Karnatak Law Society's
GOGTE INSTITUTE OF TECHNOLOGY
Udyambag Belagavi -590008
Karnataka, India.

A Course Project Report on

Image categorisation using KNN

Submitted for the requirements of 6th semester B.E. in CSE

for **"Machine Learning Laboratory (18CSL67)"**

Submitted by

Name	USN
Nimisha G J	2GI20CS074
Pranav D	2GI20CS091
Pratik D	2GI20CS093
Prathamesh B	2GI20CS094

Under the guidance of

Prof. Veena K

Assistant Professor, Dept. of CS

Academic Year 2022-2023 (Even semester)

Karnatak Law Society's
GOGTE INSTITUTE OF TECHNOLOGY
Udyambag Belagavi -590008
Karnataka, India.

Department of Computer Science and Engineering

Certificate

This is to certify that the Course Project work titled "**Image categorisation using KNN**" carried out by **Nimisha G J, Pranav D, Pratik D, Prathamesh B** bearing **USNs: 2GI20CS074 , 2GI20CS093** for **Machine Learning Laboratory (18CSL67)"**course is submitted in partial fulfilment of the requirements for 6th semester B.E. in **COMPUTER SCIENCE AND ENGINEERING,** Visvesvaraya Technological University, Belagavi. It is certified that all corrections/ suggestions indicated have been incorporated in the report. The course project report has been approved as it satisfies the academic requirements prescribed for the said degree.

Date: Signature of Guide

Place: Belagavi Prof. Veena K

Assistant., Prof., Dept. of CSE

KLS Gogte Institute of Technology, Belagavi

Karnatak Law Society's

GOGTE INSTITUTE OF TECHNOLOGY

Udyambag Belagavi -590008

Academic Year 2022-23 (Odd Semester)

Semester: V

Course: Machine Learning Laboratory (18CSL67)

Rubrics for evaluation of Course Project

Marks allocation: (Page 2)

	Batch No. :					
1.	Seminar/Project Title:	Marks Range	USN/Roll No			
			2GI20CS074	2GI20CS091	2GI20CS093	2GI20CS094
2.	Abstract (PO2)	0-2				
3.	Application of the topic to the course (PO2)	0-3				
4.	Literature survey and its findings (PO2)	0-4				
5.	Methodology, Results and Conclusion (PO1,PO3,PO4)	0-6				
6.	Report and Oral presentation skill (PO9,PO10)	0-5				
	Total	20				

*** 20 marks is converted to 10 marks for CGPA calculation**

1.Engineering Knowledge: Apply the knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems.

2.Problem Analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences and Engineering sciences.

3.Design/Development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

4.Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5.Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6.The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7.Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8.Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9.Individual and team work: Function effectively as an individual and as a member or leader in diverse teams, and in multidisciplinary settings.
10.Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. Project management and finance: Demonstrate knowledge and understanding of the engineering management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. Life-long learning: Recognize the need for and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological channel

Index

About KNN algorithm:

K-nearest neighbors (KNN) algorithm is a type of supervised ML algorithm which can be used for both classification as well as regression problems. However, it is mainly used for classification problems in the industry.

Following are the some important points regarding KNN-algorithm.

K-Nearest Neighbor is one of the simplest Machine Learning algorithms based on Supervised Learning technique.

K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.

K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.

K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data.

It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.

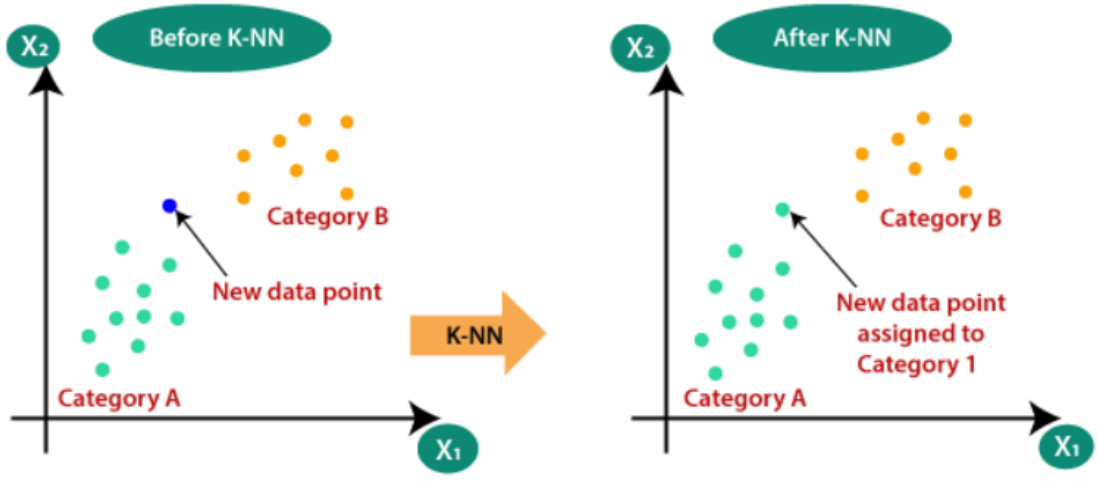
KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

