به نام خدا

دانشگاه تهران

درس امار و احتمال مهندسی

تمرین کامپیوتری سوم

محمد طاها مجلسی
۸۱۰۱۰۱۵۰۴

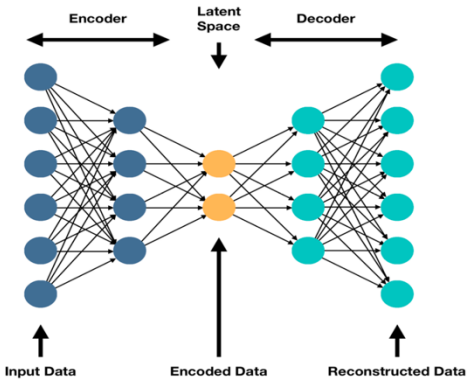
پاسخ بعضی سوالات داخل نوتبوک موجود است
سوال ۱ :

۱. Mean Squared Error **(۳۵) نمره**

مدل‌های «خود رمزگذار» (Autoencoder) یکی از مهم‌ترین مدل‌های حوزه یادگیری ماشین و یادگیری عمیق (Deep Learning) هستند. اتوانکودرها یک نوع شبکه عصبی هستند که به طور خاص برای فشرده‌سازی و بازسازی داده‌ها طراحی شده‌اند. توانایی اصلی یک اتوانکودر در بازسازی داده است؛ بدین معنا که می‌تواند یک ورودی را فشرده کرده و سپس بازسازی کند. یک اتوانکودر از دو بخش اصلی "Encoder" و "Decoder" تشکیل شده است:

- **انکودر (Encoder):** این قسمت وظیفه تبدیل داده ورودی به فضای نهان (Latent) را دارد. انکودر، ویژگی‌های مهم و معنادار، که معمولا بُعد کمتری از ورودی دارند، را از داده‌های ورودی استخراج می‌کند.
- **دیکودر (Decoder):** وظیفه‌ی این بخش، بازسازی داده از فضای نهان است. دیکودر تلاش می‌کند با استفاده از ویژگی‌های کم‌بعد (Low Dimensional Features) تولیدشده توسط انکودر، داده را به شکل اصلی ورودی بازسازی کند.

در واقع، اتوانکودرها می‌توانند با ایجاد یک نمایش کم‌بعد از داده‌های ورودی، اطلاعات مهمی از داده را بازیابی و استخراج کنند.

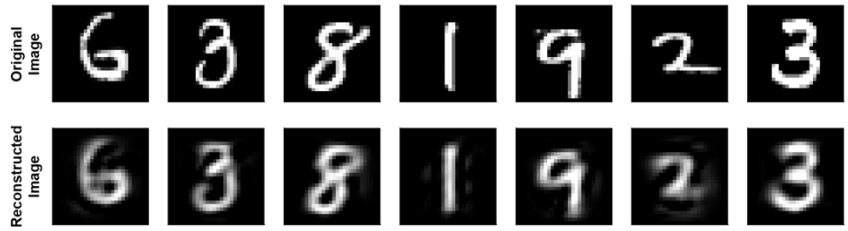


۱

۱ ـ دو مورد از کاربردهایی که اتوانکودرها می‌توانند داشته باشند را به اختصار بنویسید.

با اتو انکدر ها میتوانیم حجم تصاویر و همین طور برای فشرده سازی میتوانیم استفاده کنیم .
یعنی تا حدودی تصویر فشرده خواهد شد.
همین طور هم برای کاهش بعد میتواذ ازاتو انکدر ها استفاده کرد کاهش بعد می‌تواند برای بهبود عملکرد الگوریتم‌های یادگیری ماشین، کاهش حجم داده‌ها و تسهیل ذخیره‌سازی و انتقال داده‌ها استفاده شود.
اتوانکودرها می‌توانند برای استخراج ویژگی از داده‌ها استفاده شوند. این کار با یادگیری یک فضای نهان که ویژگی‌های مهم داده‌ها را در خود نگه می‌دارد انجام می‌شود. استخراج ویژگی می‌تواند برای بهبود عملکرد الگوریتم‌های یادگیری ماشین، تحلیل داده‌ها و تشخیص الگوها استفاده شود.
یعنی میتوانند بعضی از ویژگی هارا در فضای نهان ذخیره خواهند کرد .

۲ ـ داده‌های بازسازی‌شده به کمک اتوانکودر دارای خطا هستند. در تصویر زیر نمونه‌هایی از تصاویر دیتاست mnist، که شامل تصاویر دست‌نویس از اعداد 0 تا 9 است، به همراه تصاویر بازسازی‌شده آن‌ها توسط اتوانکودر نمایش داده شده است. مشاهده می‌کنید که تصاویر بازسازی‌شده نسبت به تصاویر اصلی تار (Blur) هستند.



در مورد دلیل وجود این خطا و ارتباط آن با اندازه فضای پنهان (Latent) تحقیق کنید و نتیجه را به طور خلاصه بیان کنید.

میدانیم که اتو اندکدر ها یک الگوریتم یادگیری ماشین هستند که بیشتر برای کاهش عمق تصاویر کاربرد دارند .
در فرآیند تبدیل داده ها به فضای کم بعدی، ممکن است برخی از اطلاعات از دست بروند. این امر منجر به خطا در بازسازی داده ها می شود.
ارتباط اندازه فضای پنهان با خطا در بازسازی داده ها

اندازه فضای پنهان یکی از عواملی است که بر میزان خطا در بازسازی داده ها تأثیر می گذارد. هرچه اندازه فضای پنهان کوچکتر باشد، اطلاعات کمتری در آن ذخیره می شود و در نتیجه، خطا در بازسازی داده ها بیشتر می شود.

به نوعی هر چه قدر که فضای نهان بیشتری داشته باشیم مقادیر بیشتری را میتواند ذخیره کند و بعد دوباره برای دکود کردن استفاده کند .

