CURVE_FIT - ALGORITHM

Shivang Chaudhary

VERACITY TRUST NETWORK

Table of Contents

Introduction

The idea around the development of the algorithm is such to contribute to the robustness of the complete veracity machine-learning algorithm. The veracity algorithm was designed with the absolute aim to reduce fraudulent repercussions and improve user-experience on websites. Thus, in turn also benefitting the involved first party company. Bots have been quite nuisance for the business conducted by various websites but they have seemingly evolved much farther in recent times.

Just as veracity focuses on protection of websites from bots, the security design that can be an addition to the complete system also focuses on improving the subjected component by catalysing the detection of bots from humans via extracted data.

Plotting a curve on the extracted coordinates and using that kind of data to further verify whether the extracted data is accurate enough to put the veracity algorithm in the good books of their buyers, is the main purpose of this algorithm. It is therefore capable of producing the required data to check whether the input graph is that of a bot and human through scattering the residuals and setting a threshold on the same graph.

Aims

1. **To input the coordinates and plot a quadratic curve on the generated graph.**

2. **To produce residual elements by scattering the data on separate graph.**

3. **To set a threshold and measure farthest data points.**

4. **Output the results as bot and human.**

Methodology

At first, the development began with producing files for plotting a curve on the given set of coordinates.
The next component was to split the file in individual components.
The third approach was to rotate the plotted graph. This was considered in the case where plotting a curve which has negative gradient or is complex in its nature, will further make the process of plotting the curve on the graph more complex and further on result in producing the complex equations for comparison.

After considering all the individual approaches, the final approach taken towards it was to input the pair of coordinates to build the graph and then plot a quadratic curve on it. Then, using the data gathered create another graph of scattering the residual elements once the list of residuals are created.
Ostensibly, the residual list consists of the same data but in different format. Marking the '0' on the given graph and then setting a threshold on a number on the graph can help in measuring the scattered elements' distance from the threshold.
The residuals list is nothing but the difference of reshaped '*y*' coordinates in *2D* array form and the predicted '*y*' points that are calculated through linear regression.

In the designed algorithm, the threshold is set at '1' and '-1' considering the both ends of the range. If the given dataset has residuals crossing this unit of threshold then it can be deemed human. Otherwise, any other residual falling in lower category can be deemed bot given the simple curve-nature of the generated graph.

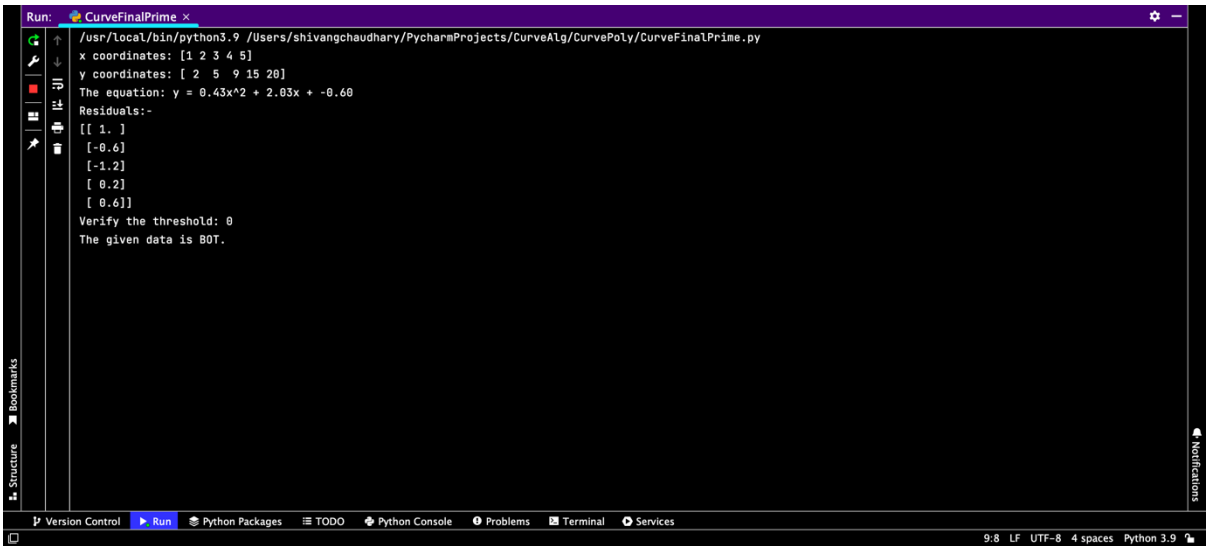
The only manual assistance in the graph is to set the threshold manually on experimented datasets and use the outcomes in the favour.