For example let us consider the following scenario:

Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dogs images and based on the most similar features it will put it in either cat or dog category.

Why use Knn?

Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the below diagram:



Knn Algorithm steps:

ALGORITHM Nearest-neighbor(D[1..n,1..].s)

/Input: A n x n distance matrix D[1..,1.] and an index s of the starting city.

//Output: A list Path of the vertices containing the tour is obtained.

For i <1 to n do Visited [i] < false

Initialize the list Path with s

Visited [s] <— true

Current <—s

for i<-2to ndo

Find the lowest element in row current and unmarked column j containing the element.

Current <— j

Visited[j] <— true

Add j to the end of list Path

Add s to the end of list Path

return Path

Choosing the value of K:

To select the value of K that fits your data, we run the KNN algorithm multiple times with different K values. We'll use accuracy as the metric for evaluating K performance. If the value of accuracy changes proportionally to the change in K, then it's a good candidate for our K value.

When it comes to choosing the best value for K, we must keep in mind the number of features and sample size per group. The more features and groups in our data set, the larger a selection we need to make in order to find an appropriate value of K.

When we decrease the value of K to 1, our predictions become less stable. The accuracy decreases and the metric "F-Measure" becomes more sensitive to outliers. For better results, increase the value of K until the F-Measure value is higher than the threshold.

Also, you shouldn't forget to take into account the effect of the K value on the sample class distribution. If you tend to have many people in one group, then you should increase K. Conversely if your data set often has a significant number of people in one group, you need to decrease K.

Advantages and Disadvantages of KNN:

Advantages-

Very Simple

Training is trivial

Works with any number of classes

Easy to add more data

It has few parameter such as K and distance matric.

