# سوال 1

کد:

nums = [

    \*[0.594, 0.928, 0.515, 0.055, 0.262, 0.797],

    \*[0.788, 0.442, 0.262, 0.797, 0.788, 0.442],

    \*[0.507, 0.351, 0.097, 0.798, 0.227, 0.127],

    \*[0.474, 0.825, 0.007, 0.182, 0.929, 0.852],

]

D = framework.Tests.ks(nums)

D\_alpha = 0.27

print(f"D = {D}")

print(f"D\_alpha = {D\_alpha}\n")

print(

    "Null hypothesis cannot be rejected. [Is uniform]"

    if D <= D\_alpha

    else "Null hypothesis is rejected. [Is not uniform]"

)

z\_0025 = 1.96

for start in range(len(nums)):

    for lag in range(1, len(nums) - start):

        z = framework.Tests.autoCorr(nums, start, lag)

        if abs(z) > 1.96:

            print("\nAutocorrelation detected with these values:")

            print(f"  start  = {start}")

            print(f"  lag    = {lag}")

            print(f"  z      = {z}")

            print(f"  z\_0025 = {z\_0025}")

رفرنس کد در فریمورک:

class Tests:

    @staticmethod

    def ks(nums: List[float | int]):

        nums = sorted(nums)

        n = len(nums)

        D\_plus = max((i + 1) / n - num for i, num in enumerate(nums))

        D\_minus = max(num - i / n for i, num in enumerate(nums))

        D = max(D\_plus, D\_minus)

        return D

    @staticmethod

    def autoCorr(nums: List[float | int], start: int, lag: int):

        pairs = (len(nums) - (start + 1)) // lag - 1

        nominator = (

            sum(

                nums[start + k \* lag] \* nums[start + (k + 1) \* lag]

                for k in range(pairs + 1)

            )

            / (pairs + 1)

            - 0.25

        )

        denominator = math.sqrt(13 \* pairs + 7) / (12 \* (pairs + 1))

        return nominator / denominator

خروجی:

D = 0.16300000000000003

D\_alpha = 0.27

Null hypothesis cannot be rejected. [Is uniform]

Autocorrelation detected with these values:

start = 1

lag = 9

z = 2.3649477803053496

z\_0025 = 1.96

Autocorrelation detected with these values:

start = 1

lag = 14

z = 2.2248984533740606

z\_0025 = 1.96

Autocorrelation detected with these values:

start = 1

lag = 18

z = 2.3385417874026904

z\_0025 = 1.96

Autocorrelation detected with these values:

start = 1

lag = 21

z = 2.7762790740314895

z\_0025 = 1.96

Autocorrelation detected with these values:

start = 1

lag = 22

z = 2.452185121431321

z\_0025 = 1.96

Autocorrelation detected with these values:

start = 5

lag = 17

z = 2.2243042932224903

z\_0025 = 1.96

Autocorrelation detected with these values:

start = 6

lag = 16

z = 2.1863823617165283

z\_0025 = 1.96

Autocorrelation detected with these values:

start = 9

lag = 13

z = 2.2243042932224903

z\_0025 = 1.96

Autocorrelation detected with these values:

start = 10

lag = 12

z = 2.1863823617165283

z\_0025 = 1.96

Autocorrelation detected with these values:

start = 11

lag = 4

z = 2.0136593004118826

z\_0025 = 1.96

Autocorrelation detected with these values:

start = 15

lag = 4

z = 2.310976254746033

z\_0025 = 1.96

Autocorrelation detected with these values:

start = 15

lag = 7

z = 2.228517841167597

z\_0025 = 1.96

Autocorrelation detected with these values:

start = 19

lag = 3

z = 2.342283635685482

z\_0025 = 1.96

Autocorrelation detected with these values:

start = 19

lag = 4

z = 2.054161317910548

z\_0025 = 1.96

Autocorrelation detected with these values:

start = 22

lag = 1

z = 2.4560494302033673

z\_0025 = 1.96

# سوال 2

کد:

for i in range(4):

    serviceTime = framework.RandomVariate.empirical(

        nums=[14, 30, 45, 60, 90, 120, 180, 300],

        frequencies=[10, 20, 25, 35, 30, 20, 10],

    )

    print(f"Customer #{i+1} service time: {serviceTime}")

