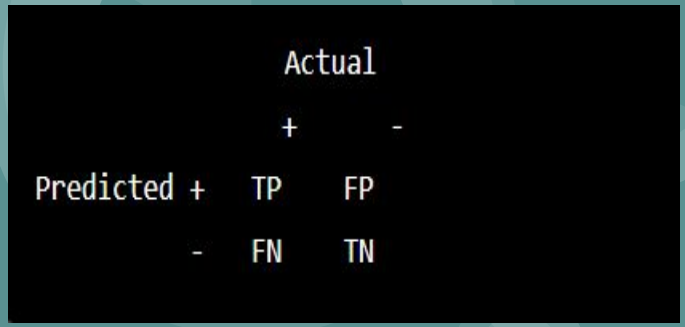
CS550 - Machine Learning and Business Intelligence



KNN + Confusion Matrix

By: Maranata Muluneh

Instructor: Dr. Chang, Henry

2023

Table of Content

Introduction

In machine learning, a confusion matrix is a table that is used to evaluate the performance of a classification model. The matrix compares the actual values of a target variable to the predicted values of the same variable. A confusion matrix is particularly useful in evaluating the accuracy of a K-nearest neighbors (KNN) algorithm.

The confusion matrix is a 2x2 table that displays four values: True Positive (TP), False Positive (FP), False Negative (FN), and True Negative (TN).

Introduction

Here is a confusion matrix structure:-

			Predicted values		
			Positive	**Negative**	**Totals**
	Actual Values	**Positive**	TP	FN	**P = (TP + FN) = Actual Total Positives**
		Negative	FP	TN	**N = (FP + TN) = Actual Total Negatives**
		Totals	Predicted Total Positives	Predicted Total Negatives	

Theory

TP (True Positives):

Actual positives in the data, which have been correctly predicted as positive by our model. Hence True Positive.

TN (True Negatives):

Actual Negatives in the data, which have been correctly predicted as negative by our model. Hence True negative.

Theory

FP (False Positives):

Actual Negatives in data, but our model has predicted them as Positive. Hence False Positive.

FN (False Negatives):

Actual Positives in data, but our model has predicted them as Negative. Hence False Negative.

Theory

Accuracy:

$$Accuracy = \frac{TP + TN}{P + N} = \frac{TP + TN}{TP + FN + TN + FP}$$

Accuracy formula

Accuracy tells the percentage of correctly predicted values out of all the data points. Often times, it may not be the accurate metric for our model performance. Specifically, when our data set is imbalanced.

Theory

$$TPR = \frac{TP}{TP + FN}$$

TPR formula

TPR (True Positive Rate) or Recall:

It tells us, out of all the ***positive*** data points, how many have been truly identified as positive by our model.

Theory

$$Precision = \frac{TP}{TP + FP}$$

Precision:

It tells use, out of all the points which have been identified as positive by our model, how many are actually true.

Implementation

For this project we have Iris flower data set to be used in for our model

Attributes of iris dataset

1. Sepal.Length: sepal length in cm
2. Sepal.Width: sepal width in cm
3. Petal.Length: petal length in cm
4. Petal.Width: petal width in cm
5. classes (species):

-- Iris Setosa

-- Iris Versicolour

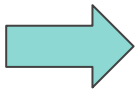
-- Iris Virginica

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Speci
5.1	3.5	1.4	0.2	I.Setosa
4.9	3.0	1.4	0.2	I.Setosa
4.7	3.2	1.3	0.2	I.Setosa
4.6	3.1	1.5	0.2	I.Setosa
5.0	3.6	1.4	0.2	I.Setosa
5.4	3.9	1.7	0.4	I.Setosa
4.6	3.4	1.4	0.3	I.Setosa
5.0	3.4	1.5	0.2	I.Setosa
4.4	2.9	1.4	0.2	I.Setosa
4.9	3.1	1.5	0.1	I.Setosa

Implementation (Using Colab)

