

CS550 - Machine Learning and Business Intelligence

Project Falling Detection: Python + kNN + Colab

By Soe Wunna (19651)

Instructor: Dr. Chang, Henry

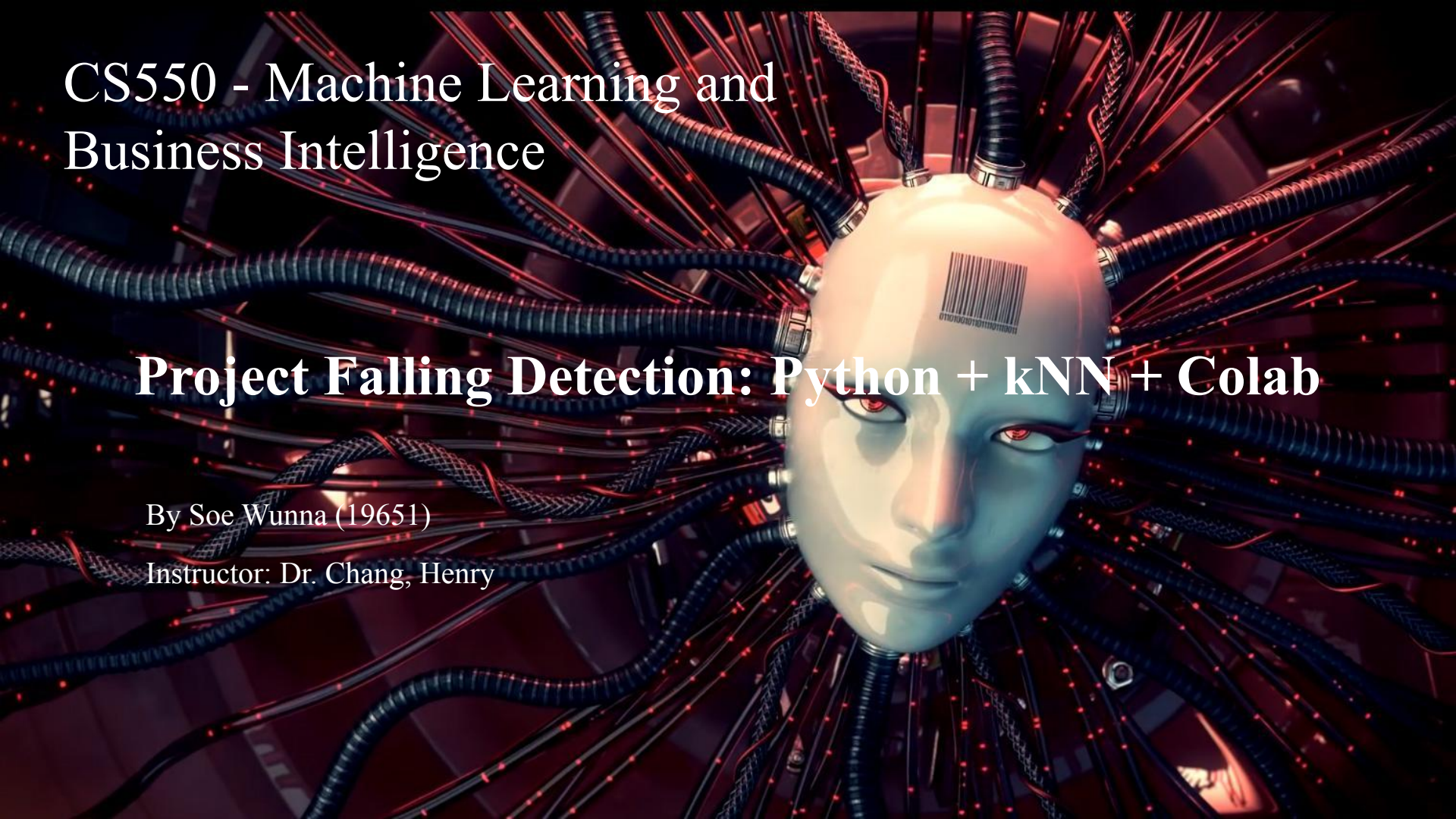


Table of Content

1. Introduction
2. Sensors
3. Accelerometer and Gyroscope
4. K-Nearest Neighbour
5. Manual Calculation
6. Calculation by Python in Google Colab
7. Comparing the Results
8. Conclusion
9. References

