

Kauno technologijos universitetas

Informatikos fakultetas

P176B101 Intelektikos pagrindai

Laboratorinis darbas nr. 3

Dirbtiniai neuroniniai tinklai

|  |
| --- |
| Darbą atliko:  IFE – 8 gr. studentas Martynas Kemežys  Darbą priėmė:  lekt. Budnikas Germanas  doc. Paulauskaitė-Tarasevičienė Agne |
|  |
|  |

Kaunas, 2021

Turinys

[Darbo užduotis 2](#_Toc72446288)

[Sprendimų eiga, pirma dalis 3](#_Toc72446289)

[Saulės dėmių aktyvumo už 1700-2014 metus grafikas 3](#_Toc72446290)

[Nupiešti 3D grafiką vaizduojantį įvesčių ir išvesčių rezultatus 3](#_Toc72446291)

[Sukuriame dirbtinį neuroną 5](#_Toc72446292)

[Pavaizduoti neurono apmokymą ir palyginti su norimu rezultatu 5](#_Toc72446293)

[Pavaizduoti neurono apmokymą su bias turimais duomenimis 5](#_Toc72446294)

[Apskaičiuoti klaidos vektorių ir jį pavaizduoti 6](#_Toc72446295)

[Autoregresijos ir tiesinio neurono modeliai(n=2) 7](#_Toc72446296)

[Autoregresijos ir tiesinio neurono modeliai(n=6) 9](#_Toc72446297)

[Autoregresijos ir tiesinio neurono modeliai(n=10) 10](#_Toc72446298)

[Atsakymas į klausimus 11](#_Toc72446299)

[Išvados 12](#_Toc72446300)

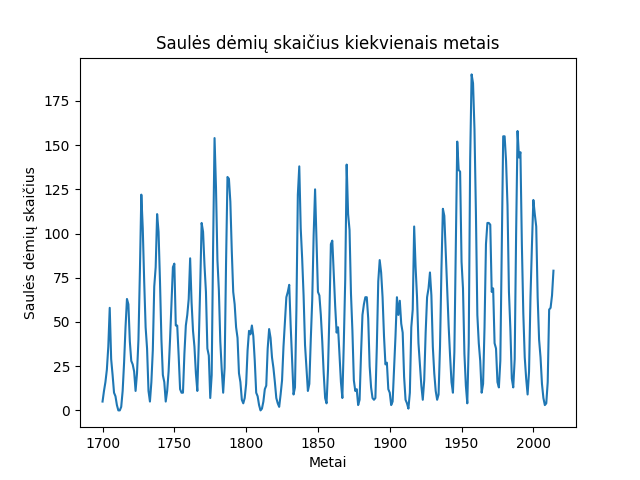
# Darbo užduotis

• Sukurti vienetinį neuroną su tiesinę aktyvavimo funkciją. Apmokinti neuroną su turimais 1700m - 2014m saulės taškų aktyvumo duomenimis. Ištirti neurono spėjimo paklaidas, apskaičiuoti vidutinės kvadratinės prognozės klaidos reikšmes, bei absoliutaus nuokrypio medianą.

• Pasirinkti iš pirmo laboratorinio darbo atributą, duomenis paruošti neurono apmokymui, apmokyti neuroną ir stebėti gautus rezultatus. Pagerinti neurono apmokymą 5%

