Logo

Description automatically generated

ALY6020 Module 1 Midweek Project

**Authored by: Prateek Singh**

Course: ALY6020 – Predictive Analytics

Submission Date: 04/17/2022

**Instructor’s name: Kristen Drobnis**

# Introduction

The assignment is focused on building and interpreting the nearest neighbor algorithm. We have used Iris flower species dataset and build the KNN model to classify the categories of flower in the dataset. Once the model is trained it can be used to test any unknown entity and compartmentalize them amongst the pre-defined categories. Model is trained and the categories have been labeled using the multiple attributes of the flowers, hence when we want to test any unknown type of Iris flower, we introduce it to the model and model calibrates it and put them in the category which is very alike to that flower. Basically, based on the attributes of the unknown flower, model tries to find under which category that flower lies in or in other words that flower is showing similarity with which group.

# Analysis

To start with the analysis, the dataset was downloaded from the sklearn library. The dataset is about Iris Flowers species and has features like sepal length and width and petal length and width in centimeters. There are three Target or as well call them categories of flowers under which all the flowers are classified namely Setosa, Versicolor and Virginica. We will use the dataset to train our model and using the nearest neighbor technique will try to fit all the flowers under either of these three categories. Based on the attributes sepal and petal length and width, we distribute them to understand a behavior and hence will see accordingly the groups or cluster formed based on similar behavior. This will help us see a pattern of the categories and eventually help us putting any new item under those categories using the K Nearest Neighbor model.

Sepal length minimum value is 4.3cm going as max as 7.9cm. The mean of sepal length is observed as 5.84cm. Coming to Petal length the minimum value is 1.0cm and maximum value is 6.9 with mean as 3.76cm.

Sepal width minimum value is 2.0cm going as max as 4.4cm. The mean of sepal width is observed as 3.05cm. Coming to Petal width the minimum value is 0.1cm and maximum value is 2.5cm with mean as 1.20cm.

Based on these attributes only the distribution is observed, and categories are formed.

Moving to our next steps, we set a seed of 42 which can be any number however, while creating a sample of 12 datapoints we want the same datapoints every time we run the model to avoid changing the results. We then divided our dataset into train and test dataset which will be used to train our model and then test the trained model respectively. Setting seed help us to retain the datapoints in our train and test dataset as well.

We classified our three target or categories as 0, 1 and 2 and plotted them in a 3-dimensional distribution using a scatter plot to see the clusters. All the groups have also been assigned with different color palettes to distinguish between them. Three colors used were blue, red, and green. We can see in the below figure clearly the three categories distributed. Although the distribution is separate from each other on most of the occasion, we can see few overlapping datapoints. This can affect our model accuracy to predict an unknown flower category because of the fact that suggests few datapoints even though under different category pose similar features(attributes). Thus, making it hard for our model to decide under which category to place unknown entity if it falls in those ambiguous zones.

Chart, scatter chart

Description automatically generated

Fig 1: Iris Categories Distribution

Finally, we created our KNN model using the distances between the neighboring datapoints. Then a vote function is created for our model to predict the categories of the sample datapoints. Based on the attributes, model tries to put the Iris flowers under different categories. For the 12 samples, we ran the model to see its accuracy and determine if it is a best fit model or not.

# Conclusion

Based on the results achieved on the categorization of the Iris flower species, we can clearly see that the model was able to predict the categories correctly 11 times out of 12. Well, 1 failure could be because of those ambiguous zones we discussed earlier where datapoints have similar attributes even though they belong to different categories. Below is the result from our model. We can see that our model was able to predict accurately almost all the vote values against the labels for the 12 datapoints we used as sample. However, at index 8 we observe that our model voted 1 for the datapoint [4.9 2.5 4.5 1.7] but the actual label was 2.

index: 0 , result of vote: 1 , label: 1 , data: [5.7 2.8 4.1 1.3]

index: 1 , result of vote: 2 , label: 2 , data: [6.5 3. 5.5 1.8]

index: 2 , result of vote: 1 , label: 1 , data: [6.3 2.3 4.4 1.3]

index: 3 , result of vote: 1 , label: 1 , data: [6.4 2.9 4.3 1.3]

index: 4 , result of vote: 2 , label: 2 , data: [5.6 2.8 4.9 2. ]

index: 5 , result of vote: 2 , label: 2 , data: [5.9 3. 5.1 1.8]

index: 6 , result of vote: 0 , label: 0 , data: [5.4 3.4 1.7 0.2]

index: 7 , result of vote: 1 , label: 1 , data: [6.1 2.8 4. 1.3]

index: 8 , result of vote: 1 , label: 2 , data: [4.9 2.5 4.5 1.7]

index: 9 , result of vote: 0 , label: 0 , data: [5.8 4. 1.2 0.2]

index: 10 , result of vote: 1 , label: 1 , data: [5.8 2.6 4. 1.2]



index: 11 , result of vote: 2 , label: 2 , data: [7.1 3. 5.9 2.1]

Overall, 11 out of 12 can be deemed as a good accuracy rate (92%) and hence we can conclude that our model accuracy is good and it can be considered as a good fit model.