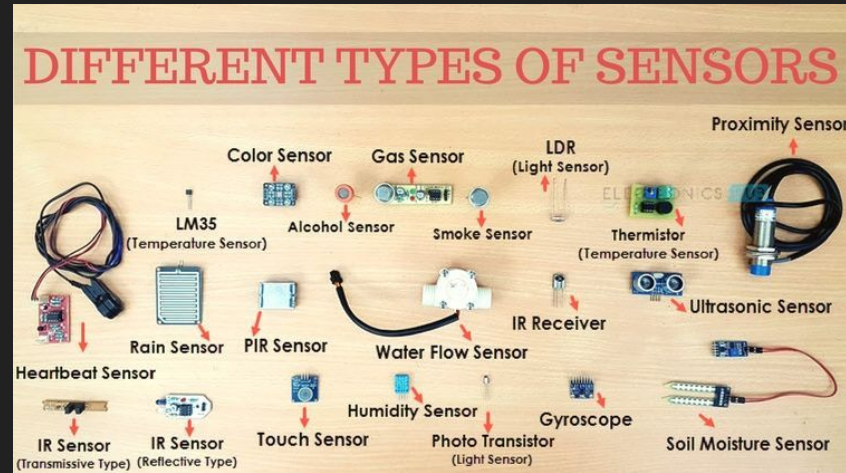


1. Introduction

In this presentation, we will discuss about K-Nearest Neighbor for Machine Learning, conduct manual calculations and counter check the manual result by implementing Python Notebook in Google Colab.

2. Sensors

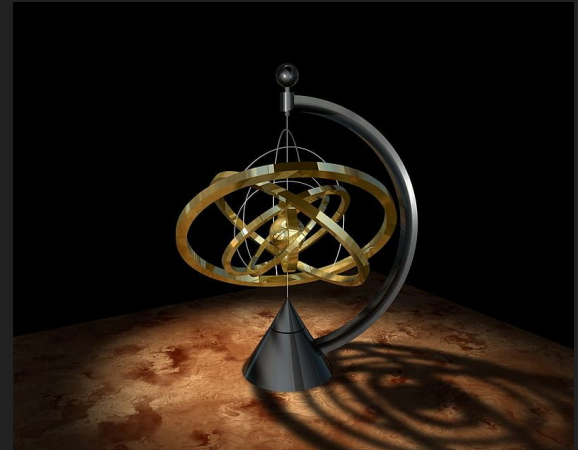
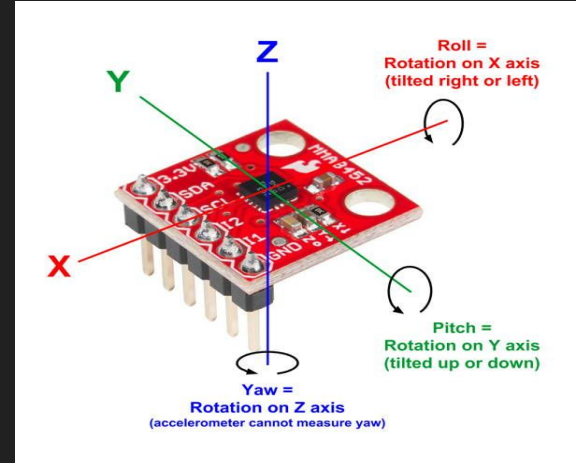
A sensor is an apparatus that generates an output signal to sense a physical occurrence. It is a device, module, machine, or subsystem that recognizes events or changes in its surroundings and transmits the data to other electronics, typically a computer processor. Always utilize sensors in conjunction with other electronics.