Environment : Colab, Tensorflow 2

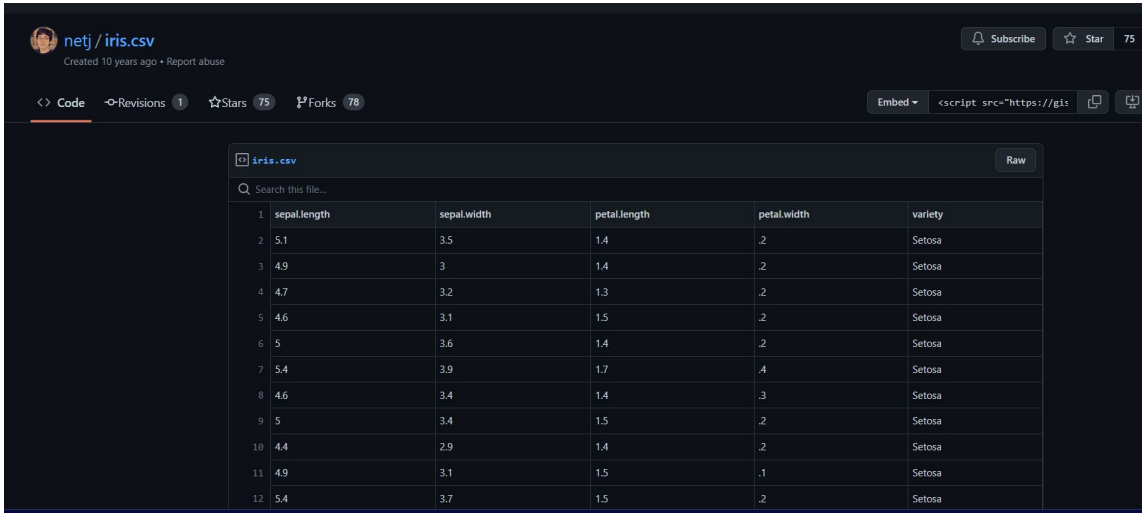
Programming Language: Python

Libraries 

```
import pandas as pd
import numpy as np
import sklearn
import seaborn as sns
import matplotlib.pyplot as plt
```

Implementation (Using Colab)

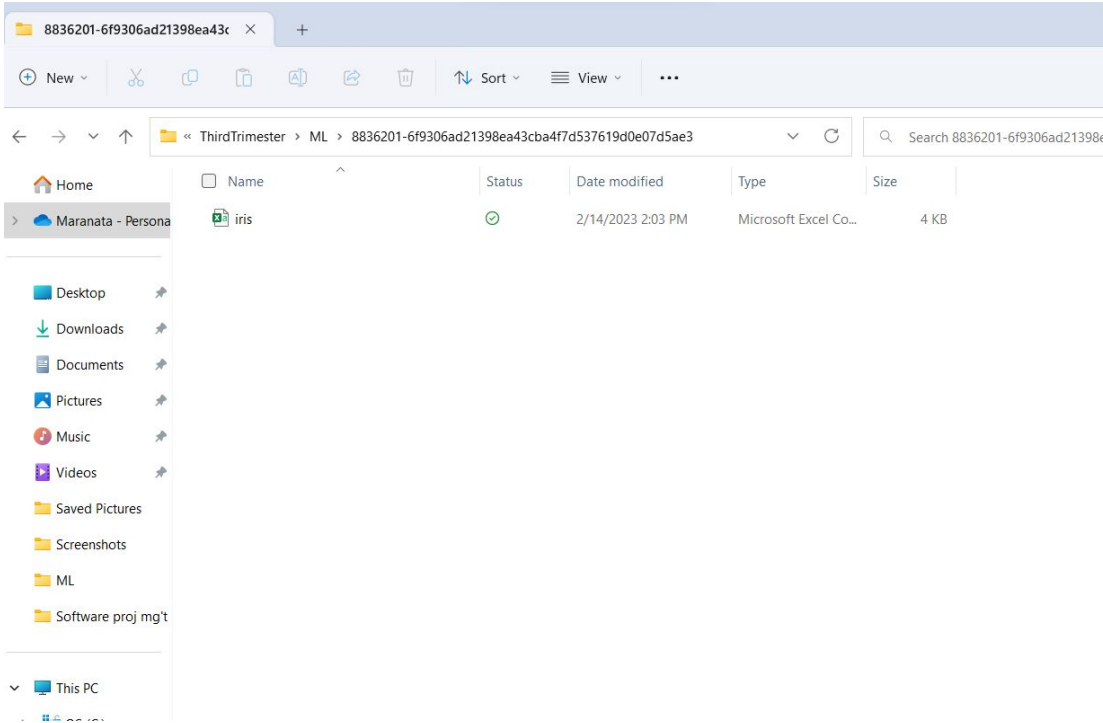
To run the Iris data set, we need to have the CSV file of the data, that is, iris.csv, from the references that are provided



