Instituto Politécnico Nacional

Escuela Superior de Cómputo

Regresión Lineal y Logística con scikit-learn

Natural Language Processing

Estudiante:
Nicolás Sayago Abigail

Profesora:
Olga Kolesnicova

Abril 21, 2020

1 Regresión Lineal

RESULTADOS TESTING SET

```
Mean squared error: 39923628078.25
Coefficient of determination: 0.72

########## Values ##########
Real: 1090000.00 Prediction: 1165484.06
Real: 350000.00 Prediction: 438332.91
Real: 520000.00 Prediction: 486911.77
Real: 679950.00 Prediction: 763665.01
Real: 1580000.00 Prediction: 954525.97
Real: 541800.00 Prediction: 738062.90
Real: 810000.00 Prediction: 1013647.42
Real: 1540000.00 Prediction: 1292503.85
Real: 467000.00 Prediction: 440868.45
Real: 224000.00 Prediction: -35035.23
Real: 507250.00 Prediction: 466247.99
Real: 429000.00 Prediction: 325006.42
Real: 610685.00 Prediction: 600855.30
Real: 1010000.00 Prediction: 823175.77
Real: 475000.00 Prediction: 363657.78
Real: 360000.00 Prediction: 483011.65
Real: 400000.00 Prediction: 438505.46
Real: 402101.00 Prediction: 142308.12
Real: 400000.00 Prediction: 385318.15
Real: 325000.00 Prediction: 144977.48
```

✓Código fuente

```
import csv
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score

def readData(nameFile):
    auxMatrix = list()  #Matrix of numpy arrays
    with open(nameFile, newline = '') as csvfile:
        reader = csv.reader(csvfile)
        headers = next(reader)  #List of headers
    with open(nameFile, newline = '') as csvfile:
```

```
        reader = csv.DictReader(csvfile)
        for row in reader:
            auxList = list()
            auxList.append(1)   #Adding 1 for convenience
            for i in range(3, len(headers)):
                intAux = float(row[headers[i]])
                auxList.append(intAux)
            auxNum = np.array(auxList)  #Numpy array
            auxNum = np.absolute(auxNum)
            auxMatrix.append(auxNum)
    matrix = np.array(auxMatrix)  #Numpy matrix
    return matrix

def getY(nameFile):
    with open(nameFile, newline = '') as csvfile:
        reader = csv.DictReader(csvfile)
        auxList = list()
        for row in reader:
            intAux = float(row["price"])
            auxList.append(intAux)
        npArray = np.array(auxList)
    return npArray

#####################################################
#                    MAIN
#####################################################
# Read data
nameFile =
 ↪  '/Users/abiga/Desktop/AbiiSnn/GitHub/Natural-Language-Processing/Practice/19/in.csv
matrix_X = readData(nameFile)
matrix_y = getY(nameFile)

# Split the data into training/testing sets
matrix_X_train = matrix_X[:-20]
matrix_X_test = matrix_X[-20:]
matrix_y_train = matrix_y[:-20]
matrix_y_test = matrix_y[-20:]

regr = linear_model.LinearRegression() #Linear regression object
regr.fit(matrix_X_train, matrix_y_train) #Training the model
matrix_y_prediction = regr.predict(matrix_X_test) #Making predctions
print('Mean squared error: %.2f' % mean_squared_error(matrix_y_test,
 ↪  matrix_y_prediction))
```

```
print('Coefficient of determination: %.2f' % r2_score(matrix_y_test,
 ↪  matrix_y_prediction))
print(" ");
print('########## Values ##########')
for i in range(0, len(matrix_y_test)):
    print("Real: %.2f" % matrix_y_test[i], "Prediction: %.2f" %
     ↪  matrix_y_prediction[i])
```

2 Regresión Logística

RESULTADOS TESTING SET

```
########## Values ##########
Real: 1.00 Prediction: 1.00
Real: 1.00 Prediction: 1.00
Real: 1.00 Prediction: 1.00
Real: 1.00 Prediction: 1.00
Real: 1.00 Prediction: 1.00
Real: 1.00 Prediction: 1.00
Real: 1.00 Prediction: 1.00
Real: 1.00 Prediction: 1.00
Real: 1.00 Prediction: 1.00
Real: 1.00 Prediction: 1.00
Real: 0.00 Prediction: 0.00
Real: 0.00 Prediction: 1.00
Real: 0.00 Prediction: 0.00
Real: 0.00 Prediction: 0.00
Real: 0.00 Prediction: 0.00
Real: 0.00 Prediction: 0.00
Real: 0.00 Prediction: 0.00
Real: 0.00 Prediction: 0.00
Real: 0.00 Prediction: 0.00
Real: 0.00 Prediction: 0.00
Real: 0.00 Prediction: 0.00
Real: 0.00 Prediction: 0.00
Real: 0.00 Prediction: 1.00
Real: 0.00 Prediction: 0.00
Real: 0.00 Prediction: 0.00
```

✓Código fuente

```
import nltk
import re
import math
from bs4 import BeautifulSoup
from pickle import dump, load
import numpy as np
from sklearn.datasets import load_iris
from sklearn.linear_model import LogisticRegression
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer

#Parameters: File path, encoding
#Return: String with only lower case letters
#Notes: path =
 ↪  '/Users/27AGO2019/Desktop/AbiiSnn/GitHub/Natural-Language-Processing/corpus/e961024
def getText(corpusRoot, code):
        f = open(corpusRoot, encoding = code) #Cod: utf-8, latin-1
        text = f.read()
        f.close()
        soup = BeautifulSoup(text, 'lxml')
        text = soup.get_text()
        text = text.lower()
        return text

#Parameters: Text
#Return: List of original tokens
def getTokens(text):
        tokens = nltk.word_tokenize(text)
        return tokens

##############################################
##############################################
##############################################

#Read file spam/ham
fpathCorpus =
 ↪  '/Users/abiga/Desktop/AbiiSnn/GitHub/Natural-Language-Processing/Practice/18/corpus
code = 'ISO-8859-1'
corpus = getText(fpathCorpus, code)
tokens = getTokens(corpus)

#Matrix and Y
```

```
xaux = list()
yaux = list()
xi = ""
for token in tokens:
        if token != 'spam' and token != 'ham':
                xi += token + " "
        else:
                typeMessage = 0
                if token == 'ham':
                        typeMessage = 1
                yaux.append(typeMessage)
                xaux.append(xi) #Create X
                xi = ""
x = np.array(xaux)
y = np.array(yaux)

vectorizer = CountVectorizer()
X = vectorizer.fit_transform(x)
matrix_X = X.toarray()
matrix_Y = y
# Split the data into training/testing sets
size = -500
matrix_X_train = matrix_X[:size]
matrix_X_test = matrix_X[size:]
matrix_y_train = matrix_Y[:size]
matrix_y_test = matrix_Y[size:]

clf = LogisticRegression(random_state = 0) #Linear regression object
clf.fit(matrix_X_train, matrix_y_train) #Training the model
matrix_y_prediction = clf.predict(matrix_X_test) #Making predctions
print('########## Values ##########')
for i in range(0, 20):
    print("Real: %.2f" % matrix_y_test[i], "Prediction: %.2f" %
     ↪  matrix_y_prediction[i])
```