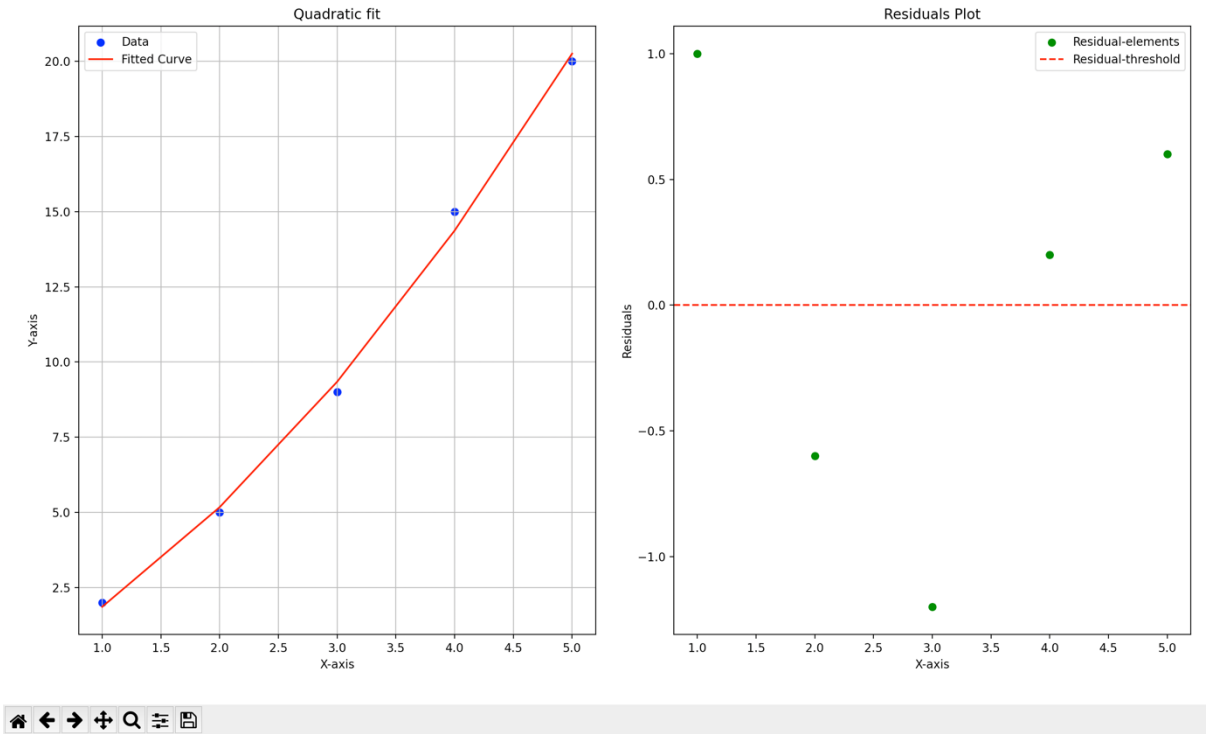
The threshold in the algorithm currently is set considering the tests performed on them till date.

