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 или отдельно следующие: pandas, numpy, matplotlib. Среда разработки – PyCharm, Spyder 3 или Jupyter.

Используемые наборы статистических данных:

– файл с журналом запросов на web-сервер «weblog.csv»;
– файл с журналом покупок в онлайн-магазине “Online Retail.csv”;
– файл с журналом поездок в такси Uber «Uber Request Data.csv».

Порядок выполнения практического занятия.

1. Откройте PyCharm. Создайте новый файл с расширением *.py* в проекте.

2. Добавьте в папку с проектом файл *rand_destribution.py*. Убедитесь, что в проекте присутствуют файлы с наборами статистических данных.

3. Пропишите секцию import в следующем виде:

```
import csv
import datetime
import math
import rand_destribution as rd
import matplotlib.pyplot as plt
```

4. Откройте файл *rand_destribution.py*. Ознакомьтесь со структурой классов, представляющих вероятностные распределения. Обратите внимание, что у каждого класса есть методы генерации случайной величины (СВ) и подбора теоретических моментов. Ниже представлен класс, реализующий данные функции для распределения Парето.

```
class Pareto_dist:
    "Распределение Парето"
    def __init__(self, params):
        """
        Принимает список параметров в следующей последовательности - alpha, K
        """
        self.a = params[0]
        self.k = params[1]
        self.params = params
        self.type = 'Pa'
```

```
    def generate(self):
        return Pareto_dist.generate_static(self.a, self.k)

    @staticmethod
    def get_pdf(t, a, k):
        """
        Probability density function
        """
        if t < 0:
            return 0
        return a*math.pow(k, a)/math.pow(t, a+1)

    @staticmethod
    def get_cdf(params, t):
        """
        Cumulative distribution function
        """
        return 1.0 - Pareto_dist.get_tail(params, t)

    @staticmethod
    def get_tail(params, t):
        """
        Complementary cumulative distribution function (tail distribution)
        """
        if t < 0:
            return 0
        a = params[0]
        k = params[1]
        return math.pow(k/t, a)

    @staticmethod
    def calc_theory_moments(a, k, max_number=3):
        f = []
        for i in range(max_number):
            if a > i+1:
                f.append(a*math.pow(k, i+1)/(a-i-1))
            else:
                return f
        return f

    @staticmethod
    def generate_static(a, k):
        return k*math.pow(np.random.rand(), -1/a)

    @staticmethod
    def get_a_k(f):
        """
        Метод возвращает параметры a и K по 2-м начальным моментам списка f
        """
        d = f[1] - f[0]*f[0]
        c = f[0]*f[0]/d
        disc = 4*(1+c)
        a = (2+math.sqrt(disc))/2
        k = (a-1)*f[0]/a
        return a, k
```

5. Ознакомьтесь с содержимым файла с журналом поездок в такси Uber «Uber Request Data.csv». Далее будем исследовать распределение интервала между заказами такси. Временные метки запросов к сервису такси Uber хранятся в поле «Request timestamp». Ниже приведена функция, возвращающая массив интервалов между соседними запросами к сервису в секундах.

```
def getIntervalsFromWeblog(file_obj):

    reader = csv.DictReader(file_obj, delimiter = ',')
    deltas_sec = []
    for line in reader:
        status = line["Status"]
        if status!= 'Trip Completed':
            continue
        dt_request = line["Request timestamp"]
        dt = dt_request.split(' ')
        if dt[0].find('/')!=-1:
            date = dt[0].split('/')
        else:
            date = dt[0].split('-')

        day = int(date[0])
        month = int(date[1])
        year = int(date[2])
        time = dt[1].split(':')
        hour = int(time[0])
        min = int(time[1])
        if len(time) == 3:
            sec = int(time[2])
        else:
            sec = 0

        request_timestamp = datetime.datetime(year, month, day, hour, min,
sec)

        dt_drop = line["Drop timestamp"]

        dt = dt_drop.split(' ')
        if dt[0].find('/')!=-1:
            date = dt[0].split('/')
        else:
            date = dt[0].split('-')

        day = int(date[0])
        month = int(date[1])
        year = int(date[2])
        time = dt[1].split(':')
        hour = int(time[0])
        min = int(time[1])
        if len(time) == 3:
            sec = int(time[2])
```

```
        else:
            sec = 0

        drop_timestamp = datetime.datetime(year, month, day, hour, min, sec)

        delta = drop_timestamp-request_timestamp
        deltas_sec.append(delta.seconds)

    return deltas_sec


fileobj = open('Uber Request Data.csv')
deltas = getIntervalsFromWeblog(fileobj)
```

