Project Falling Detection: Python + kNN + Colab

2. The value of K = The odd number closet to the square root of the number of instances.

= √8

= 2.8284

= 3

3.

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**	**??**

Accelerometer Data:

(7 – 1)^2 + (6 – 2)^2 + (5 – 3)^2 = 56

(7 – 2)^2 + (6 – 1)^2 + (5 – 3)^2 = 54

(7 – 1)^2 + (6 – 1)^2 + (5 – 2)^2 = 70

(7 – 2)^2 + (6 – 2)^2 + (5 – 3)^2 = 45

(7 – 6)^2 + (6 – 5)^2 + (5 – 7)^2 = 6 +

(7 – 5)^2 + (6 – 6)^2 + (5 – 6)^2 = 5 +

(7 – 5)^2 + (6 – 6)^2 + (5 – 7)^2 = 8

(7 – 7)^2 + (6 – 6)^2 + (5 – 7)^2 = 4 +

Accelerometer Data for (7, 6, 5) should be +

Gyroscope Data:

(5 - 2)^2 + (6 - 1)^2 + (7 – 3)^2 = 50

(5 - 3)^2 + (6 - 1)^2 + (7 – 2)^2 = 54

(5 - 3)^2 + (6 - 2)^2 + (7 – 2)^2 = 45

(5 - 3)^2 + (6 - 2)^2 + (7 – 1)^2 = 56

(5 - 5)^2 + (6 - 6)^2 + (7 – 7)^2 = 0 +

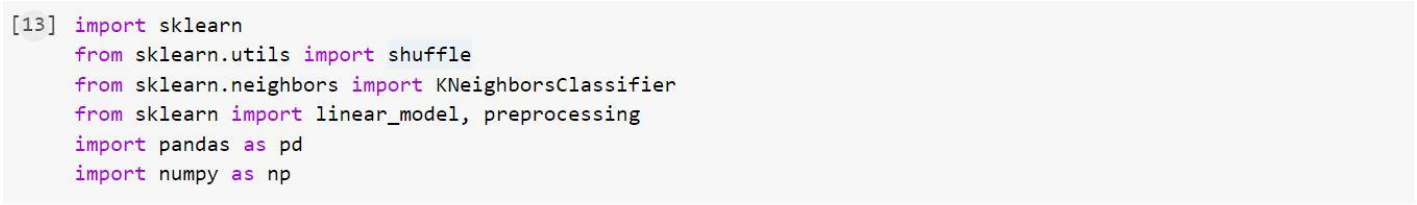
(5 - 6)^2 + (6 - 5)^2 + (7 – 7)^2 = 2 +

(5 - 5)^2 + (6 - 7)^2 + (7 – 6)^2 = 2 +

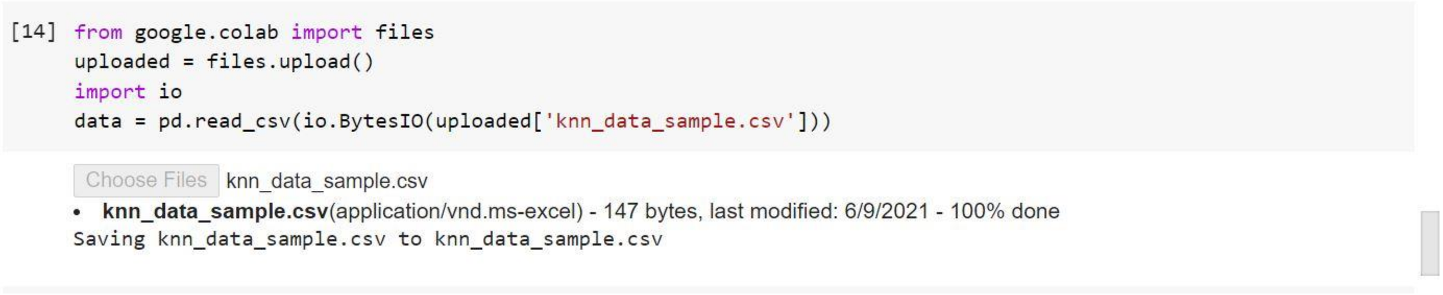
(5 - 6)^2 + (6 - 5)^2 + (7 – 6)^2 = 3

Gyroscope Data for (5, 6, 7) should be +

4. Use Python to implement the application of using kNN to predict falling.



```
[13] 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
```

