Отчет по лабораторной работе № 1

**Анализ одномерных случайных величин**

Выполнено:

Барбаков Илья Олегович,

Группа J4150

Санкт-Петербург

2022

**Оглавление**

[1. Обоснование выбора данных 1](#_Toc117510010)

[2. Построение непараметрической оценки PDF в виде гистограммы и использование функции плотности ядра 1](#_Toc117510011)

[3. Оценка порядковой статистики и её представление в виде "ящика с усами" ………………….. 3](#_Toc117510015)

[4. Подбор теоретических распределений, которые лучше всего отражают эмпирические данные 4](#_Toc117510018)

[5. Оценка параметров распределения случайной величины с помощью метода максимального правдоподобия и LS-методов 5](#_Toc117510019)

[6. Проверка эмпирических и теоретических распределений с помощью квантильного графика 11](#_Toc117510031)

[7. Статистические тесты 13](#_Toc117510034)

[8. Сэмплирование случайной величины 14](#_Toc117510038)

[Исходный код 20](#_Toc117510048)

# Обоснование выбора данных

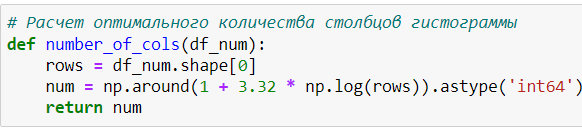
Для задачи анализа одномерных случайных величин выбран датасет с данными действий клиентов телефонной компании, а также с целевой переменной оттока каждого клиента. Выборка имеет большое количество записей, что обеспечит достаточный уровень репрезентативности данных.

Задачей работы является анализ нескольких декретных случайных величин, а также непрерывных. В качестве дискретной величины было выбрано 2 параметра: количество международных звонков, а также количество звонков в службу поддержки клиентов. Эти показатели распределятся с оптимальным количеством классов (20 и 8), и также большинство этих классов представлены как минимум 10 значениями в каждом, что позволит успешно подобрать закон распределения.

Для анализа непрерывных случайных величин были выбраны следующие показатели: плата за вечерние и ночные звонки. По первичному анализу данные распределены близко к нормальному и не имеют несколько пиков, благодаря чему можно будет подобрать известный закон распределения.

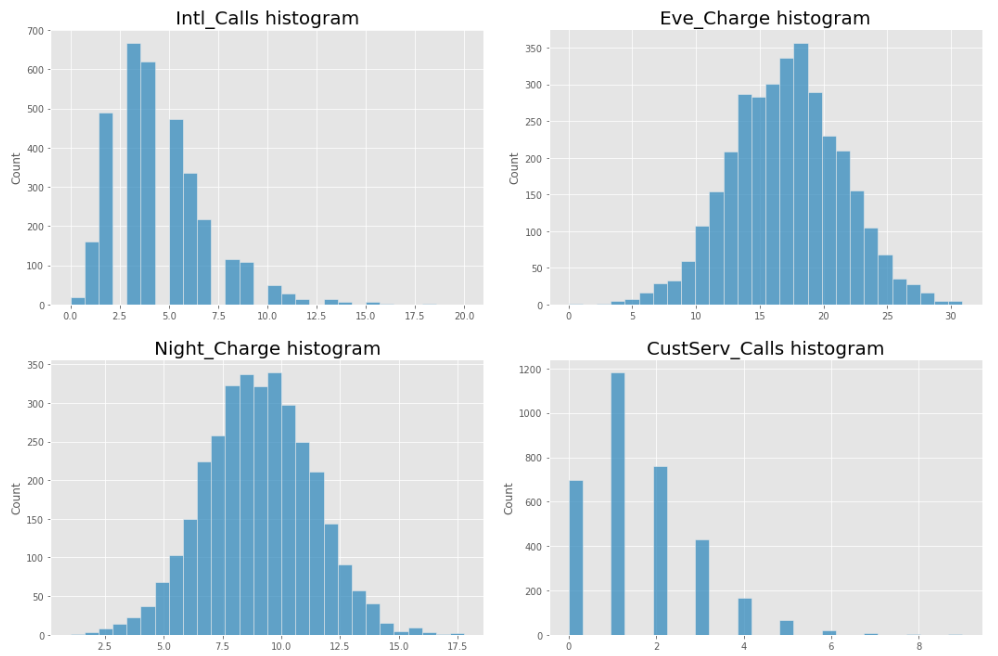
# Построение непараметрической оценки PDF в виде гистограммы и использование функции плотности ядра

Гистограмма – непараметрический способ оценки распределений, позволяющий первично оценить где находятся данные. Выборка может быть разной и поэтому чтобы отразить на графике гладкость и унимодальность необходимо подобрать количество столбцов диаграммы. Напишем для этого функцию:



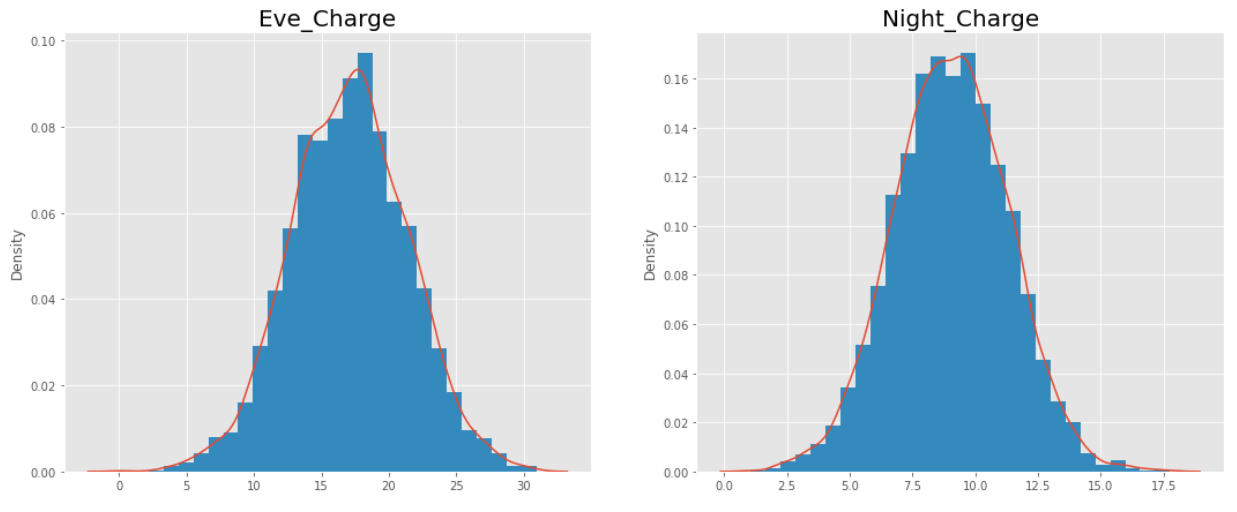
## Рисунок 1 - Функция расчета количества столбцов гистограммы

Теперь строим гистограммы для всех случайных величин:



## Рисунок 2 - Гистограммы случайных величин

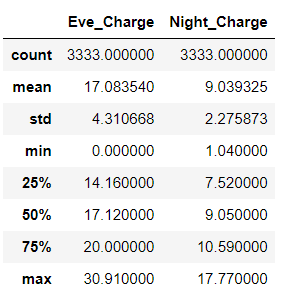
Для получения гладких оценок плотности распределения непрерывной случайной величины, применяем метод ядерного сглаживания, на основе Гауссовой функции. Для этого применяем метод sns.kdeplot, который выводит график. Параметр «bw\_adjust» - ширину окна подбираем эмпирически.



## Рисунок 3 - Ядерное сглаживание

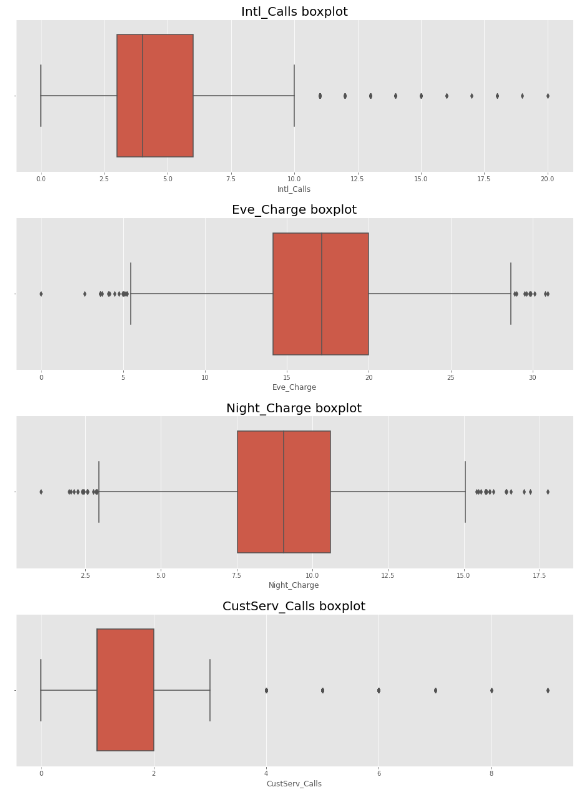
# Оценка порядковой статистики и её представление в виде "ящика с усами"

Выводим порядковую и описательную статистику для непрерывных случайных величин. Она включает в себя значения среднего, максимального, минимального, стандартного отклонения, а также квантилей.



