מסדי נתונים מטלה 4

אביה אורן - 322273301

מאיה חייט - 322515669

# 

# שאלה 1

א. הדפסת טבלה עם שמות כל הסופרים שמתחילים באות F כולל שם הסופר שם הספר ומספר השנים שעברו מאז שהוא יצא לאור:

צילום מסך של הקוד והתוצאות :





הקוד עצמו :

from pyspark.sql import SparkSession

import pyspark.sql.functions as F

# יצירת סשן של Spark

spark = SparkSession.builder.getOrCreate()

# קריאת הקובץ למסגרת דאטה פריים

df = spark.read.json("OneDrive - Ariel University/year 2/semester b/מסדי נתונים/matala 4/src/books.json" , multiLine = True)

# סינון לפי סופרים שמתחילים באות 'F'

filtered\_df = df.filter(df.author.startswith('F'))

# חישוב השנים מאז הפרסום ועד היום

filtered\_df = filtered\_df.withColumn('years', year(current\_date()) - filtered\_df.year)

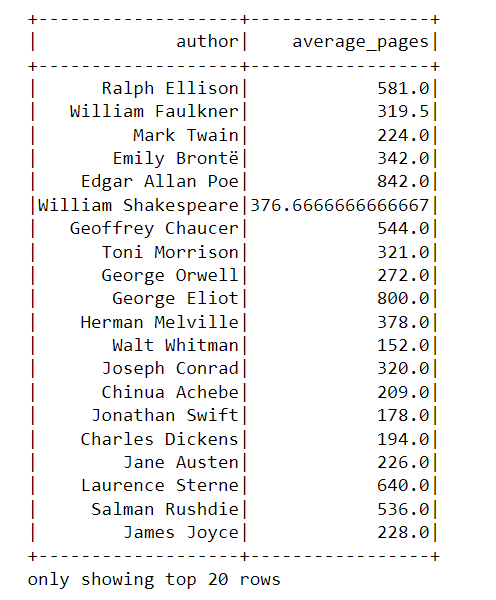
# הצגת התוצאות

filtered\_df.select("title", "author", "years").show()

ב. חישוב מספר הדפים הממוצע שכתב כל סופר, אך רק עבור ספרים באנגלית:

צילום מסך של הקוד והתוצאות :





הקוד עצמו:

from pyspark.sql import SparkSession

import pyspark.sql.functions as F

# קריאת הקובץ JSON למסגרת DataFrame

df = spark.read.json("OneDrive - Ariel University/year 2/semester b/מסדי נתונים/matala 4/src/books.json" , multiLine = True)

# סינון לפי ספרים באנגלית

english\_books\_df = df.filter(df.language == 'English')

# חישוב מספר הדפים הממוצע לכל סופר

average\_pages\_df = english\_books\_df.groupBy("author").agg(F.avg("pages").alias("average\_pages"))

# הצגת התוצאות

average\_pages\_df.show()

# שאלה 2 – LINEAR REGRESSION

יש להשתמש בפייטון ולממש את ה-classifiers מ - scratch

לא להשתמש בשום ספריה של learning machine

כמובן שאפשר להשתמש ב-numpy ולהתבסס על השקפים בשיעור

מצ"ב רשימת נתונים על מחירי דירות בעיר אחת בחו"ל:  
 הנתונים הם:

1מחירי המכירה המקומיים, במאות דולרים;  
 2, מספר חדרי האמבטיה;  
 3, שטח האתר באלפי מטרים רבועים;  
 4 גודלו של שטח המחיה באלפי מטרים רבועים;  
 5 מספר המוסכים;  
 6, מספר החדרים;  
 7, מספר חדרי שינה;  
 8, הגיל בשנים;  
 9, סוג הבנייה  
 10, סוג אדריכלות  
 11, האם יש מכבי אש (0 אין 1 יש).  
 12.מחיר המכירה

קראו את הנתונים ב- python ו:

1. בנו מודל **linear** regression בפייטון אשר חוזה את מחיר הדירה על הנתונים 1-11 .

השתמשו ב-75% מהנתונים עבור בניית המודל, ב-25% הנותרים בידקו את המודל שיצרתם .

2. דווחו את התוצאות ב-**mean squared-error .** יש להשוות את התוצאות של המודל לנתונים הקיימים .

