

Analysis of GPS receiver information

Resources



<https://clock.zone/> - universal clock time



https://content.ublox.com/sites/default/files/NEO-M8-FW3_DataSheet_UBX-15031086.pdf – UBLOX GPS receiver datasheet

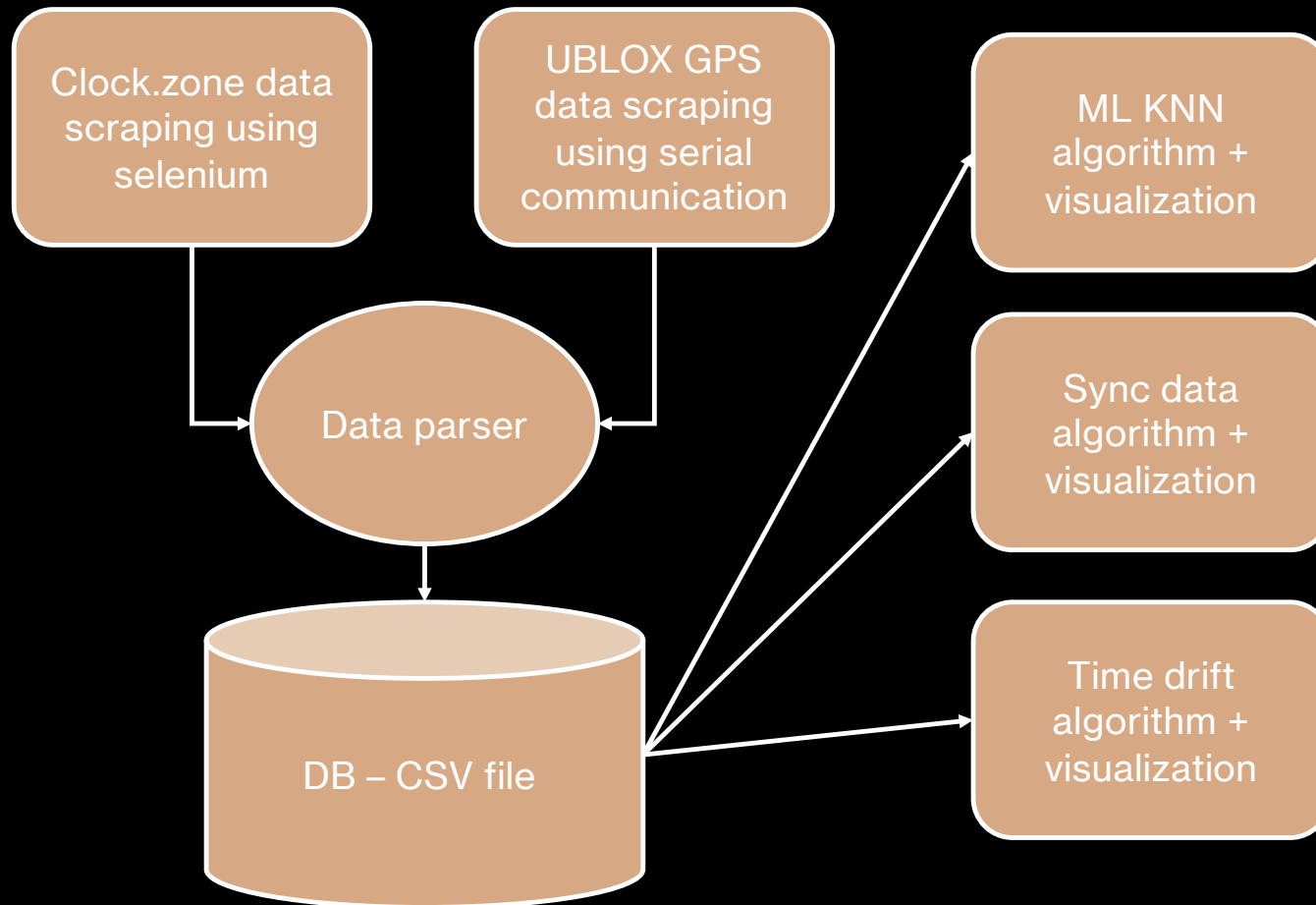


Real-time data reading

Setup

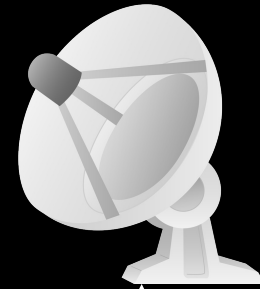
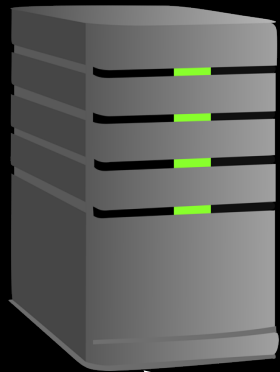


Working method

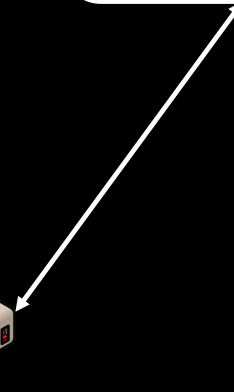
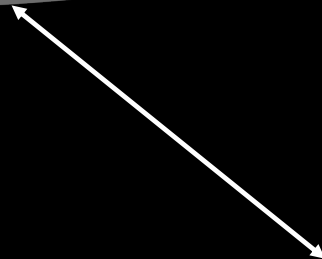


GPS setup

Clock.zone server



UBLOX GPS -
Reciver



Clock.zone Web scraping

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
```

Clock.zone Web scraping