## Рисунок 4 – Статистика

Выводим график в виде «ящика с усами» - диаграмму размаха, которая наглядно показывает медиану, нижний и верхний [квартили](https://ru.wikipedia.org/wiki/%D0%9A%D0%B2%D0%B0%D0%BD%D1%82%D0%B8%D0%BB%D1%8C#%D0%9C%D0%B5%D0%B4%D0%B8%D0%B0%D0%BD%D0%B0_%D0%B8_%D0%BA%D0%B2%D0%B0%D1%80%D1%82%D0%B8%D0%BB%D0%B8), минимальное и максимальное значение выборки и [выбросы](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D0%B1%D1%80%D0%BE%D1%81_(%D1%81%D1%82%D0%B0%D1%82%D0%B8%D1%81%D1%82%D0%B8%D0%BA%D0%B0)). Реализуем вывод графиков с помощью метода библиотеки seaborn – boxplot.



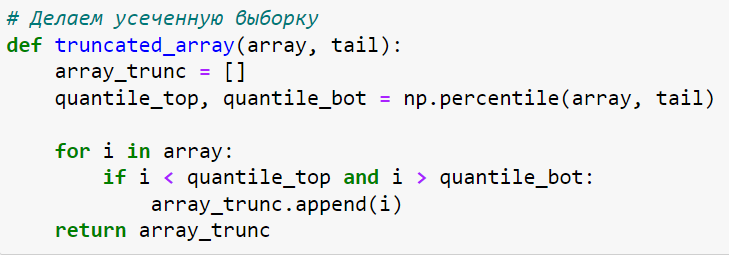
## Рисунок 5 - Ящики с усами

# Подбор теоретических распределений, которые лучше всего отражают эмпирические данные

После проведения первичного анализа, были выбраны следующие теоретические распределения случайной непрерывной величины: нормальное, логарифмически нормальное, гамма, бета и альфа. Для дискретной величины: биномиальное и распределение Пуассона.

# Оценка параметров распределения случайной величины с помощью метода максимального правдоподобия и LS-методов

Изучая распределения, по графику квантилей было заметно, что присутствуют выбросы на хвостах выборки. Такие аномальные значения не отображают логику закона распределения основных данных и могут искажать оценку. Создадим функцию, реализующую создание усеченной выборки данных.



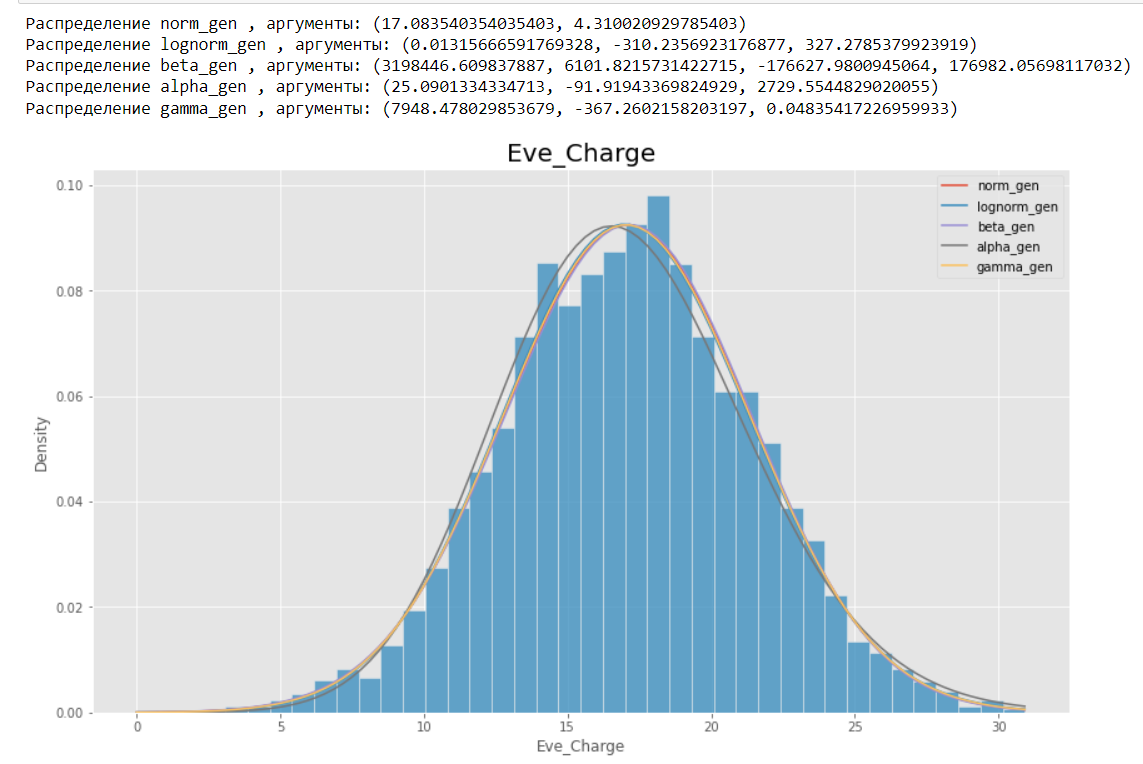
## Рисунок 6 - Усеченная выборка

Подбираем параметры с помощью метода максимального правдоподобия для каждого закона распределения. Суть данного метода заключается в оценке параметров распределения, путем максимизации функции правдоподобия. В python данный метод для непрерывных случайных величин реализован с помощью функции «scipy.stats.rv\_continuous.fit».

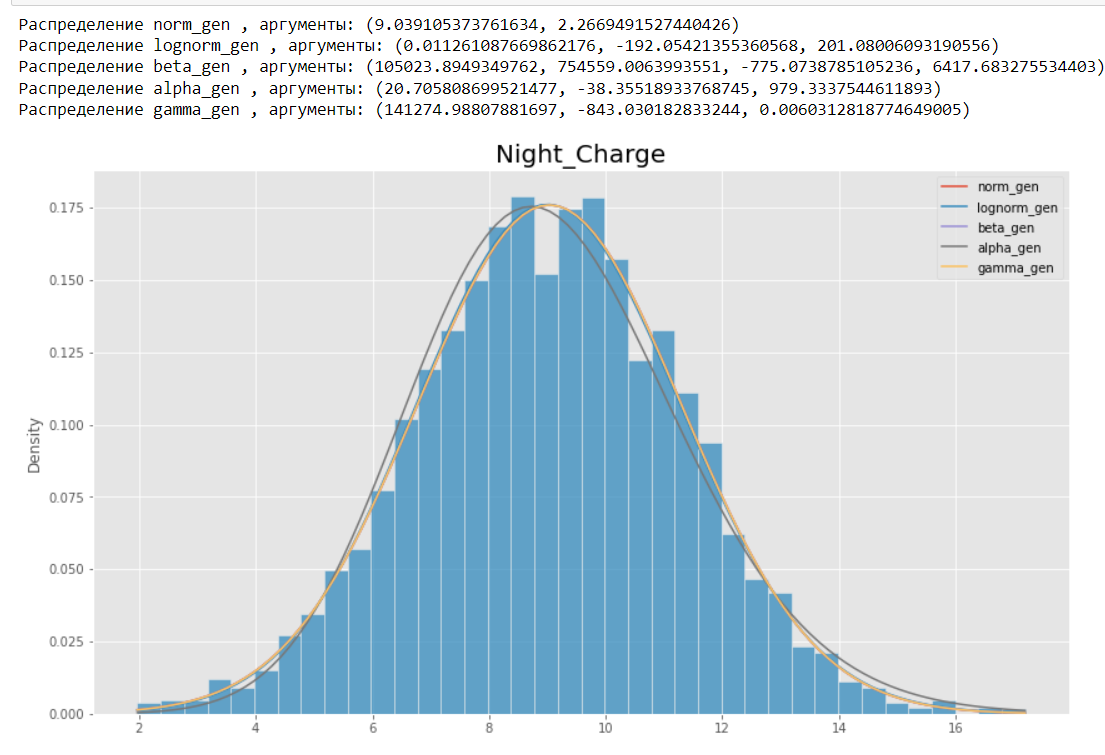


## Рисунок 7 - Метод максимального правдоподобия для непрерывных СВ

Выводим графики теоретических распределений для непрерывных случайных величин.