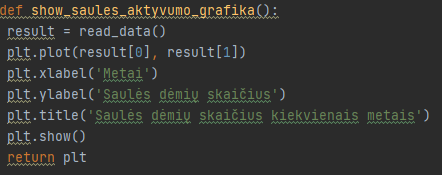
# Sprendimų eiga, pirma dalis

## Saulės dėmių aktyvumo už 1700-2014 metus grafikas



*Diagrama 1. Saulės taškų pasiskirstymas*

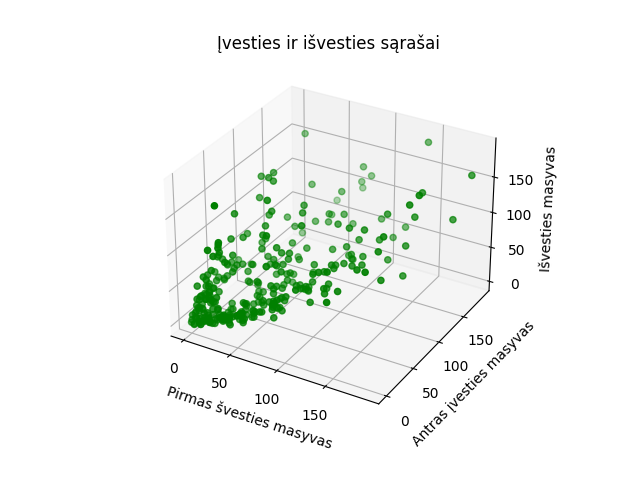
Iš šio grafiko galime pastebėti, kad saulės taškų aktyvumas turi tendenciją didėti ir staigiai sumažėti.



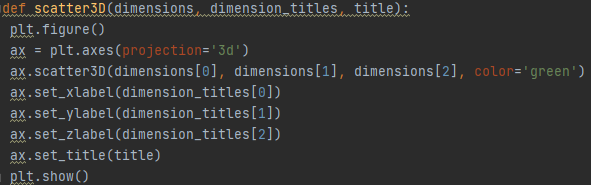
*Kodo fragmentas 1. Saulės taškų pasiskirstymas*

## Nupiešti 3D grafiką vaizduojantį įvesčių ir išvesčių rezultatus

Kai autoregresinio modelio eilė n yra lygi 2, galima nubrėžti grafiką, parodytą apačioje. Taškai sudaro plokštumą, kuri priklauso nuo svorių koeficientų. Plokštuma bus tokioje padėtyje, kad visų taškų atstumų iki plokštumos suma bus mažiausia.



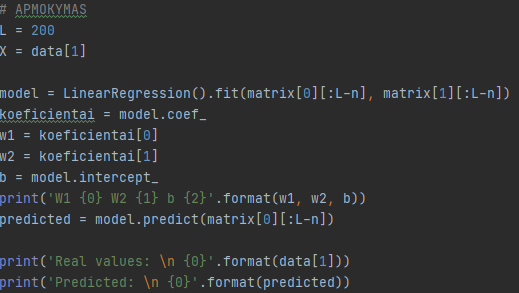
*Diagrama 2. Matricos 3D pavaizdavimas*

**

*Kodo fragmentas 2. Matricos 3D pavaizdavimas*

## Sukuriame dirbtinį neuroną

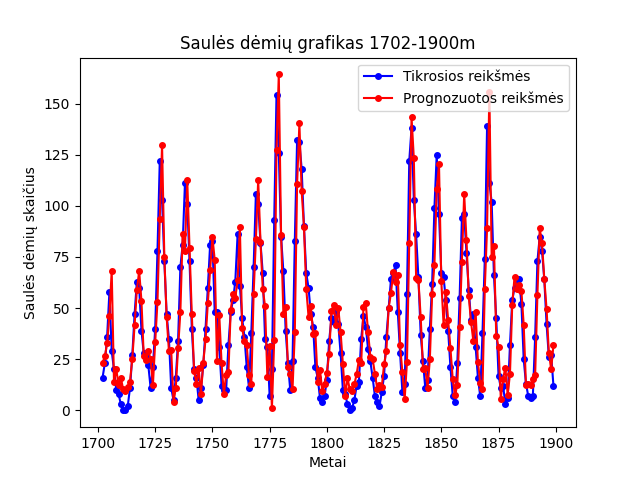
Autoregresinis tiesinis modelis parodytas apačioje. L parametru nurodoma, kiek reikia duomenų, pagal kuriuos bus atliekama prognozė. Prognozės išsaugomos sąraše predicted.



