Практическое занятие № 1

Тема: Аппроксимация распределений случайных величин.

Цель: Приобретение практических навыков подбора аппроксимирующих распределений и их параметров по статистическим данным.

Язык программирования, ПО и библиотеки: python 3.x, установленный пакет библиотек Anaconda.

Порядок выполнения практического занятия

1. Создайте новый проект. Скопируйте файл *pz1.py* в свою локальную папку.

2. Добавьте файлы с наборами статистических данных (*attendances.csv, Shared_Database.csv, Uber Request Data.csv*), а также файл с библиотекой генерации случайных величин (*rand_destribution.py*) в директорию с файлом *pz1.py*.

3. Откройте файл *rand_destribution.py*. Ознакомьтесь со структурой классов, представляющих вероятностные распределения. Обратите внимание, что у каждого класса есть методы:

- генерации случайной величины (СВ);
- подбора параметров распределения по статистическим начальным моментам;
- вычисления теоретических моментов.

Ниже представлен класс из библиотеки *rand_destribution.py*, реализующий различные функции для Гамма-распределения. Например, функция *calc_theory_moments()* вычисляет теоретические моменты распределения по заданным параметрам μ и *α*. Функция *generate_static()* возвращает псевдо-случайную величину (ПСВ). Метод *get_params()* возвращает параметры μ и *α* по заданному списку начальных моментов.

```
class Gamma:
    """
    Гамма-распределение
    """

    @staticmethod
    def get_params(b):
        """
        Статический метод аппроксимации параметров Гамма-распределения
поправочным многочленом
```

```
        b: список из произвольного числа начальных моментов
        Возвращает список параметров mu и alpha и, если число начальных
моментов больше двух, значения g[j], j=0,N, где N - число моментов
        """
        d = b[1] - b[0] * b[0]
        mu = b[0] / d
        alpha = mu * b[0]
        if len(b) > 2:
            # подбор коэффициентов g
            A = []
            B = []
            for i in range(len(b) + 1):
                A.append([])
                if i == 0:
                    B.append(1)
                else:
                    B.append(b[i - 1])
                for j in range(len(b) + 1):
                    A[i].append(Gamma.get_gamma(alpha + i + j) / (pow(mu, i +
j) * Gamma.get_gamma(alpha)))
            g = np.linalg.solve(A, B)
            return mu, alpha, g
        else:
            return mu, alpha

    @staticmethod
    def generate_static(mu, alpha):
        theta = 1 / mu
        return np.random.gamma(alpha, theta)


    @staticmethod
    def calc_theory_moments(mu, alpha, count=3):
        """
        mu, alpha - параметры распределения
        Вычисляет теоретические начальные моменты распределения. По умолчанию
- первые три
        """
        f = [0.0] * count
        for i in range(count):
            prod = 1
            for k in range(i + 1):
                prod *= alpha + k
            f[i] = prod / math.pow(mu, i + 1)
        return f
```

Добавлять данный код в файл *pz1.py* не нужно, код приведен для ознакомления. Подобные функции также реализованы для других распределений (нормального, Эрланга, Кокса, равномерного, Гамма-распределения, гиперэкспоненциального). Данные функции будут вызываться по мере необходимости из файла *pz1.py*.

4. Откройте файл *pz1.py*. Обратите внимание на код после строки

```
if __name__ == "__main__":
```

Данный код будет выполнен только в случае, если вы вызываете файл *pz1.py* непосредственно, а не используете его в качестве сторонней библиотеки.

Получите значения номера набора данных (*dataset_num*) и исследуемого распределения (*dist_num*) из таблицы 1 по вашему номеру по журналу. Подставьте их вместо соответствующих значений в коде

```
if __name__ == "__main__":

    dataset_num = 3
    dist_num = 2
```

Таблица 1. Выбор варианта выполнения практического задания

Номер по журналу	Набор данных	Распределение	Номер по журналу	Набор данных	Распределение
1	1	1	13	1	2
2	2	1	14	2	2
3	3	1	15	3	2
4	1	2	16	1	3
5	2	2	17	2	3
6	3	2	18	3	3
7	1	3	19	1	1
8	2	3	20	2	1
9	3	3	21	3	1
10	1	1	22	1	2
11	2	1	23	2	2
12	3	1	24	3	2

