به نام خدا

#تمرین
کامپیوتری سوم

درس: آمار و احتمال مهندسی

مجید صادقی نژاد
۸۱۰۱۰۱۴۵۹

- توضیحات:

کد اصلی پروژه google colab نوشته شد و آدرس آن در زیر آمده است پیشنهاد می شود برای تجربه بهتر کد در همین محیط اجرا شود هر چند که فایل ها ipynb. , py. که ذخیره شده از همین پروژه در colab هستند هم در ضمیمه فایل های تحویل شده آمده اند **قابل توجه است که در صورت اجرا کد در google colab در ابتدا باید تمامی فایل های تمرین را در آن محیط بارگذاری کنید ...**

لینک پروژه: https://colab.research.google.com/drive/۱hKZiQl۰abfQKvF_۲HRpneds۴M۹nXNfU-?usp=sharing

- سوال اول:

۱. اول : فشرده سازی : می توان با استفاده از اتوانکودر ها داده ها را فشرده سازی و به فضای نهان کم بعد تبدیل کرد و حجم آن ها را کاهش داد.

دوم : حذف نویز از داده ها :می توان اتوانکودر را به نحوی آموزش داد که نویز را از داده هایی مثل صدا یا تصویر حذف کند.

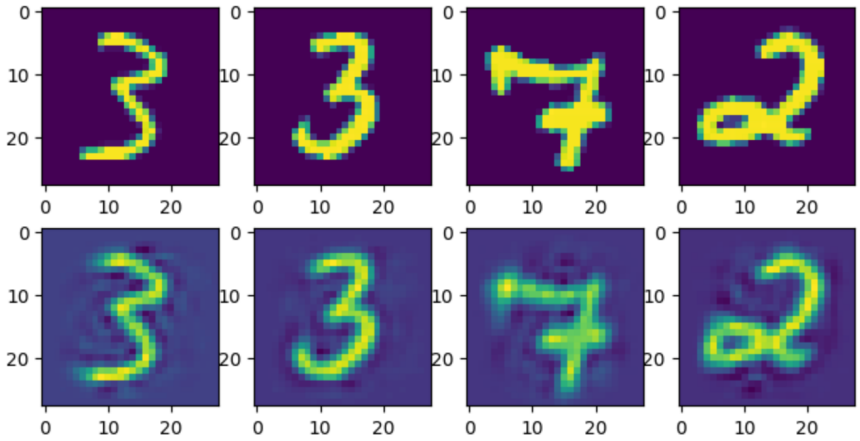
۲. اگر اندازه ی فضای نهان بسیار کوچک باشد موجب می شود انکودر بعضی از اطلاعات ضروری را در هنگام فشرده سازی از دست بدهد همچنین اگر اندازه ی فضای نهان بسیار بزرگ باشد سبب می شود تا اتوانکودر در هنگام دی کد کردن بخشی از اطلاعات جزئی را در فضای نهان نگه دارد که سبب کم شدن و Blur شدن تصویر می شود.

۳. کد نوشته شده به شکل زیر است :

```
import numpy as np, random
def set_seed(seed):
  np.random.seed(seed)
  random.seed(seed)
set_seed(810109203)

from keras.datasets import mnist
(_, _) , (test_images, _) = mnist.load_data()
test_images = test_images.reshape(test_images.shape[0] , -1)
test_images = test_images.astype('float32') / 255.0

import tensorflow as tf
autoencoder = tf.keras.models.load_model('mnist_AE.h5')
reconstructed_images = autoencoder.predict(test_images)
```

```
import matplotlib.pyplot as plt

indices = [323,3487,4654,8844]

plt.figure(figsize=(8, 4))
for i, index in enumerate(indices, 1):
    plt.subplot(2, 4, i)
    plt.imshow(test_images[index].reshape(28, 28))
    plt.subplot(2, 4, i + 4)
    plt.imshow(reconstructed_images[index].reshape(28, 28))

plt.show()
```

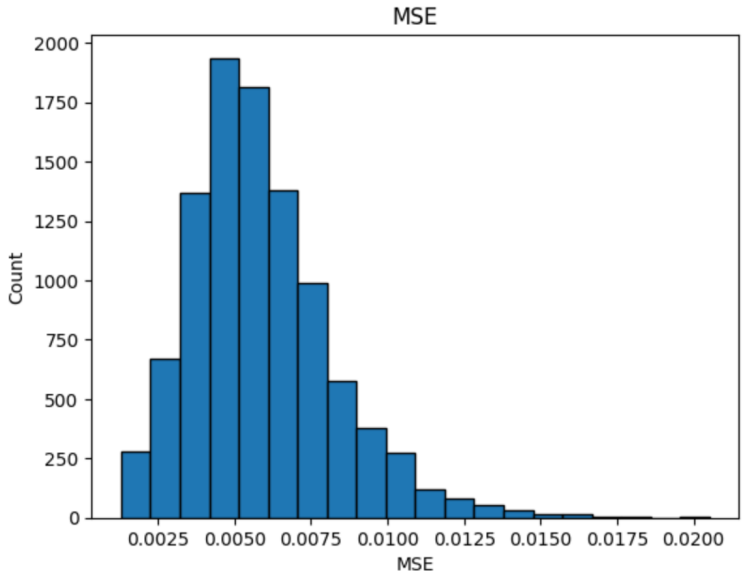
ج. نتیجه به صورت زیر است که در آن ردیف بالایی تصاویر اصلی و ردیف پایین تصاویر دیکد شده است و همانطور که مشاهده می شود تصاویر پس از اینکد و دیکد مقداری تار شده اند:



د. با کمک گرفتن از کتابخانه ی نامپای تابع MSE را می نویسیم و هیستوگرام داده های تولید شده را رسم می کنیم:

```
def mean_squared_error(original, reconstructed):
    return np.mean(np.square(original - reconstructed))

mse_values = [mean_squared_error(test_images[i], reconstructed_images[i]) for i in range(test_images.shape[0])]
plt.hist(mse_values, bins=20, edgecolor='black')
plt.title('MSE')
plt.xlabel('MSE')
plt.ylabel('Count')
plt.show()
```

نتیجه به صورت زیر است :



ه. حال پس از پیدا کردن میانگین و انحراف MSE با استفاده از قطعه کد داده شده مقدار p-value را محاسبه می کنیم :

```
from scipy import stats
mean = sum(mse_values) / len(mse_values)
std = (sum((mse - mean) ** 2 for mse in mse_values) / len(mse_values)) ** 0.5

ks_statistic, p_value = stats.kstest(mse_values, cdf='norm', args=(mean, std))
print(p_value)
```

مشاهده می شود که مقدار p-value محاسبه شده برابر است با :

4.538721695605657e-43

که بسیار کوچکتر از ۰.۰۵ است و بنابراین این توزیع به توزیع نرمال شباهتی ندارد

- **سوال دوم:**

۱- نقاط پرت: نقاطی هستند که در بازه ی x دیگر نقاط برای تشکیل منحنی رگرسیون هستند اما y به شدت متفاوتی نسبت به سایر نقاط دارند این نقاط اگر در معادله رگرسیون تاثیر داده شوند سبب خطای شدید می شوند . نقاط اهرمی: این نقاط خارج از بازه ی x های سایر نقاط هستند (در محدوده ای که نقاط دیگر در آن محدوده اطلاعاتی به ما نمی دهند) و بنابراین می توانند تاثیر بسیار زیادی در شیب منحنی رگرسیون بگذارند. نقاط با هر دو ویژگی: این نقاط هر دو ویژگی را دارا هستند یعنی هم در x و هم در y خارج از بازه ی دیگر نقاط هستند و بنابراین در شیب معادله ی رگرسیون به شدت تاثیر منفی می گذارند.

۲- ضریب تعیین میزان ارتباط خطی بین دو متغیر را اندازه گیری می کند که در مبحث رگرسیون خطی این ضریب برای بررسی نیکوئی برازش مورد استفاده قرار می گیرد بدین صورت که این ضریب مقداری بین ۰ تا ۱ را اختیار می کند هر چه ضریب به یک نزدیک تر باشد بدین معنی ست که خط برازش شده به نقاط نزدیک تر است و هر به صفر نزدیک باشد بدین معنی ست که خط برازش شده از نقاط اولیه دور تر است همچنین این ضریب به دست آمده از نسبت دو واریانس است که در آن متغیر های پیش بینی شده و اولیه حضور دارند و فرمول آن به صورت زیر است :

$$R^2 = 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}} \qquad SS_{\text{res}} = \sum_i (y_i - f_i)^2 \quad SS_{\text{tot}} = \sum_i (y_i - \bar{y})^2$$

۳- فرمول محاسبه رگرسیون خطی بر حسب MSE به شکل زیر است :

$$\widehat{\beta_1} = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2}$$

$$\widehat{\beta_0} = \bar{y} - \widehat{\beta_1}\bar{x}$$

که B۱ برابر با شیب خط و B۰ برابر با عرض از مبدا است حال با ساده سازی دو فرمول بالا به روابط زیر می رسیم که a شیب خط و b عرض از مبدا است (عملگر [ ] به معنای سیگمای i از ۱ تا n است):

$$a = \frac{n[x_i y_i] - [x_i][y_i]}{[x_i{}^2] - [x_i]^2}$$

$$b = \frac{[y_i] - a[x_i]}{n}$$

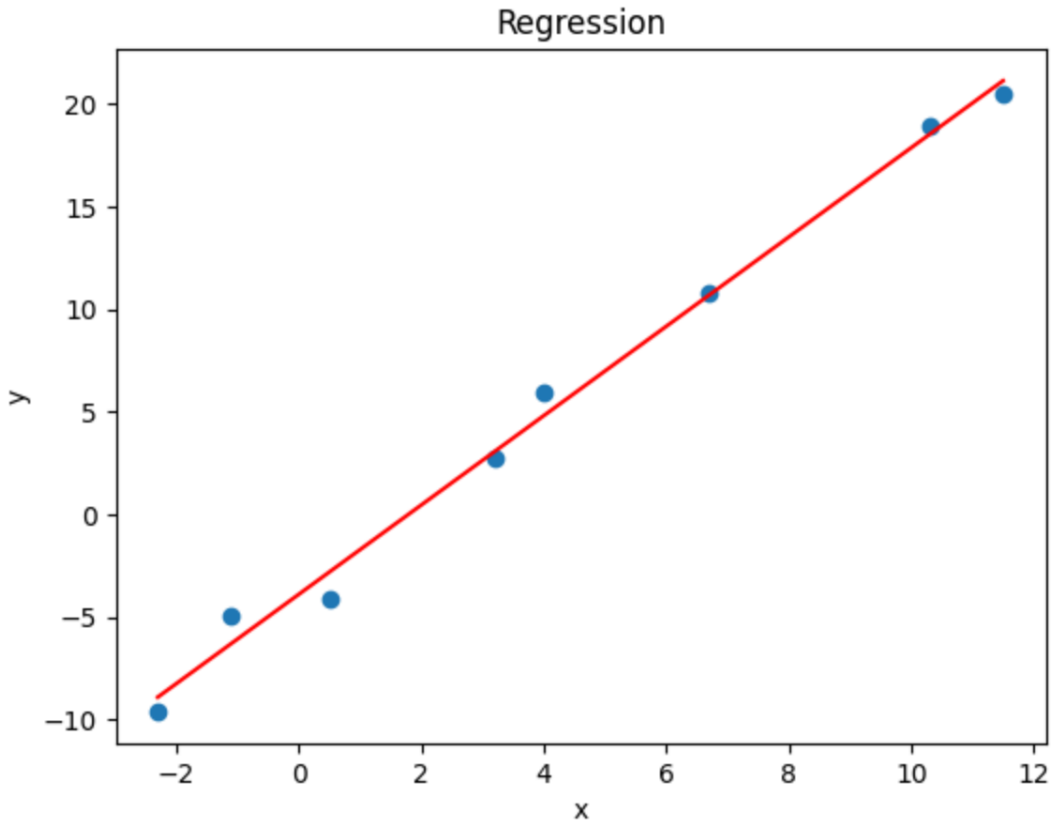
بر همین اساس تابع زیر را طراحی می کنیم تا بر اساس مجموعه ی نقاط رگرسیون خطی را محاسبه سپس بر اساس معادله به دست آمده ضریب تعیین را محاسبه و در نهایت معادله و نقاط را رسم کند :

```
import matplotlib.pyplot as plt
import numpy as np

def linear_regression(x, y):
  n = len(x)
  a = (n * np.sum(x * y) - np.sum(x) * np.sum(y)) / (n * np.sum(x**2) - (np.sum(x))**2)
  b = (np.sum(y) - a * np.sum(x)) / n
  y_new = a * x + b

  SS_res = np.sum((y - y_new)**2)
  SS_tot = np.sum((y - np.mean(y))**2)
  R_squ = 1 - (SS_res / SS_tot)

  plt.scatter(x, y)
  plt.plot(x, y_new, color='red')
  plt.title('Regression')
  plt.xlabel('x')
  plt.ylabel('y')
  plt.show()
  print(f"R^2 = {R_squ}")
```

