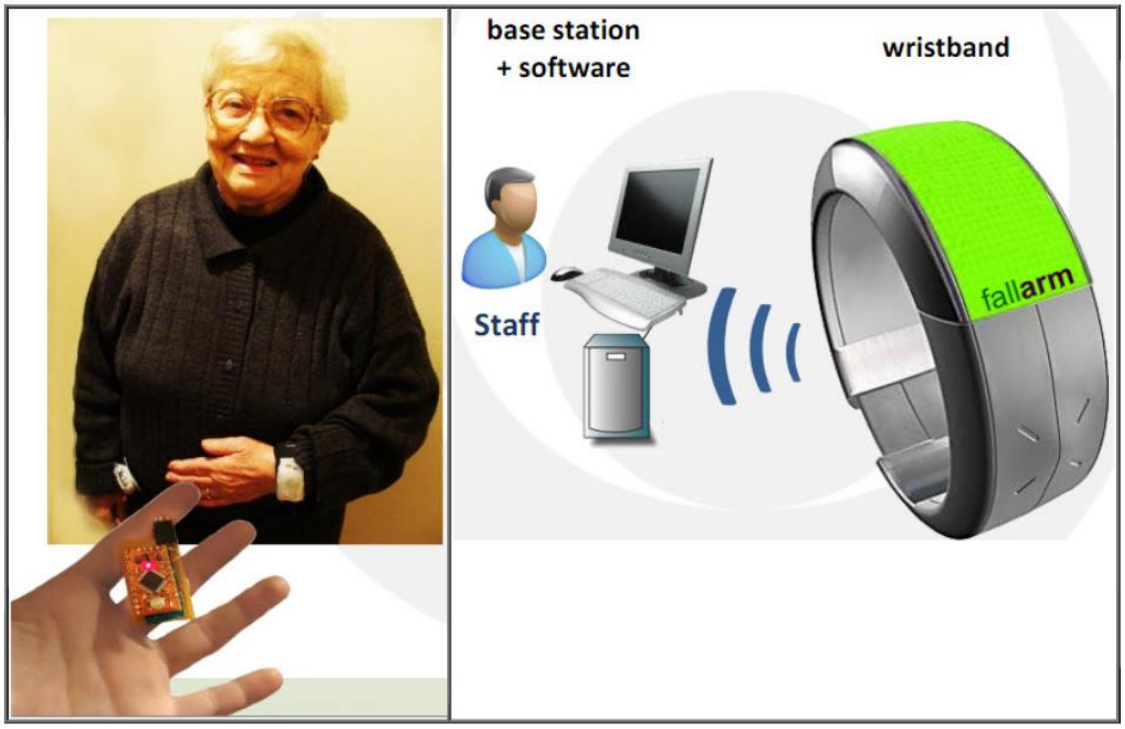
Student Name: Mahmud Omer ID: 19660

Week3_hw1

25. Project Falling Detection: Python + kNN + Colab




Process

1. Understand the project
    - Most mobile devices are equipped with different kind of sensors.
    - We can use the data sent from Gyroscope senso and Accelerometer sensor to categorize any motion:
        - 3 numbers from Accelerometer sensor
        - 3 numbers from Gyroscope sensor
    - References
        - A Review on Fall Prediction and Prevention System for Personal Devices: Evaluation and Experimental Results
        - Andorid FallArm Project
            - Sensors
                - Difference Between an Accelerometer and a Gyroscope
2. Use this heuristic to decide the value of K

    - The choice of K equal to the odd number cloest 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.

    - References
        - How can we find the optimum K in K-Nearest Neighbor?
3. Using KNN to manually calculate the distance and predict the result.
    - The first 8 rows are one set of training data, including Accelerometer Data and Gyroscope Data. The last row is for prediction.

| Accelerometer Data | | | Gyroscope Data | | | Fall (+), Not (-) |
|---|---|---|---|---|---|---|
| x | y | z | x | y | z | +/- |
| 1 | 2 | 3 | 2 | 1 | 3 | - |
| 2 | 1 | 3 | 3 | 1 | 2 | - |
| 1 | 1 | 2 | 3 | 2 | 2 | - |
| 2 | 2 | 3 | 3 | 2 | 1 | - |
| 6 | 5 | 7 | 5 | 6 | 7 | + |
| 5 | 6 | 6 | 6 | 5 | 7 | + |
| 5 | 6 | 7 | 5 | 7 | 6 | + |
| 7 | 6 | 7 | 6 | 5 | 6 | + |
| 7 | 6 | 5 | 5 | 6 | 7 | ?? |

4. Use Python to implement the application of using kNN to predict fall.
    - Create the code by modifying KNN from scratch
        - Explain the code
    - Run the code on Colab
        - References
            - Get Start with Colab

    a. References
        - Python for KNN
            - - Santhi Sree Nagalla, 2021 Summer
                - Knn.ipynb - Python code using Scikit-learn library
                - Knn_Python.ipynb - Python Program to predict kNN Value
            - Develop k-Nearest Neighbors in Python From Scratch
            - Quan Zhou - Fall, 2019
            - Wijian Xiong - Fall, 2019
            - Juilee Panse - Fall, 2019

5. Comparing the result from the Python program and the result of manual calculation.
6. Adding the project to your portofolio
        - Please use Google Slides to document the project
        - Please link your presentation on GitHub using this structure
            -
            - Machine Learning
            - - Supervised Learning
            - + Falling Prediction using KNN

    - Submit
        - The URLs of the Google Slides and GitHub web pages related to this project.
        - A PDF file of your Google Slides

Solution:

1. Use this heuristic to decide the value of K:

   $K = \sqrt{N}$ , where n = number of instances = 9

   $K = \sqrt{8} \cong 3$

2. Using KNN to manually calculate the distance and predict the result

$$Da^2 = (d^{xta} - d^{xa})^2 + (d^{yta} - d^{ya})^2 + (d^{zta} - d^{za})^2$$
$$Dg^2 = (d^{xta} - d^{xg})^2 + (d^{yta} - d^{ya})^2 + (d^{zta} - d^{za})^2$$

   The distances which are calculated with excel sheet are given below.

| Accelerometer Data | | | Gyroscope Data | | | Da | Dg | Fall (+), Not (-) |
|---|---|---|---|---|---|---|---|---|
| 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 | | | + |

For K = 3, 3 closest instances are chosen, which are 4, 5 and 6 for Dist. Acce which represent Fall(+) and 0, 2 and 2 for Dist. Gyro which are again +/Falling. Hence, the the query instance is plus **(+).**

3. Use Python to implement the application of using kNN to predict fall

```
[35] 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
```

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

Choose Files knn_data_sample.csv

- **knn_data_sample.csv**(text/csv) - 152 bytes, last modified: 2/7/2023 - 100% done

```
Saving knn_data_sample.csv to knn_data_sample (6).csv
```

```
[37] data = pd.read_csv("knn_data_sample.csv")
     print(data.loc[:, :"FallOrNot"])
```

```
   x1  y1  z1  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         +
```

```
[38] 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"])
```

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

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


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

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


```
[41] model = KNeighborsClassifier(n_neighbors=3)
```

```
[42] model.fit(x_train, y_train)
```

```
KNeighborsClassifier(n_neighbors=3)
```

```
[43] model.score(x_test, y_test)
```

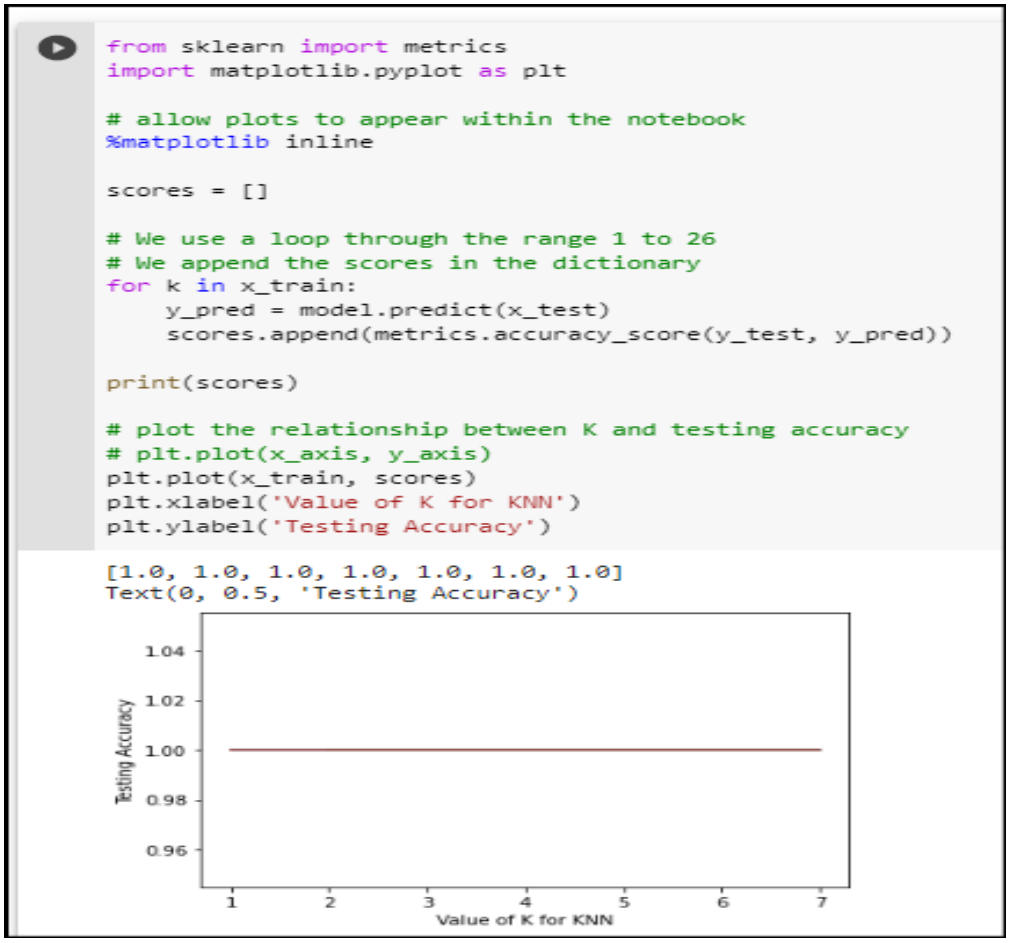
```
1.0
```

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

```
['+']
```

As its highlighted in yellow in the code above the predicted answer using the python code for KNN is the same as the one obtained manually, which is +

The screenshot below checks for the optimal K value, and as it can be seen all values of k from 1 to 7 generate the same accuracy of classification.



```
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
for k in x_train:
    y_pred = model.predict(x_test)
    scores.append(metrics.accuracy_score(y_test, y_pred))

print(scores)

# plot the relationship between K and testing accuracy
# plt.plot(x_axis, y_axis)
plt.plot(x_train, scores)
plt.xlabel('Value of K for KNN')
plt.ylabel('Testing Accuracy')
```

```
[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
Text(0, 0.5, 'Testing Accuracy')
```



Google slides and GitHub Link

https://docs.google.com/presentation/d/1D0Eo_KUxQG6bOB5M4KofjIlsujM1RvEik_G9UPXOq6A/edit?usp=sharing

https://github.com/momer22/Machine-Learning-Supervised-Learning-Falling-Prediction-using-KNN