Phatcharapon Jumruspun 6031040921

Activity X – Biometrics

Experiment steps:

1. Create a python program to track user's input of the word "Hello" and create a digraph from two consecutive keys. Then write the collected data into .csv file

```
from pynput import keyboard
import time, csv, sys

prior = ''
prior_time = -1
t0 = time.time()
g = dict()
final_res = dict()
data_file_name = "data.csv"
name = sys.argv[1]

def find_avg(g):
    for k, v in g.items():
        g[k] = sum(v)/len(v)
    return g

def writeCsv(g):
    global data_file_name, name
    with open(data_file_name, 'a', newline='') as file:
        w = csv.writer(file)
        ls = []
        for key, value in g.items():
            ls.append(value)
        transpose = list(map(list, zip(*ls)))
        print(transpose)
        for e in transpose:
            w.writerow(e + [name])

def on_press(key):
    global prior, t0, mode, prior_time, final_res
    try:
        if len(prior) == 0:
            g.clear()
            recorded = time.time() - t0
            prior_time = recorded
        else:
            recorded = time.time() - t0
            if (prior, key.char) not in g:
                g[(prior, key.char)] = [recorded - prior_time]
```

```
            else:
                g[(prior, key.char)].append(recorded - prior_time)
            prior_time = recorded
        prior = key.char
    except AttributeError:
        if key == keyboard.Key.space:
            recorded = time.time() - t0
            if (prior, 'space') not in g:
                g[(prior, 'space')] = [recorded - prior_time]
            else:
                g[(prior, 'space')].append(recorded - prior_time)
            prior_time = recorded
            prior = 'space'
        elif(key == keyboard.Key.enter):
            prior = ''
            prior_time = -1
            if len(final_res) == 0:
                for key, value in find_avg(g).items():
                    final_res[key] = [value]
                print(f"\nFinal res = {final_res}")
            else:
                for key, value in find_avg(g).items():
                    final_res[key].append(value)
                print(f"\nFinal res = {final_res}")

        elif(key == keyboard.Key.esc):
            writeCsv(final_res)
            print("Exiting...")
            exit(0)

with keyboard.Listener(
        on_press=on_press) as listener:
    listener.join()
```

collect.py

To use this program:

1.1) “sudo python3 -m collect.py [name]” where the name is your name, to identify the person who is typing and categorize data accordingly
1.2) Input root password
1.3) Once the program starts, type “Hello” (Case sensitive, no double quote) and press enter
1.4) Repeat step 1.2) until you’re satisfied
1.5) Press escape to write the data down to the file “data.csv”

Notes: If you made a typo, please do not press enter, instead just press escape and run the program again (your prior typing records will still be collected). If you accidentally press enter, please press escape and then delete the last record from your data.csv file.

2. Collect the data from myself and my mom

My mom typing “Hello” furiously on my laptop

data.csv | Preview 'data.csv'

0.124451875(	0.139750083	0.105115890	0.233129024	Pon
0.16	0.11	0.13	0.17	Pon
0.15	0.11	0.13	0.17	Pon
0.15	0.09	0.13	0.18	Pon
0.2	0.16	0.15	0.2	Pon
0.12	0.19	0.14	0.2	Pon
0.15	0.13	0.15	0.21	Pon
0.19	0.15	0.16	0.22	Pon
0.18	0.16	0.15	0.19	Pon
0.17	0.13	0.15	0.21	Pon
0.16	0.13	0.13	0.21	Pon
0.98	1.03	0.3	0.41	Noi
0.88	0.83	0.27	1.04	Noi
0.92	1	0.27	0.37	Noi
0.73	0.92	0.3	0.3	Noi
0.87	1.45	0.27	0.36	Noi
1.06	0.87	0.27	0.47	Noi
1.09	1.26	0.28	0.47	Noi
0.95	0.72	0.27	0.41	Noi
1	0.81	0.28	0.52	Noi
0.84	0.77	0.29	0.39	Noi
0.62	0.84	0.24	0.62	Noi
0.86	0.65	0.28	0.37	Noi

```
0.17476892471313477,0.13505887985229492,0.12414407730102539,0.1905360221862793,Pon
0.11726212501525879,0.12685680389404297,0.14102911949157715,0.1881880760192871,Pon
0.15600919723510742,0.1211538314819336,0.1410732269287109 4,0.1796448230743 4082,Pon
0.1539299488067627,0.11615991592407227,0.13494300842285156,0.1759490966796875,Pon
0.1470489501953125,0.11386895179748535,0.13621306419372559,0.1641550064086914,Pon
0.14756226539611816,0.09874582290649414,0.1259441375732422,0.16194891929626465,Pon
0.15011191368103027,0.10184288024902344,0.13637208938598633,0.1749269962310791,Pon
0.15470623970031738,0.10251879692077637,0.13219118118286133,0.17695283889770508,Pon
0.15764498710632324,0.10555911064147949,0.13233208656311035,0.1729259490966797,Pon
0.14981985092163086,0.10617709159851074,0.12786698341369 63,0.16501903533935547,Pon
0.14861798286437988,0.09220576286315918,0.13100719451904297,0.18191289901733398,Pon
0.19791293144226074,0.16382694244384766,0.1506350040435791,0.19840407371520996,Pon
0.12129783630371094,0.1890561580657959,0.1368732452392578,0.2044677734375,Pon
0.1514739990234375,0.1315288543701172,0.14573216438293457,0.20638370513916016,Pon
0.18530941009521484,0.14906692504882812,0.1637587547302246,0.21698832511901855,Pon
0.17737412452697754,0.16488981246948242,0.1488351821899414,0.19176197052001953,Pon
0.16977477073669434,0.1298341751098632 8,0.1542220115661621,0.20678091049194336,Pon
0.16394805908203125,0.1286609172821045,0.13471007347106934,0.21428608894348145,Pon
0.984306812286377,1.029783248901367 2,0.29959797859191895,0.4134659767150879,Noi
0.8809571266174316,0.8271079063415527,0.2679891586303711,1.036715030670166,Noi
0.9152460098266602,1.0031681060791016,0.2666471004486084,0.3720278739929199,Noi
0.7287807464599609,0.9172310829162598,0.29550909996032715,0.3035898208618164,Noi
0.8701901435852051,1.447058916091919,0.27284812927246094,0.36034202575683594,Noi
1.0605649948120117,0.8710441589355469,0.26969289779663086,0.4725780487060547,Noi
```