*Kodo fragmentas 4. Prognozavimo kodas*

## Pavaizduoti neurono apmokymą ir palyginti su norimu rezultatu

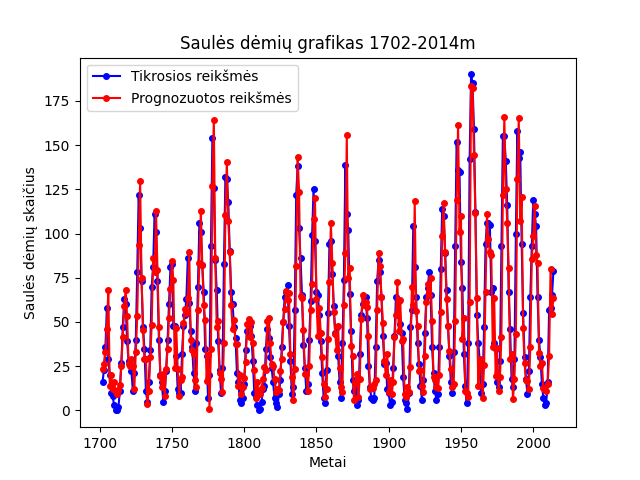
Grafike mėlyna spalva pavaizduoti saulės aktyvumo duomenys iš rinkinio. Raudona spalva pavaizduoti neurono apmokymo išvestis, kitaip sakant jo spėjimai. Galime pastebėti, jog neuronui sunkiausia buvo apskaičiuoti žemutinius taškus ir jų pakilimus. Šioje diagramoje pavaizduoti duomenys nuo 1700 iki 1900 metų.



*Diagrama 3. Neurono mokymosi palyginimas*

## Pavaizduoti neurono apmokymą su bias turimais duomenimis

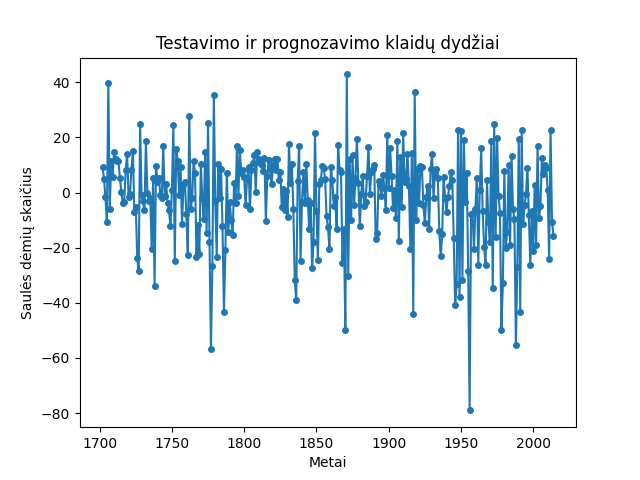
Dabar atlikome tokią pačia analizę, kaip prieš tai buvusiame punkte, tačiau su visu duomenų rinkiniu. Rezultatai apima nuo 1700 iki 2014 metų. Galima pastebėti tą pačia tendenciją, neuronas sunkiai pastebi didelius nuolydžius ir staigias viršūnes.



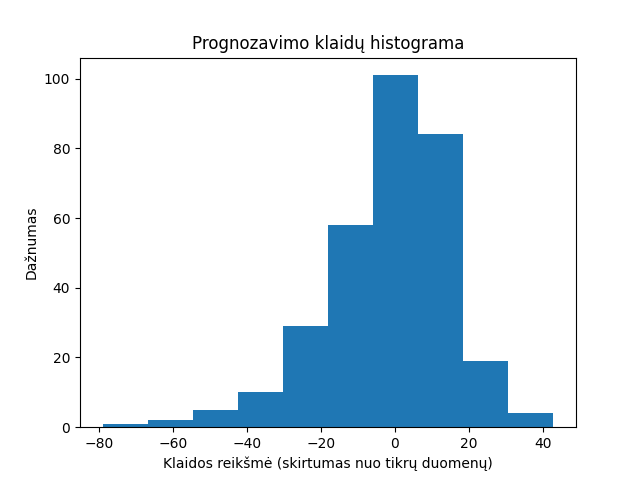
*Diagrama 4. Neurono apmokymo rezultatai su 1700-2014 duomenimis*

## Apskaičiuoti klaidos vektorių ir jį pavaizduoti

Klaidų dydžiai pateikti 5 diagramoje. Klaidų histograma parodyta 6 diagramoje. Grafikai rodo, kad didžiausia klaida yra apie 80.



*Diagrama 5. Testavimo ir prognozavimo klaidų dydžiai*

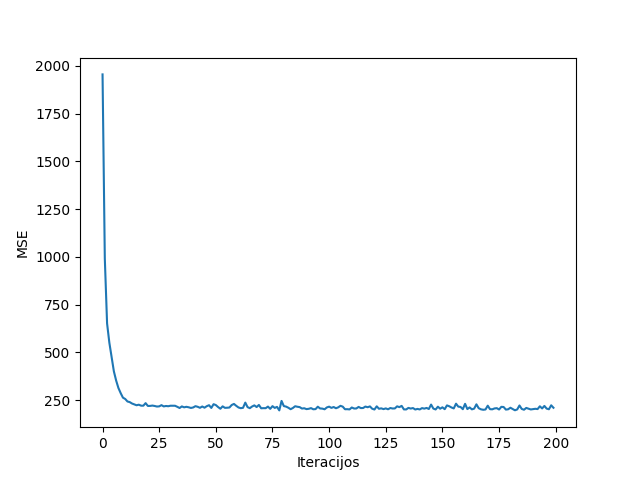


*Diagrama 6. Testavimo ir prognozavimo klaidų histograma*

## Autoregresijos ir tiesinio neurono modeliai(n=2)

Iteracijų skaičiaus ir MSE santykis:

Galime pamatyti, kad ties 10 iteracijom MSE nusistovi.



*Diagrama 7. Iteracijų ir MSE santykis*

Taip yra todėl, kad mokymosi greitis yra pakankamai didelis, todėl rezultatas nusistovi. MSE

reikšmės sumažėjimas ir jos nusistovėjimas rodo, jog procesas konverguojantis. Svorio

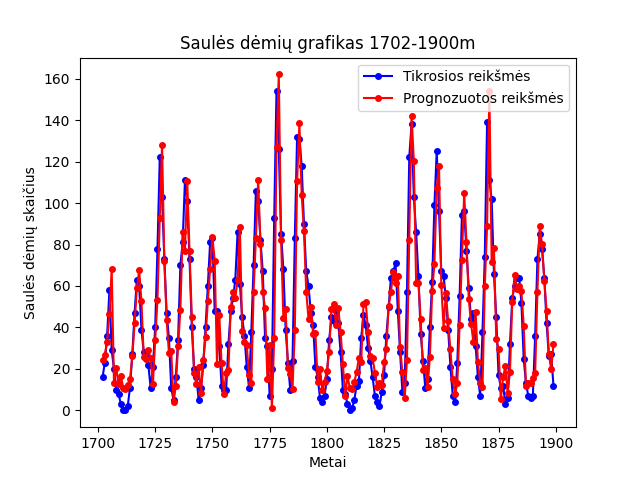
koeficientų reikšmės:

W1: -0.6677777

W2: 1.379677

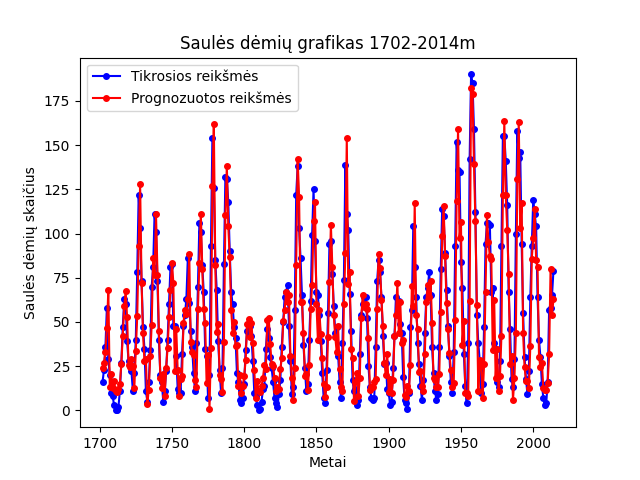
b: 12.119231

Gauti rezultatai:

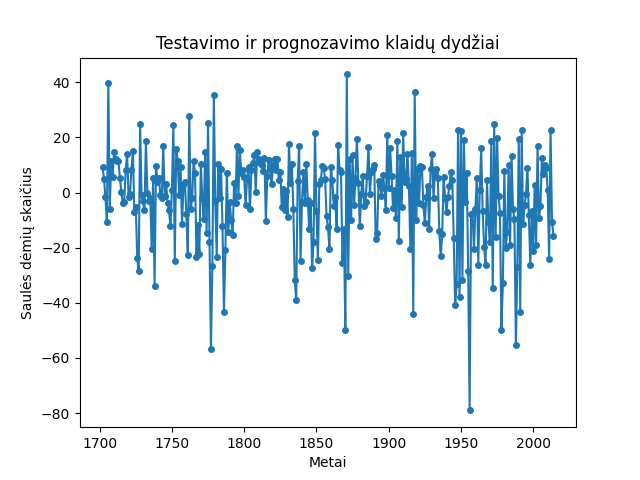


*Diagrama 8. Prognozuotos ir tikros reikšmės 1702-1900 n = 2*

Paėmus visą duomenų rinkinį:



*Diagrama 9. Prognozuotos ir tikros reikšmės 1702-2014 n = 2 iteracinis būdas*

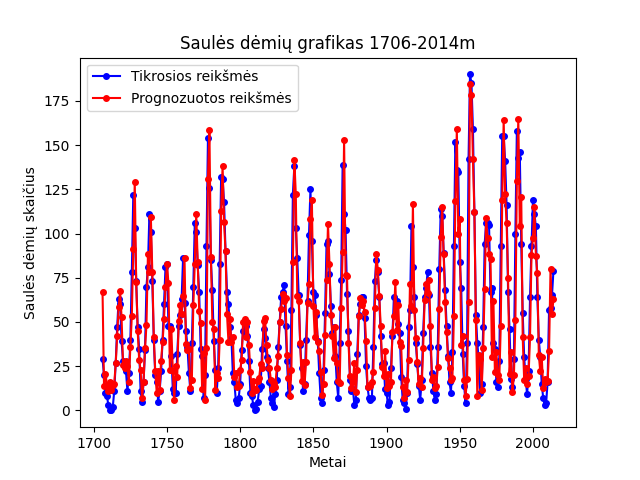


*Diagrama 10. Iteracinio metodo klaidų vektorius*

MSE: 279.175415818

MAD: 8.95595954

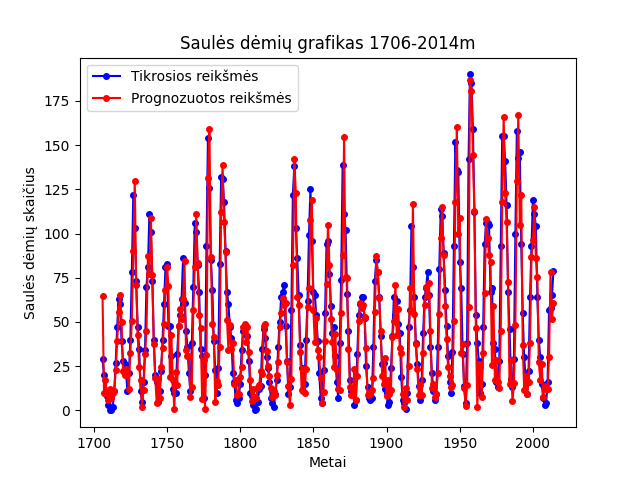
## Autoregresijos ir tiesinio neurono modeliai(n=6)