حال برای هر کدام از حالت های خواسته شده تابع را فراخوانی می کنیم (ضریب تعیین در زیر نمودار چاپ شده است):

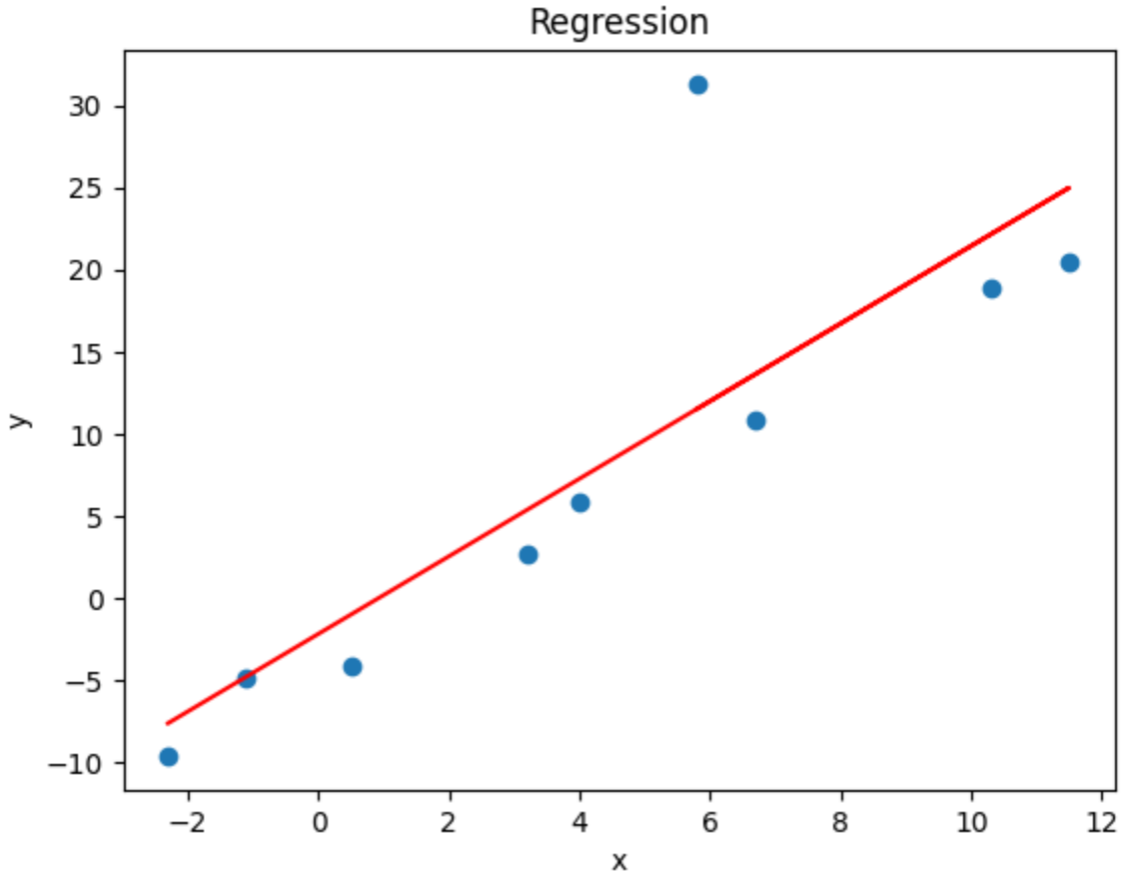
```
x = np.array([-2.3, -1.1, 0.5, 3.2, 4, 6.7, 10.3, 11.5])
y = np.array([-9.6, -4.9, -4.1, 2.7, 5.9, 10.8, 18.9, 20.5])

linear_regression(x, y)
linear_regression(np.append(x, 5.8), np.append(y, 31.3))
linear_regression(np.append(x, 20.4), np.append(y, 14.1))
linear_regression(np.append(x, 20.4), np.append(y, 31.3))
```