۳ـ در این بخش می‌خواهیم تعدادی از تصاویر دیتاست mnist را توسط یک اتوانکودر از پیش آموزش داده شده (Pre-trained) بازسازی کنیم و میزان خطای بین تصاویر بازسازی‌شده و تصاویر اصلی متناظر را بدست آوریم. در فایل mnist_AE.h5 که در ضمیمه این تمرین قرار داده شده است، یک مدل اتوانکودر از پیش آموزش داده شده، ذخیره شده است.

آ. به کمک کد زیر دیتای تست دیتاست mnist را لود و پیش پردازش کنید.

```
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
```

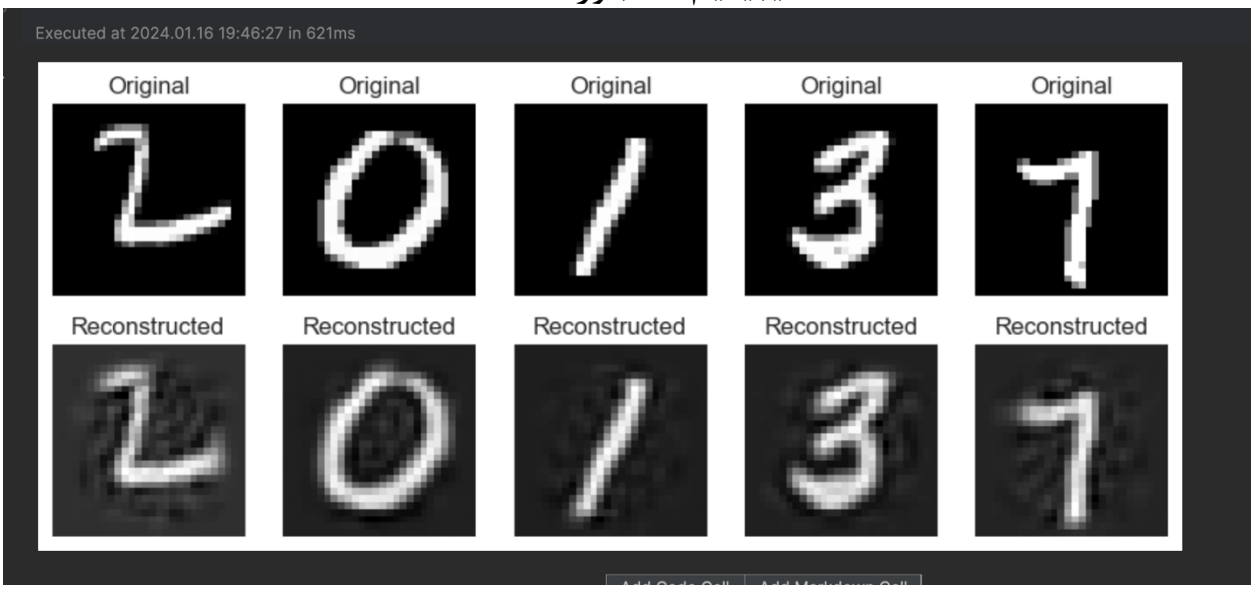
نهایتا متغیر test_images ، آرایه‌ای از 10000 تصویر که Flatten شده‌اند و مقادیر هر پیکسل آن به بازه 0 تا 1 اسکیل شده است، می‌باشد. منظور از Flatten کردن یک عکس، کنار هم قرار دادن سطرهای آن به منظور تبدیل آن به آرایه یک بعدی است. از آنجایی که اندازه هر تصویر دیتاست mnist برابر 28*28 پیکسل می‌باشد، ابعاد متغیر test_images برابر (10000 , 28*28=784) می‌باشد.

ب. به کمک قطعه کد زیر، مدل اتوانکودر Pre-trained را لود کنید و تصاویر test_images را به کمک آن بازسازی کنید.

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

ج. 4 نمونه از تصاویر test_images را به همراه تصویر بازسازی‌شده متناظر با آن رسم کنید. (توجه داشته باشید که برای نمایش تصاویر Flatten شده در test_images، باید آن‌ها را به آرایه‌ای دو بعدی با ابعاد 28*28 تغییر شکل دهید)

