

CS550 - Machine Learning and Business Intelligence

		Actual	
		+	-
Predicted	+	TP	FP
	-	FN	TN

KNN + Confusion Matrix

By: Maranata Muluneh

Instructor: Dr. Chang, Henry

2023



Table of Content

- Introduction
- Theory
- Implementation
- References



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	
Actual Values	Positive	TP	FN	$P = (TP + FN) = \text{Actual Total Positives}$
	Negative	FP	TN	$N = (FP + TN) = \text{Actual Total Negatives}$
	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

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

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

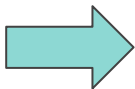


Implementation (Using Colab)

Environment : Colab, Tensorflow 2

Programming Language: Python

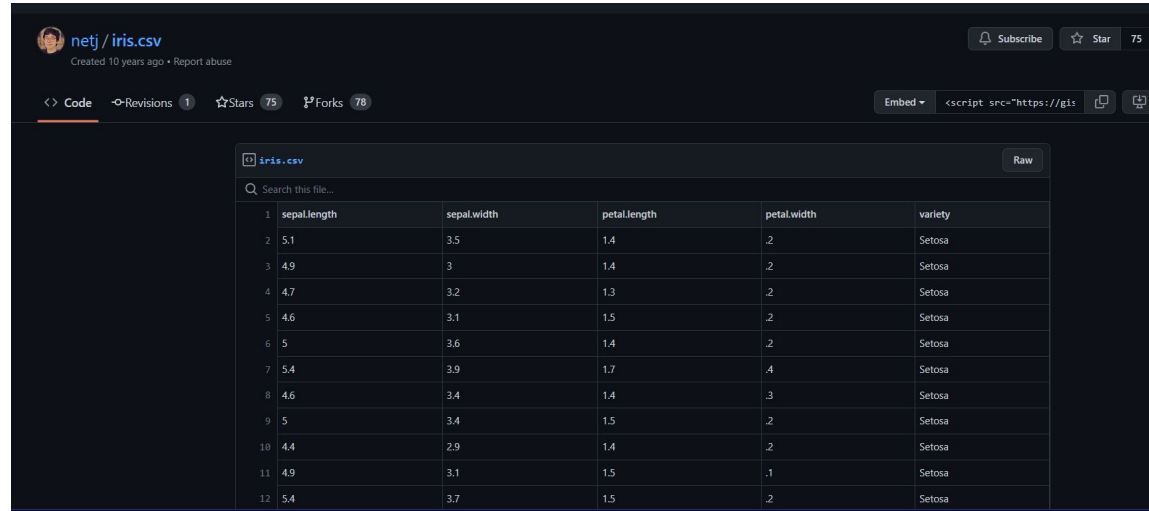
Libraries



```
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

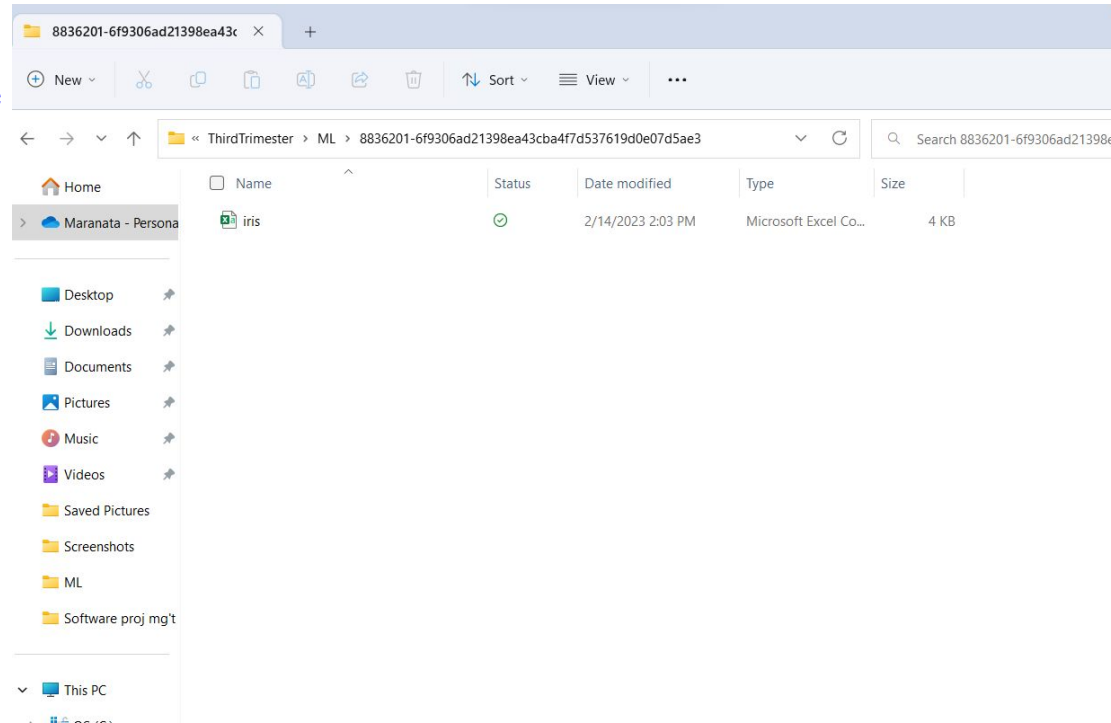