3. Accelerometer and Gyroscope

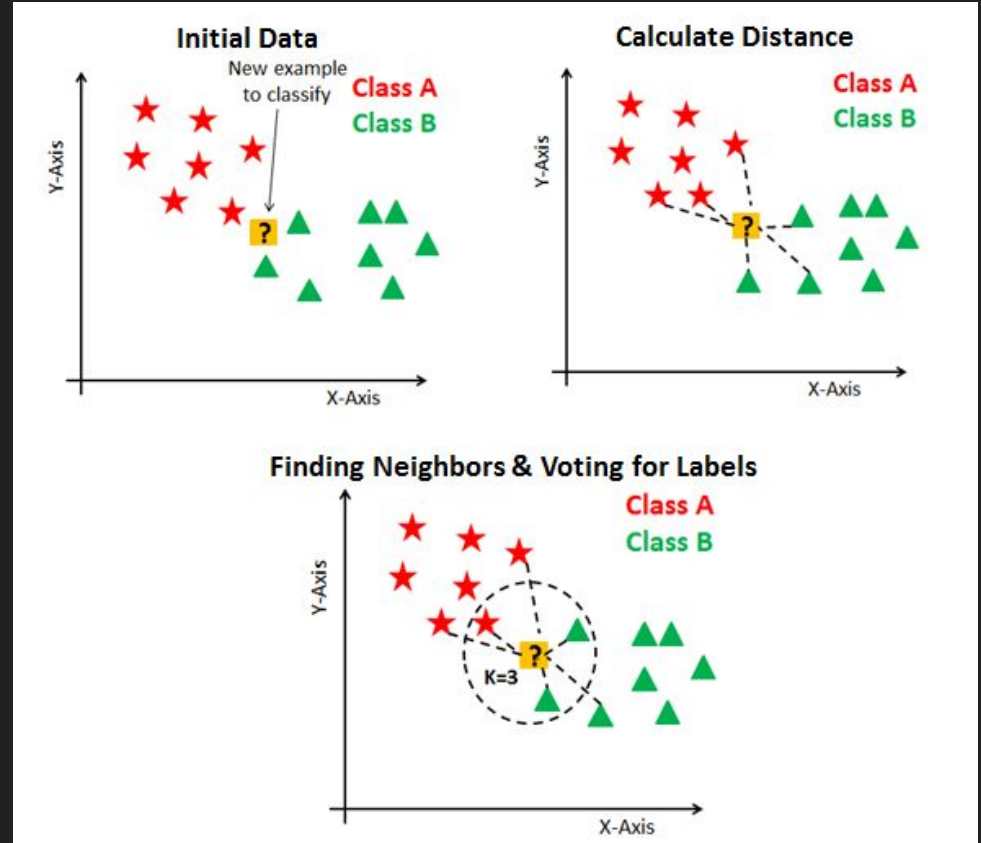
A device that accurately monitors acceleration is an accelerometer. Proper acceleration is different from coordinate acceleration, which is acceleration in a fixed coordinate system. Proper acceleration is the rate of change of velocity for a body in its own instantaneous rest frame. An accelerometer, for instance, will by definition measure an acceleration of $g = 9.81 \text{ m/s}^2$ straight upwards when it is at rest on the surface of the Earth. However, accelerometers in free fall (dropping at a speed of around 9.81 m/s^2) will register zero.

A gyroscope is a tool for measuring or maintaining orientation and angular velocity. It gets its name from the Ancient Greek words "o" gros, which means "round," and "skopé," which means "to look." It is a spinning wheel or disc where the spin axis is unrestricted in its orientation choices. According to the law of conservation of angular momentum, while the mounting is rotating or tilting, the orientation of this axis is unchanged.



4. K-Nearest Neighbour

K-Nearest Neighbor (KNN) is a simple machine learning algorithm used for classification and regression problems. In KNN, an object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors. The " k " is a user-defined constant, which is typically set using cross-validation. The algorithm is based on the idea that similar objects tend to be near each other. KNN can also be used for regression problems by predicting the average of the k nearest neighbors' output.



5. Manual Calculation

$$N = 8, K = \sqrt{N} = \sqrt{8} = 3$$

The choice of K equal to the odd number closest to the square root of the number of instances is an empirical rule-of-thumb popularized by the "Pattern Classification" book by Duda et al.

$$(\text{Target } x - \text{Data } x)^2 + (\text{Target } y - \text{Data } y)^2 + (\text{Target } z - \text{Data } z)^2$$

Since $K = 3$,

3 smallest numbers for Dist. Acce are 4, 5 and 6.