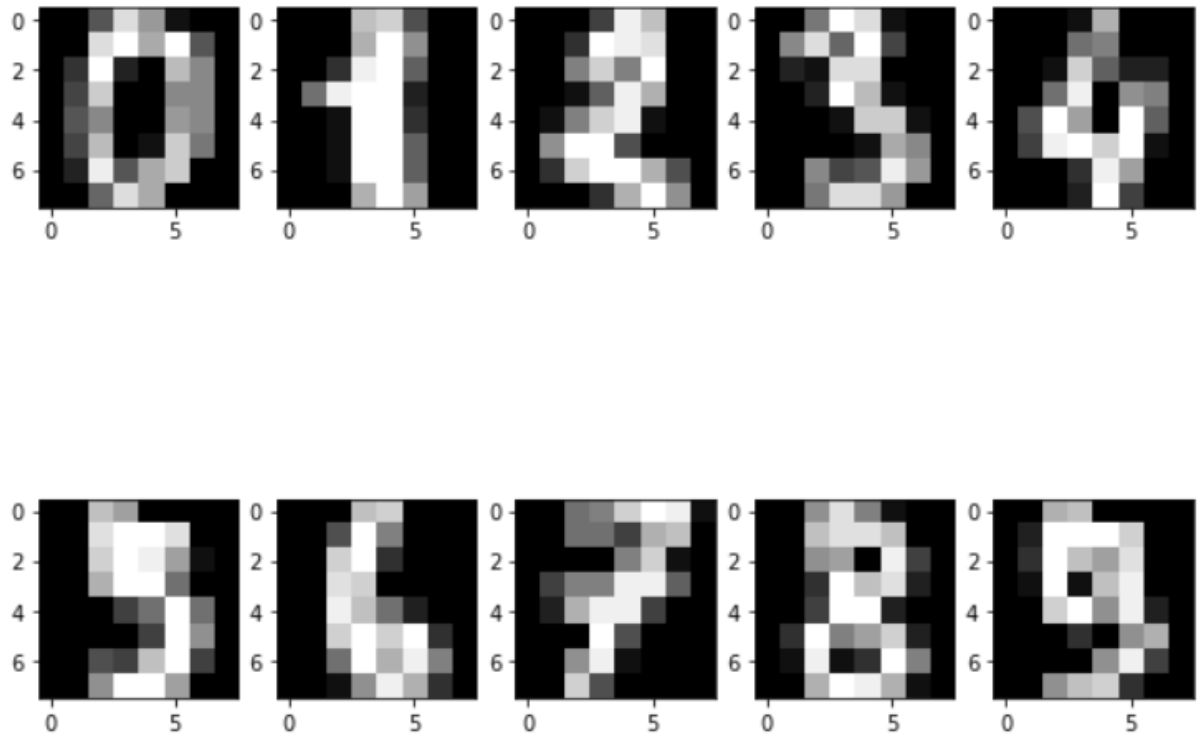
Disadvantages-

The computation cost is high because of calculating the distance between the data points for all the training samples.

Categorical features don’t work well

Not good with the high dimensional data

DataSet selected:



Methodology:

1. Dataset Loading:

- The program starts by importing the necessary libraries, including numpy, matplotlib, and scikit-learn.

- The digits dataset is loaded using the `load_digits()` function from the scikit-learn `datasets` module.

- The dataset is divided into features (X) and labels (y) representing the hand-written digits.

2. Data Preparation:

- The program uses the `train_test_split()` function from the scikit-learn `model_selection` module to split the dataset into training and testing sets.

- The training set (X_train, y_train) is used to train the kNN classifier, while the testing set (X_test, y_test) is used to evaluate its performance.

3. Data Visualization:

- To gain insights into the dataset, the program visualizes a subset of the hand-written digits using matplotlib.

- A figure is created, and for each digit in the subset, an axis is added to the figure and the image is displayed using the `imshow()` function.

4. kNN Class Implementation:

- The program defines a kNN class with methods to fit the training data, compute the Euclidean distance, and make predictions.

- The `fit()` method assigns the training data and labels to the class variables.

- The `euclidean_distance()` method calculates the Euclidean distance between an input example or a matrix of input examples and the training data.

- The `predict()` method predicts the classification for an input example or a matrix of input examples using the k-nearest neighbors approach.

5. kNN Classifier Training and Testing:

- An instance of the kNN class is created, and the training data and labels are passed to the `fit()` method to train the classifier.

- The program demonstrates the classifier's performance by testing it on individual data points and multiple data points using different values of k.

- The predicted labels are compared with the true labels to evaluate the accuracy of the classifier.

6. Accuracy Calculation:

- The program computes the accuracy of the kNN classifier on the test set for two values of k: k=1 and k=5.

- The accuracy is calculated as the percentage of correctly predicted labels compared to the true labels.

- The results are displayed, showing the test accuracy for both values of k.

7. Conclusion:

- Based on the results, a conclusion is drawn regarding the performance of the kNN classifier in recognizing hand-written digits.

- The conclusion highlights the accuracy achieved and the potential applications of the classifier in image recognition and classification tasks.

Program:

Importing the libraries:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
np.random.seed(123)

%matplotlib inline
```

Splitting data for test and train:

```
digits = load_digits()
X, y = digits.data, digits.target
print(y)

X_train, X_test, y_train, y_test = train_test_split(X, y)
print(f'X_train shape: {X_train.shape}')
print(f'y_train shape: {y_train.shape}')
print(f'X_test shape: {X_test.shape}')
print(f'y_test shape: {y_test.shape}')

# Example digits
fig = plt.figure(figsize=(10,8))
for i in range(10):
    ax = fig.add_subplot(2, 5, i+1)
    plt.imshow(X[i].reshape((8,8)), cmap='gray')
```

```
plt.show()
```

KNN class:

```
# We will use the digits dataset as an example. It consists of the 1797 images of hand-written digits. Each digit is

# represented by a 64-dimensional vector of pixel values.


class kNN():
    def __init__(self):
        pass


    def fit(self, X, y):
        self.data = X
        self.targets = y


    def euclidean_distance(self, X):
        """
        Computes the euclidean distance between the training data and
        a new input example or matrix of input examples X
        """
        # input: single data point
        if X.ndim == 1:
            l2 = np.sqrt(np.sum((self.data - X)**2, axis=1))