رفرنس کد در فریمورک:

class Rng:

    @staticmethod

    def lcg(seed: int, modulus: int, multiplier: int, increment=0, returnInts=False):

        randomInt = seed

        while True:

            yield randomInt if returnInts else randomInt / modulus

            randomInt = (multiplier \* randomInt + increment) % modulus

class RandomVariate:

    generator = Rng.lcg(seed=123\_457, modulus=2\*\*31 - 1, multiplier=7\*\*5)

    @staticmethod

    def empirical(nums, frequencies):

        randomValue = next(RandomVariate.generator)

        total = sum(frequencies)

        index = -1

        cumulativeFrequency = 0

        for frequency in frequencies:

            relativeFrequency = frequency / total

            if cumulativeFrequency + relativeFrequency <= randomValue:

                cumulativeFrequency += relativeFrequency

                index += 1

            else:

                break

        x0 = nums[index]

        x1 = nums[index + 1]

        relativeFrequency = frequencies[index + 1] / total

        return x0 + (x1 - x0) / relativeFrequency \* (randomValue - cumulativeFrequency)

خروجی:

Customer #1 service time: 40.10740459622229

Customer #2 service time: 112.87262176995753

Customer #3 service time: 22.082180094012145

Customer #4 service time: 49.92902146183374

# سوال 3

کد:

def formatTime(time: float) -> str:

    hour = math.floor(currentTime)

    minutes = math.floor(60 \* (currentTime - hour))

    return f"{hour}:{"0" if minutes < 10 else ""}{minutes}"

customersCount = framework.RandomVariate.poisson(mean=100)

print(f"Randomly generated value for customers' count: {customersCount}")

currentTime = 8

arrivalTimes = []

for \_ in range(customersCount):

    interval = framework.RandomVariate.exponential(mean=8 / 100)

    currentTime += interval

    arrivalTimes.append(formatTime(currentTime))

print("\nArrival times:")

pp(arrivalTimes)

رفرنس کد در فریمورک:

class Rng:

    @staticmethod

    def lcg(seed: int, modulus: int, multiplier: int, increment=0, returnInts=False):

        randomInt = seed

        while True:

            yield randomInt if returnInts else randomInt / modulus

            randomInt = (multiplier \* randomInt + increment) % modulus

class RandomVariate:

    generator = Rng.lcg(seed=123\_457, modulus=2\*\*31 - 1, multiplier=7\*\*5)

    @staticmethod

    def exponential(mean: float):

        randomValue = next(RandomVariate.generator)

        return -mean \* math.log(randomValue)

    @staticmethod

    def poisson(mean: int):

        if mean >= 15:

            z = RandomVariate.normal(0, 1)

            return math.ceil(math.sqrt(mean) \* z + mean - 0.5)

        n = 0

        p = 1

        while (p := p \* next(RandomVariate.generator)) >= math.exp(-mean):

            n += 1

        return n

خروجی:

Randomly generated value for cusftomers' count: 87

Arrival times:

['8:02',

'8:14',

'8:19',

'8:29',

'8:33',

'8:33',

'8:42',

'8:49',

'8:53',

'8:57',

'9:00',

'9:00',

'9:05',

'9:08',

'9:25',

'9:27',

'9:36',

'9:39',

'9:40',

'9:46',

'9:58',

'10:01',

'10:01',

'10:02',

'10:09',

'10:15',

'10:24',

'10:25',

'10:32',

'10:45',

'10:46',

'10:46',

'11:02',

'11:04',

'11:13',

'11:20',

'11:26',

'11:43',

'11:43',

'11:44',

'11:46',

'11:46',

'11:50',

'11:50',

'11:55',

'12:04',

'12:04',

'12:06',

'12:14',

'12:15',

'12:15',

'12:17',

'12:18',

'12:22',

'12:35',

'12:43',

'12:45',

'12:56',

'12:57',

'13:00',

'13:00',

'13:05',

'13:09',

'13:09',

'13:09',

'13:13',

'13:14',

'13:14',

'13:28',

'13:29',

'13:31',

'13:36',

'13:39',

'13:41',

'13:43',

'13:47',

'13:47',

'13:49',

'13:53',

'13:57',

'14:00',

'14:07',

'14:09',

'14:13',

'14:22',

'14:24',

'14:30']

# سوال 4

# سوال 5

زیرا اگر بر روی یک خط قرار بگیرد، به این معناست که روند (Trend) توزیع انباشته داده‌ها، مشابه تابع توزیع حدس زده شده است. اگر شیب آن 1 باشد، به این معناست که نه تنها روندشان مشابه است، بلکه به تقریب خوبی یکسان هستند. این یکسانی به معنای انتخاب پارامترهای درست برای تابع توزیع است.

کد:

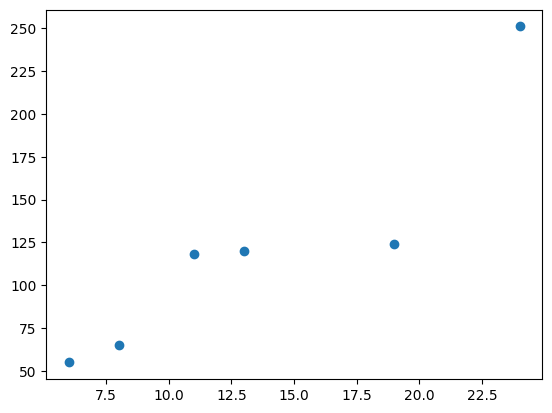
a = sorted([19, 8, 6, 11, 24, 13])

b = sorted([118, 55, 65, 251, 124, 120])

plt.scatter(a, b)

خروجی:

از یک توزیع پیروی نمی‌کنند.



# سوال 6

کد:

trucksCount = framework.RandomVariate.discreteUniform(3, 9)

print(f"Randomly generated value for trucks' count: {trucksCount}")

loadingTimes = []

for \_ in range(trucksCount):

    loadingTime = framework.RandomVariate.normal(mean=34, std=5)

    loadingTimes.append(round(loadingTime, 2))

print("\nLoading times:")

print(loadingTimes)

رفرنس کد در فریمورک:

class Rng:

    @staticmethod

    def lcg(seed: int, modulus: int, multiplier: int, increment=0, returnInts=False):

        randomInt = seed

        while True:

            yield randomInt if returnInts else randomInt / modulus

            randomInt = (multiplier \* randomInt + increment) % modulus

class RandomVariate:

    generator = Rng.lcg(seed=123\_457, modulus=2\*\*31 - 1, multiplier=7\*\*5)

    @staticmethod

    def discreteUniform(low: int, high: int):

        k = high - low + 1

        randomValue = next(RandomVariate.generator)

        return math.floor(k \* randomValue) + low

    @staticmethod

    def normal(mean: float, std: float):

        randomValue1 = next(RandomVariate.generator)

        randomValue2 = next(RandomVariate.generator)

        radius = math.sqrt(-2 \* math.log(randomValue1))

        theta = 2 \* math.pi \* randomValue2

        randomVariate1 = radius \* math.cos(theta)

        randomVariate2 = radius \* math.sin(theta)

        randomVariate1 = std \* randomVariate1 + mean

        randomVariate2 = std \* randomVariate2 + mean

        return randomVariate1 if next(RandomVariate.generator) < 0.5 else randomVariate2

خروجی:

Randomly generated value for trucks' count: 6

Loading times:

[38.04, 33.82, 31.3, 39.11, 25.37, 33.93]