Лабораториска вежба бр.1

Потребниот data set за оваа лабораториска вежба го преземав од страницата :

https://data.melbourne.vic.gov.au/Environment/Sensor-readings-with-temperature-light-humidity-ev/ez6b-syvw/data

Мерењата се направени во склоп на истражување при Универзитетот на Мелбурн за креирање на Smart city. Имено поставиле real-time сензори низ урбаната средина кои правеле мерење на светлина, температура и влажност со цел да ја видат нивната зависност во однос на тоа каде се наоѓаат, поточно дали овие вредности многу зависат од близината до шумски простри. Мерењата се направени во период од 15.12.2014 до 4.6.2015, на секои 10 минути.

Од мерењата ја земав колоната во која се претставени мерењата за температурата.

Во продолжение се претставени слики од кодот кој го напишав за соодветните алгоритми за обработка на податоците. Бидејќи мерната единица е °C, соодветно за threshold опсег избрав 3 степени со чекор 0.2

[0, 0.2, 0.4, 0.6, 0.8, 1, 1.2, 1.4, 1.6, 1.8, 2, 2.2, 2.4, 2.6, 2.8, 3]

```
list = []
with open('sensor.csv', 'r') as file:
    reader = csv.reader(file)
    next(file)
    for row in reader:
        list.append(row[4])
    list_freq3 = []
    list_freq5 = []
    for i in range(0, len(list), 3):
        list_freq3.append(list[i])
    for i in range(0, len(list), 5):
        list_freq5.append(list[i])

    # threshoold 3 stepeni
    threshold = [0, 0.2, 0.4, 0.6, 0.8, 1, 1.2, 1.4, 1.6, 1.8, 2, 2.2, 2.4, 2.6, 2.8, 3]
```

Јана Ѓелевска 171084

Кодот за трите алгоритми, соодветно во продолжение истите се применети над мерењата кои се семплирани со различни фрекфеници. Како пример ги земав фрекфенција 3 и фрекфенција 5 (секое 3то и 5то мерење соодветно). Прикажани се кодовите за имплементација на алгоритмите на несемплирани податоци по фрекфенција, за различна фрекфенција е користен истиот код само на различно множество податоци.

LA

```
# LA
list_sent = []
list_MSE = []
for t in threshold:
    # graf algoritam
    sent = 1
    list_values = []
    list_values.append(list[0])
    # graf MSE
    MSE = 0
    for m in range(1, len(list)):
        if math.fabs(float(list[m]) - float(list_values[m - 1])) >= t:
            sent += 1
            list_values.append(list[m])
        else:
            list_values.append(list_values[m - 1])
        MSE =float(MSE)+ pow(float(list[m]) - float(list_values[m]), 2)

    list_MSE.append(float(MSE) / len(list))
    list_sent.append(sent * 100 / len(list))
```

MA(2)

```
# MA(2)
list_sent1 = []
list_MSE1 = []
for t in threshold:
    # algoritam
    list_values1 = []
    list_values1.append(list[0])
    list_values1.append(list[1])
    sent1 = 2
    # MSE graf
    MSE1 = 0
    for k in range(2, len(list)):
        avg = round(float((float(list_values1[k - 2]) + float(list_values1[k - 1])) / 2), 1)
        if math.fabs(avg - float(list[k])) >= t:
            sent1 += 1
            list_values1.append(list[k])
        else:
            list_values1.append(avg)
        MSE1=float(MSE1)+ pow(float(list[k]) - float(list_values1[k]), 2)
    list_MSE1.append(float(MSE1) / len(list))

    list_sent1.append(sent1 * 100 / len(list))
```

MA(3)