6. Вычислим начальные моменты статистического распределения интервалов между запросами к сервису Uber, дисперсию и коэффициент вариации данной СВ.

```
f = [0, 0, 0]
N = len(deltas)

for i in range(len(deltas)):
    deltas[i] = deltas[i]/60

for d in deltas:
    for j in range(3):
        f[j] += math.pow(d, j+1)

for j in range(3):
    f[j] /= N

variance = f[1]-f[0]**2
coev = math.sqrt(variance)/f[0]

print("Статистические начальные моменты:")
print("{0:<15.3f}{1:<15.3f}{2:<15.3f}\n".format(*f))
print("Коэффициент вариации:")
print("{0:<15.3f}\n".format(coev))
```

Поскольку коэффициент вариации меньше единицы, для аппроксимации данной СВ могут подойти такие распределения, как распределение Эрланга, Гамма-распределение. Аппроксимируем интервалы между соседними запросами Гамма-распределением. Для этого вычислим параметры Гамма распределения. Для проверки правильности подбора параметров вычислим теоретические начальные моменты и сравним их с полученными ранее статистическими.

```
print("Аппроксимация Гамма - распределением:")

params = rd.Gamma.get_mu_alpha(f)
mu = params[0]
alpha = params[1]

f_teor = rd.Gamma.calc_theory_moments(mu, alpha)
print("Параметры Гамма-распределения:")
print("alpha = {0:<15.3f}    mu = {1:<15.3f}\n".format(alpha, mu))

print("Теоретические начальные моменты:")
for mom in f_teor:
    print("{0:<15.3f}".format(mom), end="   ")
```

Сделайте выводы о близости полученных значений.

7. Сгенерируем интервалы между соседними запросами подчиненные аппроксимирующему Гамма-распределению. Построим гистограммы для первоначальных статистических данных и аппроксимирующего распределения.

```
gamma_data = []
for i in range(len(deltas)):
    gamma_data.append(rd.Gamma.generate_static(mu, alpha))

plt.hist([deltas, gamma_data], label=['Stat', 'Gamma'], density=True)
plt.legend()
plt.show()
```

Должен получиться график, представленный на рисунке 1.

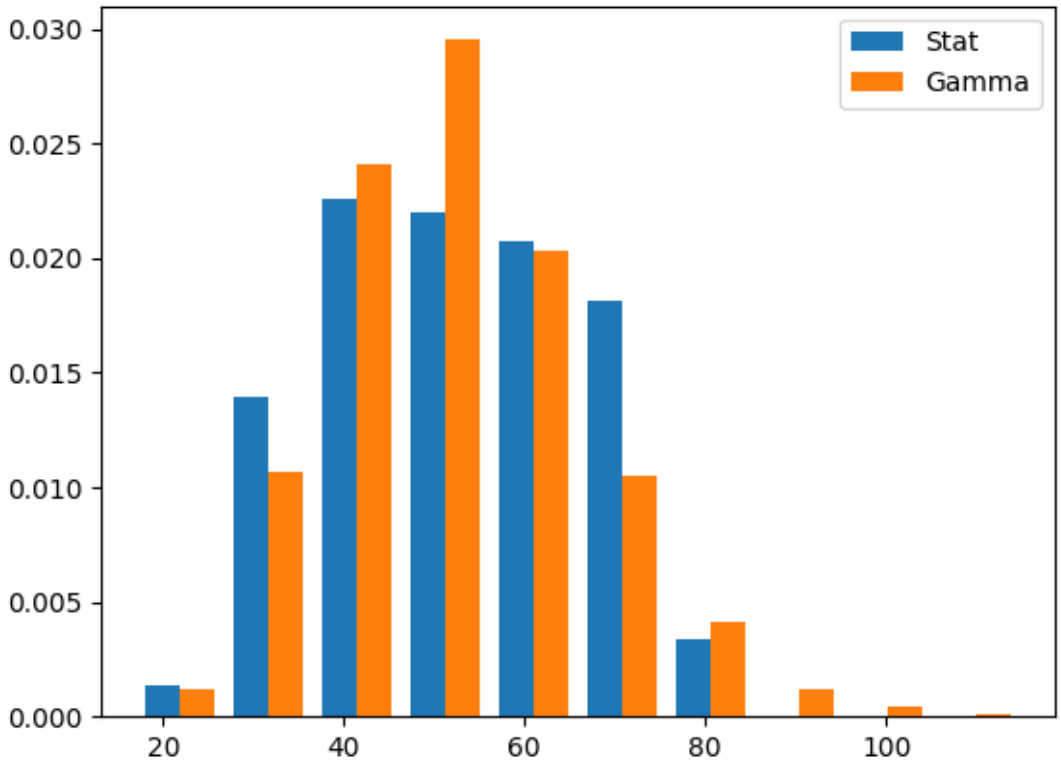

Рисунок 1. Гистограммы статистического и аппроксимирующего распределений

Сделайте выводы о близости полученных распределений.