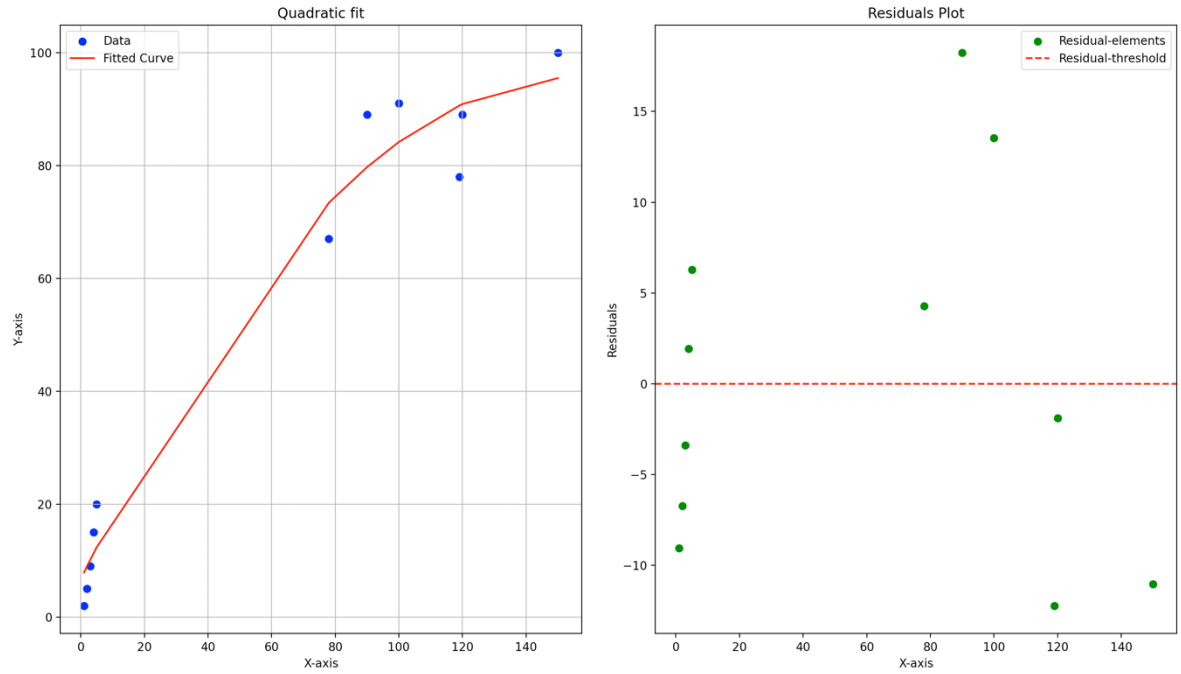
Results

For the following pair of coordinates, are the resultant output of the algorithm to classify the input graph as bot or human.

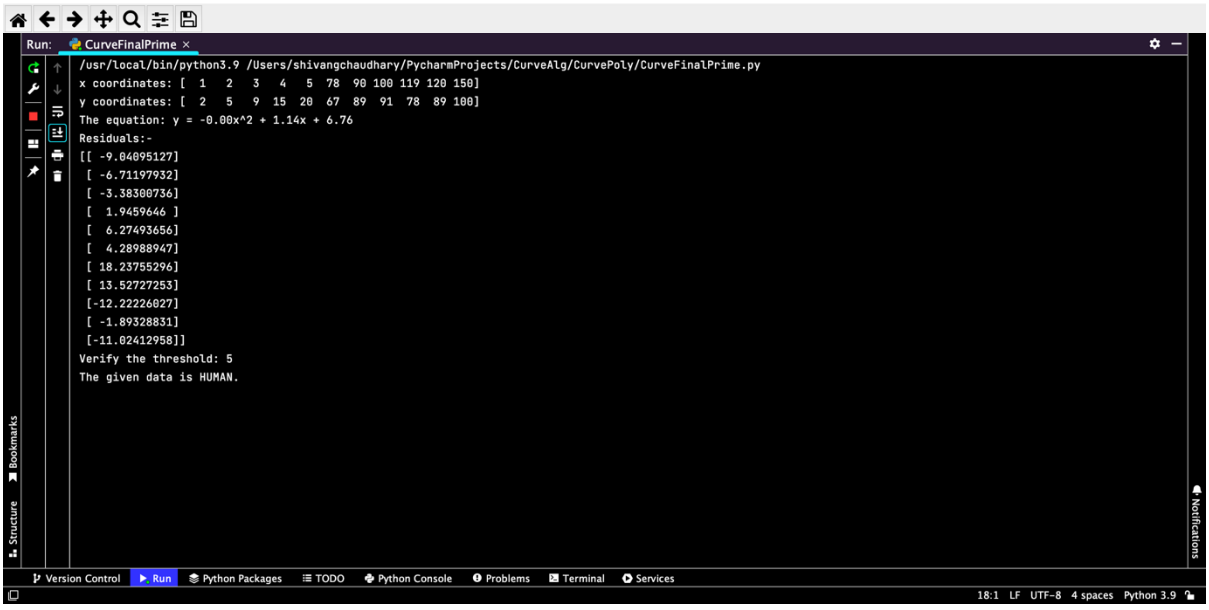
1.