که بدین شکل میشوند :
میبینیم که بلور شده اند

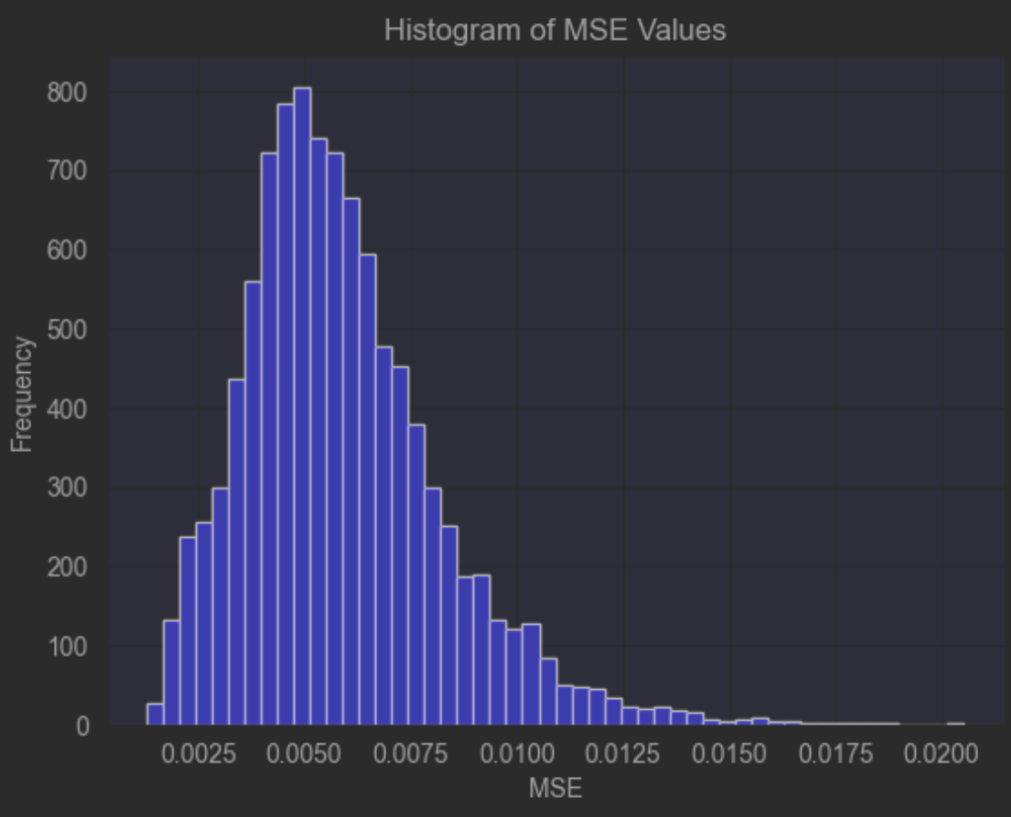


د. تابعی برای محاسبه «میانگین مربع خطاها» (Mean Squared Error) بنویسید. سپس میزان MSE برای تمامی این 10000 تصویر بازسازی شده را بدست آورید. در نهایت هیستوگرام MSE ها را رسم کنید. (توجه داشته باشید که استفاده از کتابخانه یا تابع آماده برای محاسبه MSE مجاز نیست)

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

def calculate_mse(original_images, reconstructed_images):
    mse_values = np.mean((original_images - reconstructed_images)**2, axis=(1,))
    return mse_values

mse_values = calculate_mse(test_images, reconstructed_images)

average_mse = np.mean(mse_values)
print(f"Average MSE: {average_mse}")

plt.hist(mse_values, bins=50, color='blue', edgecolor='black')
plt.title('Histogram of MSE Values')
plt.xlabel('MSE')
plt.ylabel('Frequency')
plt.show()
```

Executed at 2024.01.16 19:46:27 in 221ms

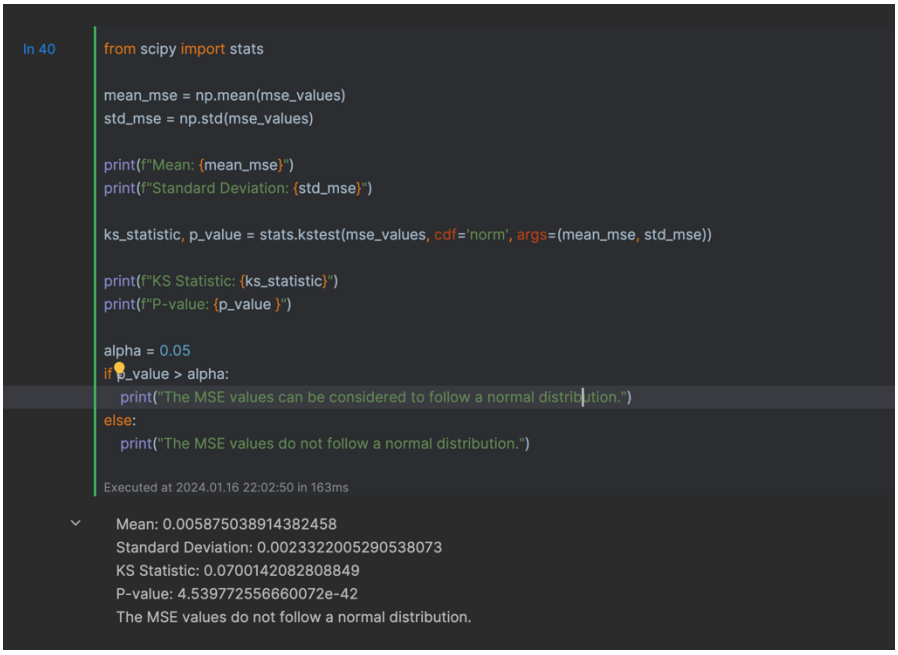
```
Average MSE: 0.005875038914382458
```



ه. در ادامه به کمک «آزمون کولموگروف–اسمیرنوف» میخواهیم بررسی کنیم آیا 10000 MSE تصویر بازسازی شده دارای توزیع نرمال هستند یا خیر. آزمون کولموگروف–اسمیرنوف نوعی آزمون نیکوئی برازش (Goodness of Fit) برای مقایسه یک توزیع نظری با توزیع مشاهده شده است. ابتدا با محاسبه میانگین و انحراف معیار نمونه‌ای MSE ها، μ و σ توزیع نرمال فرضی را بدست آورید. سپس به کمک دستور زیر آزمون نیکویی برازش کولموگروف–اسمیرنوف را روی داده‌های MSE انجام دهید.