8. Повторите проделанные шаги для новых наборов данных. Варианты данных следует выбрать следующим образом – четный номер по журналу соответствует файлу с журналом запросов на web-сервер «weblog.csv», нечетный – файлу с журналом покупок в онлайн-магазине “Online Retail.csv”. Для первого варианта в качестве исследуемой случайной величины будет интервал между соседними запросами на сервер. Время запросов хранится в колонке «Time». Для второго варианта – интервал между соседними обращениями в онлайн-магазин. Время обращения содержится в колонке «InvoiceDate».

9. . Итоговый отчет должен быть представлен в виде программы. Секции отображения графиков должны быть закомментированы, и готовы к отображению по просьбе принимающего. Возможен вариант предварительного формирования графиков – для этого используйте в конце программы вместо *plt.show()* метод *plt.savefig(«имя_файла.jpg»)* для сохранения графика в директорию проекта.

10. Примечание. Для вычисления интервалов между соседними запросами на сервер для первого варианта можете воспользоваться функцией, приведенной ниже. Изучите реализованные здесь подходы.

```
def getIntervalsFromWeblog(file_obj):
    """
    Получает интервалы из файла, содержащего время поступления запросов на
сервер (
    поле 'Time')
    """
    reader = csv.DictReader(file_obj, delimiter = ',')
    timestamps = []
    for line in reader:
        dt = line["Time"]
        if dt[0] != '[':
            continue
        dt = dt.split('/')
        day = int(dt[0][1:])
        month = switch_month(dt[1])
        yeartime = dt[2].split(":")
        year = int(yeartime[0])
        hour = int(yeartime[1])
        min = int(yeartime[2])
        sec = int(yeartime[3])
        timestamps.append(datetime.datetime(year, month, day, hour, min,
sec))

    deltas_sec = []
    for i in range(len(timestamps)-1):
        delta = timestamps[i+1]-timestamps[i]
        deltas_sec.append(delta.seconds)

    return deltas_sec

def switch_month(month_text):
    switcher = {
        "Jan": 1,
        "Feb": 2,
        "Mar": 3,
        "Apr": 4,
        "May": 5,
        "Jun": 6,
        "Jul": 7,
        "Aug": 8,
        "Sep": 9,
        "Oct": 10,
        "Nov": 11,
        "Dec": 12
    }
    return switcher.get(month_text, "Invalid month")

fileobj = open('weblog.csv')
deltas = getIntervalsFromWeblog(fileobj)
```

Приведем аналогичный листинг для другого варианта

```
def getIntervalsFromWeblog(file_obj):
    """
    Получает интервалы из файла, содержащего время поступления запросов на
сервер (
    поле 'Time')
    """
    reader = csv.DictReader(file_obj, delimiter = ';')
    timestamps = []
    for line in reader:
        dt = line["InvoiceDate"].split(' ')
        date = dt[0].split(".")
        time = dt[1].split(":")
        day = int(date[0])
        month = int(date[1])
        year = int(date[2])
        hour = int(time[0])
        min = int(time[1])
        if len(time) == 3:
            sec = int(time[2])
        else:
            sec = 0

        timestamps.append(datetime.datetime(year, month, day, hour, min,
sec))

    deltas_sec = []
    for i in range(len(timestamps)-1):
        delta = timestamps[i+1]-timestamps[i]
        delta_sec = delta.seconds
        if delta_sec != 0:
            deltas_sec.append(delta.seconds)

    return deltas_sec


fileobj = open('Online Retail.csv')
deltas = getIntervalsFromWeblog(fileobj)
```

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

1) что понимается под аппроксимацией СВ?

2) что такое функция распределения? Дополнительная функция распределения?

3) что понимается под плотностью распределения?

4) что такое начальные и центральные моменты распределения СВ? Напишите формулы для их вычисления для случая дискретной и непрерывной СВ.

5) Что такое дисперсия, коэффициент вариации СВ?

6) Экспоненциальное распределение. Функция и плотность распределения. Начальные моменты. Особое свойство экспоненциального распределения. Формула для генерации СВ, распределенной по данному закону.

7) 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.

8) 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?

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

10) Распределение Парето. Дополнительная функция распределения. Начальные моменты. При каких значениях параметров существуют начальные моменты распределения Парето? В чем сложности использования данного распределения?