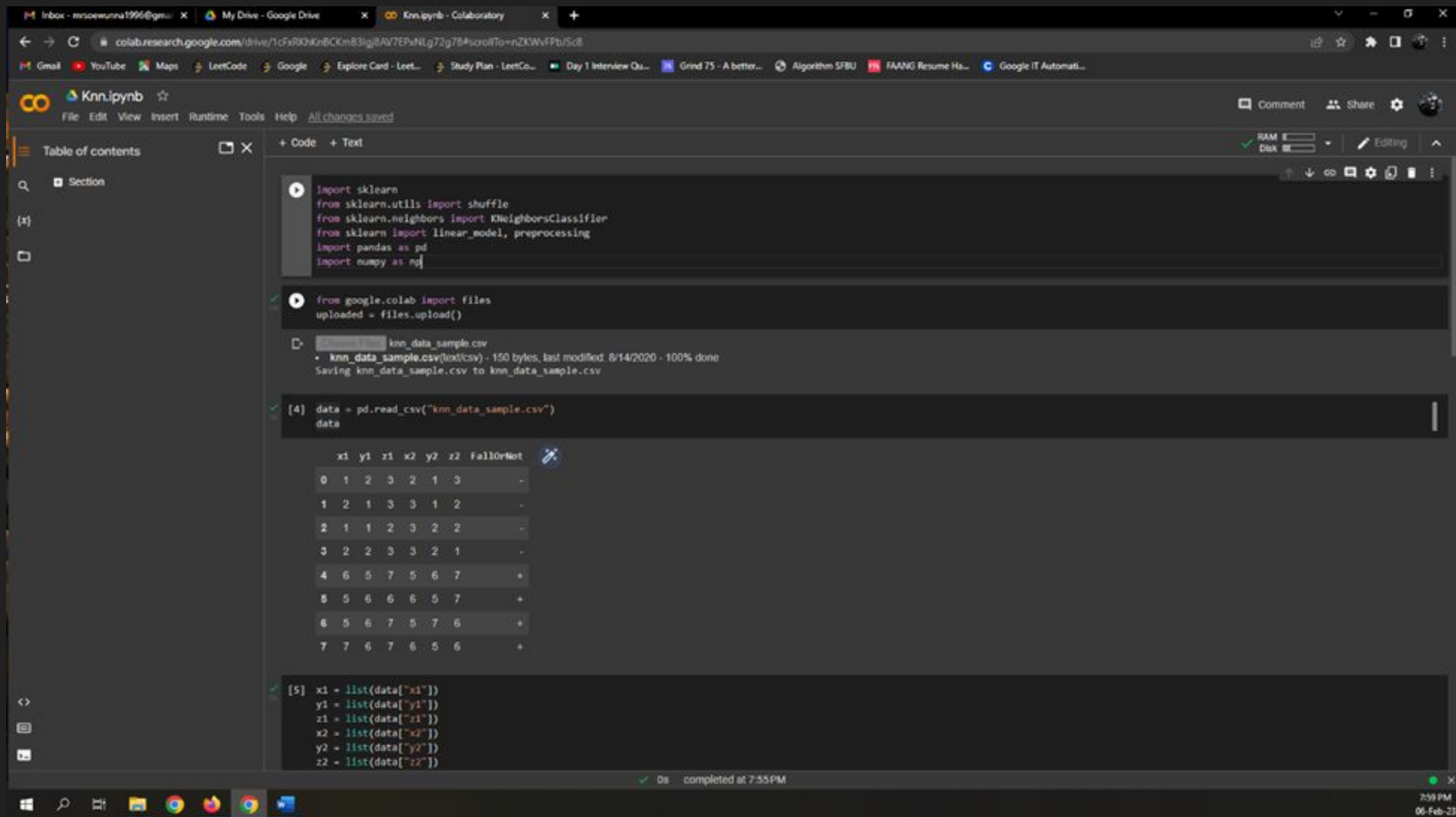
Since all numbers represent positive, the query instance is plus (+).

3 smallest numbers for Dist. Gyro are 0, 2 and 2.

Since all numbers represent positive, the query instance is plus (+).

Accelerometer Data			Gyroscope Data			Fall(+), Not (-)	Dist. Acce	Dist. Gyro
x	y	z	x	y	z	+/-		
1	2	3	2	1	3	-	56	50
2	1	3	3	1	2	-	54	54
1	1	2	3	2	2	-	70	45
2	2	3	3	2	1	-	45	56
6	5	7	5	6	7	+	6	0
5	6	6	6	5	7	+	5	2
5	6	7	5	7	6	+	8	2
7	6	7	6	5	6	+	4	3
7	6	5	5	6	7	+		

6. Calculation by Python in Google Colab



The screenshot shows a Google Colab notebook titled "Knn.ipynb". The notebook contains the following code cells:

```
import sklearn
from sklearn.utils import shuffle
from sklearn.neighbors import KNeighborsClassifier
from sklearn import linear_model, preprocessing
import pandas as pd
import numpy as np
```

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

The output of the second cell shows a file named "knn_data_sample.csv" uploaded.