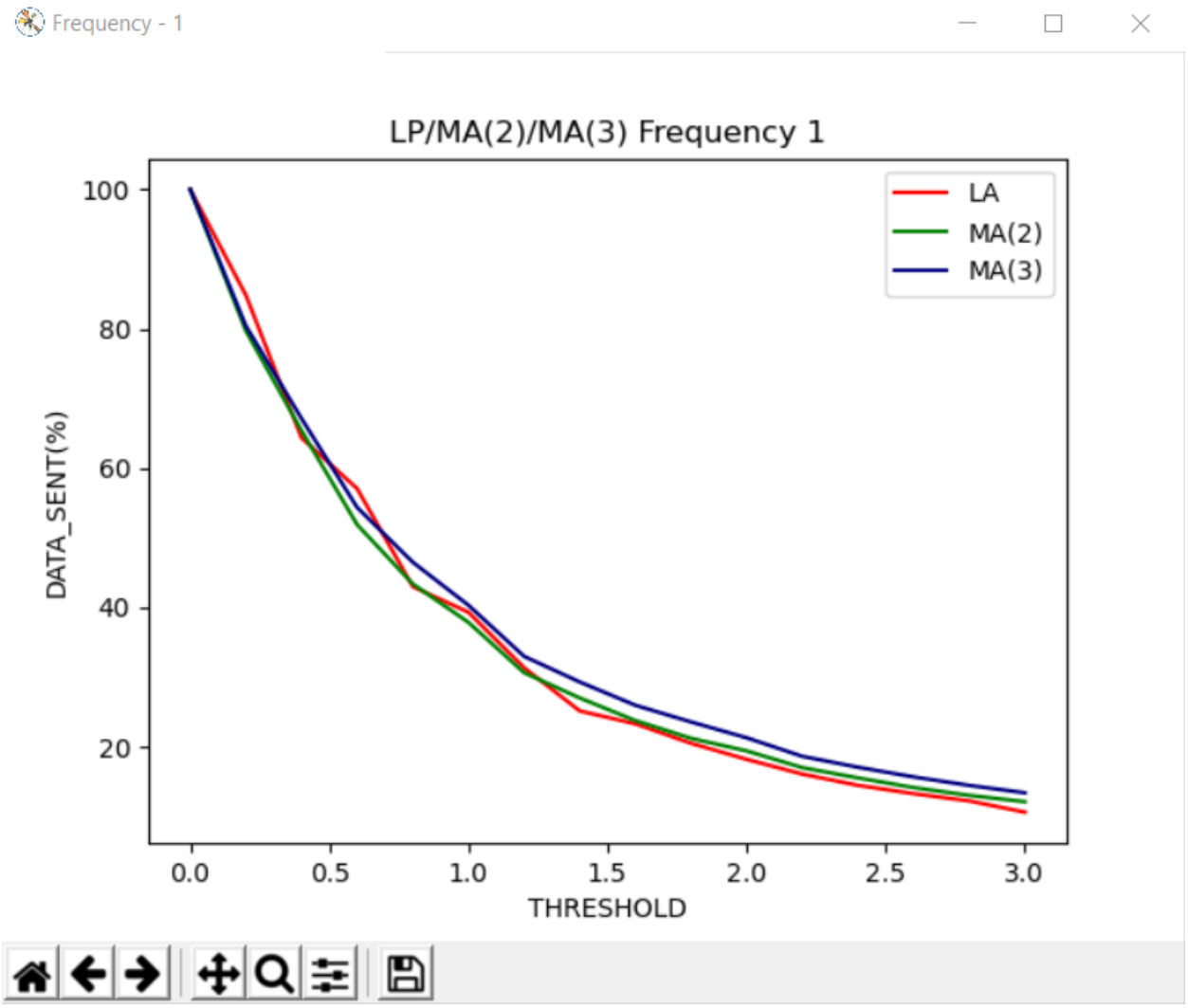
```
#MA(3)
list_sent2 = []
list_MSE2 = []
for t in threshold:
    # algoritam
    list_values1 = []
    list_values1.append(list[0])
    list_values1.append(list[1])
    list_values1.append(list[2])
    sent1 = 3
    # MSE graf
    MSE2 = 0
    for k in range(3, len(list)):
        avg = round(float((float(list_values1[k-3])+float(list_values1[k - 2]) + float(list_values1[k - 1])) / 3), 1)
        if math.fabs(avg - float(list[k])) >= t:
            sent1 += 1
            list_values1.append(list[k])
        else:
            list_values1.append(avg)
        MSE2 = float(MSE2) + pow(float(list[k]) - float(list_values1[k]), 2)
    list_MSE2.append(float(MSE2) / len(list))

    list_sent2.append(sent1 * 100 / len(list))
```