*Diagrama 11. Prognozuotos ir tikros reikšmės 1706-2014 n = 6*

MSE: 270.9948

MAD: 9.086



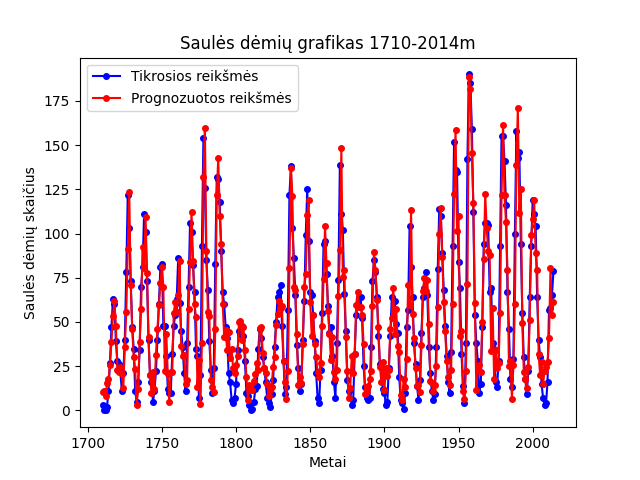
*Diagrama 12. Prognozuotos ir tikros reikšmės 1706-2014 n = 6 Iteracinis būdas*

MSE: 281.4012

MAD: 9.6854

Autoregresijos modelyje galime pastebėti kad vidutinės kvardratinės prognozės klaidos reikšmės, tiek medianos reikšmės pagerėjimas. O tiesinio neurono modelyje atvirkščiai.

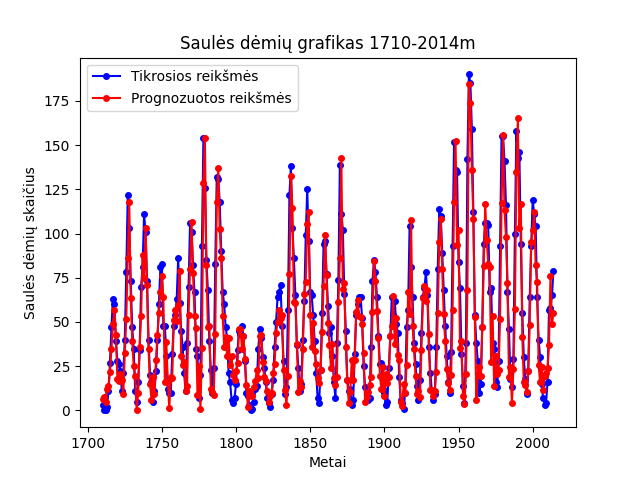
## Autoregresijos ir tiesinio neurono modeliai(n=10)



*Diagrama 13. Prognozuotos ir tikros reikšmės 1706-2014 n = 10*

MSE: 232.7721

MAD: 8.74132



*Diagrama 14. Prognozuotos ir tikros reikšmės 1706-2014 n = 10 Iteracinis būdas*

MSE: 234.5193

MAD: 9.22614

Didinant n skaičių iki 10 tiesinio autoregresijos modelio rezultatai gerėja, o toliau didinant blogėja.

## Atsakymas į klausimus

* Ar mokymosi procesas yra konverguojantis? Jeigu ne, pamąstyti kas gali būti priežastimi ir pakeisti atitinkamą parametrą.

Mokymoso procesas yra konverguojantis.

* Kokios yra naujos neurono svorių koeficientų reikšmės?

n = 2

b = 11.916184

W2 = 1.3827134

W1 = -0.66102415



*Pav 1. Svorių koeficientai*

* Kokia yra neurono darbo kokybės įverčio MSE ir MAD reikšmės ?

n=2



*Pav 2. MSE ir MAD reikšmės*

n=6



*Pav 3. MSE ir MAD reikšmės*

n=10



*Pav 4. MSE ir MAD reikšmės*

MSE yra jautri nuokrypiams, o MAD reikšmė paima vidurinę reikšmę, todėl nuokrypiai nedaro didelės įtakos

# Išvados

* Didesnis iteracijų skaičius nebūtinai reiškia optimaliausia rezultatą, tinklas gali persimokyti jei jų per daug, arba nedasimokinti jei jų per mažai.
* Norint užtikrinti neurono kokybišką spėjimą klaidos vektorius turi būti kuo mažesnis, kuo arčiau 0 yra MSE tuo geriau.
* Duomenys turi būti paruošti ir abdoroti prieš juo paleidžiant į dirbtinį tinklą, išmesti triukšmai, papildytos tuščios reikšmės. Taip užtikrinamas tikslesnis neurono prognozavimas.
* Keičiant neuroninio tinklo struktūrą, galima pagerinti jo tikslumą.
* MSE reikšmė yra jautri nuokrypiams, o MAD yra tam atspari.