الف) رگرسیون بر پایه هشت داده ی اصلی :



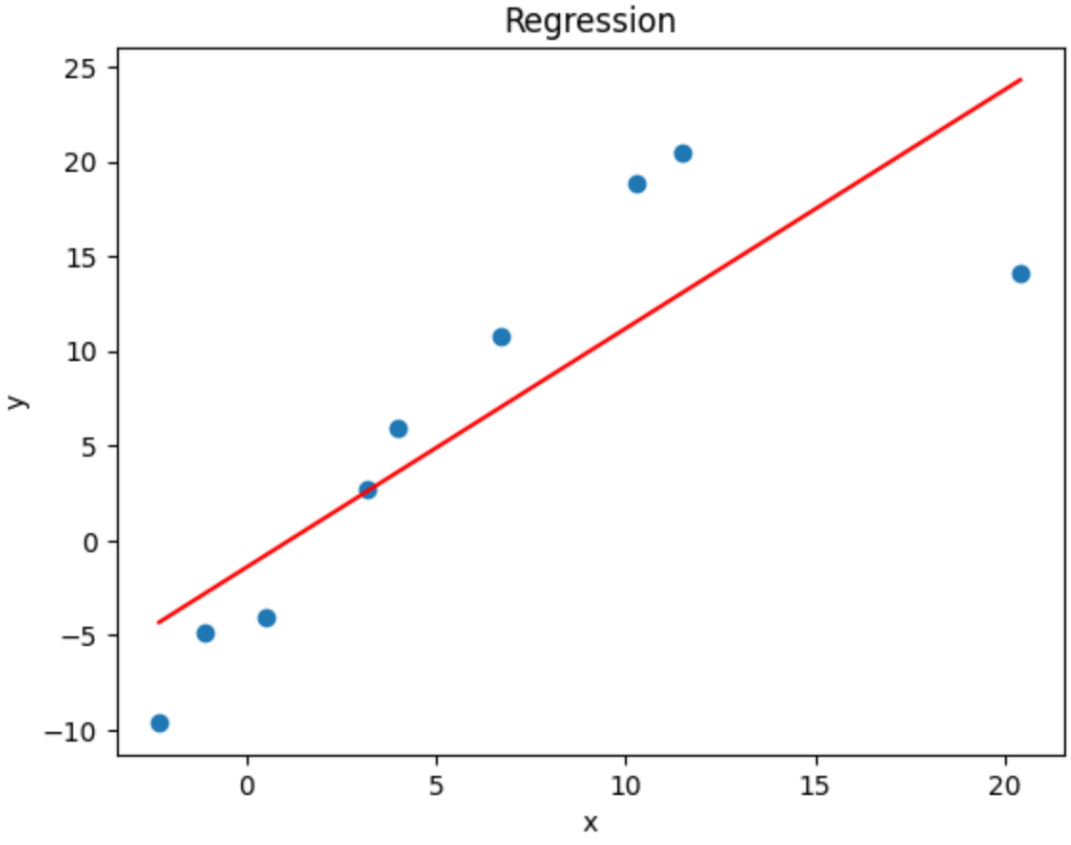

```
R^2 = 0.9931142293628097
```

ب) رگرسیون بر پایه هشت داده اصلی به اضافه نقطه دور افتاده:



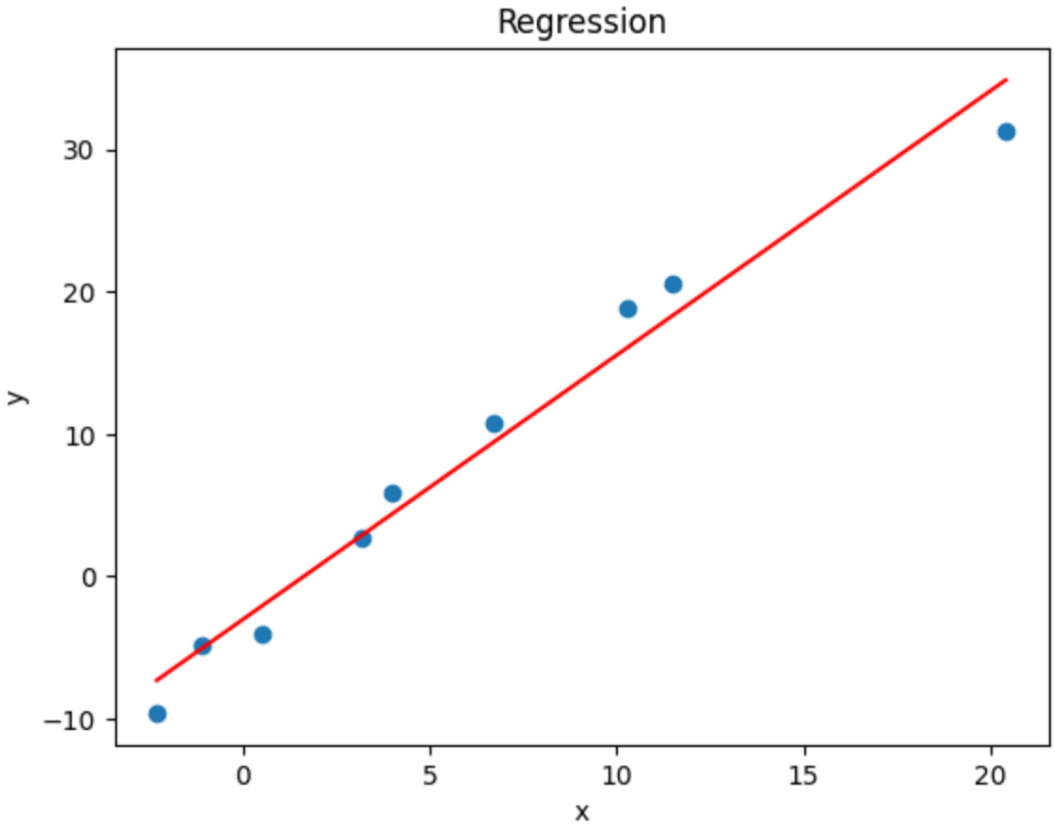

```
R^2 = 0.6943381680789323
```

ج) رگرسیون بر پایه هشت داده اصلی به اضافه نقطه اهرمی:




```
R^2 = 0.7069879724740985
```