```
from scipy import stats
ks_statistic, p_value = stats.kstest(data, cdf='norm', args=(mean, std))
```

بر اساس p__value بدست آمده، تعیین کنید که آیا میتوان پذیرفت که داده‌های MSE از توزیع نرمال با μ و σ برآورد شده پیروی می‌کنند یا خیر؟



چون که مقدار p_value خیلی کم شد پس نمیتواند یک توزیع نرمال داشته باشد .

سوال ۲ :

هشت نقطه اصلی و سه نقطه دیگر را که در جدول‌های زیر داده شده‌اند در نظر بگیرید. این سه نقطه به ترتیب از چپ به راست، نقطه «پرت» [۱] ، نقطه «اهرمی» (نافذ) [۲] و نقطه‌ای با هر دو ویژگی «دور افتادگی» و «اهرمی» هستند.

x	-2.3	-1.1	0.5	3.2	4.0	6.7	10.3	11.5
y	-9.6	-4.9	-4.1	2.7	5.9	10.8	18.9	20.5

x	5.8	20.4 (L)	20.4 (L)
y	31.3 (O)	14.1	31.3 (O)

۱ ـ در مورد نقاط پرت و نقاط اهرمی و نقاطی با هر دو ویژگی تحقیق کنید و تاثیر منفی این نقاط را بر معادله رگرسیونی توضیح دهید.

نقاط پرت

نقاط پرت، نقاطی هستند که از سایر نقاط داده در یک مجموعه داده فاصله زیادی دارند. این نقاط می توانند بر معادله رگرسیون تأثیر منفی بگذارند، زیرا می توانند خط را به سمت خود بکشانند و باعث شوند که خط کمتر دقیق باشد.

نقاط اهرم

نقاط اهرم، نقاطی هستند که دارای مقادیر بزرگی از متغیر مستقل هستند. این نقاط می توانند بر معادله رگرسیون تأثیر منفی بگذارند، زیرا می توانند خط را به سمت خود بکشانند و باعث شوند که خط کمتر دقیق باشد.

به طور مثال در بالا ایکس مقدار بزرگی دارد

نقاطی با هر دو ویژگی پرت و اهرم

نقاطی با هر دو ویژگی پرت و اهرم، تأثیر منفی بیشتری بر معادله رگرسیون نسبت به هر کدام از این نقاط به تنهایی دارند. این نقاط می توانند خط را به سمت خود بکشانند و باعث شوند که خط بسیار نادرست باشد.

۲ ـ «ضریب تعیین» [۳] (R^2) یکی از شاخص‌هایی است که میزان ارتباط خطی بین دو متغیر را اندازه‌گیری می‌کند. این ضریب می‌تواند به عنوان شاخصی برای بررسی نیکوئی برازش رگرسیون خطی استفاده شود. در مورد این ضریب تحقیق و به صورت خلاصه آن را توضیح دهید.

این ضریب بیانگر میزان رگرسیون میباشد

ضریب تعیین نشان می‌دهد که چند درصد از تغییرات متغیر وابسته توسط متغیرهای مستقل توضیح داده می‌شود یکی از شاخص‌های برازش مدل است که قدرت پیش‌بینی متغیر وابسته (ملاک) براساس متغیرهای مستقل (پیش‌بین) را نشان می‌دهد. مقدار این شاخص بین صفر تا یک می‌باشد و اگر از ۶/۰ بیشتر باشد نشان می‌دهد متغیرهای مستقل تا حد زیادی توانسته‌اند تغییرات متغیر وابسته را تبیین کنند. به نوعی در حال توضیح این است که مدل چه مقدر قابلیت پیش بینی دارد

به این دو صورت میتواند تعریف شود :

$$R^2 = \frac{\text{تغییرات توضیح داده شده در Y}}{\text{تغییرات کل در Y}} = \frac{\sum (\hat{Y}_t - \bar{Y})^2}{\sum (Y_t - \bar{Y})^2}$$

تغییرات توضیح داده نشده + تغییرات توضیح داده شده = کل تغییرات

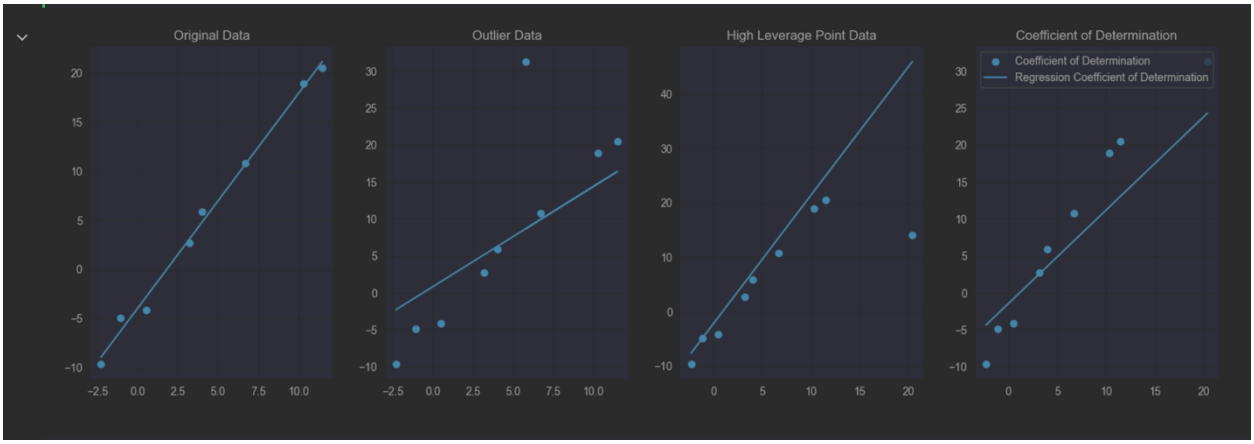
TSS = ESS + RSS

$$R^2 = \frac{ESS}{TSS} = \frac{TSS - RSS}{TSS} = 1 - \frac{RSS}{TSS} = 1 - \frac{\sum \hat{u}_i^2}{\sum (Y_t - \bar{Y})^2}$$

۳ـ تاثیر نقاط غیر عادی را با اجرای 4 رگرسیون خطی جداگانه به شرح زیر بررسی کنید.

- رگرسیون بر پایه هشت داده اصلی
- رگرسیون بر پایه هشت داده اصلی به اضافه نقطه دور افتاده
- رگرسیون بر پایه هشت داده اصلی به اضافه نقطه اهرمی
- رگرسیون بر پایه هشت داده اصلی به اضافه نقطه دور افتاده_اهرمی

از روش رگرسیون خطی مبتنی بر روش کمترین مربعات خطا (Least Squares) استفاده کنید. در هر یک از چهار رگرسیون فوق، نمودار داده‌ها همراه با خط رگرسیون را در یک صفحه رسم کنید و ضریب تعیین (R^2)هر یک را بیان کنید. (توجه داشته باشید که استفاده از کتابخانه یا تابع آماده برای پیاده سازی رگرسیون خطی مجاز نیست)



همین گونه که میبینیم اگر دیتا ها درست باشند و دیتای اضافه ای نداشته باشیم در ا
صورت تقریبا به شیب خوبی میرسیم و روی دیتا هایی که داریم مچ خواهند شد

مطابق شکل پایین که میبینیم شیب هایی که دارند با هم تفاوت دارند