	sepal.length	sepal.width	petal.length	petal.width	variety
2	5.1	3.5	1.4	.2	Setosa
3	4.9	3	1.4	.2	Setosa
4	4.7	3.2	1.3	.2	Setosa
5	4.6	3.1	1.5	.2	Setosa
6	5	3.6	1.4	.2	Setosa
7	5.4	3.9	1.7	.4	Setosa
8	4.6	3.4	1.4	.3	Setosa
9	5	3.4	1.5	.2	Setosa
10	4.4	2.9	1.4	.2	Setosa
11	4.9	3.1	1.5	.1	Setosa
12	5.4	3.7	1.5	.2	Setosa

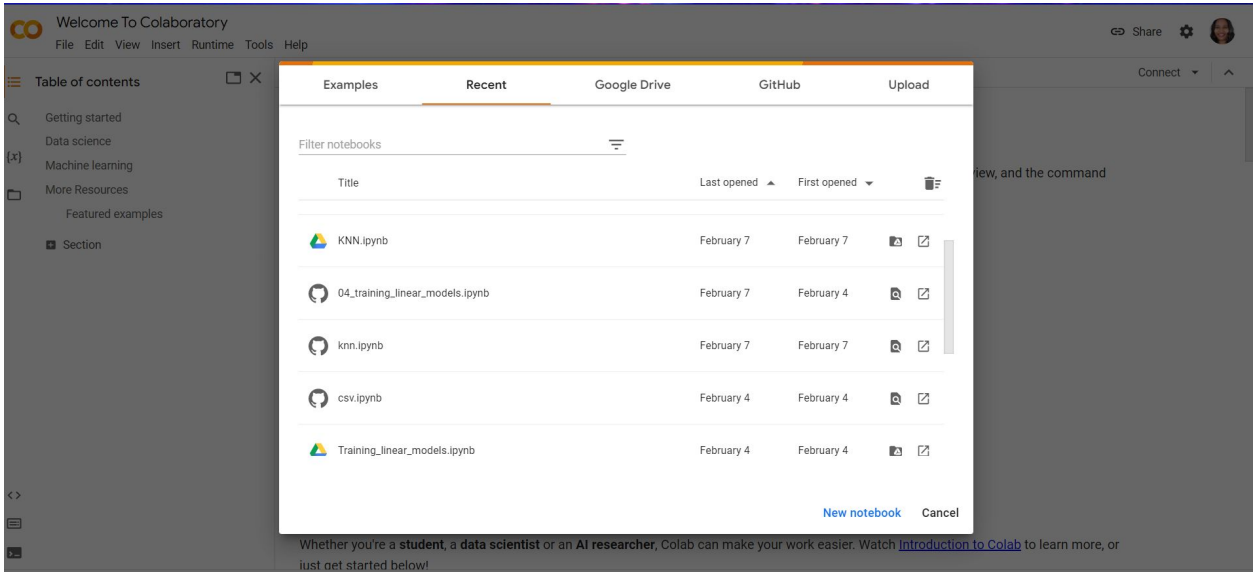
Implementation (Using Colab)