**Kodas**

Pirma dalis:

import matplotlib.pyplot as plt  
import numpy as np  
from sklearn.linear\_model import LinearRegression  
import tensorflow as tf  
from tensorflow import keras  
  
R = []  
years = []  
sunspots = []  
  
def read():  
  
 f = open('sunspot.txt', 'r')  
  
 lines = f.readlines()  
 for line in lines:  
 temp=line.split('\t')  
 years.append(int(temp[0]))  
 sunspots.append(int(temp[1]))  
 f.close()  
  
 R.append(years)  
 R.append(sunspots)  
 return R  
data = read()  
print(data[1])  
  
plt.plot(data[0], data[1], marker='o', markersize=4)  
plt.title('Saulės dėmių grafikas 1700-2014m.')  
plt.xlabel('Metai')  
plt.ylabel('Saulės dėmių skaičius')  
plt.show()  
# --------------------------------------------------------------------------------------  
n = 10  
p = []  
t = []  
  
def split(data, n):  
 sunspots = data[1]  
  
 for i in range(len(sunspots) - n):  
 temporary = []  
 for j in range(i, i + n):  
 temporary.append(sunspots[j])  
 else:  
 t.append((sunspots[j + 1]))  
 p.append(temporary)  
 R = list()  
 R.append(p)  
 R.append(t)  
 return R  
  
matrix = split(data, n)  
fig = plt.figure()  
ax = fig.add\_subplot(111, projection='3d')  
X = [i[0] for i in matrix[0]]  
Y = [i[1] for i in matrix[0]]  
print(matrix[0])  
ax.scatter(X, Y, matrix[1])  
ax.set\_xlabel('Saulės dėmių skaičius(x-2) metais')  
ax.set\_ylabel('Saulės dėmių skaičius(x-1) metais')  
ax.set\_zlabel('Saulės dėmių skaičius x metais')  
  
plt.show()  
  
# TRAINING  
L = 200  
X = data[1]  
  
model = LinearRegression().fit(matrix[0][:L-n], matrix[1][:L-n])  
coef = model.coef\_  
w1 = coef[0]  
w2 = coef[1]  
b = model.intercept\_  
print('W1 {0} W2 {1} b {2}'.format(w1, w2, b))  
predicted = model.predict(matrix[0][:L-n])  
  
print('Real values: \n {0}'.format(data[1]))  
print('Predicted: \n {0}'.format(predicted))  
plt.title('Saulės dėmių grafikas {0}-{1}m'.format(1700 + n, 1900))  
plt.xlabel('Metai')  
plt.ylabel('Saulės dėmių skaičius')  
plot1, = plt.plot(data[0][n:L], data[1][n:L], marker='o', markersize=4,  
color='blue')  
plot2, = plt.plot(data[0][n:L], predicted, marker='o', markersize=4,  
color='red')  
plt.legend([plot1,plot2],["Tikrosios reikšmės", "Prognozuotos reikšmės"])  
plt.show()  
  