5. В наборах данных 1 и 2 содержатся различная информация о приеме пациентов в палате неотложной помощи. В наборе данных 3 – данные по такси Uber. С помощью функции *reader()* из всего набора данных собираются значения времен ожиданий и возвращаются в виде списка. Для наборов данных 1 и 2 речь идет о времени ожидания в очереди пациентов, для 3 – времени ожидания клиентов. На рисунке 1 представлен фрагмент набора данных *attendances.csv*.

timestamp	emergency_room	priority	examined_patients	waiting_patients	waiting_time
1,61645E+12	Pronto Soccorso Burlo	Giallo	1	0	0:02:00
1,61645E+12	Pronto Soccorso Burlo	Verde	0	0	0:00:00
1,61645E+12	Pronto Soccorso Burlo	Bianco	0	0	0:00:00
1,61645E+12	Pronto Soccorso Burlo	Rosso	0	0	0:00:00
1,61645E+12	Pronto Soccorso Spilimbergo	Giallo	0	0	0:00:00
1,61645E+12	Pronto Soccorso Spilimbergo	Verde	7	0	1:05:00
1,61645E+12	Pronto Soccorso Spilimbergo	Bianco	2	0	2:46:00
1,61645E+12	Pronto Soccorso Spilimbergo	Rosso	0	0	0:00:00
1,61645E+12	Pronto Soccorso Pordenone	Giallo	3	0	0:04:00
1,61645E+12	Pronto Soccorso Pordenone	Verde	7	0	0:32:00
1,61645E+12	Pronto Soccorso Pordenone	Bianco	0	0	0:00:00
1,61645E+12	Pronto Soccorso Pordenone	Rosso	1	0	0:02:00
1,61645E+12	Pronto Soccorso Pediatrico Pordenone	Giallo	0	0	0:00:00
1,61645E+12	Pronto Soccorso Pediatrico Pordenone	Verde	0	0	0:00:00
1,61645E+12	Pronto Soccorso Pediatrico Pordenone	Bianco	0	0	0:00:00
1,61645E+12	Pronto Soccorso Pediatrico Pordenone	Rosso	0	0	0:00:00
1,61645E+12	Pronto Soccorso San Vito al Tagliamento	Giallo	1	0	0:24:00

Рисунок 1. Фрагмент набора данных *attendances.csv*

Далее мы будем аппроксимировать статистическое распределение, заданное набором данных, одним из следующих, в зависимости от заданного номера распределения:

1- Гамма-распределением;
2- Парето;
3- нормальным.

6. Далее в коде есть такие строки:

```
moments = get_moments(wait_times)
variance, coev = get_variance(moments)
```

В них вызываются две функции:

1) функция *get_moments()*, которая должна по заданному массиву (списку) времен ожиданий вернуть массив начальных моментов распределения;
2) функция *get_variance()*, которая должна по заданному массиву (списку) начальных моментов распределения вернуть значение дисперсии и коэффициент вариации.

В коде выше есть заготовки для этих функций:

```
def get_moments(data, num_of_moments=3):
    """
    Формирует из массива данных статистические начальные моменты
    data - массив времен ожидания
    num_of_moments - требуемое число начальных моментов

    """
    moments = [0.0] * num_of_moments
    #  добавить код ниже (вместо pass):
    pass
```

```
    return moments


def get_variance(moments):
    """
    Возвращет значение дисперсии и коэффициента вариации по массиву начальных
моментов распределения moments
    """

    #  заменить код ниже:
    variance = 0
    coev = 0

    return variance, coev
```

Необходимо дописать их самостоятельно. Вы можете отладить эти функции отдельно, создав еще один файл с расширением *.py*. После завершения написания кода, запустите его на выполнение. В зависимости от выбранного варианта должен получиться график, похожий на представленный на рисунке 2.