```
Original Data: Slope=2.18, Intercept=-3.90
Outlier Data: Slope=1.35, Intercept=0.86
High Leverage Point Data=2.36, Intercept=-2.19
Coefficient of Determination Data=1.26, Intercept=-1.42
```

۴ـ راهکارهایی برای یافتن مدل رگرسیونی بهتر (نسبت به مدل مبتنی بر کمترین مربعات خطا) در حضور نقطه دور افتاده و یا اهرمی پیشنهاد کنید.

بخش سوم :

در حضور نقطه دور افتاده و یا اهرمی، مدل رگرسیون مبتنی بر کمترین مربعات خطا می‌تواند به‌طور غیرواقعی به سمت نقطه دور افتاده یا اهرمی متمایل شود .این امر می‌تواند منجر به تخمین نادرست پارامترهای معادله رگرسیون و کاهش دقت مدل شود.

برای یافتن مدل رگرسیون بهتر در حضور نقطه دور افتاده و یا اهرمی، می‌توان از روش‌های زیر استفاده کرد:

- حذف نقاط دور افتاده :این روش ساده‌ترین روش برای کاهش تأثیر نقاط دور افتاده است .در این روش، نقاط دور افتاده از مجموعه داده حذف می‌شوند.
- استفاده از توابع وزنی :در این روش، وزنهای متفاوتی به نقاط داده اختصاص داده می‌شود .وزن نقاط دور افتاده کمتر از وزن نقاط دیگر است .این امر باعث می‌شود که تأثیر نقاط دور افتاده بر معادله رگرسیون کاهش یابد.
- استفاده از روش‌های رگرسیون مقاوم :این روش‌ها به‌گونه‌ای طراحی شده‌اند که تأثیر نقاط دور افتاده و اهرمی را کاهش دهند.

در ادامه، به بررسی هر یک از این روش‌ها می‌پردازیم:

حذف نقاط دور افتاده

حذف نقاط دور افتاده می‌تواند تأثیر قابل‌توجهی بر دقت مدل رگرسیون داشته باشد .با این حال حذف نقاط دور افتاده می‌تواند منجر به از دست رفتن اطلاعات شود .بنابراین، قبل از حذف نقاط دور افتاده، باید اطمینان حاصل کنید که این نقاط واقعاً نقطه دور افتاده هستند و حذف آنها منجر به از دست رفتن اطلاعات مهم نمی‌شود.

برای تشخیص نقطه دور افتاده، می‌توان از روش‌های زیر استفاده کرد:

- آزمون فاصله :در این آزمون فاصله بین نقطه داده و خط رگرسیون محاسبه می‌شود .اگر این فاصله بیش از یک مقدار آستانه باشد، نقطه داده به‌عنوان نقطه دور افتاده در نظر گرفته می‌شود.
- آزمون همبستگی :در این آزمون همبستگی بین نقطه داده و متغیر مستقل محاسبه می‌شود .اگر این همبستگی بیش از یک مقدار آستانه باشد، نقطه داده به‌عنوان نقطه دور افتاده در نظر گرفته می‌شود.

استفاده از توابع وزنی

در این روش، وزنهای متفاوتی به نقاط داده اختصاص داده می‌شود .وزن نقاط دور افتاده کمتر از وزن نقاط دیگر است .این امر باعث می‌شود که تأثیر نقاط دور افتاده بر معادله رگرسیون کاهش یابد.

سوال ۳ :

۳. Central Limit Theorem & Sampling **(۴۰) نمره**

تصویر یک **«تابلوی گالتون» (Galton Board)** ـ در این تخته، تعداد زیادی توپ از بالا به پایین سرازیر میشوند. در طی مسیر چندین لایه از موانع وجود دارند که هر توپ در هر مرحله با برخورد به این موانع، به یکی از دو سمت راست یا چپ منحرف می‌شوند. این توپ‌ها نهایتا توزیعی شبیه توزیع نرمال ایجاد می‌کنند.

دیتاست ضمیمه‌شده FIFA2020.csv شامل اطلاعات مربوط به بهترین بازیکنان تاریخ فوتبال جهان تا سال 2020 می‌باشد که شامل ستون‌هایی مانند: ملیت (nationality)، امتیاز (overall)، وزن (weight)، قد (height) ، توانایی شوت زدن (shooting) ، توانایی دریبل زدن (dribbling) ، سرعت (pace) و... می باشد. در واقع هر یک از ستون‌ها یک متغیر تصادفی می‌باشد.
برای لود کردن دیتاست از دستور زیر استفاده کنید.