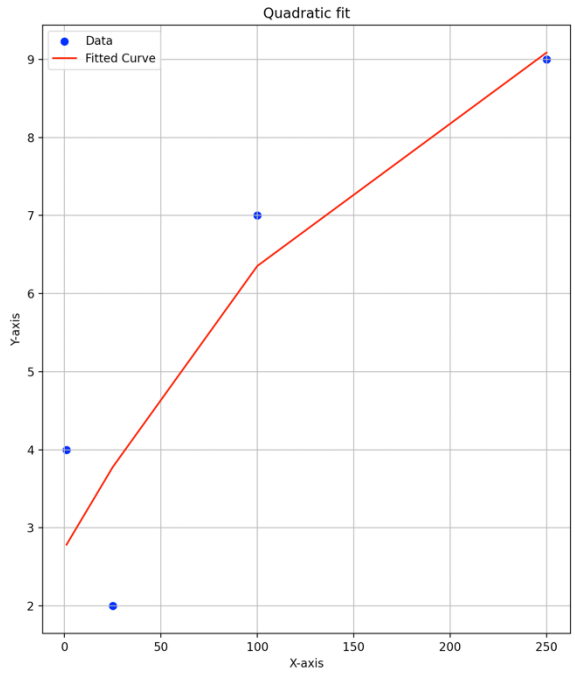


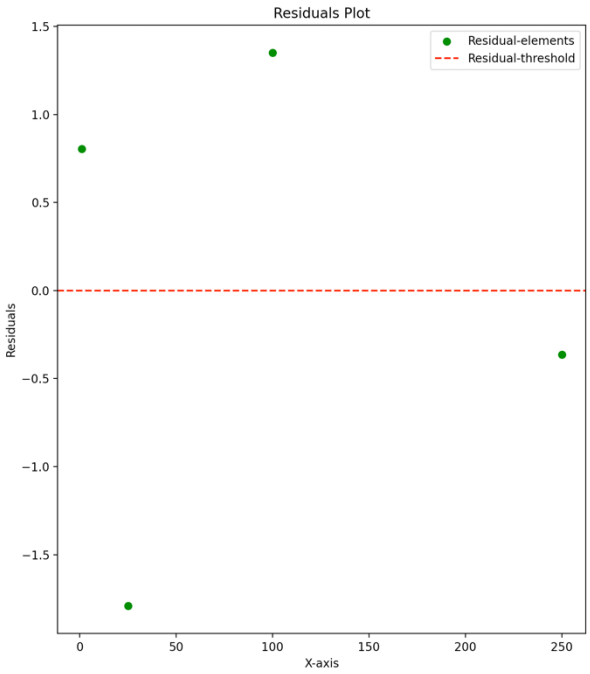



Quadratic fit
Data
Fitted Curve
Y-axis
X-axis
Residuals Plot
Residual-elements
Residual-threshold
Residuals
X-axis