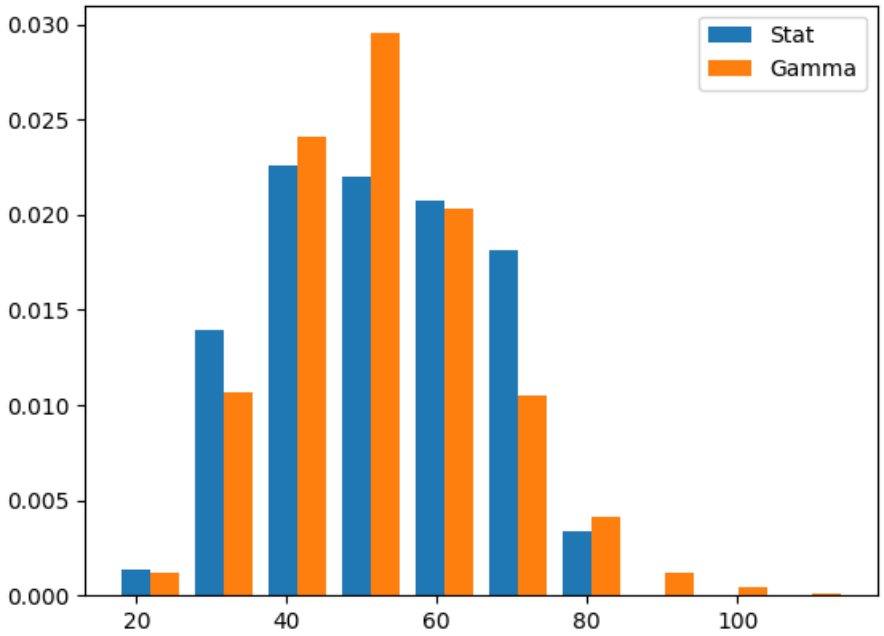

Рисунок 2. Гистограммы статистического и аппроксимирующего распределений

Также в результате выполнения программы выводится информация вида:

```
-----------------------------------
Аппроксимация Гамма-распределением:
-----------------------------------
Параметры Гамма-распределения:
mu = 0.005              alpha = 14.325

Коэффициент вариации:
0.264
```

```
Начальные моменты:
------------------------------------------------------------
       -       |       1       |       2       |       2
------------------------------------------------------------
     Стат      |   3144.825    | 10580311.381  |37617175561.609
     Теор      |   3144.825    | 10580311.381  |37918638950.049
Значение параметра D = 0.66761 меньше порога 1.35810, распределения близки
```

В последней строке выводится результат теста Колмогорова-Смирнова о близости статистического и аппроксимирующего распределений.

7. Попробуйте аппроксимировать ваши данные другими распределениями. Для этого измените параметр *dist_num*.

8. Итоговый отчет должен быть представлен в виде программы, при запуске которой производятся соответствующие расчеты, выводится график гистограмм, в зависимости от вашего варианта.

9. Будьте готовы ответить на контрольные вопросы по практическому занятию:

1) Что понимается под аппроксимацией статистического распределения?
2) Что такое функция распределения? Дополнительная функция распределения? Что понимается под плотностью распределения?
3) Начальные и центральные моменты распределения случайной величины (СВ). Напишите формулы для их вычисления для случая дискретной и непрерывной СВ.
4) Дисперсия, коэффициент вариации СВ.
5) Экспоненциальное распределение. Функция и плотность распределения. Начальные моменты. Особое свойство экспоненциального распределения. Формула для генерации СВ, распределенной по данному закону.
6) Распределение Эрланга. Плотность распределения. Начальные моменты. При каких значениях параметров вырождается в экспоненциальное распределение? При каких значениях коэффициента вариации используется в качестве аппроксимирующего? Фазовая интерпретация данного распределения. Подходы к генерации СВ.
7) Гамма-распределение. Плотность распределения. Начальные моменты. При каких значениях параметров вырождается в распределение

Эрланга и экспоненциальное распределение? При каких значениях коэффициента вариации используется в качестве аппроксимирующего? В чем сложности использования данного распределения?

8) Гиперэкспоненциальное распределение. Дополнительная функция распределения. При каких значениях коэффициента вариации используется в качестве аппроксимирующего? Фазовая интерпретация данного распределения. Подходы к генерации СВ.

9) Распределение Парето. Дополнительная функция распределения. Начальные моменты. При каких значениях параметров существуют начальные моменты распределения Парето? Какие процессы обычно аппроксимируются распределением Парето?