# Reference

1. Grosz, J. (n.d.). Working With Nearest Neighbors. Canvas. Retrieved April 16, 2022, from <https://northeastern.instructure.com/courses/105748/pages/lesson-1-5-running-the-algorithm?module_item_id=7143680>
2. Load and return the iris dataset. scikit. (n.d.). Retrieved April 16, 2022, from <https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_iris.html>
3. Pingulkar, H. (2021, October 22). Scikit-Learn: NAMEERROR: Name 'Load\_iris' is not defined. Stack Overflow. Retrieved April 16, 2022, from <https://stackoverflow.com/questions/69675761/scikit-learn-nameerror-name-load-iris-is-not-defined>
4. Tournoij, M. (2015, March 17). TypeError: 'function' object is not subscriptable - Python. Stack Overflow. Retrieved April 16, 2022, from <https://stackoverflow.com/questions/29101836/typeerror-function-object-is-not-subscriptable-python>

# Appendix

# Importing the libraries

import numpy as np

from sklearn import datasets

import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d import Axes3D

from collections import Counter

# Loading the dataset Iris which classifies the types of flowers

Iris = datasets.load\_iris()

# Initializing independant variables and separating it from label

Iris\_Data = Iris.data

# Initializing labels as target on what we will be predicting the classification of flowers

Iris\_Labels = Iris.target

Iris

#Setting seed to randomly select the sample data from the dataset

np.random.seed(42)

# Initializing the sample size

indices = np.random.permutation(len(Iris\_Data))

TrainSampleSize = 12

# Creating a Train Dataset to train our model

Train\_Iris\_Data = Iris\_Data[indices[:-TrainSampleSize]]

Train\_Iris\_Labels = Iris\_Labels[indices[:-TrainSampleSize]]

# Creating a Test Dataset to test our model accuracy and prediction

Test\_Iris\_Data = Iris\_Data[indices[-TrainSampleSize:]]

Test\_Iris\_Labels = Iris\_Labels[indices[-TrainSampleSize:]]

# Plotting the three different Species or Categories of flowers in Iris Dataset which is our Target as well to see their distribution.

X = []

for iclass in range (3):

X.append([[],[],[]])

for i in range(len(Train\_Iris\_Data)):

if Train\_Iris\_Labels[i] == iclass:

X[iclass][0].append(Train\_Iris\_Data[i][0])

X[iclass][1].append(Train\_Iris\_Data[i][1])

X[iclass][2].append(sum(Train\_Iris\_Data[i][2:]))

colours = ("b", "r", "g")

plot = plt.figure()

ax = plot.add\_subplot(111, projection = '3d')

for iclass in range (3):

ax.scatter(X[iclass][0], X[iclass][1], X[iclass][2], c = colours[iclass])

plt.show()

# Calculating the KNN or the K nearest neighbour distance

def distance(point1, point2):

point1 = np.array(point1)

point2 = np.array(point2)

return np.linalg.norm(point1 - point2)

# Building KNN Model to correctly calculate the distance and predict the random datapoint how it can be classified as

def get\_neighbor(training\_set, labels, test\_instance, k, distance = distance):

distances = []

for index in range(len(training\_set)):

dist = distance(test\_instance, training\_set[index])

distances.append((training\_set[index], dist, labels[index]))

distances.sort(key = lambda x: x[1])

neighbors = distances[:k]

return(neighbors)

# Determine which are the closest datapoints to the test value which is getting predicted

def vote(neighbors):

class\_counter = Counter()

for neighbor in neighbors:

class\_counter[neighbor[2]] += 1

return class\_counter.most\_common(1)[0][0]

# Display vote and the labels for the sample data points and see the accuracy of the model

for i in range(TrainSampleSize):

neighbors = get\_neighbor(Train\_Iris\_Data, Train\_Iris\_Labels, Test\_Iris\_Data[i], 3, distance = distance)

print("index: ", i,

", result of vote: ", vote(neighbors),

", label: ", Test\_Iris\_Labels[i],

", data: ", Test\_Iris\_Data[i])