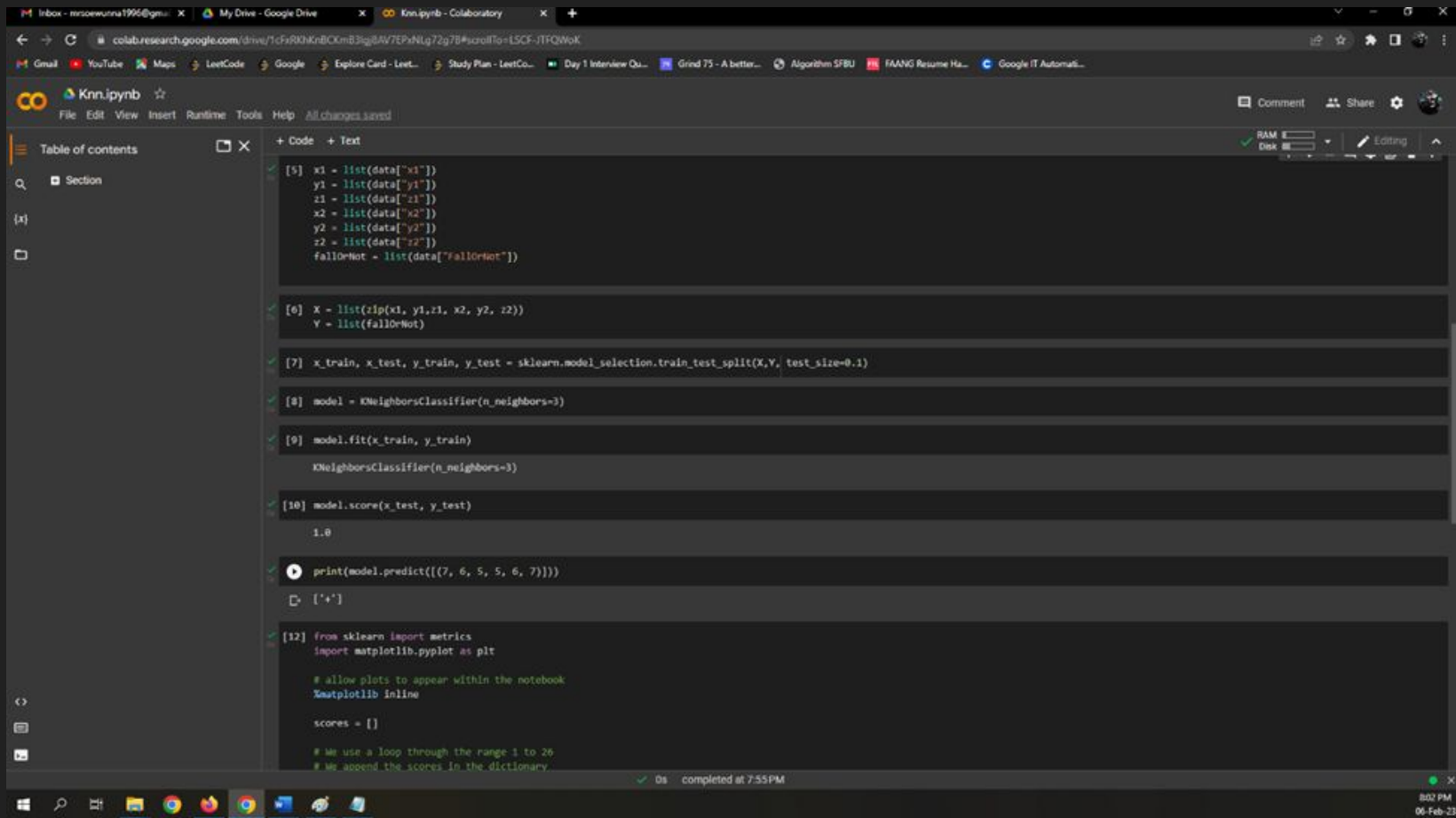
```
[4] data = pd.read_csv("knn_data_sample.csv")
data
```

	x1	y1	x1	x2	y2	z2	fallOrNot
0	1	2	3	2	1	3	-
1	2	1	3	3	1	2	-
2	1	1	2	3	2	2	-
3	2	2	3	3	2	1	-
4	6	5	7	5	6	7	+
5	5	6	6	6	5	7	+
6	5	6	7	5	7	6	+
7	7	6	7	6	5	6	+

```
[5] x1 = list(data["x1"])
y1 = list(data["y1"])
x1 = list(data["x1"])
x2 = list(data["x2"])
y2 = list(data["y2"])
z2 = list(data["z2"])
```

The notebook interface shows the "Table of contents" on the left, the "Code" editor in the center, and the "Output" on the right. The status bar at the bottom indicates "0s completed at 7:55PM".