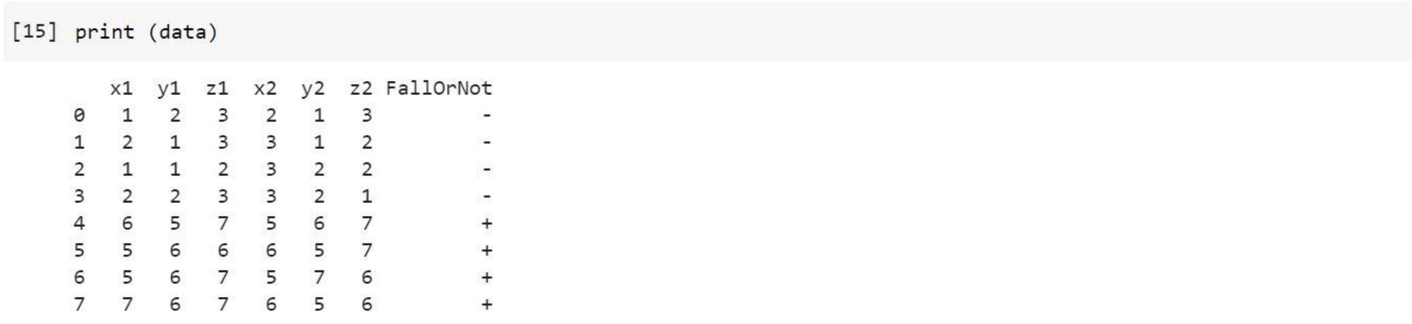


```
[14] from google.colab import files
     uploaded = files.upload()
     import io
     data = pd.read_csv(io.BytesIO(uploaded['knn_data_sample.csv']))
```

Choose Files knn_data_sample.csv

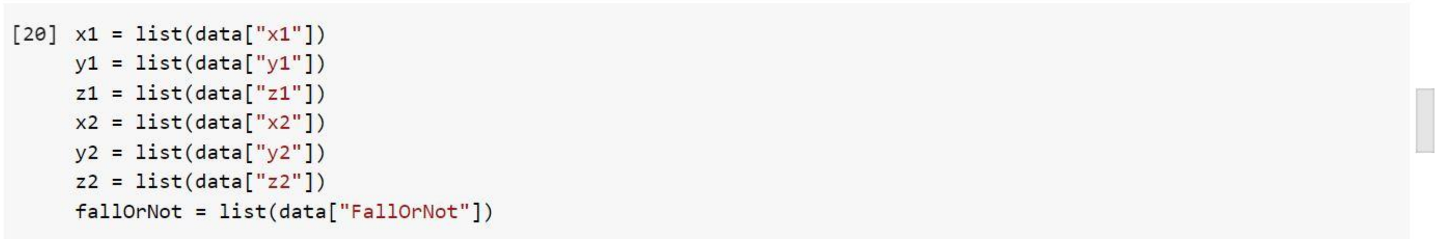
- **knn_data_sample.csv**(application/vnd.ms-excel) - 147 bytes, last modified: 6/9/2021 - 100% done

```
Saving knn_data_sample.csv to knn_data_sample.csv
```