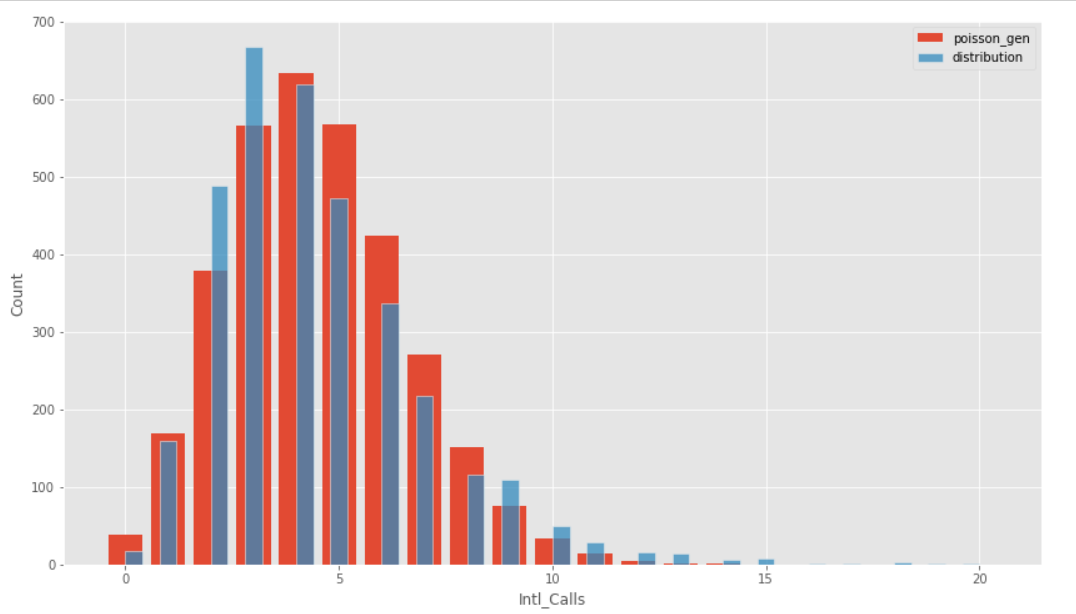


## Рисунок 8 – Метод MLE для «Eve\_Charge»

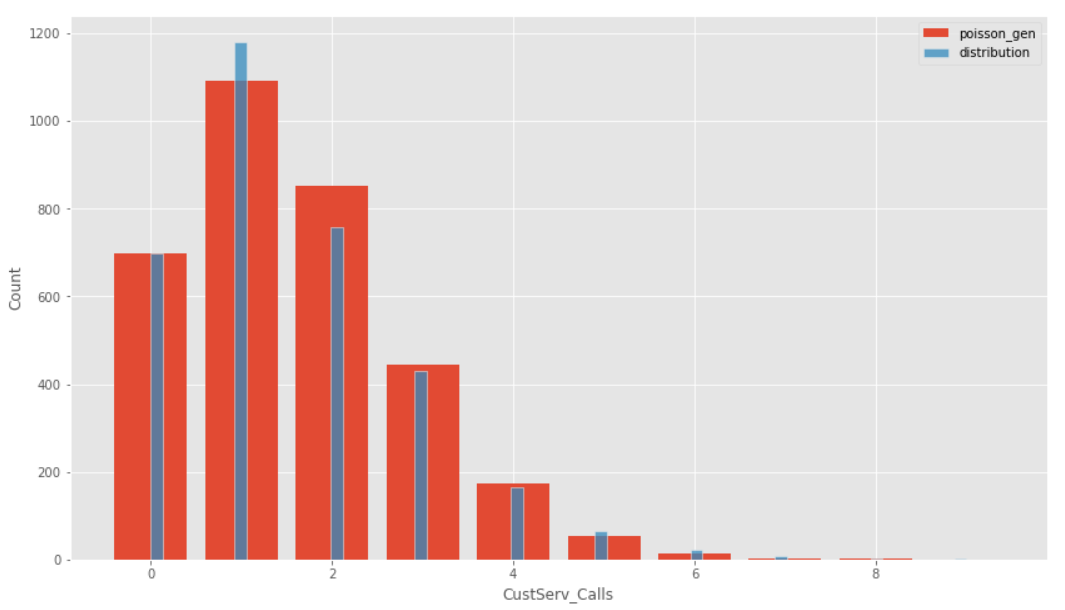


## Рисунок 9 – Метод MLE для «Night\_Charge»

Оценка параметров распределений для дискретных случайных величин представлена в пакете scipy методом «scipy.stats.fit». Выводим графики теоретического Пуассоновского распределения для показателей «Intl\_Calls» и «CustServ\_Calls».

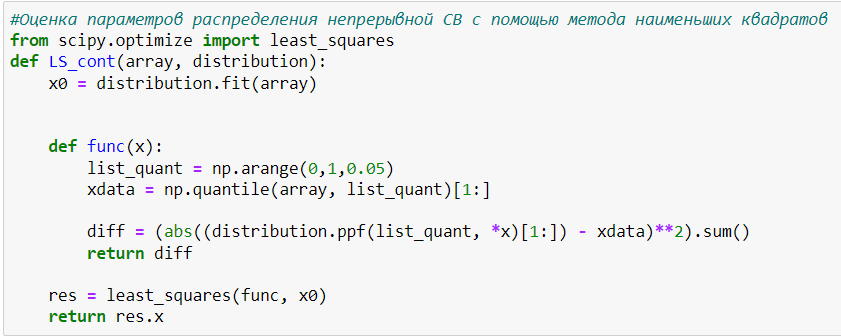


## Рисунок 10 - Метод MLE для «Intl\_Calls»



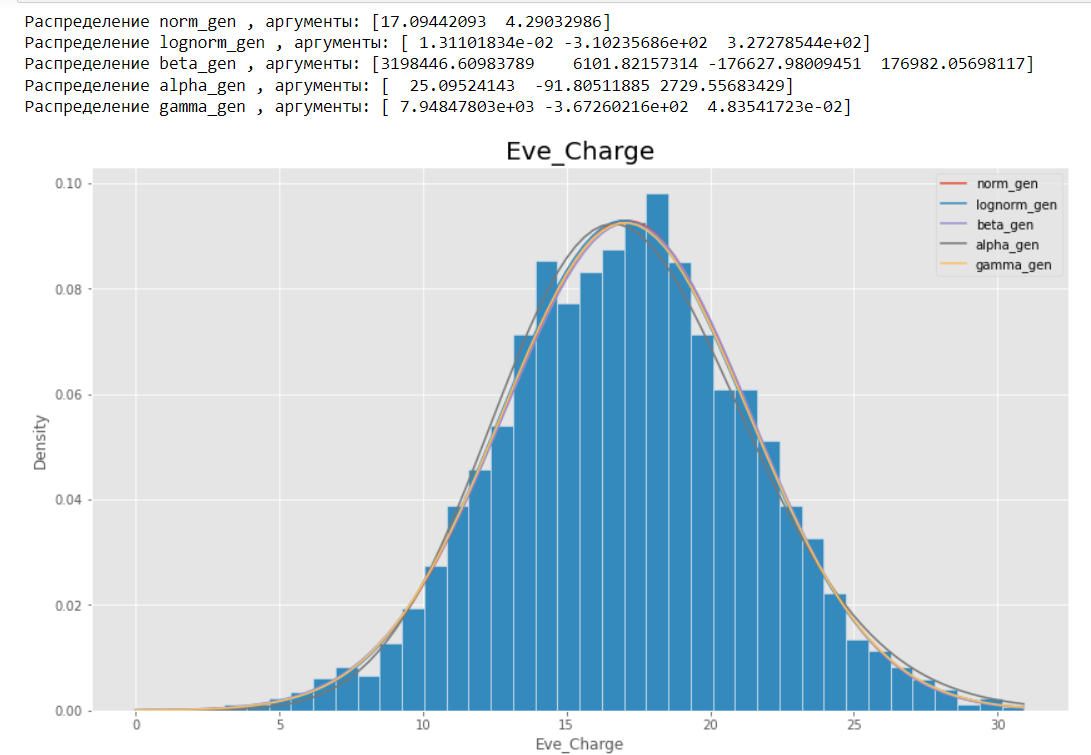
## Рисунок 11 - Метод MLE для «CustServ\_Calls»

Оценивать параметры теоретического распределения также возможно с помощью метода минимальных квадратов, суть которого заключается в минимизации суммы квадратов ошибки по определенным квантилям выборки.