Final result of data.csv file

3. Train and evaluate the data using scikit-learn KNN

Link to .ipynb file used for KNN:
https://colab.research.google.com/drive/1q2eZxshbg26QfBwot30i2kUX2qpGI5iq?usp=sharing

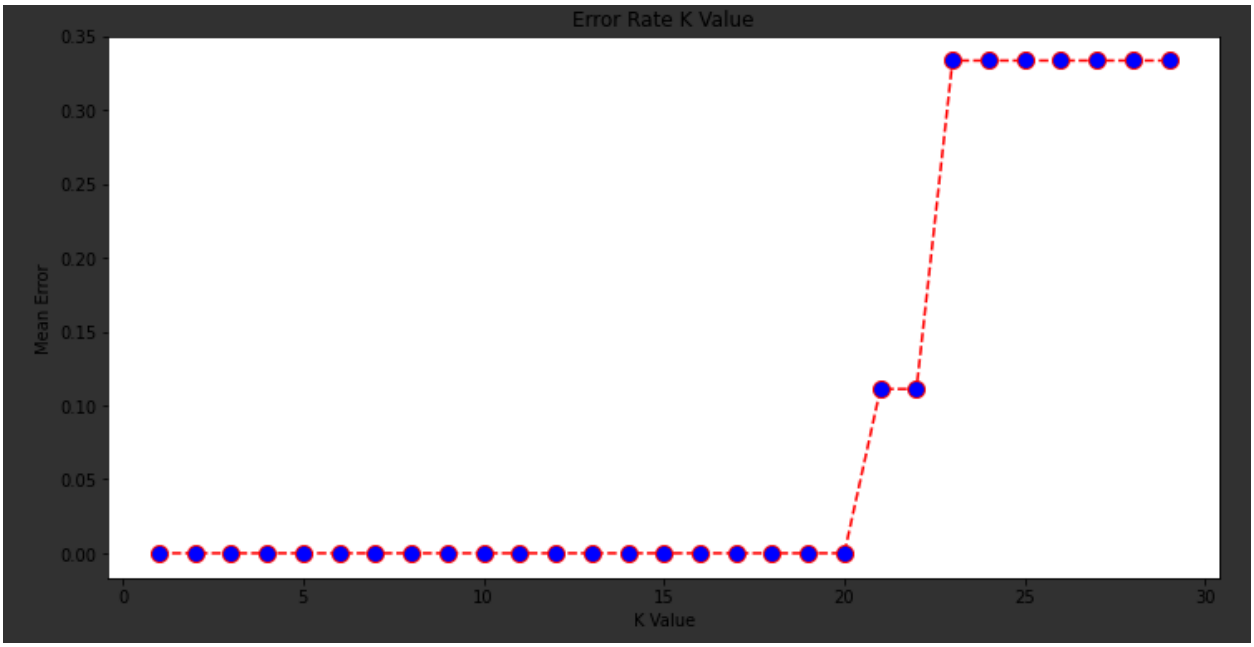
Evaluation: The precision with just two person and 5-letters word is 100% with the best nearest neighbor value being from 1 to 20 (which yields 0% mean error rate)

```
[[3 0]
 [0 6]]
              precision    recall  f1-score   support

         Noi       1.00      1.00      1.00         3
         Pon       1.00      1.00      1.00         6

    accuracy                           1.00         9
   macro avg       1.00      1.00      1.00         9
weighted avg       1.00      1.00      1.00         9
```

Confusion matrix and classification result from the KNN algorithm



K-value and its mean error rate (Optimal at 1-20)

Phatcharapon Jumruspun 6031040921

Questions:

1. How many words do we need to correctly identify the person?

   In this case, I used 41 words (rows) in the dataset, which is sufficient for classifying 2 persons.

2. Do you think this method is scalable? (to thousand persons) for either recognition system or identification system. Please provide your analysis.

   KNN algorithm generally requires exponentially high computing power as the number of data increases. Additionally, we would need a larger word for identification to enable more dimensions, in order to increase spaces for each additional class (person). So, in my opinion this method does not scale well.

   Besides the fact that this method does not scale well, there also might be some errors as the number of classes goes up. Which might lead to more false acceptance and resulting in a failure of the system's security eventually.

3. Will you use this kind of authentication in your system? Please also provide a reason.

   As stated in the answer of the question 2., This method does not scale well and might yield a false acceptance which let unauthenticated people accessing the system, doing unauthorized actions. So I would not use this typing pattern for the authentication in my system.