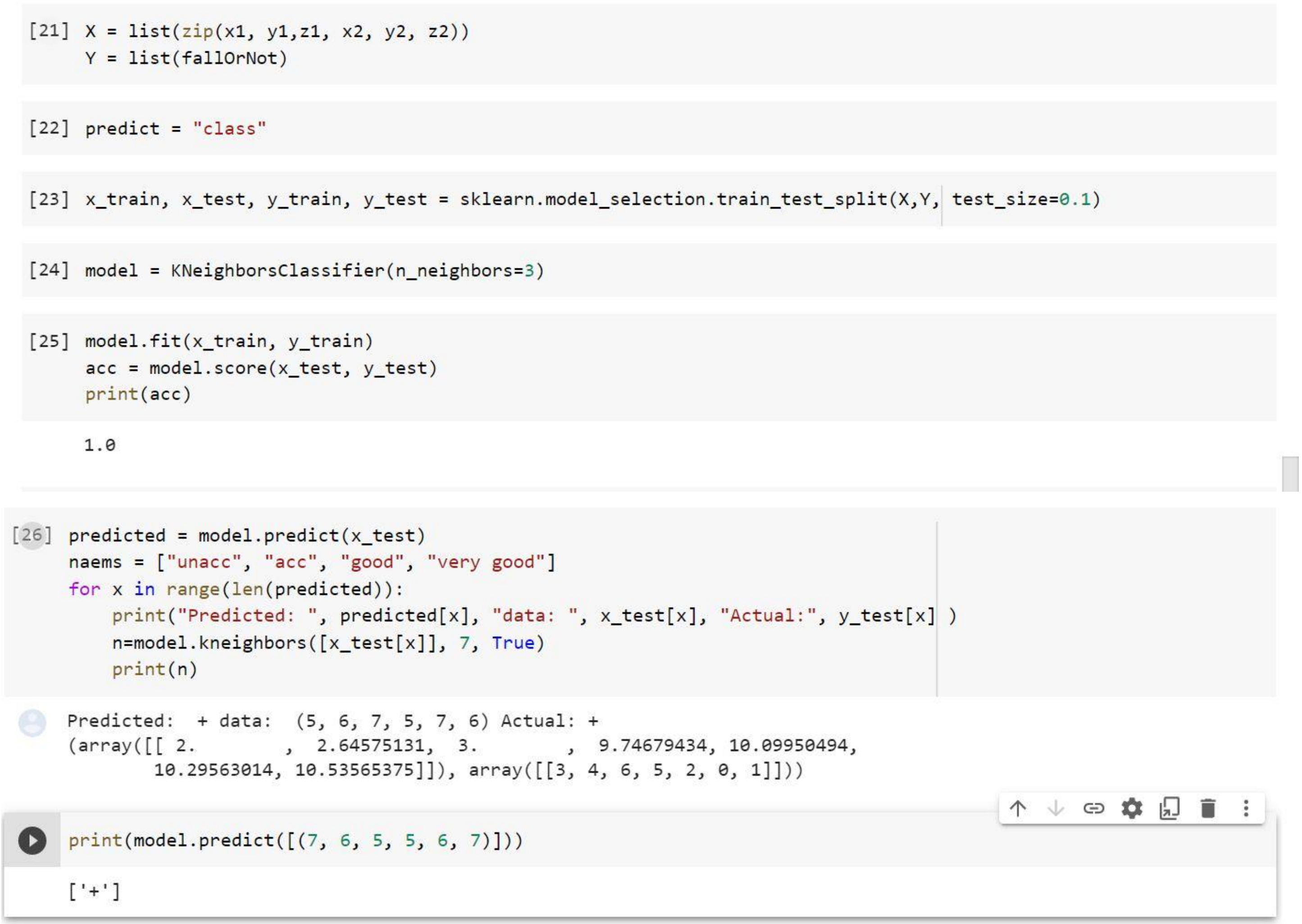


```
[15] print (data)
```

```
   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         +
```



```
[20] 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"])
```



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

```
[22] predict = "class"
```

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

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

```
[25] model.fit(x_train, y_train)
     acc = model.score(x_test, y_test)
     print(acc)
```

```
1.0
```

```
[26] predicted = model.predict(x_test)
     naems = ["unacc", "acc", "good", "very good"]
     for x in range(len(predicted)):
         print("Predicted: ", predicted[x], "data: ", x_test[x], "Actual:", y_test[x] )
         n=model.kneighbors([x_test[x]], 7, True)
         print(n)
```

```
Predicted:  + data:  (5, 6, 7, 5, 7, 6) Actual: +
(array([[ 2.        ,  2.64575131,  3.        ,  9.74679434, 10.09950494,
        10.29563014, 10.53565375]]), array([[3, 4, 6, 5, 2, 0, 1]]))
```

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

```
['+']
```

5. Comparing the result from the Python program and the result of manual calculation.

The Python program result from Colab match with the predication of hand calculation.