```
import pandas as pd
df = pd.read_csv('FIFA2020.csv', encoding = "ISO-8859-1")
```

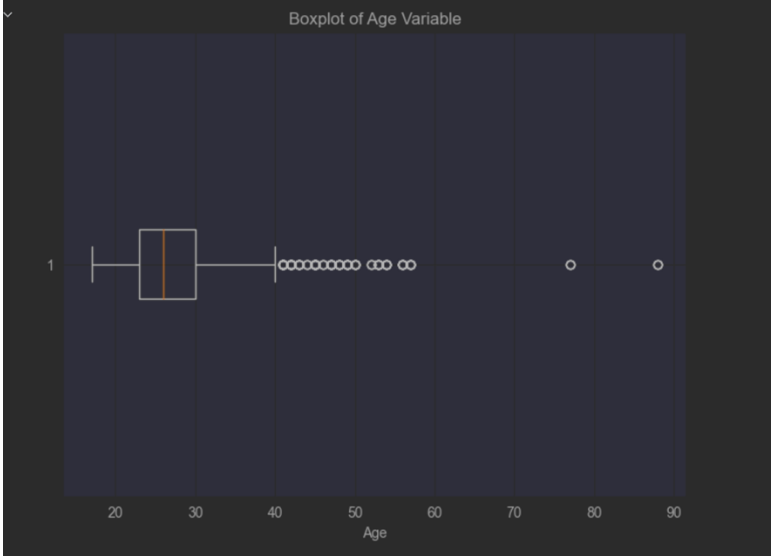
۱ ـ ،در این دیتاست، تعدادی از داده‌های کمّی N/A (Not A Number) هستند و همچنین تعدادی از داده‌های کیفی، Icons هستند که نشان‌دهنده نامعلوم بودن این مقادیر می‌باشد. برای جایگزین کردن داده‌های کمّی نامعلوم چه راهکاری پیشنهاد می‌کنید؟ راهکار خود را برای داده‌های ستون (pace) و ستون (dribbling) پیاده کنید و دیتاست جدید را جایگزین دیتاست قبل کنید.

```
In 9    df = pd.read_csv('FIFA2020.csv', encoding='ISO-8859-1')


        print("Number of missing values before imputation:")
        print(df[['pace', 'dribbling']].isnull().sum())


        df['pace'].fillna(df['pace'].mean(), inplace=True)
        df['dribbling'].fillna(df['dribbling'].mean(), inplace=True)


        print("\nNumber of missing values after imputation:")
        print(df[['pace', 'dribbling']].isnull().sum())
        Executed at 2024.01.16 19:46:28 in 375ms

        Number of missing values before imputation:
        pace        2182
        dribbling   2182
        dtype: int64
```

۲ـ نمودار جعبه‌ای متغیر تصادفی age را رسم کنید و مقادیر (min, Q1, Q2, Q3, max) را بدست آورید. به صورت خلاصه توضیح دهید هر کدام از این مقادیر به چه معنا هستند.

```python
import matplotlib.pyplot as plt
plt.figure(figsize=(8, 6))
plt.boxplot(df['age'], vert=False)
plt.title('Boxplot of Age Variable')
plt.xlabel('Age')
plt.show()


age_stats = df['age'].describe()
print("\nSummary Statistics for 'age':")
print(age_stats)
```

Executed at 2024.01.16 19:46:29 in 224ms



۳ـ متغیر تصادفی weight را در نظر بگیرید و به صورت تصادفی و بدون جایگذاری، $n = 100$ نمونه از این متغیر انتخاب کنید:

آ. میانگین، واریانس و انحراف معیار این نمونه‌ها را بیابید.

ب. یکی از ابزارهایی که برای مقایسه شهودی دو توزیع به کار می‌رود، نمودار Q-Q می‌باشد. نحوه استفاده از این نمودار را در یک یا دو جمله توضیح دهید.

ج. یک نمونه‌ی $n=100$ تایی از توزیع نرمال با μ و σ (میانگین و واریانس نمونه‌ای n نمونه) برآورد شده در قسمت " آ " ایجاد کنید. سپس با استفاده این دو مجموعه n تایی و نمودار Q-Q، توزیع آماری وزن بازیکنان را با توزیع نرمال مقایسه کنید و نتیجه را تحلیل کنید.

د. در ادامه به کمک آزمون Shapiro-Wilk مشخص کنید که آیا توزیع آماری وزن 100 بازیکن انتخاب شده از توزیع نرمال پیروی می‌کند یا نه. آزمون «شاپیرو ویلک» (Shapiro-Wilk Test) از آزمون‌های برازش توزیع نرمال محسوب می‌شود. به کمک این آزمون می‌توان مشخص کرد که آیا داده‌ها از توزیع نرمال پیروی می‌کنند یا خیر. برای پیاده سازی این آزمون از کد زیر استفاده کنید:

```python
import scipy.stats as stats
statistic, p_value = stats.shapiro(data)
```

ه. سپس قسمت‌های " آ، ب، ج " را به ازای $n = 500, 2000$ تکرار کنید، چه نتیجه‌ای می‌گیرید؟

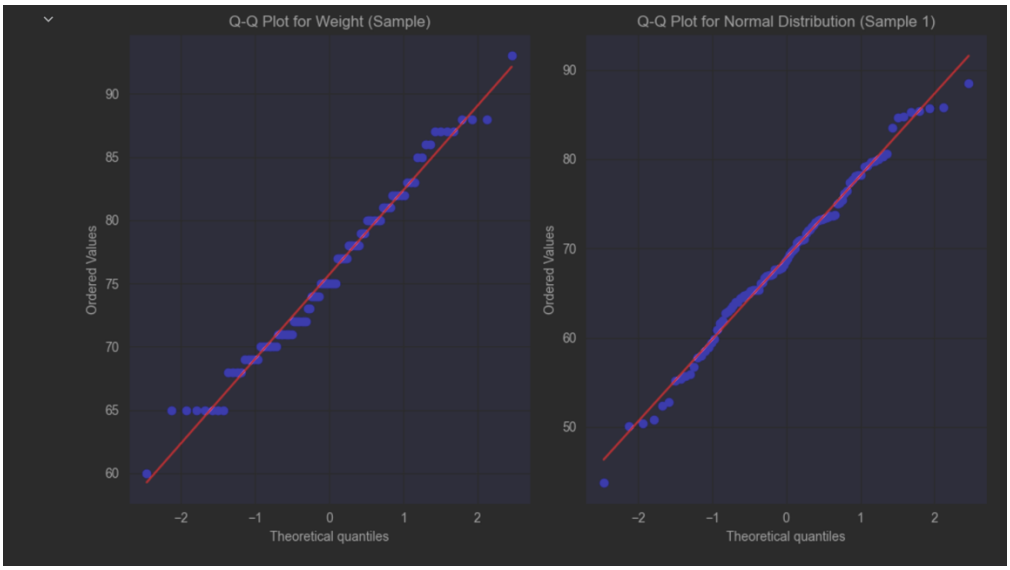
```
random_samples = np.random.choice(df['weight'].dropna(), size=100, replace=False)
```
Executed at 2024.01.16 19:46:29 in 267ms