ت) رگرسیون بر پایه هشت داده اصلی به اضافه نقطه دور افتاده-اهرمی:




```
R^2 = 0.9738367949787371
```

۴- در ارتباط با رگرسیون ، مبحثی به نام رگرسیون با ثبات وجود دارد رگرسیون های باثبات روش هایی هستند که نسبت به روش حداقل مربعات مانده عملکرد بهتری در مواجهه با داده های پرت دارند یکی از روش های رگرسیون باثبات استفاده از قدر مطلق به جای توان دو است بدین معنی که در روش مربعات مانده هر کجا که از توان دو استفاده می شد حالا از قدر مطلق استفاده شود.

- سوال سوم:

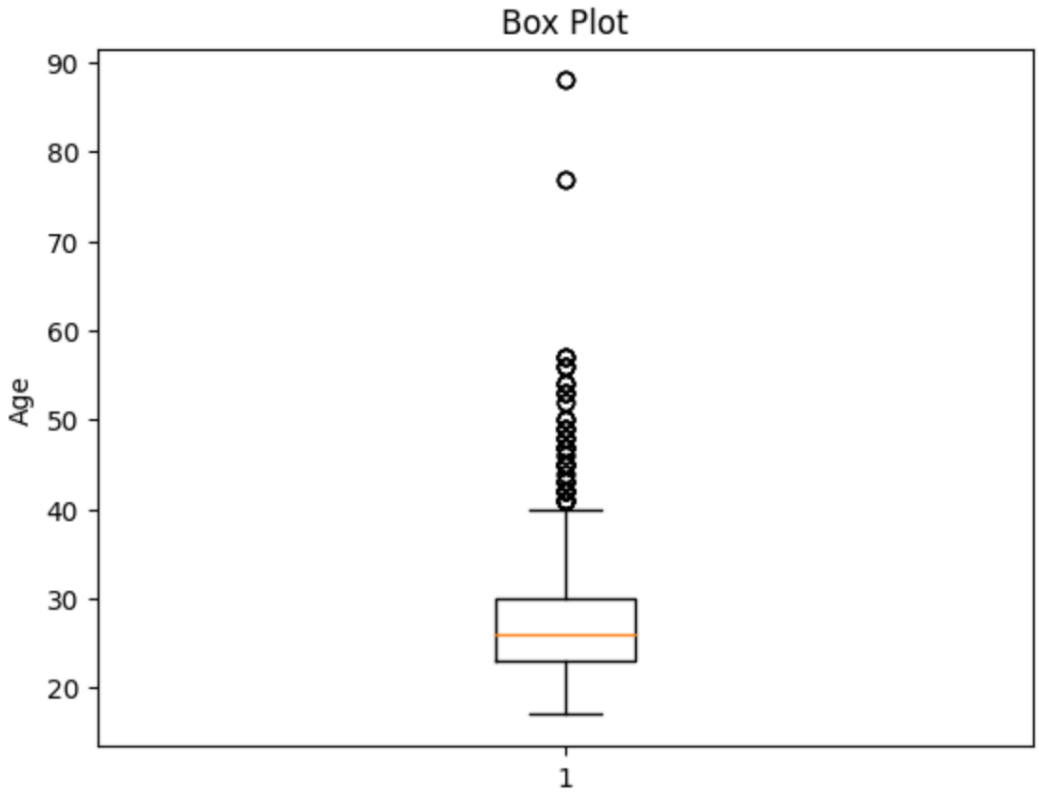
۱. یکی از راه حل ها بدین صورت است تا میانگین مقدار کنونی آن ستون را جایگزین مقادیر نامعلوم کنیم بدین صورت در نمونه برداری هایی که با میانگین سر و کار دارند دچار کمترین خطا می شویم.

```
import pandas as pd

df = pd.read_csv('FIFA2020.csv', encoding = "ISO-8859-1")
df.replace("N/A", np.nan, inplace=True)

df['dribbling'] = df['dribbling'].fillna(df['dribbling'].mean())
df['pace'] = df['pace'].fillna(df['pace'].mean())
```

۲. مقدار Min و Max به معنای کمترین و بیشترین سن ممکنه هستند . همچنین Q۱ نشان دهنده ی آن است که از چه سنی به قبل شامل ۲۵ درصد داده هاست (چارک اول).همچنین Q۲ برای ۵۰٪ داده ها (میانه) و Q۳ برای ۷۵٪ داده هاست (چارک سوم)

```
plt.boxplot(df['age'])
plt.title('Box Plot')
plt.ylabel('Age')
plt.show()

print("Min=", df['age'].describe()['min'])
print("Q1=", df['age'].describe()['25%'])
print("Q2=", df['age'].describe()['50%'])
print("Q3=", df['age'].describe()['75%'])
print("Max=", df['age'].describe()['max'])
```

```
Min= 17.0
Q1= 23.0
Q2= 26.0
Q3= 30.0
Max= 88.0
```



۳. برای این بخش تابع می نویسیم تا به ازای n (اندازه ی نمونه) نمونه برداری و محاسبه میانگین و انحراف معیار را انجام دهد و نمودار Q-Q را رسم کرده و در نهایت p_value آزمون شاپیرو ویلک را نمایش دهد تابع به صورت زیر است :