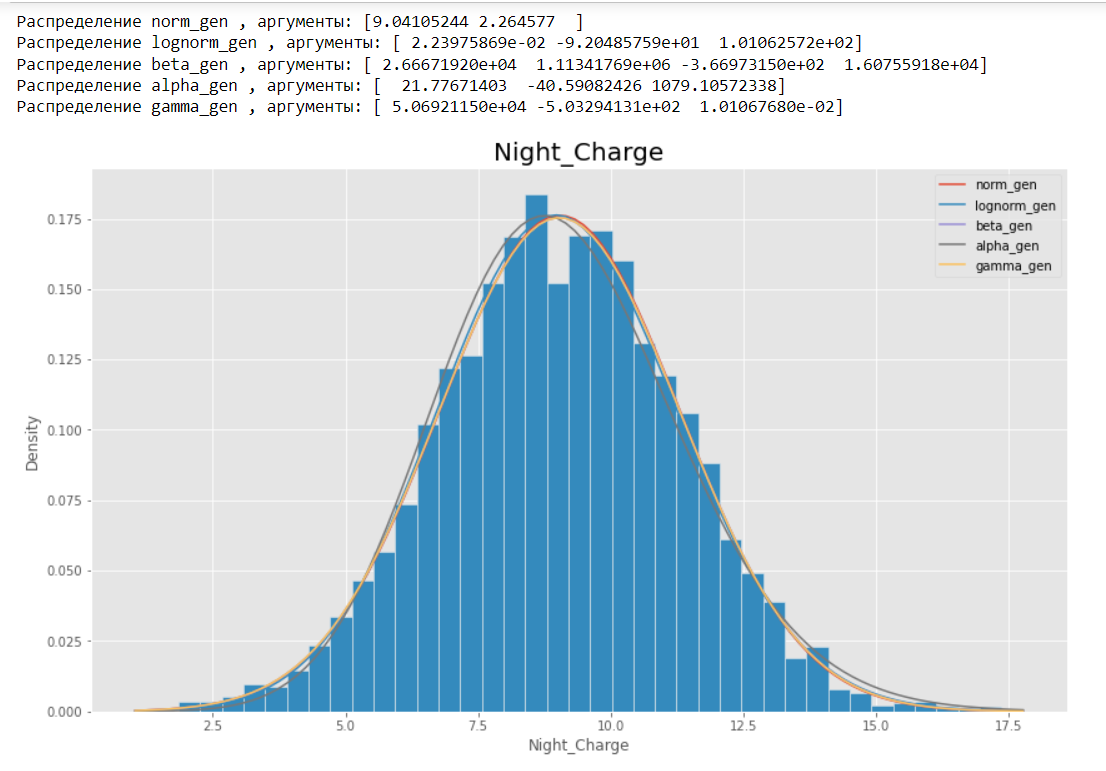


## Рисунок 12 - Метод наименьших квадратов

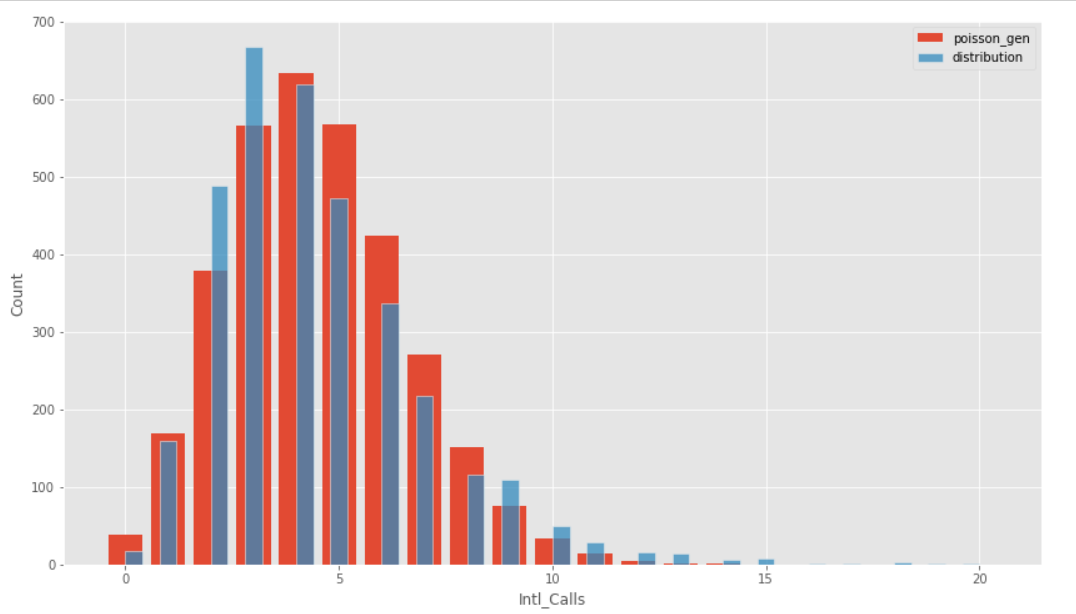
Выведем графики теоретических распределений по параметрам, подобранным с помощью МНК.



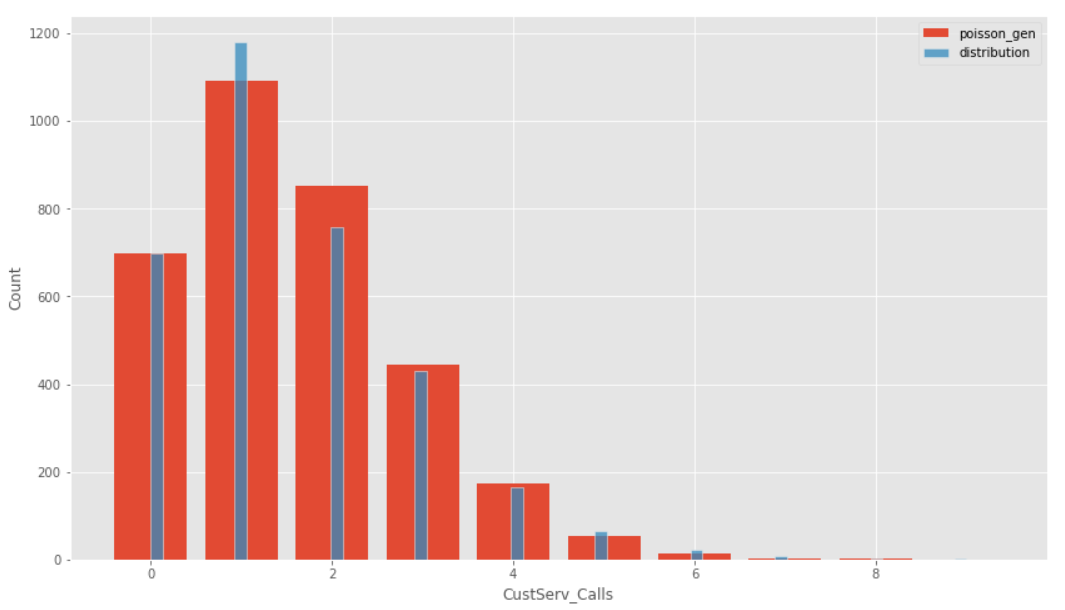
## Рисунок 13 - Метод МНК для «Eve\_Charge»