Here we have the extracted file for the Iris.csv saved in our local file



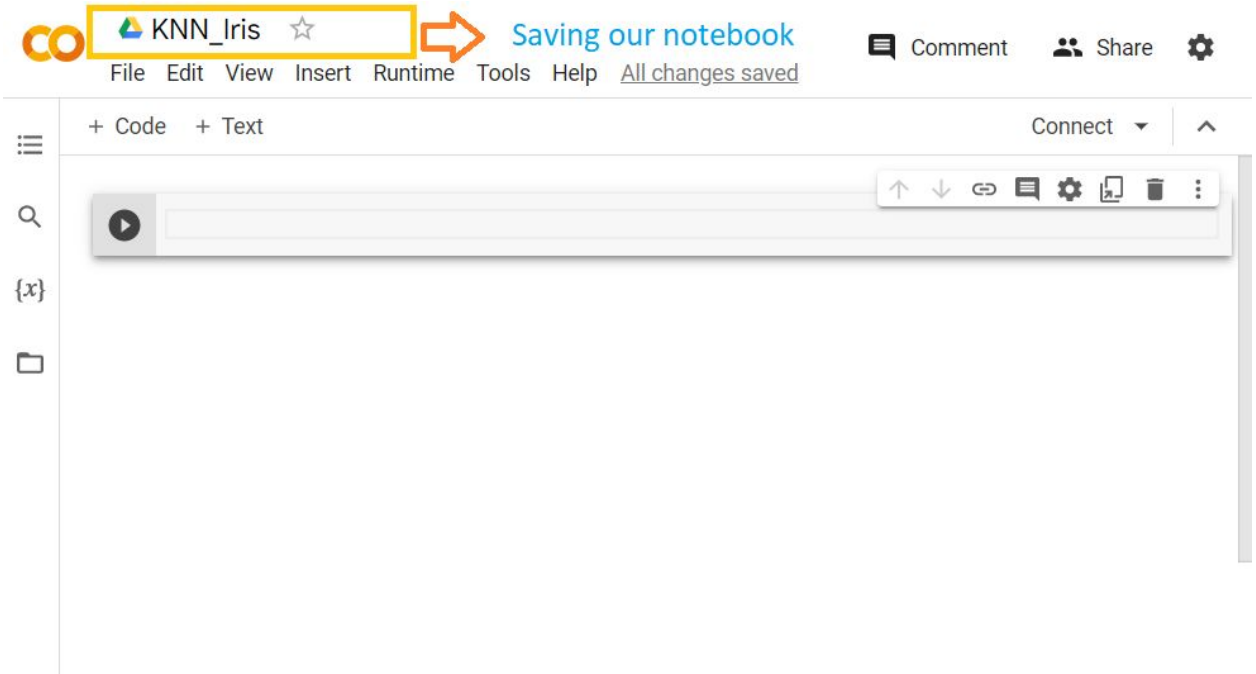
Implementation (Using Colab)

Opening a new notebook on Colab



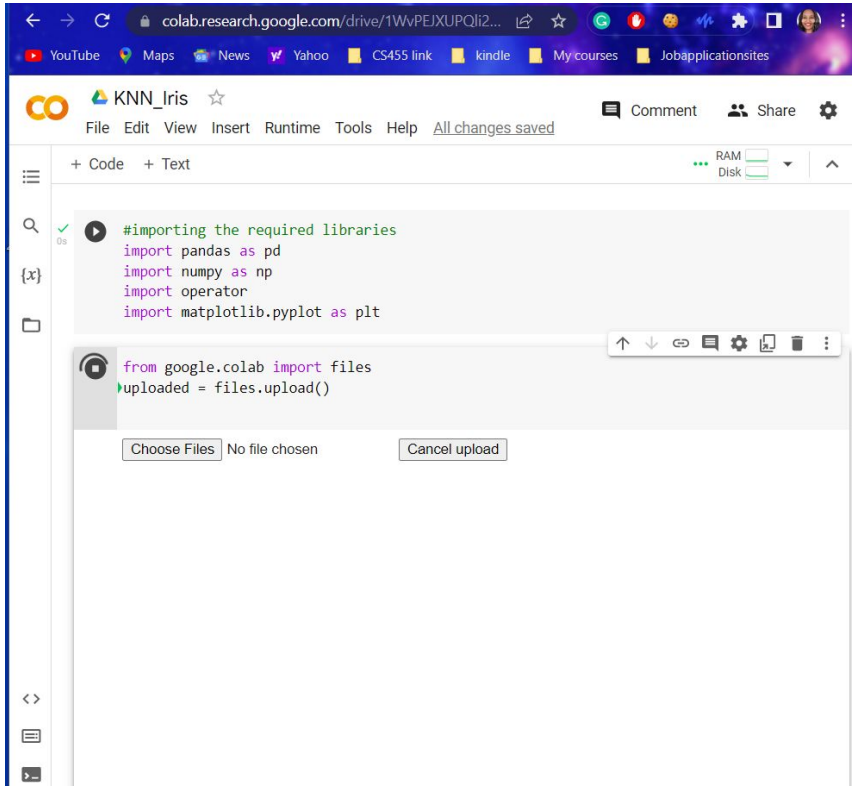
Implementation (Using Colab)