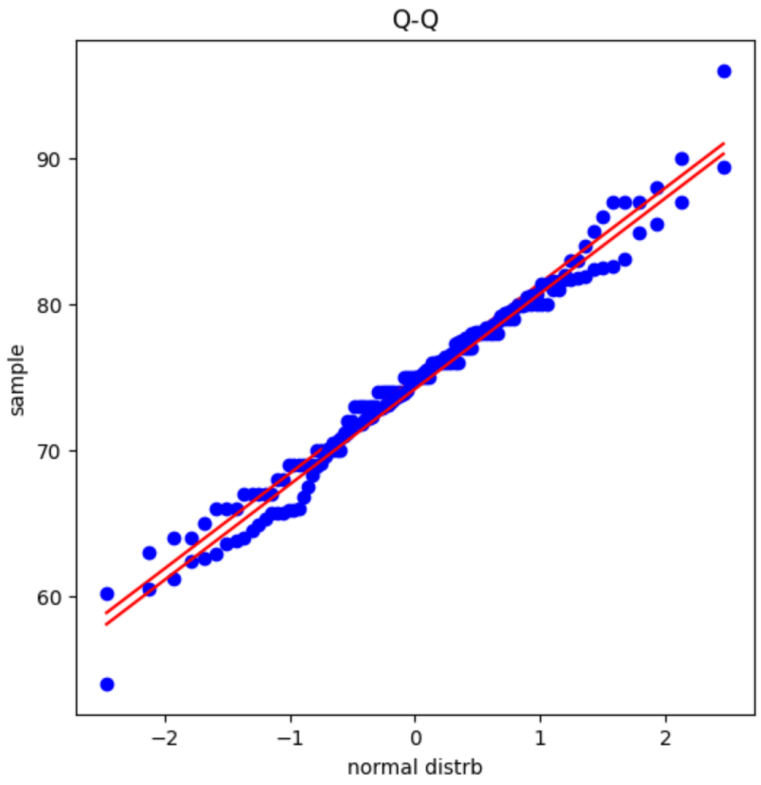
```
from scipy.stats import probplot
import scipy.stats as stats

def compare(n):
  sample = np.random.choice(df['weight'].dropna(), size=n)
  sample_mean = np.mean(sample)
  sample_std = np.var(sample)**(0.5)
  normal_distrb = np.random.normal(loc=sample_mean, scale=sample_std, size=n)
  print("mean=",sample_mean)
  print("std=",sample_std)

  plt.figure(figsize=(6, 6))
  probplot(sample, plot=plt)
  probplot(normal_distrb, plot=plt)
  plt.title('Q-Q')
  plt.xlabel('normal distrb')
  plt.ylabel('sample')
  plt.show()

  statistic, p_value = stats.shapiro(sample)
  print("p-value=",p_value)
```

آ.

```
mean= 74.9
std= 6.475337829024832
```

ب. به اندازه ای که این نمودار شبیه خط شود یعنی نمونه داده با توزیع مورد نظر شباهت دارد.

ج. نمودار Q-Q به دست آمده به شکل زیر است همانطور که مشاهده می شود این نمودار تا اندازه ای به خط صاف شباهت دارد پس به صورت شهودی می توان گفت نمونه به توزیع نرمال شباهت دارد.



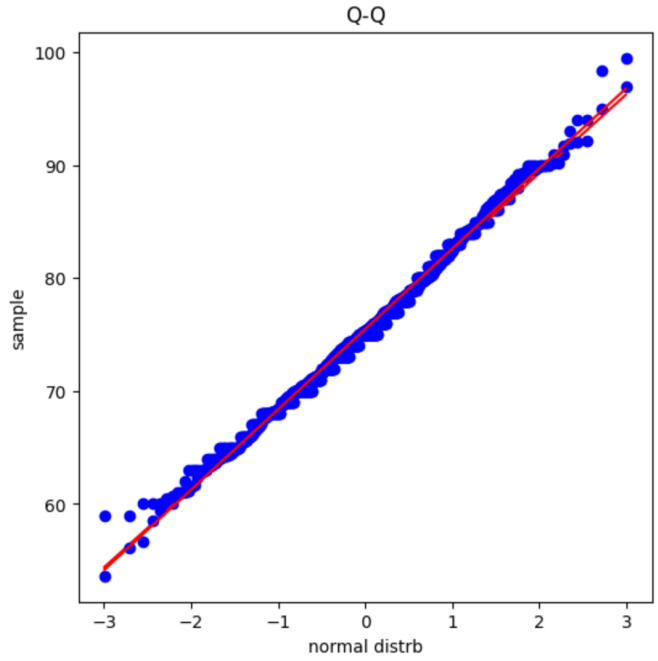
د. p-value به دست آمده از آزمون برابر با :

```
p-value= 0.16473481059074402
```

که مقداری بیشتر از ۰.۰۵ دارد و بنابراین می توان پذیرفت که این نمونه به توزیع نرمال شباهت دارد.