The screenshot shows a GitHub repository page for 'netj/iris.csv'. The repository was created 10 years ago and has 75 stars and 78 forks. The file 'iris.csv' is displayed in a table view with the following data:

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

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

co

Welcome To Colaboratory

File Edit View Insert Runtime Tools Help

Table of contents

Getting started

Data science

Machine learning

More Resources

Featured examples

Section

Share

Settings

Profile

Examples

Recent

Google Drive

GitHub

Upload

Filter notebooks

Title	Last opened	First opened	
KNN.ipynb	February 7	February 7	
04_training_linear_models.ipynb	February 7	February 4	
knn.ipynb	February 7	February 7	
csv.ipynb	February 4	February 4	
Training_linear_models.ipynb	February 4	February 4	

New notebook

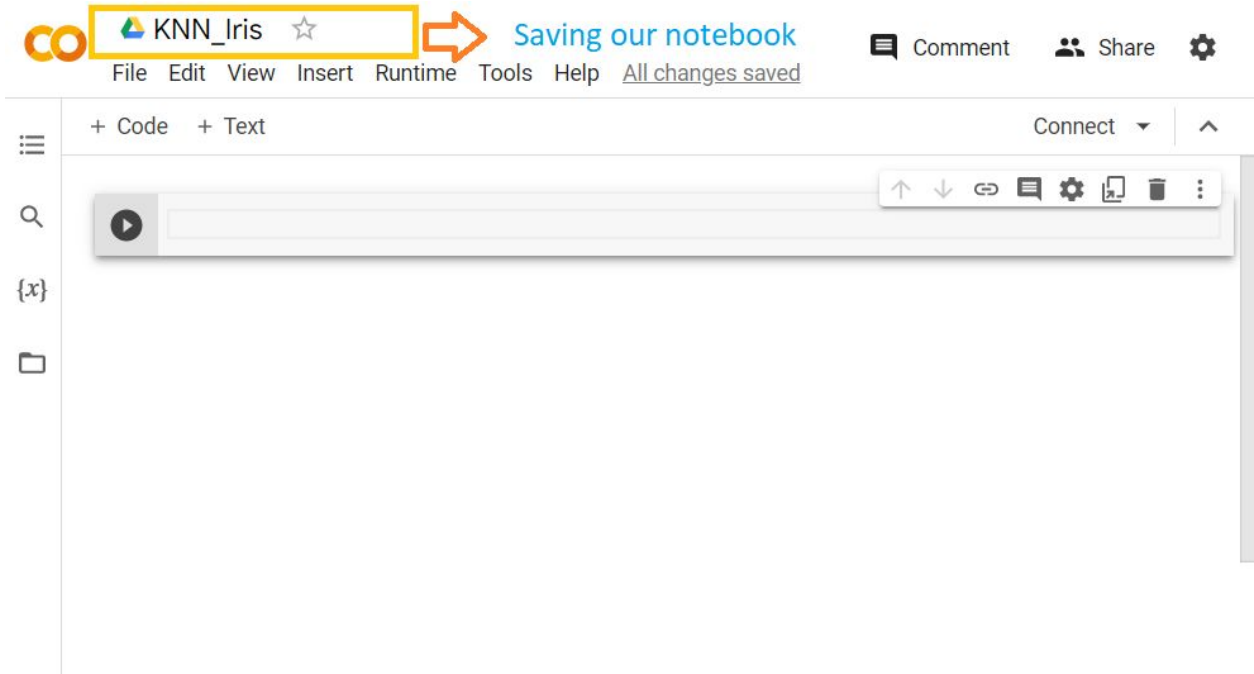
Cancel

Whether you're a student, a data scientist or an AI researcher, Colab can make your work easier. Watch [Introduction to Colab](#) to learn more, or just get started below!