## Рисунок 14 - Метод МНК для «Night\_Charge»



## Рисунок 15 - Метод МНК для «Intl\_Calls»

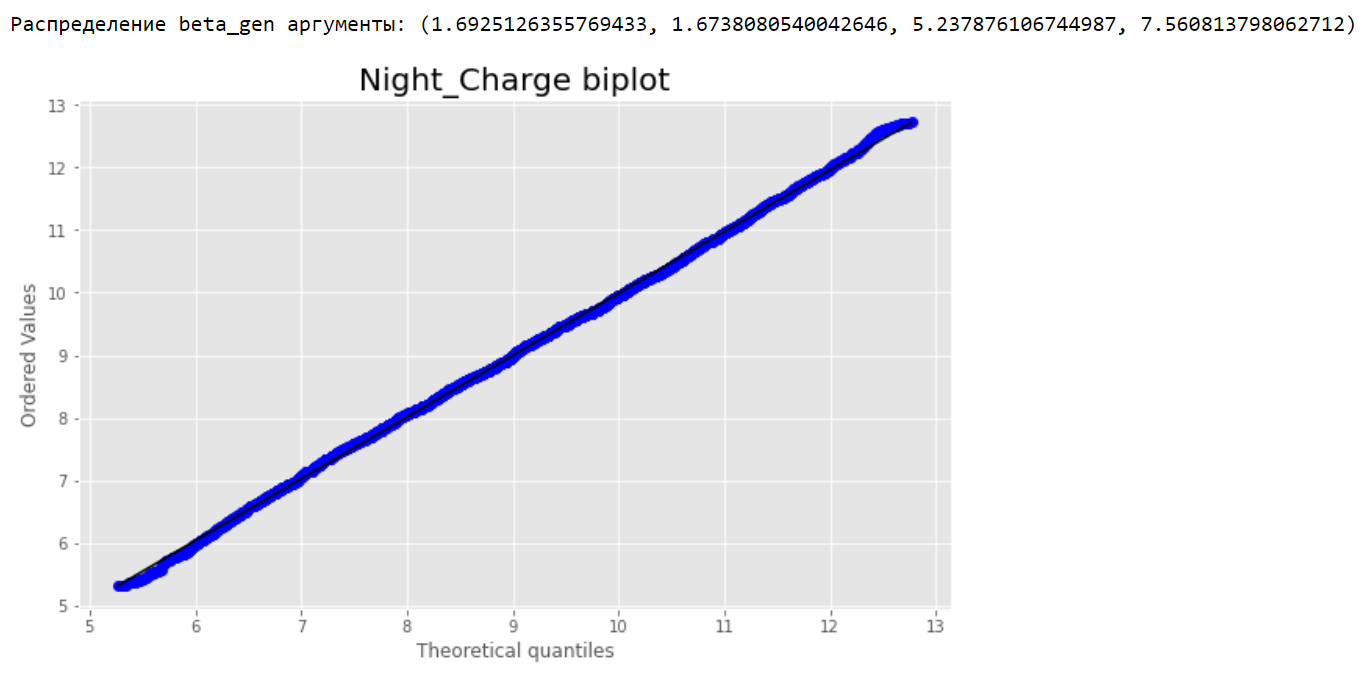


## Рисунок 16 - Метод МНК для «CustServ\_Calls»

С первого взгляда практически все законы довольно хорошо описывают распределения, поэтому необходимо провести дальнейшие проверки.

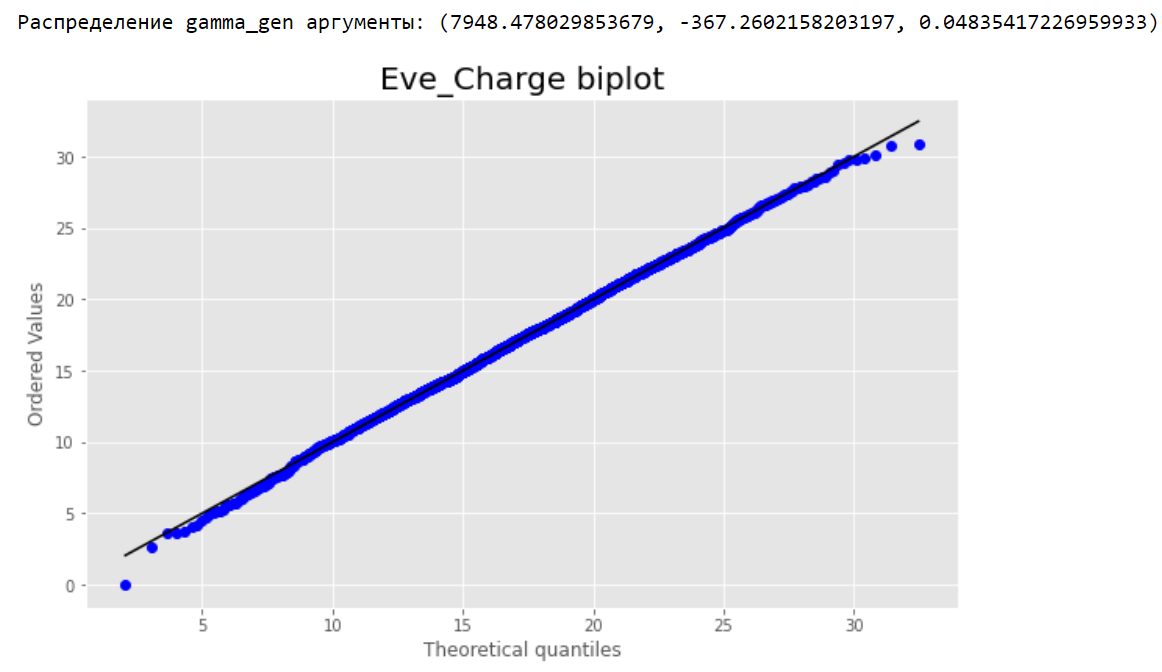
# Проверка эмпирических и теоретических распределений с помощью квантильного графика

Проведем оценку подбора распределения с помощью квантильного биплота. Чем ближе точки значений квантилей лежат на главной диагонали, тем лучше подобрано теоретическое распределение. Для параметра «Night\_Charge» наилучшим распределением выступает beta-распределение.



## Рисунок 17 - Квантильный биплот «Night\_Charge»

Для показателя «Eve\_Charge» оптимальным распределением выступает gamma-распределение.

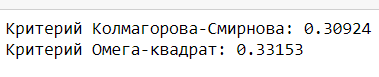


## Рисунок 18 - Квантильный биплот «Eve\_Charge»

Точки значений теоретических квантилей лежат практически на главной диагонали, что порождает вывод о том, что подобран оптимальный закон распределения.

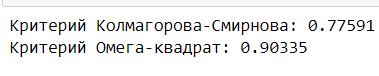
# Статистические тесты

Оценим критерии согласия - критерии для сравнения эмпирического распределения и теоретического. Для непрерывных случайных величин применим критерии Колмогорова-Смирнова и Омега-квадрат. Для статистических тестов примем уровень достоверности равным 95%. Таким образом, результат для «Night\_Charge»:



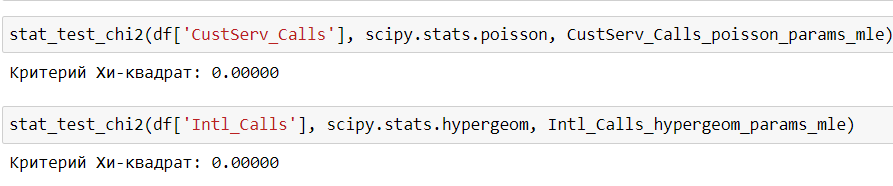
## Рисунок 19 - Статистический тест «Night\_Charge»

P-значение превышает 0.05, таким образом мы принимаем нулевую гипотезу, о том что данные распределены по beta закону. Далее применим эти же критерии для «Eve\_Charge». Значение p также превышает 0.05, что говорит нам о том, что данные распределены по gamma закону.



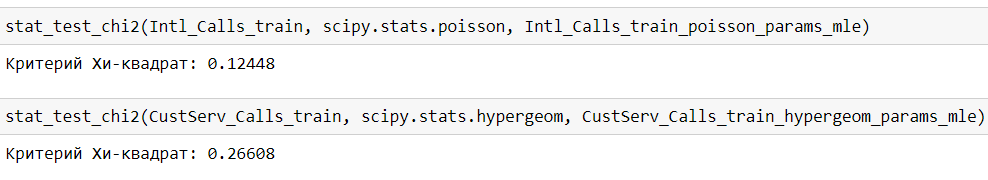
## Рисунок 20 - Статистический тест «Eve\_Charge»

Для дискретных случайных величин воспользуемся тестом Хи-квадрат, который проверяет нулевую гипотезу о том, что категориальные данные имеют заданные частоты.



## Рисунок 21 - Тест Хи-квадрат

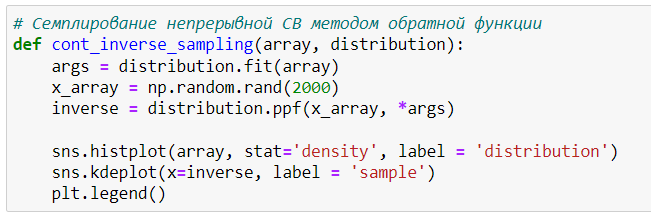
Результаты теста для всего датасета получаются нулевыми, поэтому попробуем сократить объем выборки и опять сравнить данные с помощью критерия Хи-квадрат. Теперь теоретическое распределение подобрано хорошо.