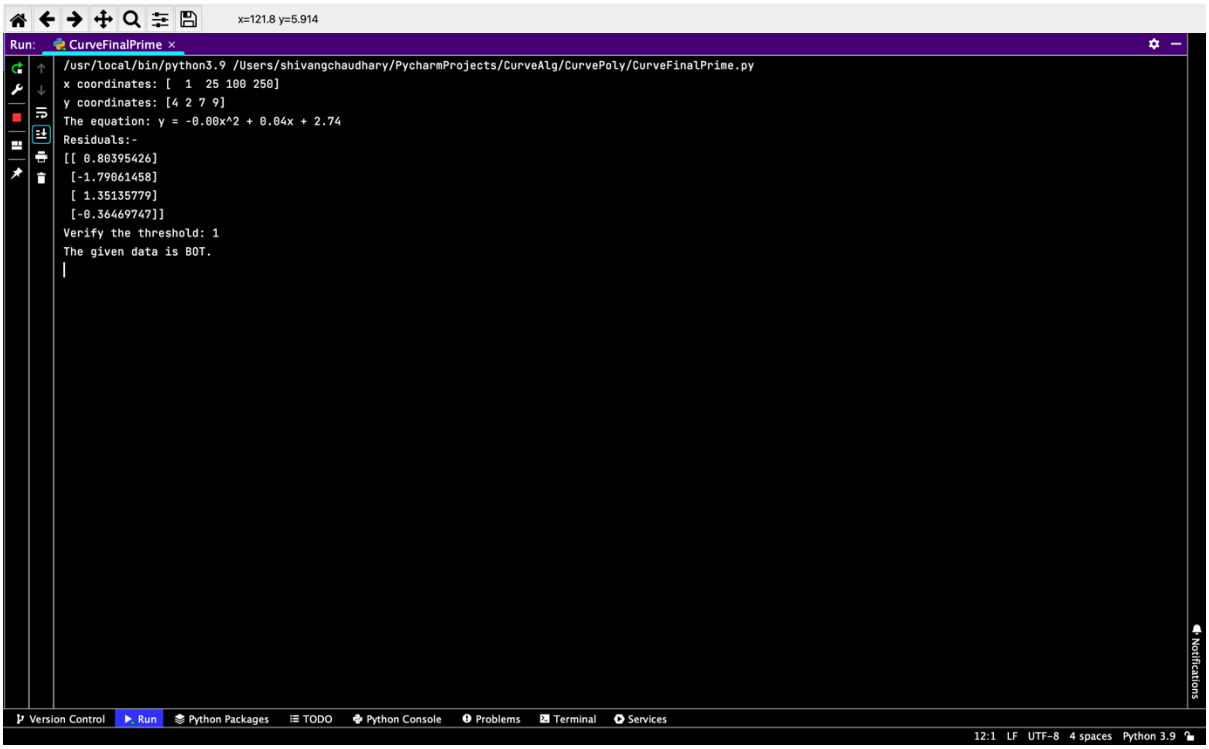
2.