6. Calculation by Python in Google Colab



The screenshot displays a Google Colab notebook titled "Knn.py - Colaboratory". The notebook contains a series of code cells for implementing a K-Nearest Neighbors (KNN) classifier. The code is as follows:

```
[5] x1 = list(data["x1"])
    y1 = list(data["y1"])
    z1 = list(data["z1"])
    x2 = list(data["x2"])
    y2 = list(data["y2"])
    z2 = list(data["z2"])
    fallOrNot = list(data["fallOrNot"])

[6] X = list(zip(x1, y1, z1, x2, y2, z2))
    Y = list(fallOrNot)

[7] x_train, x_test, y_train, y_test = sklearn.model_selection.train_test_split(X, Y, test_size=0.1)

[8] model = KNeighborsClassifier(n_neighbors=3)

[9] model.fit(x_train, y_train)

    KNeighborsClassifier(n_neighbors=3)

[10] model.score(x_test, y_test)

    1.0

[11] print(model.predict([(7, 6, 5, 5, 6, 7)]))

    ['+']

[12] from sklearn import metrics
    import matplotlib.pyplot as plt

    # allow plots to appear within the notebook
    %matplotlib inline

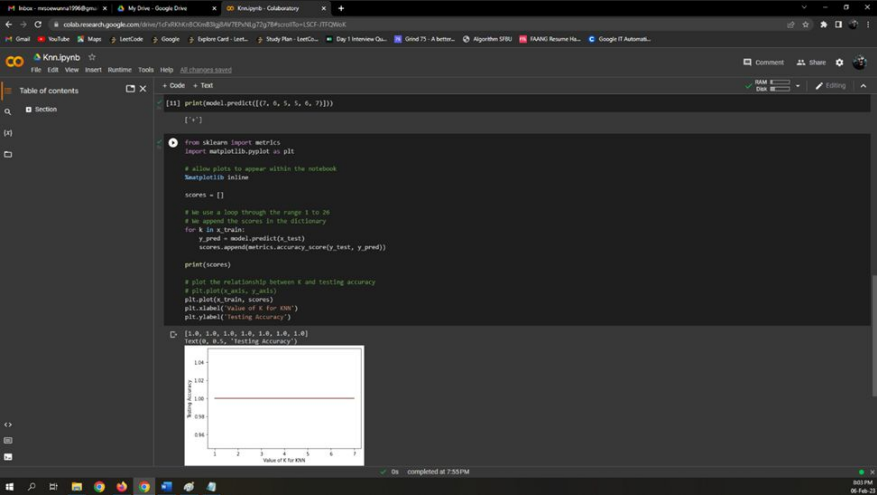
    scores = []

    # We use a loop through the range 1 to 26
    # We append the scores in the dictionary
```

The notebook interface includes a "Table of contents" on the left, a "Code" editor on the right, and a "Runtime" status bar at the bottom indicating "0s completed at 7:55PM". The bottom of the screen shows a Windows taskbar with various application icons.

7. Comparing the Results

Accelerometer Data			Gyroscope Data			Fall(+), Not (-)	Dist. Acce	Dist. Gyro
x	y	z	x	y	z	+/-		
1	2	3	2	1	3	-	56	50
2	1	3	3	1	2	-	54	54
1	1	2	3	2	2	-	70	45
2	2	3	3	2	1	-	45	56
6	5	7	5	6	7	+	6	0
5	6	6	6	5	7	+	5	2
5	6	7	5	7	6	+	8	2
7	6	7	6	5	6	+	4	3
7	6	5	5	6	7	+		



```
[ ] print(model.predict([(7, 6, 5, 5, 6, 7)]))

['+']
```

We got the same result.

8. Conclusion

In conclusion, K-Nearest Neighbor is a simple yet powerful algorithm that is widely used in machine learning. It is easy to implement and has the advantage of being non-parametric, meaning that it doesn't make assumptions about the underlying distribution of the data. KNN is well-suited for small datasets and can be used for both classification and regression problems. However, it may not perform well with large datasets or those with many features, as it requires a large amount of computation to find the nearest neighbors. Despite its limitations, KNN remains a popular choice for many applications due to its ease of use and interpretability.



9. References

1. https://hc.labnet.sfbu.edu/~henry/sfbu/course/data_science/algorithm/slide/k_nn_example.html
2. <https://www.ibm.com/topics/knn>
3. <https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761>
4. <https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html>