ه. با تکرار مراحل برای n=۵۰۰ نتایج و نمودار زیر به دست می آید :

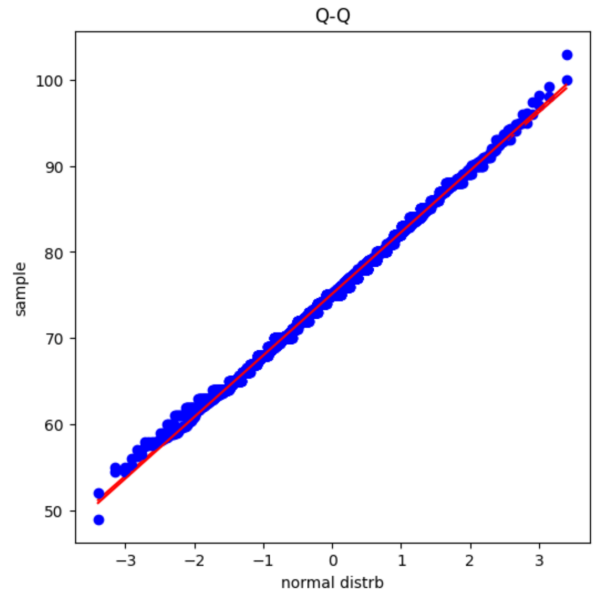
```
mean= 75.326
std= 6.985679923958727
```




```
p-value= 0.0025564595125615597
```

که همانطور که دیده می شود نمودار بیشتر به خط مستقیم شبیه شده است اما p-value کاهش یافته که این پدیده ناشی از آن است که با افزایش اندازه ی نمونه نمونه به طور شهودی شباهت بیشتری به توزیع نرمال پیدا می کند اما در عوض با افزایش نمونه آزمون شاپیرو ویلک حساسیت بیشتری به تطابق کامل نمونه و توزیع نرمال پیدا می کند که سبب کاهش p-value می شود. همچنین این مشاهدات برای نمونه با اندازه ی ۲۰۰۰ تکرار می شود :

```
mean= 75.101
std= 7.044274199660318
```

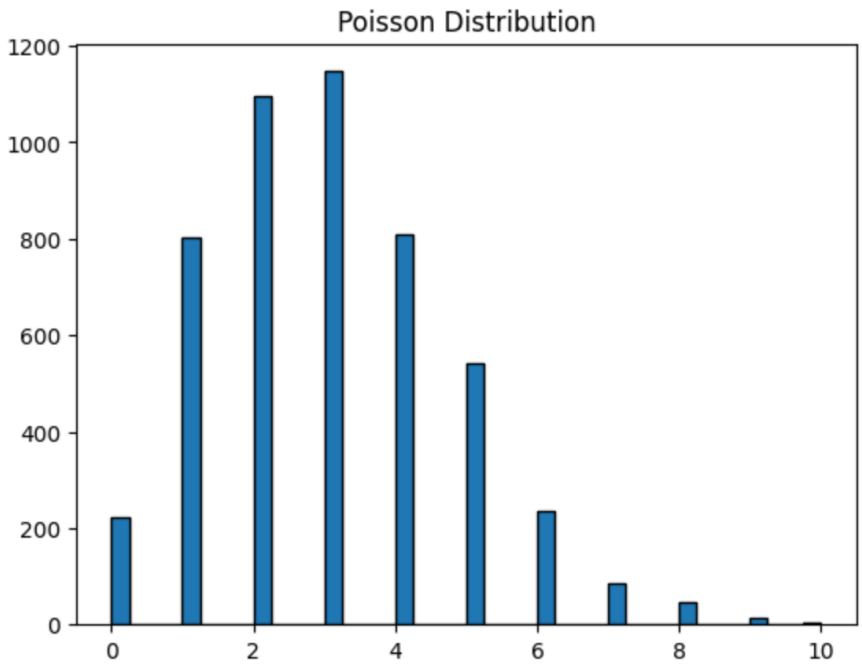



```
p-value= 6.333292276394786e-06
```

۴.

آ. به وسیله ی کد زیر نمونه برداری کرده و هیستوگرام مورد نظر را رسم می کنیم:

```
import numpy as np
import matplotlib.pyplot as plt

plt.hist(np.random.poisson(3, size=5000), bins=40, edgecolor='black')
plt.title('Poisson Distribution')
plt.show()
```



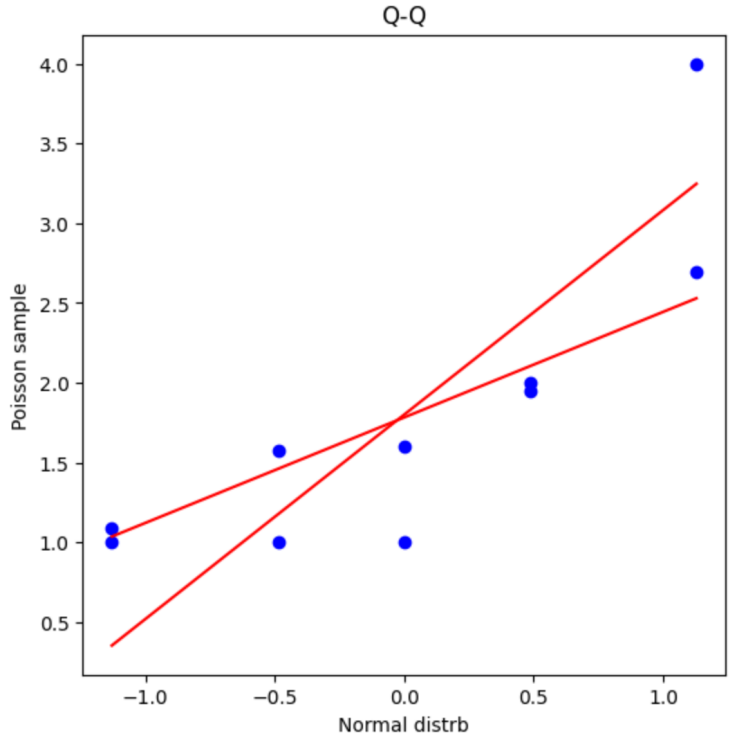
ب. تابعی برای نمونه برداری و محاسبه p-value و رسم نمودار Q-Q به شکل زیر طراحی می کنیم و آن را با مقادیر ۵ و ۵۰ و ۵۰۰۰ نمایش می دهیم:

```
def Poisson_Normal_compare(n):
  poisson_sample = np.random.poisson(3, size=n)
  normal_distrb = np.random.normal(loc=np.mean(poisson_sample), scale=np.var(poisson_sample)**(0.5), size=n)

  plt.figure(figsize=(6, 6))
  probplot(poisson_sample, plot=plt)
  probplot(normal_distrb, plot=plt)
  plt.title('Q-Q')
  plt.xlabel('Normal distrb')
  plt.ylabel('Poisson sample')
  plt.show()

  statistic, p_value = stats.shapiro(poisson_sample)
  print("p-value=",p_value)
  print(".")

Poisson_Normal_compare(5)
Poisson_Normal_compare(50)
Poisson_Normal_compare(5000)
```