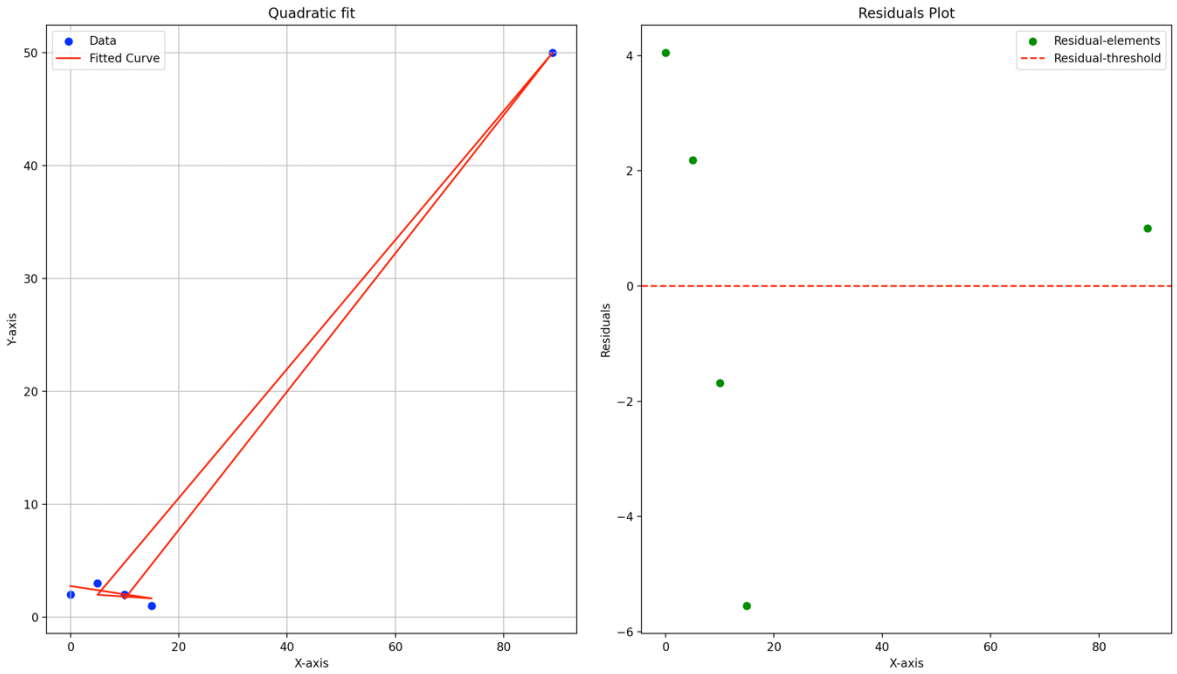

Run: CurveFinalPrime
/usr/local/bin/python3.9 /Users/shivangchaudhary/PycharmProjects/CurveAlg/CurvePoly/CurveFinalPrime.py
x coordinates: [1 2 3 4 5 78 90 100 119 120 150]
y coordinates: [2 5 9 15 20 67 89 91 78 89 100]
The equation: y = -0.00x^2 + 1.14x + 6.76
Residuals:-
[[-9.04095127]
[-6.71197932]
[-3.38300736]
[1.9459646]
[6.27493656]
[4.28988947]
[18.23755296]
[13.52727253]
[-12.22226027]
[-1.89328831]
[-11.02412958]]
Verify the threshold: 5
The given data is HUMAN.
Version Control
Run
Python Packages
TODO
Python Console
Problems
Terminal
Services
18:1 LF UTF-8 4 spaces Python 3.9


Quadratic fit
Data
Fitted Curve
Y-axis
X-axis
0
50
100
150
200
250
2
3
4
5
6
7
8
9

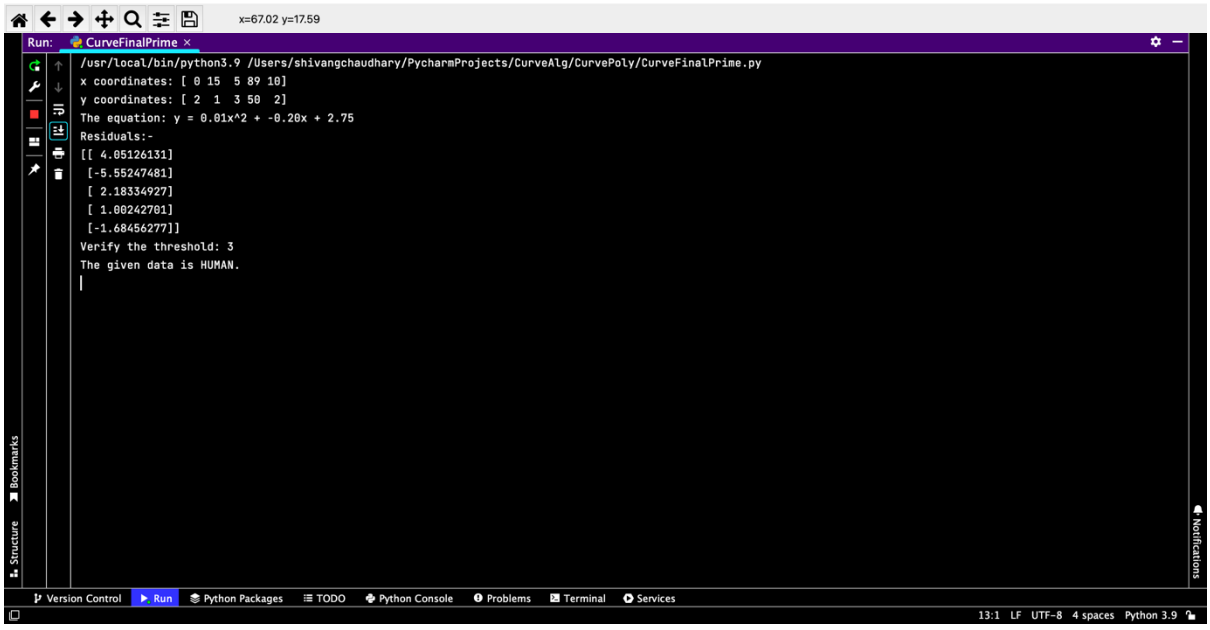

Residuals Plot
Residual-elements
Residual-threshold
Residuals
X-axis
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
0
50
100
150
200
250

3.


x=121.8 y=5.914
Run:
CurveFinalPrime
/usr/local/bin/python3.9 /Users/shivangchaudhary/PycharmProjects/CurveAlg/CurvePoly/CurveFinalPrime.py
x coordinates: [1 25 100 250]
y coordinates: [4 2 7 9]
The equation: y = -0.00x^2 + 0.04x + 2.74
Residuals:-
[[0.80395426]
[-1.79061458]
[1.35135779]
[-0.36469747]]
Verify the threshold: 1
The given data is BOT.
Version Control
Run
Python Packages
TODO
Python Console
Problems
Terminal
Services
Notifications
12:1 LF UTF-8 4 spaces Python 3.9



4.