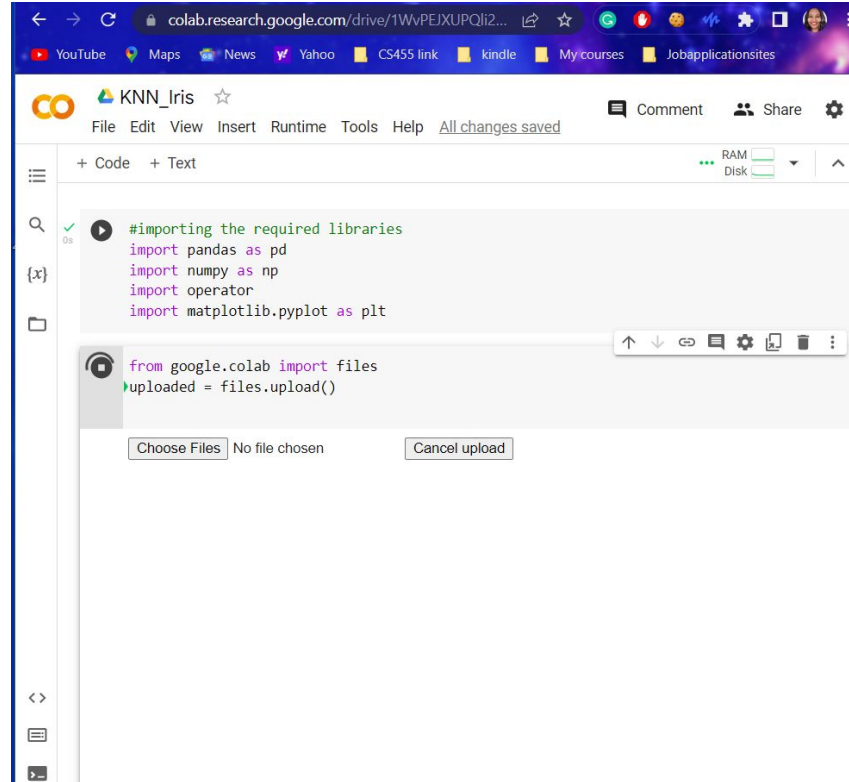
Implementation (Using Colab)

Saving our new notebook



Implementation (Using Colab)

For more explanation of the code refer the github link:



The screenshot shows a Google Colab notebook interface. The browser address bar displays `colab.research.google.com/drive/1WvPEJXUPQI2...`. The notebook title is `KNN_Iris`. The menu bar includes `File`, `Edit`, `View`, `Insert`, `Runtime`, `Tools`, and `Help`, with a status indicator `All changes saved`. On the left sidebar, there are icons for a file explorer, search, and a code editor. The main code area contains two code cells. The first cell, marked with a green checkmark, contains the following code:

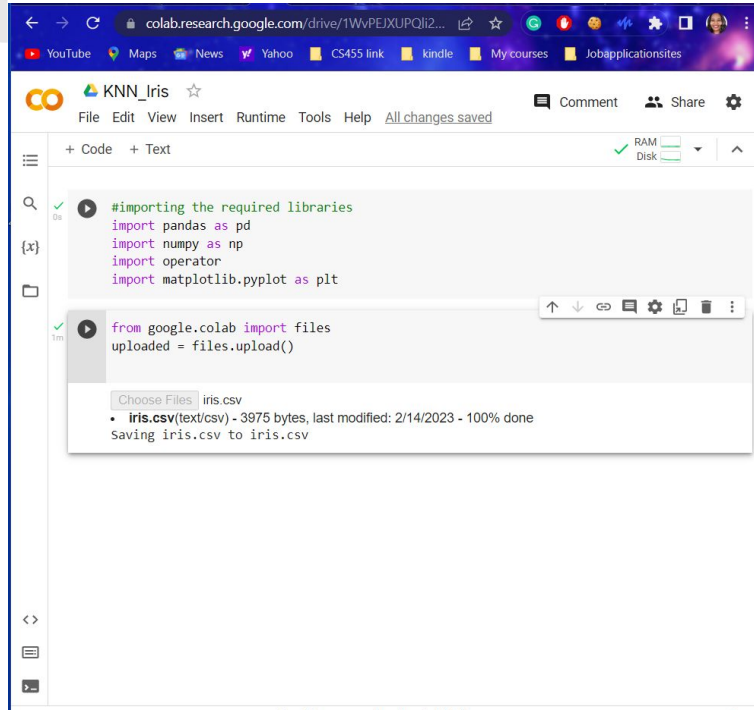
```
#importing the required libraries
import pandas as pd
import numpy as np
import operator
import matplotlib.pyplot as plt
```

 The second cell, marked with a play button icon, contains the code:

```
from google.colab import files
uploaded = files.upload()
```

 Below the code cells, there is a file upload interface with a `Choose Files` button, the text `No file chosen`, and a `Cancel upload` button. The top right of the interface shows `Comment` and `Share` options, along with a settings gear icon. The bottom left corner shows a code editor icon.

Implementation (Using Colab)

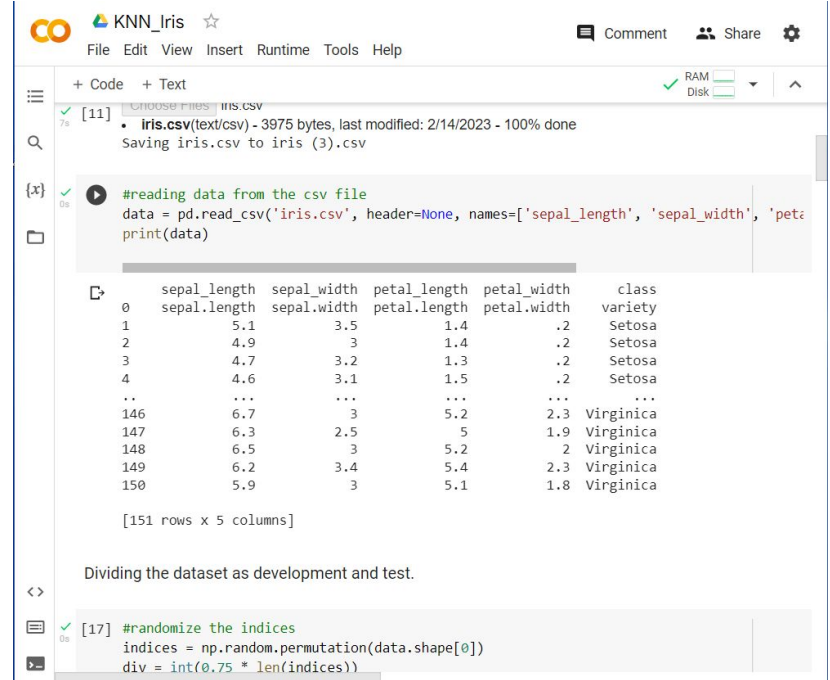


The screenshot shows a Google Colab notebook titled "KNN_Iris". The code cell [10] contains the following code:

```
#importing the required libraries
import pandas as pd
import numpy as np
import operator
import matplotlib.pyplot as plt

from google.colab import files
uploaded = files.upload()
```

A file upload dialog is open, showing "iris.csv" (3975 bytes) being uploaded. The status bar indicates "All changes saved".



The screenshot shows the same Google Colab notebook. The code cell [11] contains the following code:

```
#reading data from the csv file
data = pd.read_csv('iris.csv', header=None, names=['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'class'])
print(data)
```

The output of the code is a table with 151 rows and 5 columns. The first few rows are shown below:

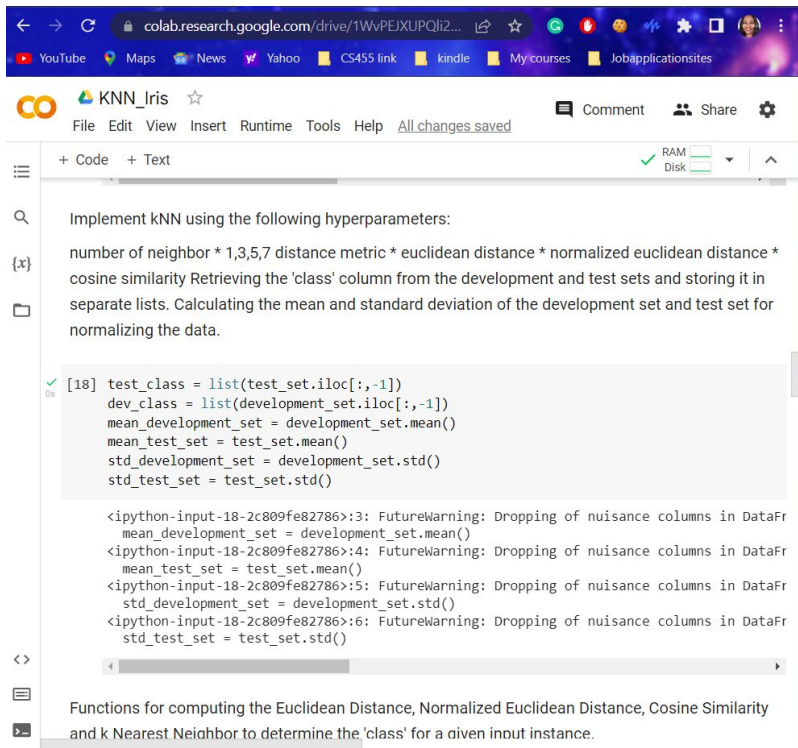
	sepal_length	sepal_width	petal_length	petal_width	class
0	sepal.length	sepal.width	petal.length	petal.width	variety
1	5.1	3.5	1.4	.2	Setosa
2	4.9	3	1.4	.2	Setosa
3	4.7	3.2	1.3	.2	Setosa
4	4.6	3.1	1.5	.2	Setosa
...
146	6.7	3	5.2	2.3	Virginica
147	6.3	2.5	5	1.9	Virginica
148	6.5	3	5.2	2	Virginica
149	6.2	3.4	5.4	2.3	Virginica
150	5.9	3	5.1	1.8	Virginica

The output is summarized as "[151 rows x 5 columns]". Below the table, the text "Dividing the dataset as development and test." is visible.

The code cell [17] contains the following code:

```
#randomize the indices
indices = np.random.permutation(data.shape[0])
div = int(0.75 * len(indices))
```

Implementation (Using Colab)



colab.research.google.com/drive/1WvPEJXUPQli2...

KNN_Iris

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

Implement KNN using the following hyperparameters:

- number of neighbor * 1,3,5,7
- distance metric * euclidean distance * normalized euclidean distance * cosine similarity

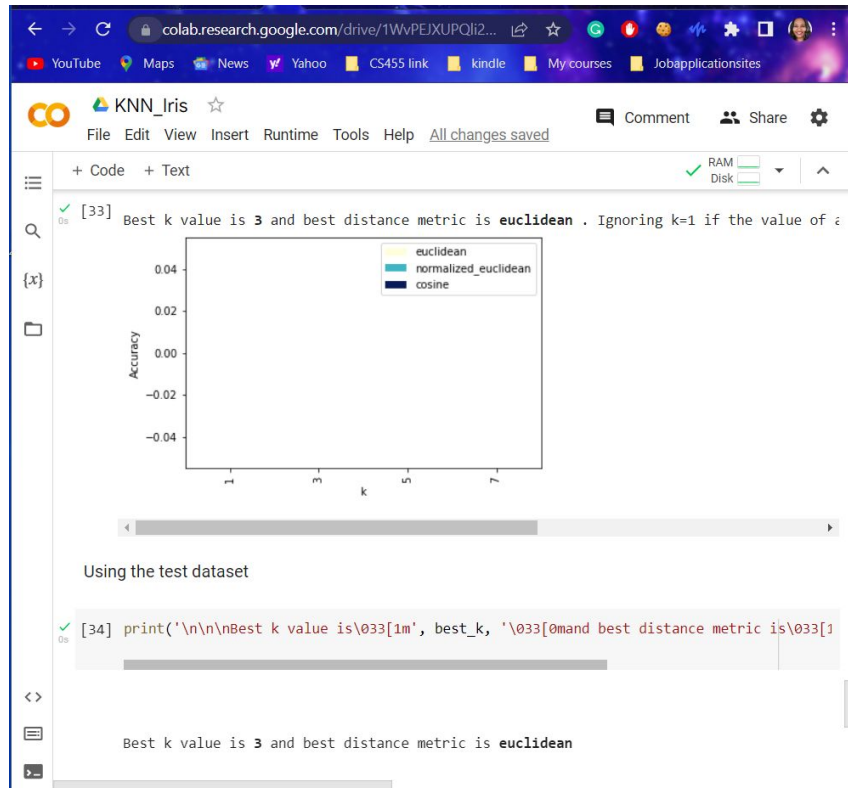
Retrieving the 'class' column from the development and test sets and storing it in separate lists. Calculating the mean and standard deviation of the development set and test set for normalizing the data.

```
[18] test_class = list(test_set.iloc[:, -1])
dev_class = list(development_set.iloc[:, -1])
mean_development_set = development_set.mean()
mean_test_set = test_set.mean()
std_development_set = development_set.std()
std_test_set = test_set.std()
```

<ipython-input-18-2c809fe82786>:3: FutureWarning: Dropping of nuisance columns in DataFrame

```
mean_development_set = development_set.mean()
<ipython-input-18-2c809fe82786>:4: FutureWarning: Dropping of nuisance columns in DataFrame
mean_test_set = test_set.mean()
<ipython-input-18-2c809fe82786>:5: FutureWarning: Dropping of nuisance columns in DataFrame
std_development_set = development_set.std()
<ipython-input-18-2c809fe82786>:6: FutureWarning: Dropping of nuisance columns in DataFrame
std_test_set = test_set.std()
```

Functions for computing the Euclidean Distance, Normalized Euclidean Distance, Cosine Similarity and k Nearest Neighbor to determine the 'class' for a given input instance.



colab.research.google.com/drive/1WvPEJXUPQli2...

KNN_Iris

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

Best k value is 3 and best distance metric is euclidean . Ignoring k=1 if the value of a

Accuracy

k

euclidean
normalized_euclidean
cosine

Using the test dataset

```
[34] print('\n\nBest k value is\033[1m', best_k, '\033[0mand best distance metric is\033[1m'
```

Best k value is 3 and best distance metric is euclidean



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