# TESTING  
length = len(data[0])  
predicted = model.predict(matrix[0][:length-n])  
plt.title('Saulės dėmių grafikas {0}-{1}m'.format(1700 + n, 2014))  
plt.xlabel('Metai')  
plt.ylabel('Saulės dėmių skaičius')  
plot1, = plt.plot(data[0][n:length], data[1][n:length], marker='o',  
markersize=4, color='blue')  
plot2, = plt.plot(data[0][n:length], predicted, marker='o', markersize=4,  
color='red')  
plt.legend([plot1,plot2],["Tikrosios reikšmės", "Prognozuotos reikšmės"])  
plt.show()  
  
def errorVector(RR, PR, years):  
 error = PR - RR  
  
 print("Mediana su testiniais duomenim {0} {1}".format(years, PR.flatten()))  
  
 plt.title('Testavimo ir prognozavimo klaidų dydžiai')  
 plt.plot(years, error, marker='o', markersize=4, label='Testavimo ir prognozavimo klaidų dydžiai')  
  
 plt.xlabel('Metai')  
 plt.ylabel('Saulės dėmių skaičius')  
 plt.show()  
  
 return error  
  
e = errorVector(data[1][n:length], predicted, data[0][n:length])  
  
plt.hist(e)  
plt.title('Prognozavimo klaidų histograma')  
plt.xlabel('Klaidos reikšmė (skirtumas nuo tikrų duomenų)')  
  
plt.ylabel('Dažnumas')  
plt.show()  
  
def MSE(count, error):  
 mse\_Sum = 0  
 for i in error:  
 mse\_Sum += i \* i  
  
 value = 1 / count \* mse\_Sum  
  
 return value  
  
mse = MSE(length - n, e)  
print('MSE {0}'.format(mse))  
  
def MAD(error):  
 mediana = np.median(np.absolute(error))  
 return mediana  
  
mad = MAD(e)  
print('MAD {0}'.format(mad))  
a = [[2,5], [3, 6]]  
ats = np.dot(a, [5, 4])  
print(ats)  
  
#ITERACION METHOD  
#TRAINING  
X = matrix[0][:L-n]  
Y = matrix[1][:L-n]  
model = tf.keras.models.Sequential() # sukuria layer  
model.add(tf.keras.layers.Dense(1, input\_dim = n)) # kiek ivesciu  
opt = keras.optimizers.Adam(learning\_rate=0.01)  
model.compile(optimizer = opt, loss = 'mean\_squared\_error', metrics =  
['mse'])  
before = model.get\_weights()  
print("Svoriai prieš apmokymą: {0}".format(before))  
history = model.fit(X, Y, epochs=200, batch\_size=10, verbose=1)  
mse = history.history['mse']  
plt.plot(mse)  
plt.xlabel("Iteracijos")  
plt.ylabel("MSE")  
plt.show()  
after = model.get\_weights()  
print("Svoriai po apmokymo: {0}".format(after))  
predictions = model.predict(X)  
print("Spėjimai {0}".format(predictions))  
plt.title('Saulės dėmių grafikas {0}-{1}m'.format(1700 + n, 1900))  
plt.xlabel('Metai')  
plt.ylabel('Saulės dėmių skaičius')  
  
plot1, = plt.plot(data[0][n:L], data[1][n:L], marker='o', markersize=4,  
color='blue')  
plot2, = plt.plot(data[0][n:L], predictions, marker='o', markersize=4,  
color='red')  
plt.legend([plot1,plot2],["Tikrosios reikšmės", "Prognozuotos reikšmės"])  
plt.show()  
e = predictions - data[1][n:L]  
mad = np.median(np.absolute(e))  
  
#ITERATION METHOD  
#TESTING  
predictions = model.predict(matrix[0][:length-n])  
plt.title('Saulės dėmių grafikas {0}-{1}m'.format(1700 + n, 2014))  
plt.xlabel('Metai')  
plt.ylabel('Saulės dėmių skaičius')  
plot1, = plt.plot(data[0][n:length], data[1][n:length], marker='o',  
markersize=4, color='blue')  
plot2, = plt.plot(data[0][n:length], predictions, marker='o', markersize=4,  
color='red')  
plt.legend([plot1,plot2],["Tikrosios reikšmės", "Prognozuotos reikšmės"])  
plt.show()  
e = errorVector(data[1][n:length], predictions.flatten(),  
data[0][n:length])  
mse = MSE(length - n, e)  
print('MSE {0}'.format(mse))  
mad = MAD(e)  
print('MAD {0}'.format(mad))