Saving our new notebook

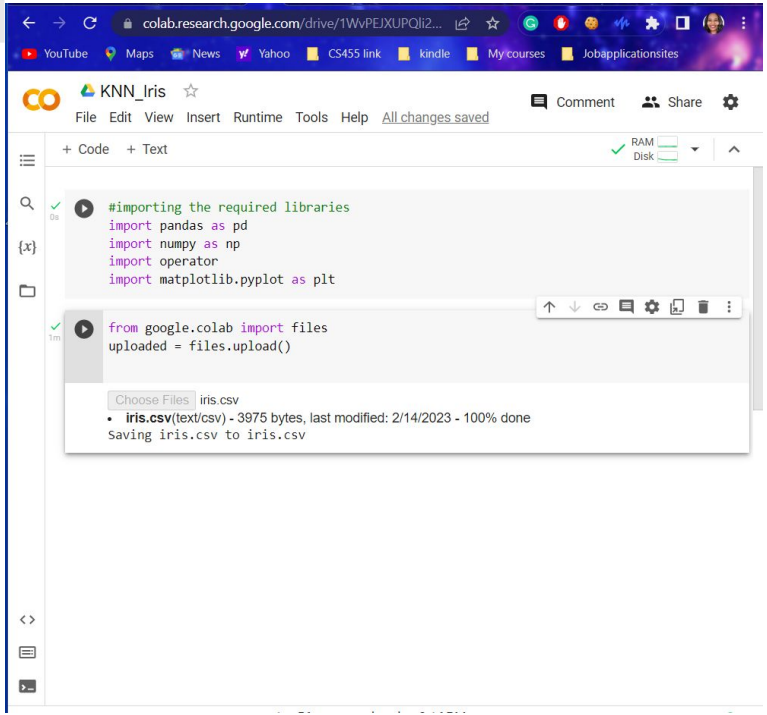


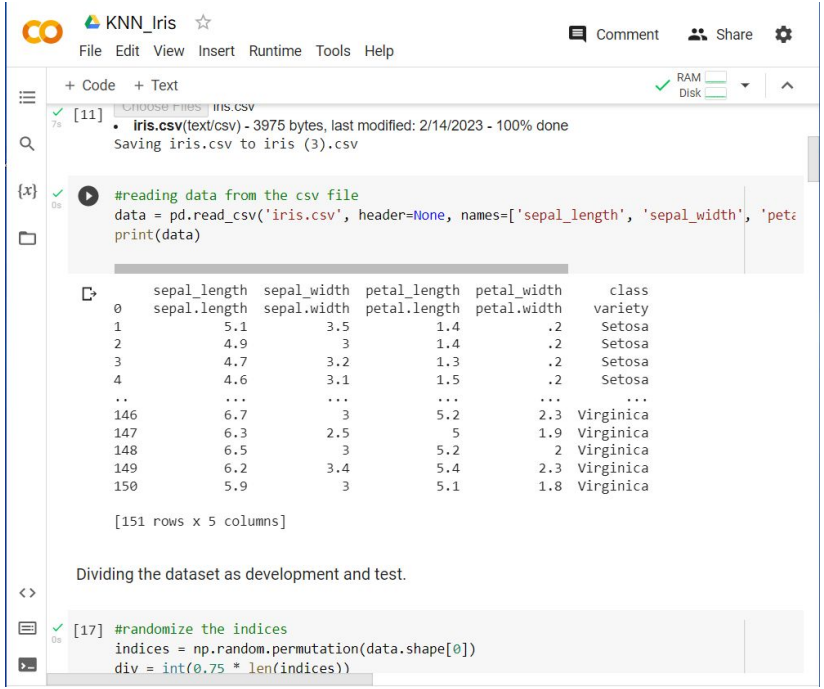
Implementation (Using Colab)

For more explanation of the code refer the github link:

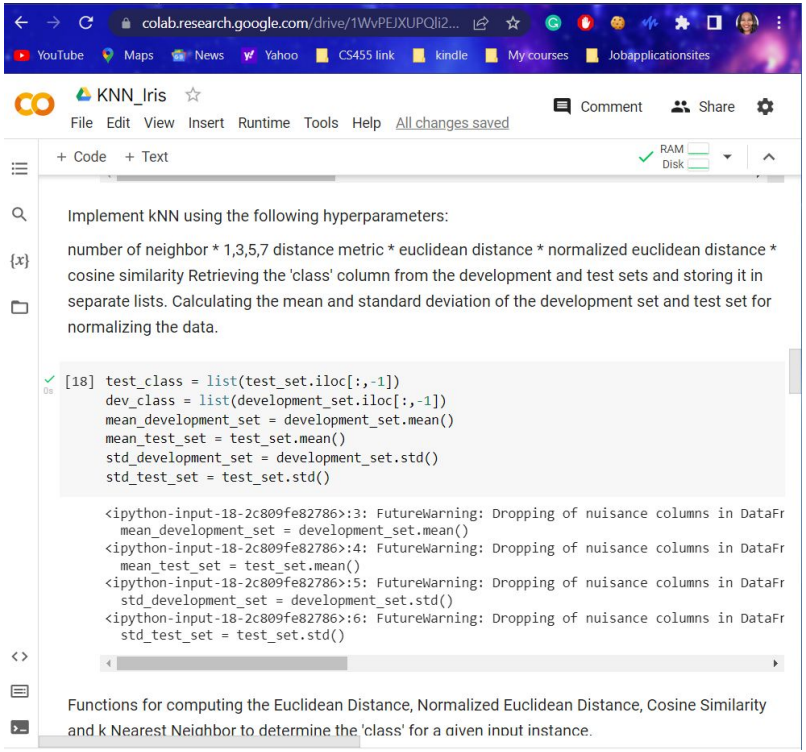


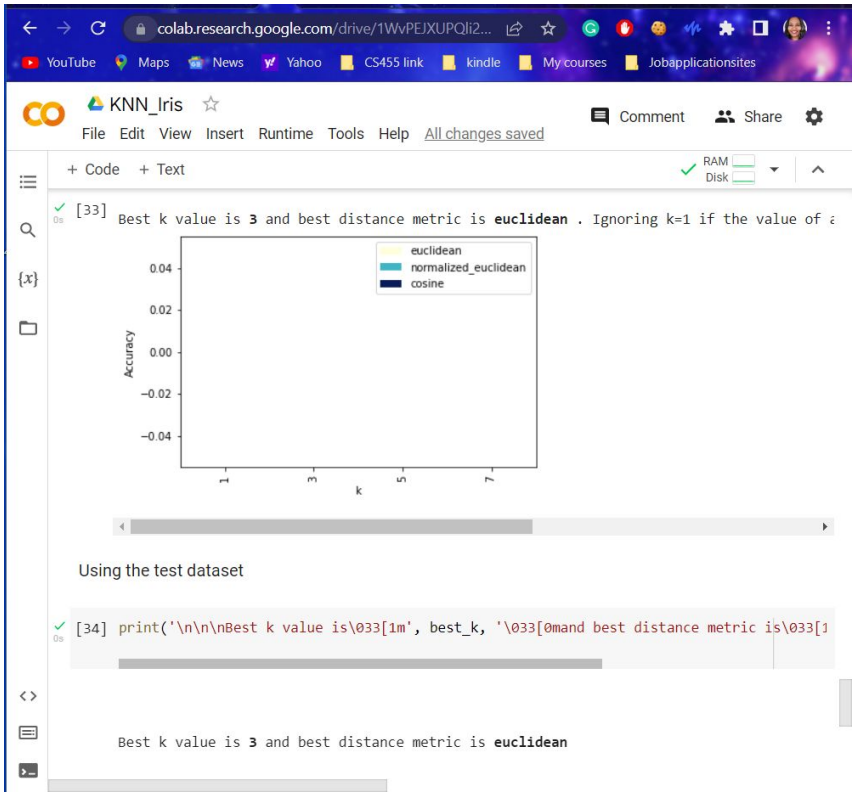
Implementation (Using Colab)





Implementation (Using Colab)





References

- Beena, V. (2020, May 13). *Understanding confusion matrix and applying it on KNN-classifier on Iris data set*. plainenglish.io/blog/understanding-confusion-matrix-and-applying-it-on-knn-classifier-on-iris-dataset-b57f85d05cd8. Retrieved February 14, 2023, from https://plainenglish.io/blog/understanding-confusion-matrix-and-applying-it-on-knn-classifier-on-iris-dataset-b57f85d05cd8
- Ishita-Kapur. (2020, May 27). K-nn-on-iris-dataset/knn_iris.ipynb at master · Ishita-Kapur/K-nn-on-iris-dataset. GitHub. Retrieved February 14, 2023, from https://github.com/ishita-kapur/k-NN-on-Iris-Dataset/blob/master/kNN_iris.ipynb