```
print("Randomly selected 100 samples from the 'weight' variable:")
print(random_samples)
```
Executed at 2024.01.16 19:46:29 in 255ms

```
print("Mean of the 100 samples:")
print(random_samples.mean())
print("Standard deviation of the 100 samples:")
print(random_samples.std())
print("Variance of the 100 samples:")
print(random_samples.var())
```
Executed at 2024.01.16 19:46:29 in 247ms

```
Mean of the 100 samples:
75.79
Standard deviation of the 100 samples:
6.861916641872007
Variance of the 100 samples:
47.0859
```

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import probplot

mu = 70
sigma = 10

np.random.seed(42)
sample_normal1 = np.random.normal(mu, sigma, 100)

df = pd.read_csv('FIFA2020.csv', encoding='ISO-8859-1')
sample_weight = df['weight'].dropna().sample(100, replace=False).values

plt.figure(figsize=(10, 6))

plt.subplot(1, 2, 1)
probplot(sample_weight, dist='norm', plot=plt)
plt.title('Q-Q Plot for Weight (Sample)')

plt.subplot(1, 2, 2)
probplot(sample_normal1, dist='norm', plot=plt)
plt.title('Q-Q Plot for Normal Distribution (Sample 1)')

plt.tight_layout()
plt.show()
```



نمودار چندک چندک یا Q-Q plot به منظور مقایسه دو توزیع به کار گرفته می‌شود. از چنین نمودارهایی حتی می‌تواند مطابقت توزیع داده‌ها را با یک توزیع مشخص، مورد بررسی قرار داد. توسط نمودار چندک چندک یا Q-Q plotشکل توزیع‌ها مقایسه می‌شود و یک تصویر گرافیکی یا نمودار برای نمایش میزان مطابقت آن دو توزیع نشان داده می‌شود. چولگی، پارامتر مرکزی و پراکندگی در نحوه مقایسه دو توزیع مشکلی ایجاد نمی‌کنند و به راحتی می‌توان هم توزیع (Equal Distributed) بودن داده‌ها را دو گروه مقایسه کرد.
به طور کلی برای مقایسه کردن دو توزیع از این نمودار مینتوانیم استفاده کنیم.

```
data = sample_weight

import scipy.stats as stats
statistic, p_value = stats.shapiro(data)
print(statistic, p_value)
Executed at 2024.01.16 19:46:29 in 7ms

  0.9821552634239197 0.19498354196548462


 n = 500

n = 500
sample_weight = np.random.choice(df['weight'].dropna(), size=n, replace=False)

statistic, p_value = stats.shapiro(sample_weight)
print(statistic, p_value)
Executed at 2024.01.16 19:46:29 in 40ms

  0.9921923875808716 0.01013781689107418


n = 2000

n= 2000
sample_weight = np.random.choice(df['weight'].dropna(),size=n , replace=False)

statistic, p_value = stats.shapiro(sample_weight)
print(statistic, p_value)
Executed at 2024.01.16 19:46:29 in 35ms

  0.9957238435745239 1.8126154827768914e-05
```

هر چه میزان سمپل ها بیشتر باشد در نهایت مقدار p_valur کمتر خواهد شد که از نتایج معلوم است .

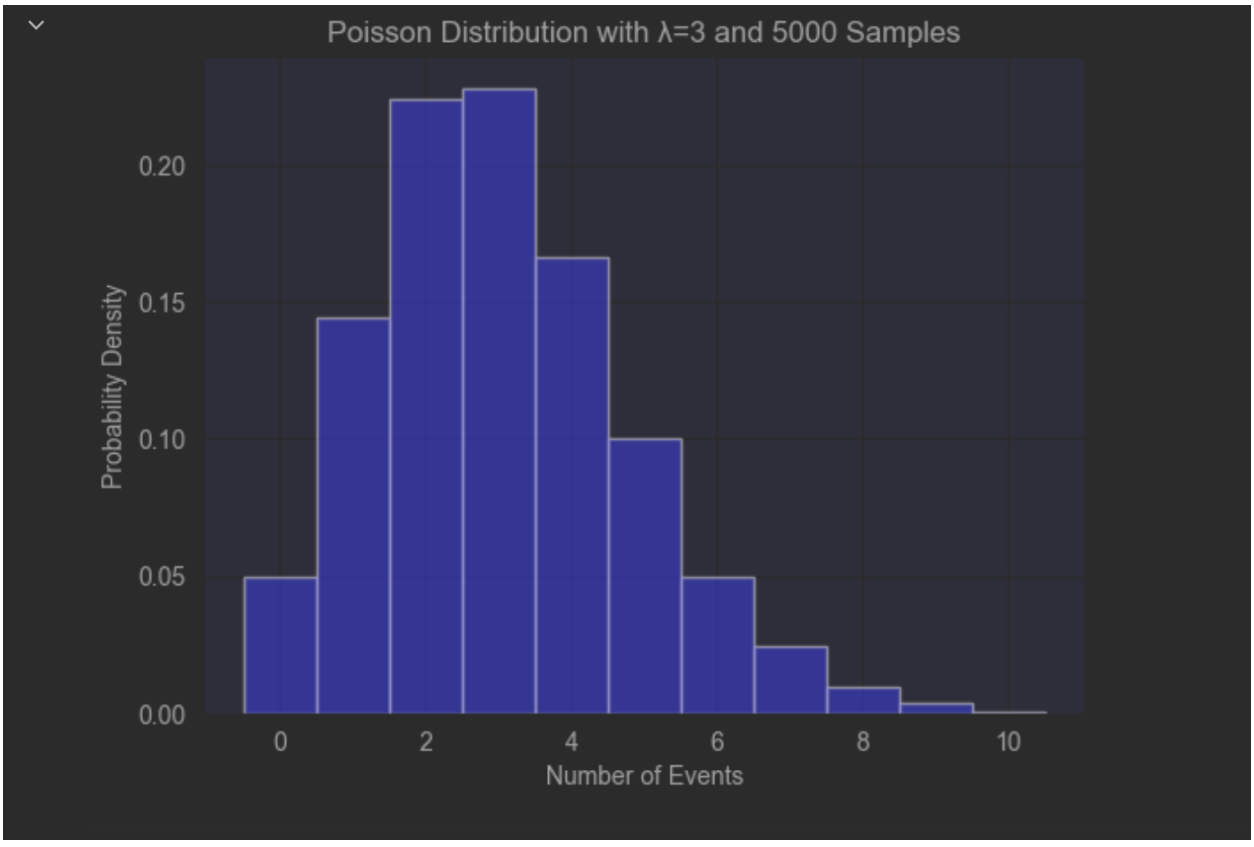
۴ــ یکی از توزیع‌های آماری مهم، «توزیع پواسون» (Poisson) است. این توزیع بیانگر رویدادهایی است که در طول زمان اتفاق می‌افتند و فقط میانگین فاصله‌ی بین این رویدادها را از داده‌های گذشته می‌دانیم:

$$P(X = x) = \frac{\lambda^x e^{-\lambda}}{x!} \qquad (x \in \mathbb{Z})$$

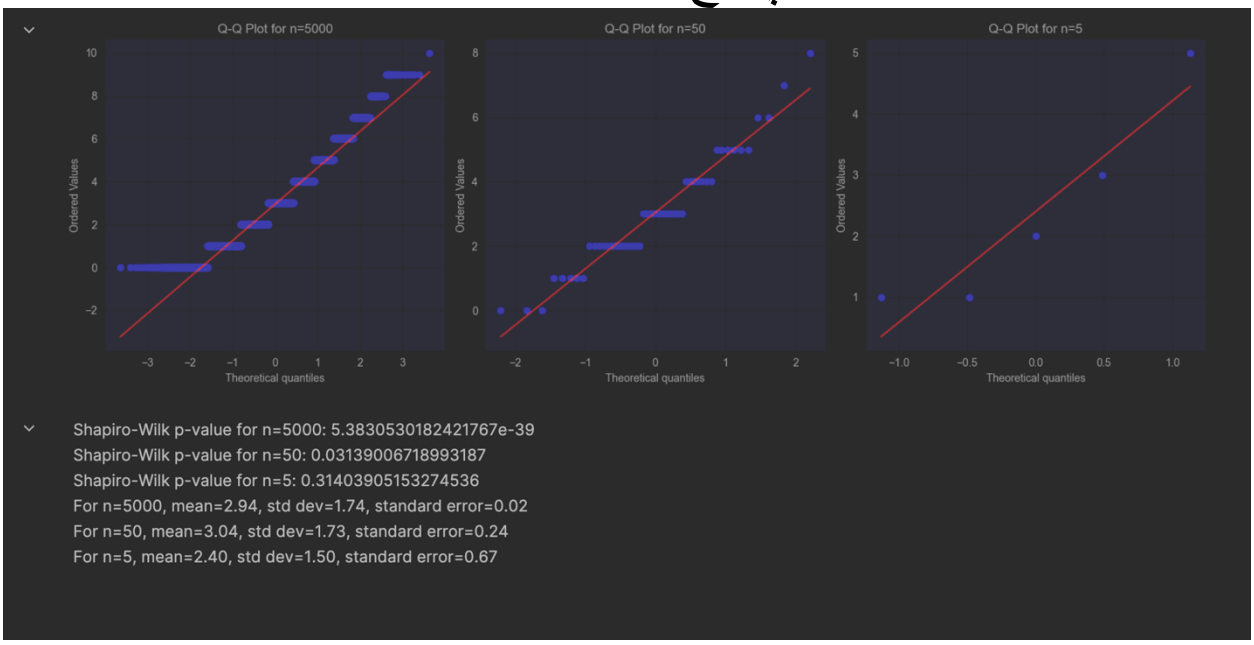
آ. به ازای $\lambda = 3$ تعداد $n = 5000$ از این توزیع بدون جایگذاری نمونه‌برداری کنید و هیستوگرام آن را رسم کنید.

ب. به ازای $n = 5, 50, 5000$ و $\lambda = 3$ با استفاده از نمودار Q-Q توزیع این نمونه‌ها را با توزیع نرمال مقایسه کنید. سپس p_value آزمون Shapiro-Wilk را برای هر یک بدست آورید و فرضیه نرمال بودن توزیع هر یک از این نمونه‌ها را آزمون کنید. نهایتا نتایج بدست آمده برای این 3 نمونه را بر اساس قضیه‌ی حد مرکزی (CLT) توجیه کنید.

پاسخ قسمت الف:



پاسخ قسمت ب :



همان طور که میبینیم هر چه که تعداد سمپل ها زیاد تر باشد میزان ارور کمتری پیدا میکنیم
همین طور هم به مقدار میانگین بهتر و واریانس بهتری هم نزدیک خواهیم شد .
مثلا برای ۵۰۰۰ مقدار میانگین بسیار به مقدار میانگین واقعی نزدیک است .