```
with webdriver.Chrome() as driver:
    try:
        remove_file('test.csv')
        driver.get(website_URL)

        clock_element = WebDriverWait(driver,
1).until(EC.visibility_of_element_located((By.ID, 'MyClockDisplay')))
        clock_stats_element = WebDriverWait(driver,
1).until(EC.visibility_of_element_located((By.CLASS_NAME, 'clock-stats')))

        while True:
            current_time = datetime.now()
            web_timestamp = parsing_timestamp(clock_element)
            sync_precision = parsing_sync_precision(clock_stats_element)
            write_to_csv("test.csv", current_time, web_timestamp, sync_precision)
            time.sleep(0.5)

    except KeyboardInterrupt:
        pass
```

UBLOX GPS data scraping - NMEA messages data

\$GNRMC ,122642.00 ,A ,3151.39315 ,N ,03443.70047 ,E ,0.125 , ,010623 , , ,A ,V*1A

\$GNGGA ,122642.00 ,3151.39315 ,N ,03443.70047 ,E ,1,09 ,1.25 ,39.1 ,M ,17.1 ,M , ,*7F

UBLOX GPS data scraping

```
def get_all_ublox_data():
    baudrate = 9600
    serial_port = serial.Serial("/dev/tty.usbmodem112401", baudrate, timeout=1)
    serial_port.isOpen()

    output=''
    for i in range(3):
        ubr = serial_port.readline().decode().strip()
        output+= ubr + '\n'

    timestamp = None
    number_of_sat = None
    fix_quality = None

    for line in output.splitlines():
        splited_line = line.split(',')
        if splited_line[0] == '$GNRMC':
            timestamp = splited_line[1]
        if splited_line[0] == '$GNGGA':
            number_of_sat = splited_line[7]
            fix_quality = splited_line[6]

    return parse_to_UNIX(timestamp) ,fix_quality,number_of_sat
```

KNN - machine learning method

- The machine learning method used in our code is K-Nearest Neighbors (KNN) regression.
- In this case, it is applied for regression to predict the "Drift" variable based on the features "GPS Time" and "clock.zone Time".
- To evaluate the performance of the model, the mean squared error (MSE) and R-squared score (R2) are calculated. MSE measures the average squared difference between the predicted and actual values, with lower values indicating better performance. R2 score measures the proportion of variance in the target variable explained by the model, with higher values indicating a better fit.

KNN accuracy values

Mean Squared Error: 0.1705009182421782
R2 Score: 0.9898681793140732

Simulate GNSS block



Data

1	Current Time,	GPS Time,	clock.zone Time,	Drift	,clock.zone Sync Precision,	Fix Quality,	Number of SAT
2	2023-06-15 18:03:02.792106,	1686830583,	1686798182.78,	- 0.7799999713897705,	0.188,	1,	12
3	2023-06-15 18:03:03.400460,	1686830584,	1686798183.4,	- 0.40000009536743164,	0.188,	1,	12
4	2023-06-15 18:03:04.411855,	1686830585,	1686798184.41,	- 0.4100000858306885,	0.185,	1,	12
5	2023-06-15 18:03:05.413151,	1686830586,	1686798185.41,	- 0.4100000858306885,	0.185,	1,	12
6	2023-06-15 18:03:06.406565,	1686830587,	1686798186.4,	- 0.40000009536743164,	0.185,	1,	12
7	2023-06-15 18:03:07.405540,	1686830588,	1686798187.4,	- 0.40000009536743164,	0.188,	1,	12
8	2023-06-15 18:03:08.414132,	1686830589,	1686798188.41,	- 0.4100000858306885,	0.188,	1,	12
9	2023-06-15 18:03:09.410086,	1686830590,	1686798189.41,	- 0.4100000858306885,	0.188,	1,	12
10	2023-06-15 18:03:10.404372,	1686830591,	1686798190.4,	- 0.40000009536743164,	0.183,	1,	12
11	2023-06-15 18:03:11.409282,	1686830592,	1686798191.41,	- 0.4100000858306885,	0.183,	1,	12
12	2023-06-15 18:03:12.409332,	1686830593,	1686798192.41,	- 0.4100000858306885,	0.183,	1,	12
13	2023-06-15 18:03:13.404985,	1686830594,	1686798193.4,	- 0.40000009536743164,	0.185,	1,	12
14	2023-06-15 18:03:14.399871,	1686830595,	1686798194.39,	- 0.3900001049041748,	0.185,	1,	12
15	2023-06-15 18:03:15.409504,	1686830596,	1686798195.4,	- 0.40000009536743164,	0.185,	1,	12
16	2023-06-15 18:03:16.415719,	1686830597,	1686798196.41,	- 0.4100000858306885,	0.188,	1,	12
17	2023-06-15 18:03:17.398652,	1686830598,	1686798197.39,	- 0.3900001049041748,	0.188,	1,	12
18	2023-06-15 18:03:18.400469,	1686830599,	1686798198.39,	- 0.3900001049041748,	0.188,	1,	12
19	2023-06-15 18:03:19.396711,	1686830600,	1686798199.38,	- 0.38000011444091797,	0.2,	1,	12
20	2023-06-15 18:03:20.396451,	1686830601,	1686798200.39,	- 0.3900001049041748,	0.2,	1,	12
21	2023-06-15 18:03:21.406996,	1686830602,	1686798201.4,	- 0.40000009536743164,	0.2,	1,	12
22	2023-06-15 18:03:22.411309,	1686830603,	1686798202.4,	- 0.40000009536743164,	0.186,	1,	12
23	2023-06-15 18:03:23.409954,	1686830604,	1686798203.41,	- 0.4100000858306885,	0.186,	1,	12
24	2023-06-15 18:03:24.406002,	1686830605,	1686798204.4,	- 0.40000009536743164,	0.186,	1,	12

EDA