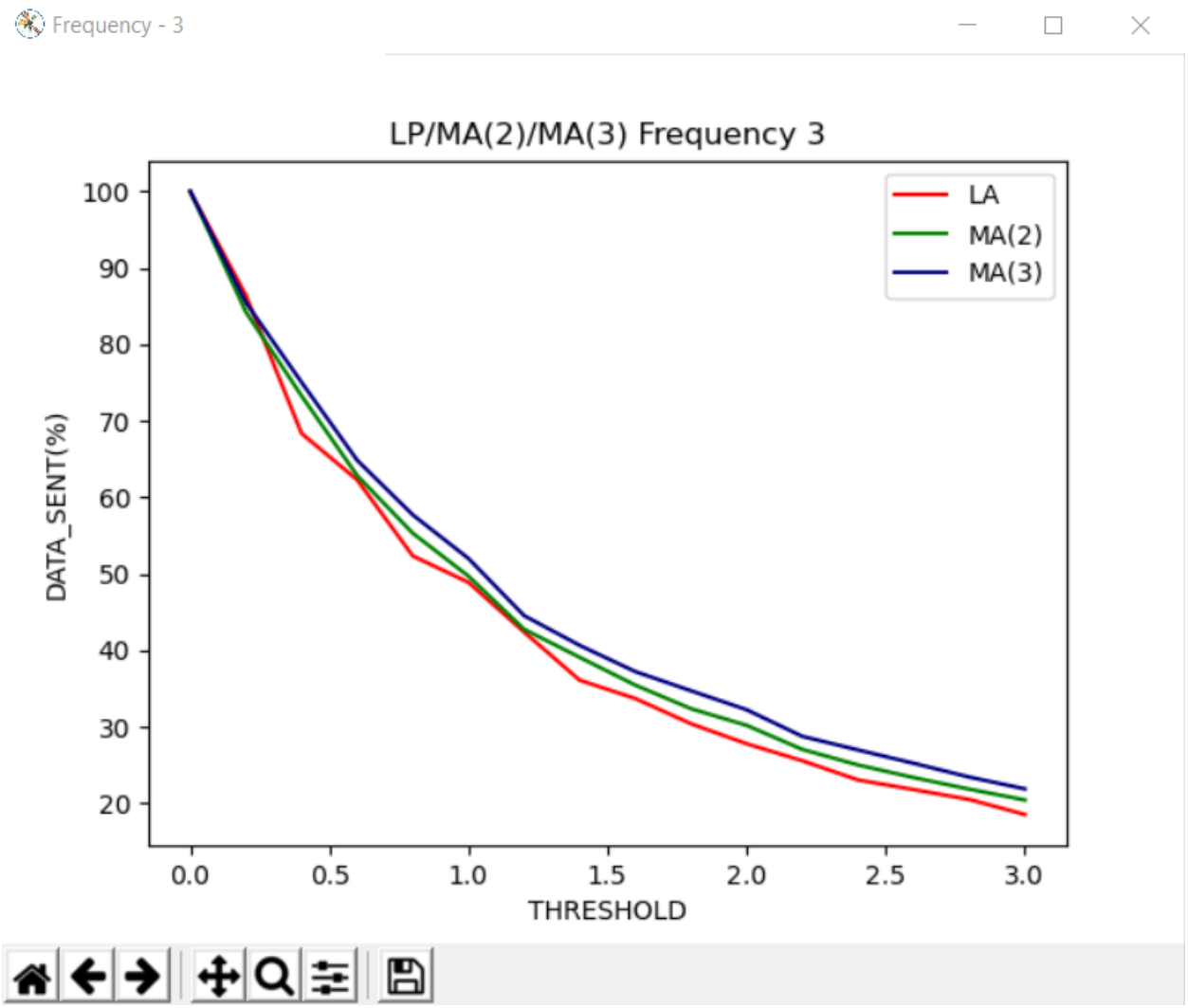
На крај графиците ги исцртав со помош на библиотеката matplotlib.pyplot во python,во продолжение е прикажан кодот за еден од графиците. Во продолжение има коментари за соодветните графици во кои за подобар алгоритам се смета тој што праќа помалку податоци.

```
# x oska ->threshold
# y oska ->list_sent i list_sent1 i list_sent_2
plt.figure(num='Frequency - 1 ')
plt.plot(threshold, list_sent, label='LA', color='red')
plt.plot(threshold, list_sent1, label='MA(2)', color='green')
plt.plot(threshold, list_sent2, label='MA(3)', color='navy')
plt.legend()
plt.xlabel('THRESHOLD')
plt.ylabel('DATA_SENT(%)')
plt.title('LP/MA(2)/MA(3) Frequency 1')
plt.show()
```