## Рисунок 22 - Рисунок 23 - Тест Хи-квадрат для усеченной выборки

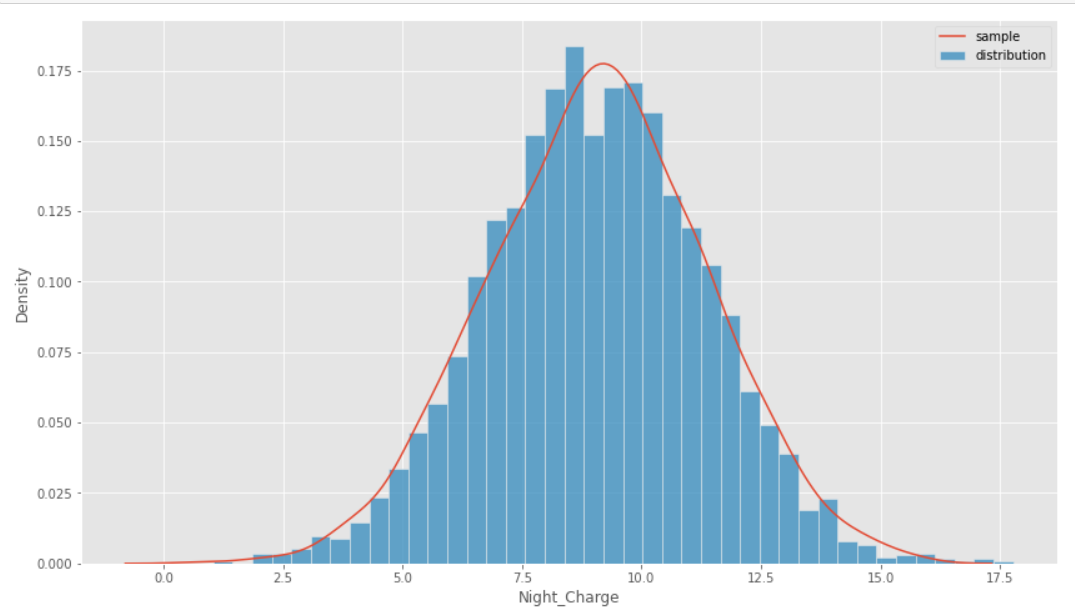
# Сэмплирование случайной величины

Далее необходимо произвести сэмплирование случайной величины, основываясь на уже оценённых распределениях. Первым способом выступает сэмплирование обратной функцией. Его смысл заключается в нахождении обратной функции подобранного распределения. Далее генерация равномерной случайной величины в диапазоне от 0 до 1 и далее применение этих данных в уже найденную функцию.

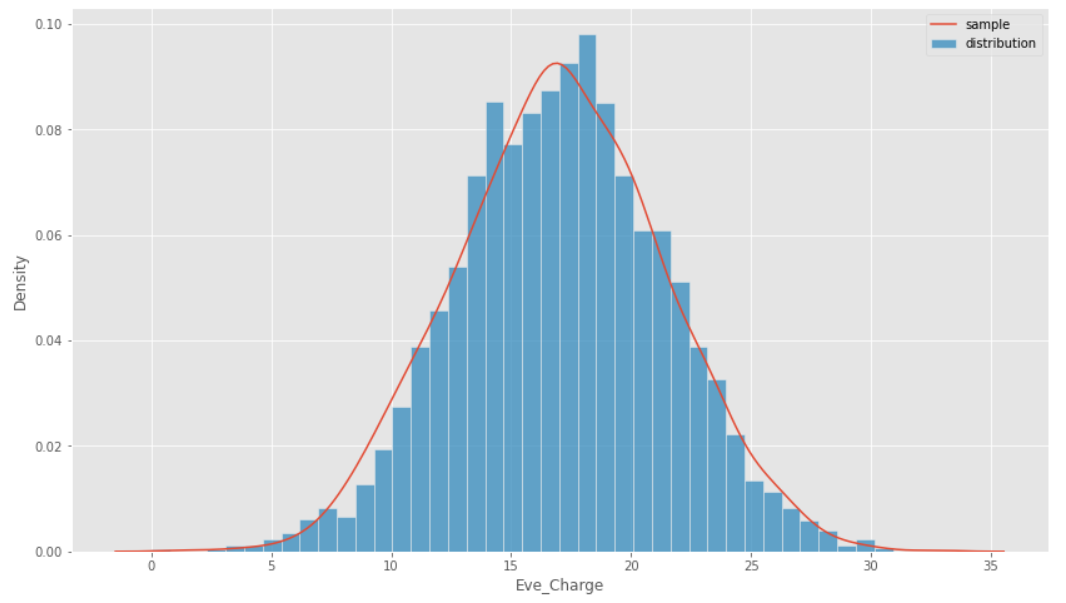


## Рисунок 24 - Сэмплирование методом обратной функции

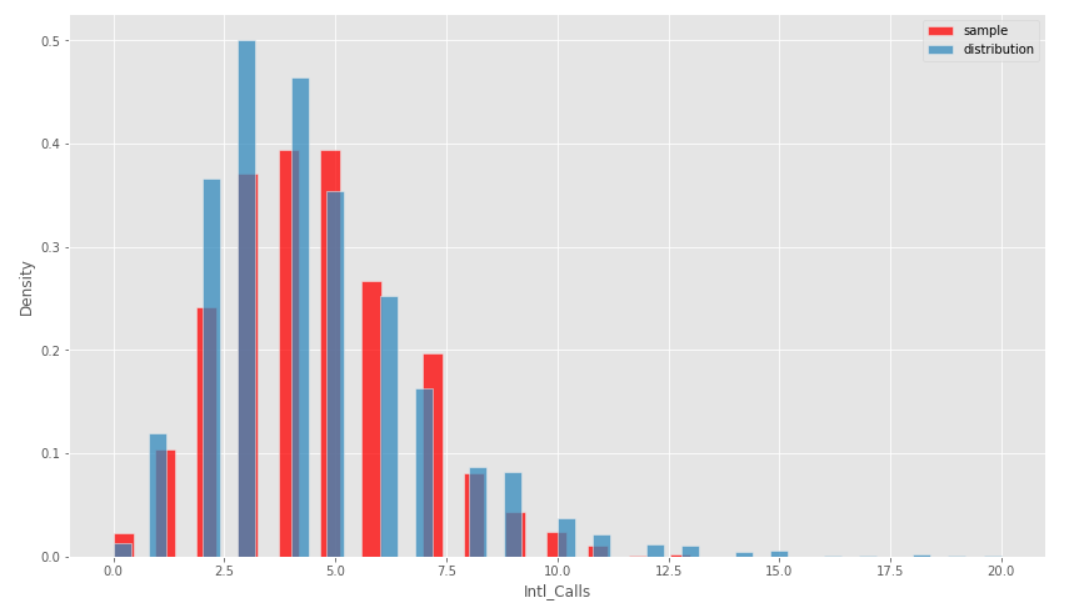
Выведем на график результат сэмплирование, чтобы оценить приближенность выборки к выбранному распределению.



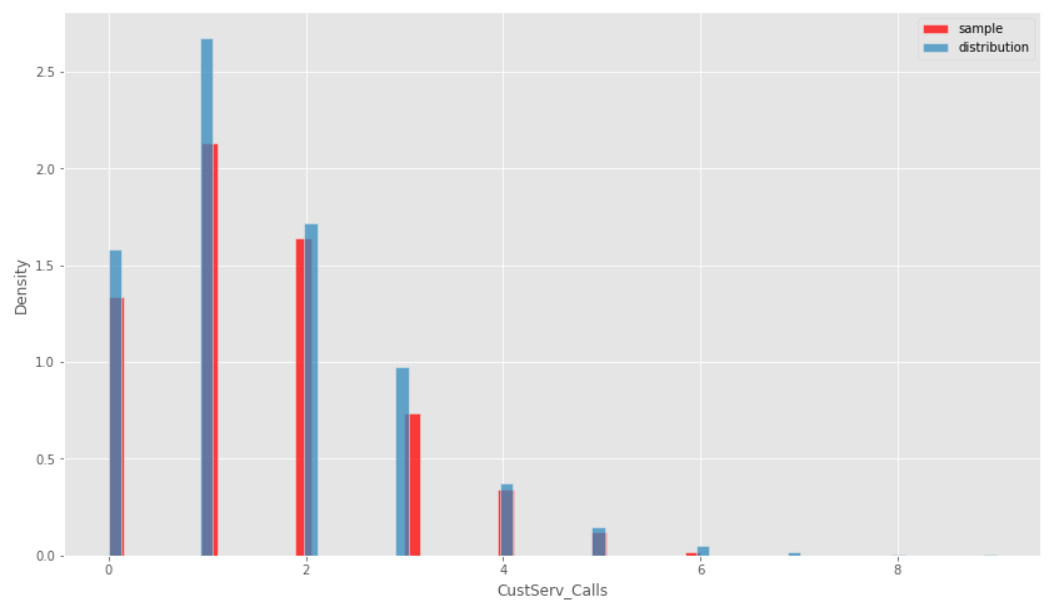
## Рисунок 25 - Сэмплирование обратной функцией для «Night\_Charge»



## Рисунок 26 - Сэмплирование обратной функцией для «Eve\_Charge»



## Рисунок 27 - Сэмплирование обратной функцией для «Intl\_Calls»

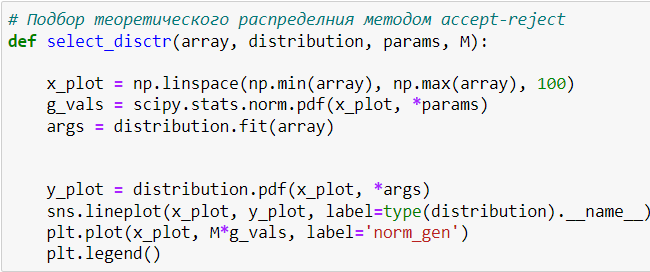


## Рисунок 28 - Сэмплирование обратной функцией для «CustServ\_Calls»

Исходя из графиков, сэмплированная выборка подчиняется выбранным законам распределения и может быть использована в дальнейших процедурах.