**We’ve added below our code as well as screenshots:**

import numpy as np

# set "data" to the data given

data = np.array([

[4.9176,1.0,3.4720,0.998,1.0,7,4,42,3,1,0,25.9],

[5.0208,1.0,3.5310,1.500,2.0,7,4,62,1,1,0,29.5],

[4.5429,1.0,2.2750,1.175,1.0,6,3,40,2,1,0,27.9],

[4.5573,1.0,4.0500,1.232,1.0,6,3,54,4,1,0,25.9],

[5.0597,1.0,4.4550,1.121,1.0,6,3,42,3,1,0,29.9],

[3.8910,1.0,4.4550,0.988,1.0,6,3,56,2,1,0,29.9],

[5.8980,1.0,5.8500,1.240,1.0,7,3,51,2,1,1,30.9],

[5.6039,1.0,9.5200,1.501,0.0,6,3,32,1,1,0,28.9],

[16.4202,2.5,9.8000,3.420,2.0,10,5,42,2,1,1,84.9],

[14.4598,2.5,12.8000,3.000,2.0,9,5,14,4,1,1,82.9],

[5.8282,1.0,6.4350,1.225,2.0,6,3,32,1,1,0,35.9],

[5.3003,1.0,4.9883,1.552,1.0,6,3,30,1,2,0,31.5],

[6.2712,1.0,5.5200,0.975,1.0,5,2,30,1,2,0,31.0],

[5.9592,1.0,6.6660,1.121,2.0,6,3,32,2,1,0,30.9],

[5.0500,1.0,5.0000,1.020,0.0,5,2,46,4,1,1,30.0],

[5.6039,1.0,9.5200,1.501,0.0,6,3,32,1,1,0,28.9],

[8.2464,1.5,5.1500,1.664,2.0,8,4,50,4,1,0,36.9],

[6.6969,1.5,6.9020,1.488,1.5,7,3,22,1,1,1,41.9],

[7.7841,1.5,7.1020,1.376,1.0,6,3,17,2,1,0,40.5],

[9.0384,1.0,7.8000,1.500,1.5,7,3,23,3,3,0,43.9],

[5.9894,1.0,5.5200,1.256,2.0,6,3,40,4,1,1,37.5],

[7.5422,1.5,4.0000,1.690,1.0,6,3,22,1,1,0,37.9],

[8.7951,1.5,9.8900,1.820,2.0,8,4,50,1,1,1,44.5],

[6.0931,1.5,6.7265,1.652,1.0,6,3,44,4,1,0,37.9],

[8.3607,1.5,9.1500,1.777,2.0,8,4,48,1,1,1,38.9],

[8.1400,1.0,8.0000,1.504,2.0,7,3,3,1,3,0,36.9],

[9.1416,1.5,7.3262,1.831,1.5,8,4,31,4,1,0,45.8],

[12.0000,1.5,5.0000,1.200,2.0,6,3,30,3,1,1,41.0]

])

#divide the data into features and labels (labels is the last column)

features = data[: , :-1]

labels = data[:, -1]

#dividing the data into train and test set

train\_set = data[int(len(data) \* 0.75)]

test\_set = data[int(len(data) \* 0.75):]

w = np.array([0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0])

b = 0.0

learning\_rate = 0.001

for iteration in range(4300):

deriv\_b = np.mean(1\*((np.dot(features, w) + b)-labels))

deriv\_w = 1.0 / len(labels) \* np.dot(((np.dot(features, w) + b) - labels), features)

b -= learning\_rate \* deriv\_b

w -= learning\_rate \* deriv\_w

print("---------------PART A----------------------")

#Calculating the estimated price per apartment

apt1 = np.dot(np.array([7.5422,1.5,4.0000,1.690,1.0,6,3,22,1,1,0]), w)+ b

apt2 = np.dot(np.array([8.7951,1.5,9.8900,1.820,2.0,8,4,50,1,1,1]), w)+ b

apt3 = np.dot(np.array([6.0931,1.5,6.7265,1.652,1.0,6,3,44,4,1,0]), w)+ b

apt4 = np.dot(np.array([8.3607,1.5,9.1500,1.777,2.0,8,4,48,1,1,1]), w)+ b

apt5 = np.dot(np.array([8.1400,1.0,8.0000,1.504,2.0,7,3,3,1,3,0]), w)+ b

apt6 = np.dot(np.array([9.1416,1.5,7.3262,1.831,1.5,8,4,31,4,1,0]), w)+ b

apt7 = np.dot(np.array([12.0000,1.5,5.0000,1.200,2.0,6,3,30,3,1,1]), w)+ b

#printing the estimated price vs the actual price

print("Estimated price for apt1 is: ", apt1 , " and the actual price is " , 37.9)

print("Estimated price for apt2 is: ", apt2 , " and the actual price is " , 44.5)

print("Estimated price for apt3 is: ", apt3 , " and the actual price is " , 37.9)

print("Estimated price for apt4 is: ", apt4 , " and the actual price is " , 38.9)

print("Estimated price for apt5 is: ", apt5 , " and the actual price is " , 36.9)

print("Estimated price for apt6 is: ", apt6 , " and the actual price is " , 45.8)

print("Estimated price for apt7 is: ", apt7 , " and the actual price is " , 41.0)

print("---------------PART B----------------------")

# Calculating the MSE

#first calculate the squared error of each apartment

cost1 = (apt1 - 37.9)\*\*2

cost2 = (apt2 - 44.5)\*\*2

cost3 = (apt3 - 37.9)\*\*2

cost4 = (apt4 - 38.9)\*\*2

cost5 = (apt5 - 36.9)\*\*2

cost6 = (apt6 - 45.8)\*\*2

cost7 = (apt7 - 41.0)\*\*2

#Adding all errors and multiplying by a 1/2\*7

MSE = (1/(2\*len(test\_set)))\*(cost1+cost2+cost3+cost4+cost5+cost6+cost7)

print("The final mean squared error is ", MSE)

This is a screenshot of the results we got when running our code:

