In this research, the author developed a program to perform classification using k-nearest neighbors in Python. The developed program performed the basic steps of classification using k-nearest neighbors: preparing observed data, preparing test data, computing distances, sorting distances, computing neighbors, performing majority voting, computing predictions, computing confusion matrix, and computing model accuracy.

The program was tested on an experimental dataset and provided the required results: distances, neighbors, predictions, confusion matrix, and model accuracy.

In future work, more research is needed to improve and develop the current methods of classification using k-nearest neighbors. In addition, they should be more investigated on different fields, domains, and datasets.

6. REFERENCES

[1] Sammut, C., & Webb, G. I. (2011). "Encyclopedia of Machine Learning". Springer Science & Business Media.

[2] Jung, A. (2022). "Machine Learning: The Basics". Singapore: Springer.

[3] Kubat, M. (2021). "An Introduction to Machine Learning". Cham, Switzerland: Springer.

[4] Li, H. (2023). "Machine Learning Methods". Springer Nature.

[5] Mohammed, M., Khan, M. B., & Bashier, E. B. M. (2016). "Machine Learning: Algorithms and Applications". Crc Press.

[6] Dey, A. (2016). "Machine Learning Algorithms: A Review". International Journal of Computer Science and Information Technologies, 7 (3), 1174-1179.

[7] Bonaccorso, G. (2018). "Machine Learning Algorithms: Popular Algorithms for Data Science and Machine Learning". Packt Publishing.

[8] Jo, T. (2021). "Machine Learning Foundations: Supervised, Unsupervised, and Advanced Learning". Springer.

[9] Chopra, D., & Khurana, R. (2023). "Introduction to Machine Learning with Python". Bentham Science Publishers.

[10] Müller, A. C., & Guido, S. (2016). "Introduction to Machine Learning with Python: A Guide for Data Scientists". O'Reilly Media.

[11] Raschka, S. (2015). "Python Machine Learning". Packt Publishing.

[12] Forsyth, D. (2019). "Applied Machine Learning". Cham, Switzerland: Springer.

[13] Sarkar, D., Bali, R., & Sharma, T. (2018). "Practical Machine Learning with Python". Apress.

[14] Swamynathan, M. (2019). "Mastering Machine Learning with Python in Six Steps: A Practical Implementation Guide to Predictive Data Analytics using Python". Apress.

[15] Kong, Q., Siauw, T., & Bayen, A. (2020). "Python Programming and Numerical Methods: A Guide for Engineers and Scientists". Academic Press.

[16] Unpingco, J. (2022). "Python for Probability, Statistics, and Machine Learning". Cham, Switzerland: Springer.

[17] Brandt, S. (2014). "Data Analysis: Statistical and Computational Methods for Scientists and Engineers". Springer.

[18] VanderPlas, J. (2017). "Python Data Science Handbook: Essential Tools for Working with Data". O'Reilly Media.

[19] James, G., Witten, D., Hastie, T., Tibshirani, R., & Taylor, J. (2023). "An Introduction to Statistical Learning: With Applications in Python". Springer Nature.

[20] Aggarwal, C. C. (2020). "Data Classification: Algorithms and Applications". CRC Press

[21] Kotsiantis, S. B., Zaharakis, I., & Pintelas, P. (2007). "Supervised Machine Learning: A Review of Classification Techniques". Emerging Artificial Intelligence Applications in Computer Engineering, 160(1), 3-24.

[22] Neal, R.M. (2006). "Pattern Recognition and Machine Learning". Technometrics, 49, 366 - 366.

[23] Kotsiantis, S.B. (2007). "Supervised Machine Learning: A Review of Classification Techniques". Informatica. 31, 249-268.

[24] Sharma, A., Kaur, A., & Semwal, A. (2022). "Supervised and Unsupervised Prediction Application of Machine Learning". In 2022 International Conference on Cyber Resilience (ICCR), (pp. 1-5). IEEE.

[25] Bao, W. (2016). "Introduction to Machine Learning: K-Nearest Neighbors". Annals of Translational Medicine, 4 (11), 218.

[26] Matloff, N. (2017). "Statistical Regression and Classification: from Linear Models to Machine Learning". Chapman and Hall/CRC.

[27] Soofi, A. A., & Awan, A. (2017). "Classification Techniques in Machine Learning: Applications and Issues". Journal of Basic & Applied Sciences, 13(1), 459-465.

[28] Fix, E., & Hodges, J. L. (1951). "Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties". USAF School of Aviation Medicine, Randolph Field, Texas.

[29] Cover, Thomas M.; Hart, Peter E. (1967). "Nearest Neighbor Pattern Classification". IEEE Transactions on Information Theory. 13(1), 21–27.

[30] Python: https://www.python.org

[31] Numpy: https://www.numpy.org

[32] Pandas: https:// pandas.pydata.org

[33] Matplotlib: https://www. matplotlib.org

[34] NLTK: https://www.nltk.org

[35] SciPy: https://scipy.org

[36] SK Learn: https://scikit-learn.org

[37] Kaggle: https://www.kaggle.com