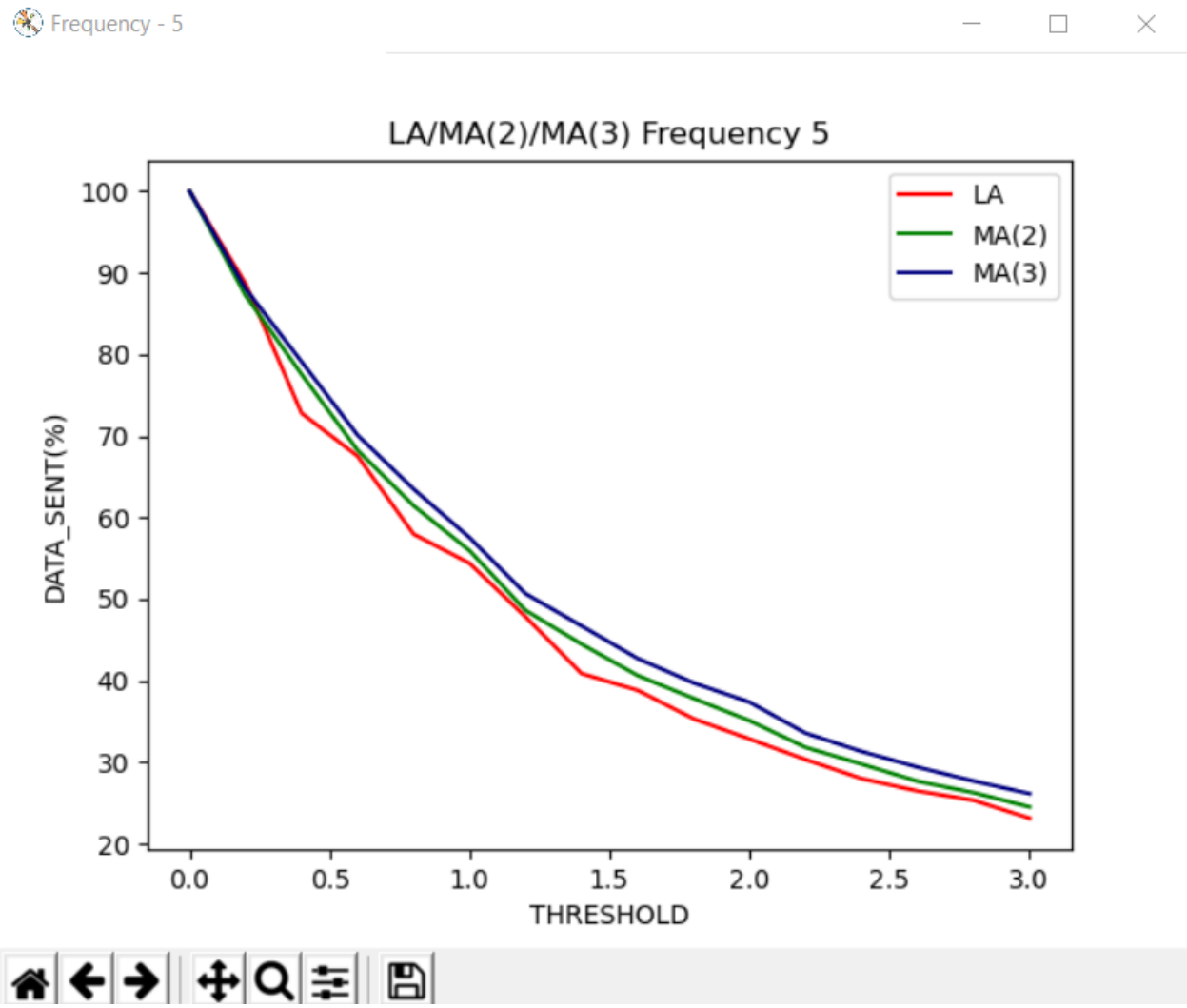
Во овој случај разликата не е многу воочлива, сепак при голем threshold се воочува дека MA(3) е многу подобар од останатите два, исто може да се воочи и дека MA(2) е за нијанса подобар од LA, сепак за мали вредности на threshold алгоритмите праќаат приближно ист процент на податоци .



Во овој случај со фрекфенција 3, повоочлива е разликата дури и за помали вредности на threshold, истото може да се воочи дека MA(3) е најдобар, додека пак тука се воочува подобро дека MA(2) е подобар од LA.