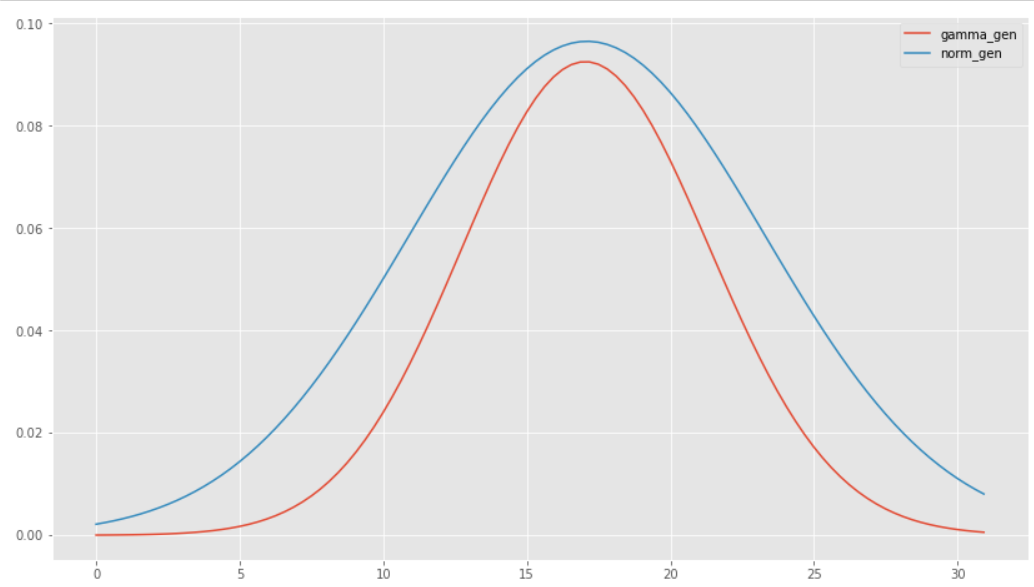
Вторым способом сэмплирование случайной величины является accept-reject метод, суть которого заключается в выборе теоретического распределение, которое по графику накрывает выборку. Далее мы сэмплируем значения из уже известного распределения и выбираем только те значения, которые попадают и в исходную выборку.

Первым шагом подберем теоретическое распределение, которое оптимально накроет наши данные.

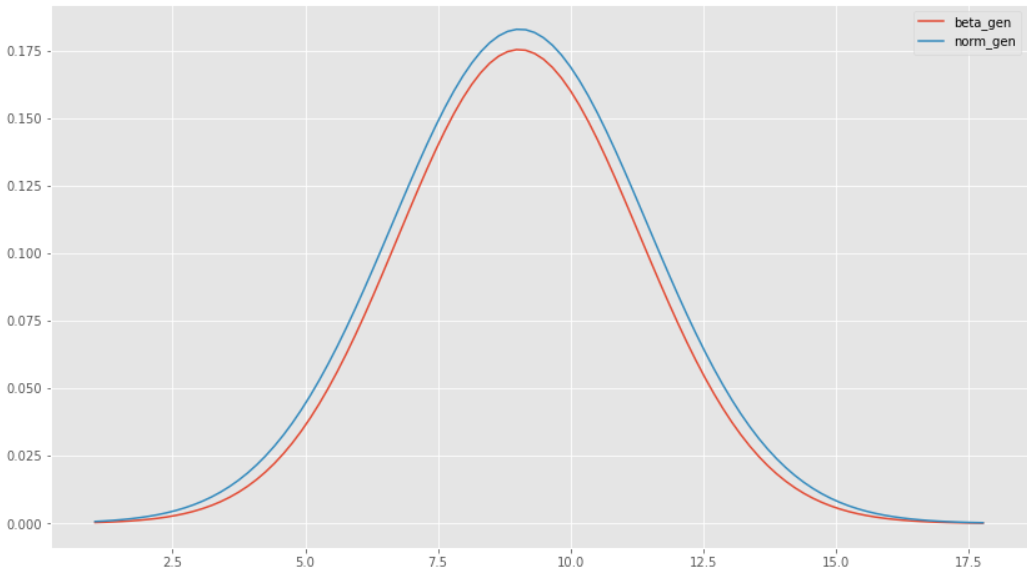


## Рисунок 29 - Подбор теоретического распределения методом accept-reject

В качестве теоретического распределения будем использовать нормальное, так как оно наиболее оптимально подходит по графику. Подбираем параметры нормального распределения так, чтобы оно накрывало исходное распределение, но зазор между ними был минимальным.

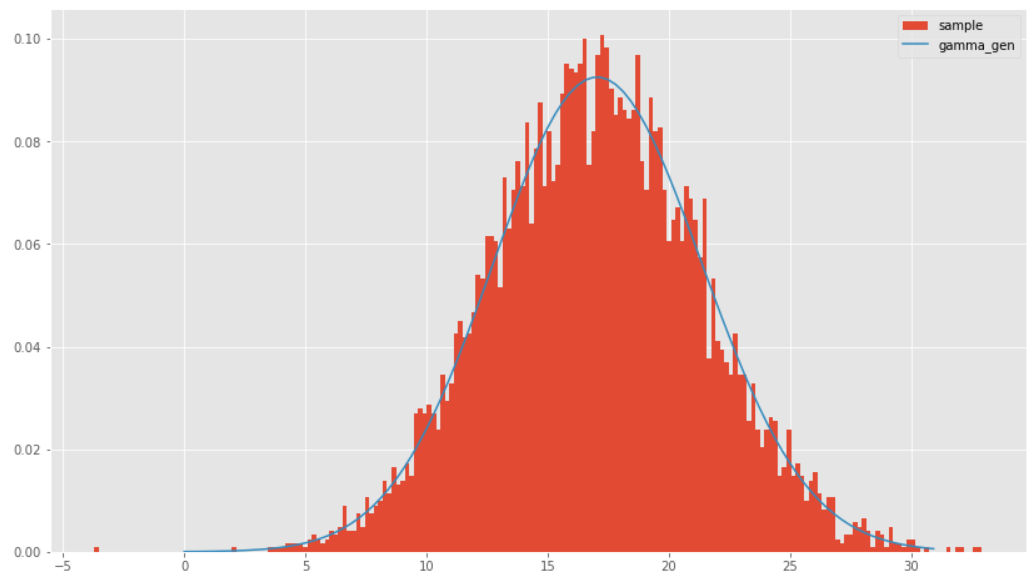


## Рисунок 30 - Подбор теоретического распределения методом accept-reject для «Eve\_Charge»



## Рисунок 31 - Подбор теоретического распределения методом accept-reject для «Night\_Charge»

Далее реализуем accept-reject сэмплирование.



## Рисунок 32 - Accept-reject сэмплирование для «Eve\_Charge»

## C:\Users\Test\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Снимок экрана (84).png

## Рисунок 33 - Accept-reject сэмплирование для «Night\_Charge»

Сэмплированная выборка попадает под изначальное распределение случайной величины, таким образом, параметры подобраны верно и сэмплирование проведено успешно.

