Міністерство освіти та науки України
Львівський національний університет імені Івана Франка

Звіт

Про виконання лабораторної роботи №3

Обчислення термофізичних показників будинку

Виконав:
студент групи ФеС-31
Мацевко Олександр
Перевірив:
Сінкевич О.О.

Львів 2019

Мета : “Порахувати потужність(Вт) котла(чи якогось іншого опалювального пристрою) та спрогнозувати настільки необхідно збільшити/зменшити потужність опалювального пристрою на 1-3 кроки наперед використовуючи метод для прогнозування часових рядів ARIMA”

Теоретичні відомості :

Систему з декількох пристрої яку буде представлено нижче можна буде використовувати в технологіях реального часу за допомогою фрейморка FLASK(python) який в свою чергу буде підключений до Rasberry Pi - який вимірює температуру та вологість за допомогою сенсора

Система полягає в наступному, ми сенсором міряємо температуру на 100(інтервал сталий допустим 15 хв.) кроків вперед. Для кожного кроку вимірюємо потужність - Q. Берем перші 50 кроків та для них прогнозуємо потужність на крок вперед. Далі так продовжуємо циклічно для кожного кроку вимірюємо потужність, та для кожного наступного 50 кроку прогнозуємо. Це все підключене до Flask.

Реалізація : Все реалізовано на python використовуючи такі бібліотеки як numpy(для розв’язування систем),pandas(Для читання csv даних), sympy(для розв’язування диференціальних рівнянь). Почнемо з ARIMA :

```
def ARIMA_with_Prediction(step, x,y,p,q,d):
    new_y = [0]
    new_y1 = []

    resut_arr = []
    MA_plust_ar = 0

    if d == 0:
        new_x = []
        new_y = []
        new_x = x
        new_y = y
        my_x_x = 199
        for i in range(step):
            MA_res = MA_foresee(new_y,q)
            AR_res = auto_reg(new_x,new_y,p=p)
            MA_plust_ar = MA_res + AR_res
            resut_arr.append(MA_plust_ar)
            new_y.append(MA_plust_ar)
            my_x_x +=1
            new_x.append(my_x_x)
        return resut_arr
# data = pd.read_csv("mainData.csv")
```

Тут реалізована ARIMA:

```
def minMNK(a,b):
    function = 0
    w0, w1 = symbols('x y', real=True)

    for i in range(len(a)):
        function +=(w1*a[i] + w0 - b[i]) ** 2

    #print(function)

    function1W0 = diff(function,w0)
    function2W1 = diff(function,w1)

    #print(function1W0)
    # print(function2W1)

    function1W0 = str(function1W0)
    function2W1 = str(function2W1)


    result1 = re.findall(r'\d+',function1W0)
    result2 = re.findall(r'\d+',function2W1)

    for i in range(len(result1)):
        result1[i] = int(result1[i])

    for i in range(len(result2)):
        result2[i] = int(result2[i])


    matrix = [result1[0:2],result2[0:2]]
    matrix_result = [result1[2],result2[2]]

    resultAll = np.linalg.solve(matrix,matrix_result)

    new_w0, new_w1 = resultAll

    return new_w0, new_w1
```

Тут реалізований метод найменших квадратів для оптимізації аріми:

```
import new_arima
import newq
import pandas as pd

paht_to_data = "meinData.csv"

def CountQAndArima(step,path,p,q,d=0):
        data = pd.read_csv(path)
        temp_y = []
        temp_y.extend(data['value'])
        y = []
        for i in range(len(temp_y)):
            temp = temp_y[i]
            y.append(temp)

        q_list = newq.gloabalTemperature(y,1)
        x = []
        for i in range(len(q_list)):
            x.append(i)

        result = new_arima.ARIMA_with_Prediction(step=step,x=x,y=q_list,p=p,q=q,d=0)
        return result

result = CountQAndArima(2,paht_to_data,1,7)
print(' Predict = ',result)
```

Результат роботи :

```
File  Edit  View  Search  Terminal  Help
ostap@ostap:~/Documents/LNU/kiberg/QWithArima$ python3 main_laboratorna2.py
Predict =  [0.9806108368302218, 0.982208968511009]
ostap@ostap:~/Documents/LNU/kiberg/QWithArima$
```

Висновок :

В цій лабораторній роботі я ознайомився з написанням коду для машин реального часу на опрацювання даних.