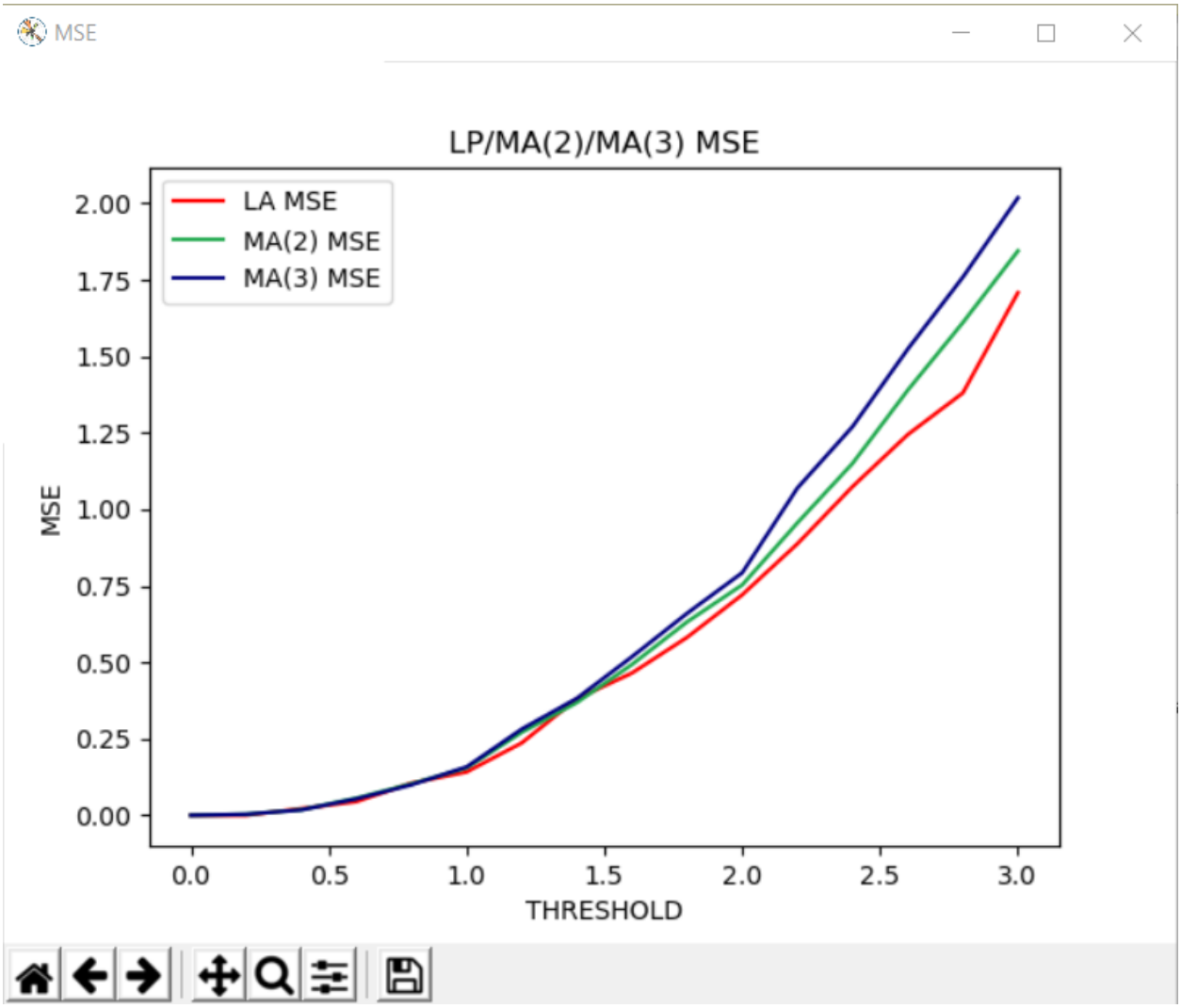


Во овој случај со фрекфенција 5, колку е поголем threshold така се гледа разликата,на почеток речиси нема разлика во процент на пратени податоци но со поголем threshold MA(3) е најдобар па MA(2) и на крај LA.



Јана Ѓелевска 171084

MSE е пресметан за првичните податоци и јасно се гледа дека кривата на MA(3) побрзо расте отколку оние на MA(2) и LA, што значи дека MA(3) врши најдобра предикција а со тоа и праќа најмалку податоци.



Линк до кодот за лабораториската :

https://github.com/GJJana/Lab_Senzorski_Sistemi

Јана Ѓелевска 171084