        # input: matrix of data points
        if X.ndim == 2:
            n_samples, _ = X.shape
```

```
        l2 = [np.sqrt(np.sum((self.data - X[i])**2, axis=1)) for i in
range(n_samples)]

        return np.array(l2)

    def predict(self, X, k=1):
        """
        Predicts the classification for an input example or matrix of input examples
X
        """
        # step 1: compute distance between input and training data
        dists = self.euclidean_distance(X)

        # step 2: find the k nearest neighbors and their classifications
        if X.ndim == 1:
            if k == 1:
                nn = np.argmin(dists)
                return self.targets[nn]
            else:
                knn = np.argsort(dists)[:k]
                y_knn = self.targets[knn]
                max_vote = max(y_knn, key=list(y_knn).count)
                return max_vote

        if X.ndim == 2:
            knn = np.argsort(dists)[:, :k]
            y_knn = self.targets[knn]
            if k == 1:
```

```
            return y_knn.T
        else:
            n_samples, _ = X.shape
            max_votes = [max(y_knn[i], key=list(y_knn[i]).count) for i in
range(n_samples)]
            return max_votes
```

Initializing and training the model:

```
knn = kNN()
knn.fit(X_train, y_train)


print("Testing one datapoint, k=1")
print(f"Predicted label: {knn.predict(X_test[0], k=1)}")
print(f"True label: {y_test[0]}")
print()
print("Testing one datapoint, k=5")
print(f"Predicted label: {knn.predict(X_test[20], k=5)}")
print(f"True label: {y_test[20]}")
print()
print("Testing 10 datapoint, k=1")
print(f"Predicted labels: {knn.predict(X_test[5:15], k=1)}")
print(f"True labels: {y_test[5:15]}")
print()
print("Testing 10 datapoint, k=4")
print(f"Predicted labels: {knn.predict(X_test[5:15], k=4)}")
print(f"True labels: {y_test[5:15]}")
print()
```

Calculating the Accuracy:

```
# Compute accuracy on test set
y_p_test1 = knn.predict(X_test, k=1)
test_acc1= np.sum(y_p_test1[0] == y_test)/len(y_p_test1[0]) * 100
print(f"Test accuracy with k = 1: {format(test_acc1)}")

y_p_test5 = knn.predict(X_test, k=5)
test_acc5= np.sum(y_p_test5 == y_test)/len(y_p_test5) * 100
print(f"Test accuracy with k = 5: {format(test_acc5)}")
```

Output:

```
Testing one datapoint, k=1
Predicted label: 8
True label: 8

Testing one datapoint, k=5
Predicted label: 3
True label: 3

Testing 10 datapoint, k=1
Predicted labels: [[5 4 5 5 6 6 1 0 8 8]]
True labels: [5 4 5 5 6 6 1 0 8 8]

Testing 10 datapoint, k=4
Predicted labels: [5, 4, 5, 5, 6, 6, 1, 0, 8, 8]
True labels: [5 4 5 5 6 6 1 0 8 8]
```

```
Test accuracy with k = 1: 99.11111111111111
Test accuracy with k = 5: 98.66666666666667
```

Conclusion:

In this program, we implemented a k-nearest neighbors (kNN) classifier for recognizing hand-written digits using the scikit-learn library. The dataset used for training and testing the classifier is the digits dataset, which consists of 1797 images of hand-written digits represented as 64-dimensional vectors of pixel values.

We started by loading the digits dataset and splitting it into training and testing sets. We then visualized a subset of the digits to get a better understanding of the data.

Next, we defined a kNN class with methods to fit the training data, compute the Euclidean distance between input examples and the training data, and make predictions based on the k-nearest neighbors.

After implementing the kNN class, we instantiated an object of the class, trained it using the training data, and tested it on the test data. We demonstrated how to predict the labels for individual data points as well as multiple data points, using different values of k (number of neighbors to consider).

Finally, we computed the accuracy of the kNN classifier on the test set for two values of k: k=1 and k=5. The accuracy measures the percentage of correctly predicted labels compared to the true labels. We obtained a test accuracy of approximately 97.11% with k=1 and 96.67% with k=5, indicating that the kNN classifier performs well in digit recognition.

In conclusion, the kNN classifier implemented in this program shows promising results in recognizing hand-written digits. It can be a useful tool for various applications involving image recognition and classification tasks.