The input data is used keeping in mind that they be diverse and giving the idea of supposed working of the algorithm. Thus, various kinds of similar datasets can be experiment with.

Conclusion

Arriving at the conclusion, the algorithm is currently capable of plotting a simple quadratic curve for visual verification and then further using residual elements to perfectly classify the input as bot or human.

There were many different approaches discussed while undertaking it's development but none of them seem to provide any immediate solution but have a very good potential in the future uses if needed to further advance the algorithm.

Thus, the graphs available to the user by the end is : -

- Curve plotting on the given graph.
- Residual measuring from a manually set threshold.

Future work

Considering all the approaches that were discussed previously along with their limitations are as follows. They do not prove any immediate need as of now but if needed in future, can be of great use inside the same algorithm with a little modifications regardless of their complex nature.

1. **Chebyshev Polynomial**

   Chebyshev polynomials are a family of orthogonal polynomials with various applications in diverse fields; including numerical analysis, polynomial approximation and interpolation, signal processing, and solving differential equations. They are named after the Russian mathematician Pafnuty Chebyshev and are denoted by *Tn(x)* where *n* is the degree of polynomial. Chebyshev polynomials *Tn(x)* and *Un(x)* of the degree *n* are of the first and second kinds respectively, of the range [-1, 1] of *x*. The functions

   $$T_n(x) = \cos\left(n \cos^{-1} x\right) \tag{1}$$

   or

   $$T_n(\cos\theta) = \cos(n\theta) \tag{2}$$

   for

   $$x = \cos\theta \tag{3}$$

   are called the Chebyshev polynomials of the first kind. They are solutions of the Chebyshev differential equation for non-negative numbers.

   Advantages: -

   - This property ensures that the coefficients of different terms in the polynomial do not interfere with each other, hence simplifying the process of finding the best-fit polynomial and leading to more accurate approximations.

- Chebyshev polynomials exhibit a superiority in polynomial approximation to the Fourier series due to its accuracy.

This method only has few severe challenges when taking the pragmatic perspective such as that it needs to be set a degree for the polynomial and number of terms, which can be used for equations' comparison. Though complex, serves as potential for future advancements (if any).

2. Markov Chain

The continuous valued Markov Chain can also be used for identifying hidden patterns within trajectories. They used the smooth SVM for classification, within the context of identifying whether a user is indeed the user themselves or an intruder.
Given the Markov chain method deals with real time data as it's coming in, it could be a powerful tool in that it can perform the classification as the user moves their cursor, whereas the *Chebyshev* polynomial can only be applied once the user has finished creating their trajectory. Though not of the usage now, the method has potential.

References

- *Lynnette Hui Xian Ng, Dawn C. Robertson, and Kathleen M. Carley. "Stabilizing a supervised bot detection algorithm: How much data is needed for consistent predictions?" In: Online Social Networks and Media 28 (2022), p. 100198. issn: 2468-6964. doi: https://doi.org/10.1016/ j.osnem.2022.100198.*
- *url: https://www.sciencedirect.com/science/article/pii/ S2468696422000027.*

- *Natanael Karjanto. Properties of chebyshev polynomials. arXiv preprint arXiv:2002.01342, 2020.*

- Source file destination ('CurveFinalPrime.py') *: -* *https://github.com/Shiv716/CurveFit_veracity/blob/main/CurveFinalPrime.py*

Contributors

- **Shivang Chaudhary (shivang.chaudhary@vtn.live)**
- **Andrea Hagiu (ge22657@bristol.ac.uk)**
- **Winfred Gatua (winfred.gatua@bristol.ac.uk)**
- **Muheeb Ahmed (muheeb.ahmed@vtn.live)**