نتایج به دست آمده به شکل زیر است :

نمونه با اندازه ی ۵ :

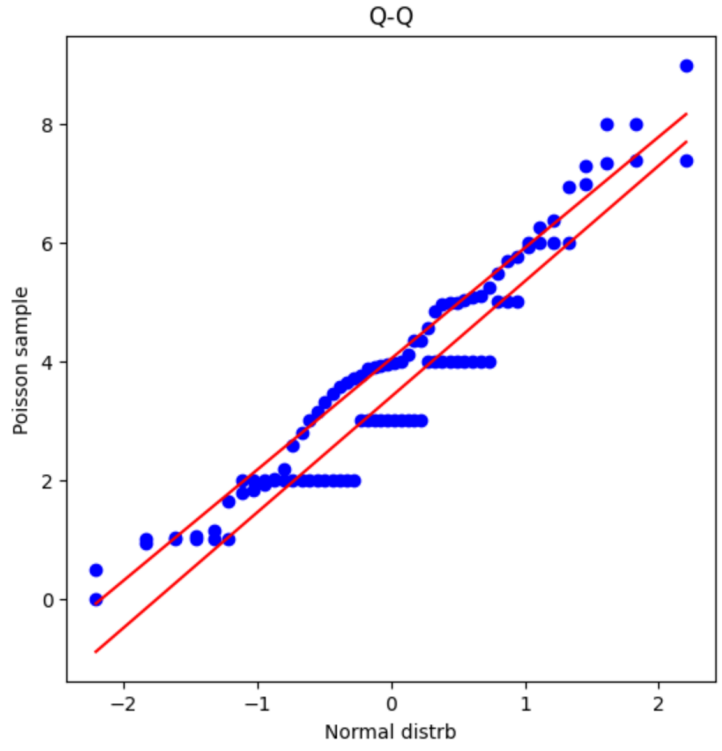
همانطور که در پایین مشاهده می شود این نمونه چه از لحاظ شهودی و چه از لحاظ آزمون فاصله ی زیادی با توزیع نرمال دارد که علت این موضوع اندازه کوچک نمونه است که در قضیه حد مرکزی بررسی شده است با افزایش اندازه نمونه بر اساس قضیه ی حد مرکزی شباهت این توزیع به توزیع نرمال افزایش خواهد یافت.



p-value= 0.021380668506026268

نمونه با اندازه ی ۵۰ :

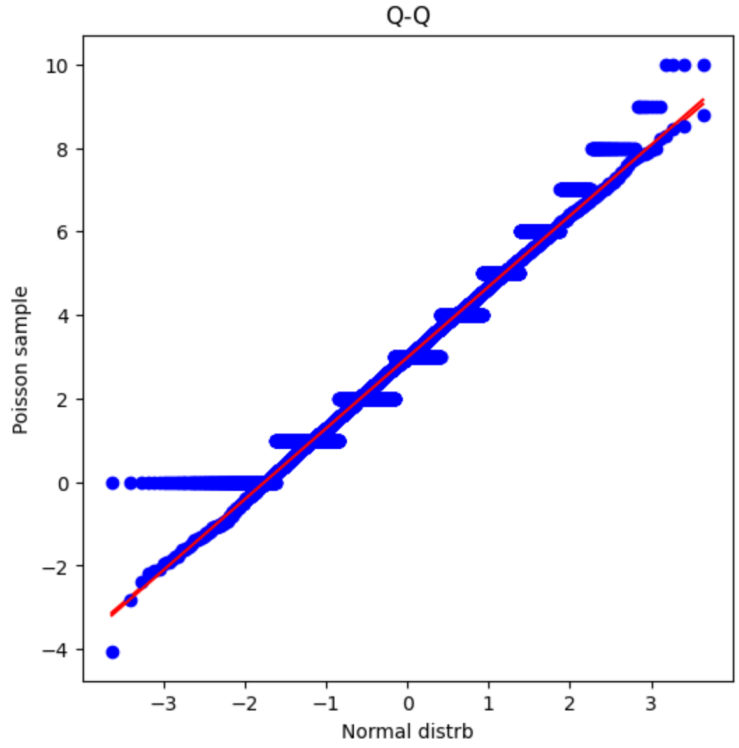
نمودار کاملا به شکل خط درنیامده ست اما شباهت شکل کلی آن به خط افزایش یافته است علت کاهش p-value اما این است که این آزمون با افزایش اندازه ی نمونه حساسیت بیشتری به شباهت کامل نمونه به توزیع نرمال پیدا می کند (فقط شکل شبه نرمال کافی نیست.) و بنابراین p-value به دست آمده از آزمون کمتر می شود. بر اساس قضیه حد مرکزی به علت افزایش اندازه ی نمونه شباهت شکل کلی نمونه به توزیع نرمال افزایش یافته است.



p-value= 0.0012321919202804565

نمونه با اندازه ی ۵۰۰۰:

مجددا همانند افزایش نمونه از ۵ به ۵۰ روند تکرار می شود اما شکل نمودار مقدار بیشتری از خط مستقیم تخطی می کند که نشانه از دور شدن نمونه از توزیع نرمال است.



p-value= 1.5552724389355958e-38