## 

# Исходный код

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import scipy

plt.style.use('ggplot')

%matplotlib inline

plt.rcParams["figure.figsize"] = (14,8)

import warnings

warnings.filterwarnings(action='ignore')

df = pd.read\_csv('Датасеты/DataSetTelecom\_churn.csv',delimiter=';')

df = df[['Intl Calls','Eve Charge','Night Charge','CustServ Calls']]

df.rename(columns={"Intl Calls": "Intl\_Calls",'Eve Charge':'Eve\_Charge','Night Charge':'Night\_Charge','CustServ Calls':'CustServ\_Calls'},inplace=True)

# Расчет оптимального количества столбцов гистограммы

def number\_of\_cols(df\_num):

rows = df\_num.shape[0]

num = np.around(1 + 3.32 \* np.log(rows)).astype('int64')

return num

# Гистограммы распределений

import seaborn as sns

plt.figure(figsize=(18,12))

for i in range(1,5):

plt.subplot(2,2,i)

plt.subplots\_adjust(wspace=0.15, hspace=0.2)

plt.title(df.columns[i-1] + ' histogram', fontsize=20)

plt.xlabel(' ')

sns.histplot(data=df, x=df.columns[i-1],bins=number\_of\_cols(df))

# Ядерная оценка плотности распределения

plt.figure(figsize=(18,7))

plt.subplot(1,2,1)

plt.title('Eve\_Charge', fontsize=20)

plt.xlabel(' ')

sns.kdeplot(data=df, x='Eve\_Charge',bw\_adjust=0.9)

plt.hist(df['Eve\_Charge'], bins=number\_of\_cols(df), density=True)

plt.subplot(1,2,2)

plt.title('Night\_Charge', fontsize=20)

plt.xlabel(' ')

sns.kdeplot(data=df, x='Night\_Charge',bw\_adjust=0.88)

plt.hist(df['Night\_Charge'], bins=number\_of\_cols(df), density=True)

df[['Eve\_Charge','Night\_Charge']].describe()

plt.figure(figsize=(16,22))

for i in range(1,5):

plt.subplot(4,1,i)

plt.subplots\_adjust(hspace=0.3)

plt.title(str(df.columns[i-1]) + ' boxplot', fontsize=20)

sns.boxplot(data=df, x=df.columns[i-1])

# Делаем усеченную выборку

def truncated\_array(array, tail):

array\_trunc = []

quantile\_top, quantile\_bot = np.percentile(array, tail)

for i in array:

if i < quantile\_top and i > quantile\_bot:

array\_trunc.append(i)

return array\_trunc

# Выборка для дисретных величин

from sklearn.model\_selection import train\_test\_split

Intl\_Calls\_train, Intl\_Calls\_test = train\_test\_split(df['Intl\_Calls'], test\_size=0.9, random\_state=42)

Intl\_Calls\_train = Intl\_Calls\_train[Intl\_Calls\_train < 10]

CustServ\_Calls\_train, CustServ\_Calls\_test = train\_test\_split(df['CustServ\_Calls'], test\_size=0.9, random\_state=42)

CustServ\_Calls\_train = CustServ\_Calls\_train[CustServ\_Calls\_train < 5]

# Строим графики теоретических распределений, подобранных с помощью метода максимального правдоподобия для непрерывной случайной величины

def mle\_distplot(array, label):

plot = sns.histplot(array, stat='density')

plt.title(label, fontsize=20)

list\_distr = [scipy.stats.norm, scipy.stats.lognorm,scipy.stats.beta,scipy.stats.alpha,scipy.stats.gamma]

for distribution in list\_distr:

x\_plot = np.linspace(np.min(array), np.max(array), 100)

args = distribution.fit(array)

print('Распределение', type(distribution).\_\_name\_\_, ', аргументы:', args)

y\_plot = distribution.pdf(x\_plot, \*args)

sns.lineplot(x\_plot, y\_plot, label=type(distribution).\_\_name\_\_)

plt.legend()

mle\_distplot(df.Eve\_Charge, 'Eve\_Charge')

mle\_distplot(df.Night\_Charge, 'Night\_Charge')

#Оценка параметров распределения непрерывной СВ с помощью метода наименьших квадратов

from scipy.optimize import least\_squares

def LS\_cont(array, distribution):

x0 = distribution.fit(array)

def func(x):

list\_quant = np.arange(0,1,0.05)

xdata = np.quantile(array, list\_quant)[1:]

diff = (abs((distribution.ppf(list\_quant, \*x)[1:]) - xdata)\*\*2).sum()

return diff

res = least\_squares(func, x0)

return res.x

# Строим графики теоретических распределений, подобранных с помощью метода наименьших квадратов для непрерывной

# случайной величины

def ls\_distplot(array, label):

list\_distr = [scipy.stats.norm, scipy.stats.lognorm,scipy.stats.beta,scipy.stats.alpha,scipy.stats.gamma]

for distr in list\_distr:

args = LS\_cont(array, distr)

print('Распределение', type(distr).\_\_name\_\_, ', аргументы:', args)

sns.histplot(array, stat='density')

x\_plot = np.linspace(np.min(array), np.max(array), 100)

y\_plot = distr.pdf(x\_plot, \*args)

sns.lineplot(x\_plot, y\_plot, label=type(distr).\_\_name\_\_)

plt.title(label, size=20)

plt.legend()

ls\_distplot(df['Eve\_Charge'], 'Eve\_Charge')

ls\_distplot(df.Night\_Charge, 'Night\_Charge')

# Оцениваем параметры распределения для дискретных величин c помощью метода максимального правдободобия

def discr\_fit(array, distribution, bounds):

res = scipy.stats.fit(distribution, array, bounds)

args = tuple(res.params)

return args

Intl\_Calls\_poisson\_params\_mle = discr\_fit(df['Intl\_Calls'], scipy.stats.poisson, [(0, 300)])

Intl\_Calls\_hypergeom\_params\_mle = discr\_fit(df['Intl\_Calls'], scipy.stats.hypergeom, [(3333, 3333),(0, 200),(0, 200)])

CustServ\_Calls\_poisson\_params\_mle = discr\_fit(df['CustServ\_Calls'], scipy.stats.poisson, [(0, 300)])

CustServ\_Calls\_hypergeom\_params\_mle = discr\_fit(df['CustServ\_Calls'], scipy.stats.hypergeom, [(3333, 3333),(0, 200),(0, 200)])

# Параметры усеченных выборок

Intl\_Calls\_train\_poisson\_params\_mle = discr\_fit(Intl\_Calls\_train, scipy.stats.poisson, [(0, 300)])

Intl\_Calls\_hypergeom\_params\_mle = discr\_fit(Intl\_Calls\_train, scipy.stats.hypergeom, [(3333, 3333),(0, 200),(0, 200)])

CustServ\_Calls\_train\_poisson\_params\_mle = discr\_fit(CustServ\_Calls\_train, scipy.stats.poisson, [(0, 300)])

CustServ\_Calls\_train\_hypergeom\_params\_mle = discr\_fit(CustServ\_Calls\_train, scipy.stats.hypergeom, [(3333, 3333),(0, 200),(0, 200)])

# Строим графики теоретических распределений для дискретной случайной величины

def discr\_distplot(array, distribution, args):

val\_sum = array.shape[0]

categories = array.unique()

categories.sort()

pmf = distribution(\*args).pmf(categories)

pmf = (pmf \* val\_sum).round().astype('int')

plt.bar(categories, pmf, label=type(distribution).\_\_name\_\_)

sns.histplot(array, label = 'distribution')

plt.legend()

discr\_distplot(df['Intl\_Calls'], scipy.stats.poisson, Intl\_Calls\_poisson\_params\_mle)

discr\_distplot(df['CustServ\_Calls'], scipy.stats.poisson, CustServ\_Calls\_poisson\_params\_mle)

# Оцениваем параметры распределения для дискретных величин c помощью метода наименьших квадратов

def LS\_discr(array, distribution, bounds):

res = scipy.stats.fit(distribution, array, bounds)

x0 = tuple(res.params)

val\_sum = array.shape[0]

freq = np.array(array.groupby(array).count())

categories = array.unique()

categories.sort()

pmf = distribution(\*x0).pmf(categories)

pmf = (pmf \* val\_sum).round().astype('int')

def func(x):

diff = (abs(freq - pmf)\*\*2).sum()

return diff

res = least\_squares(func, x0)

return res.x

Intl\_Calls\_poisson\_params\_ls = LS\_discr(df['Intl\_Calls'], scipy.stats.poisson, [(0, 300)])

Intl\_Calls\_hypergeom\_params\_ls = LS\_discr(df['Intl\_Calls'], scipy.stats.hypergeom, [(3333, 3333),(0, 200),(0, 200)])

CustServ\_Calls\_poisson\_params\_ls = LS\_discr(df['CustServ\_Calls'], scipy.stats.poisson, [(0, 300)])

CustServ\_Calls\_hypergeom\_params\_ls = LS\_discr(df['CustServ\_Calls'], scipy.stats.hypergeom, [(3333, 3333),(0, 200),(0, 200)])

discr\_distplot(df['Intl\_Calls'], scipy.stats.poisson, Intl\_Calls\_poisson\_params\_ls)

discr\_distplot(df['CustServ\_Calls'], scipy.stats.poisson, CustServ\_Calls\_poisson\_params\_ls)

# Квантильный биплот непрерывных случайных величин

def cont\_biplot(array, distribution, label):

figure, ax = plt.subplots(1, 1, figsize=(10, 6))

args = distribution.fit(array)

print('Распределение', type(distribution).\_\_name\_\_, 'аргументы:', args)

scipy.stats.probplot(array, dist=distribution, sparams=(args), plot=ax)

ax.get\_lines()[1].set\_color('black')

plt.title(f'{label} biplot', fontsize=20)

plt.show()

cont\_biplot(df.Eve\_Charge, distribution=scipy.stats.gamma, label='Eve\_Charge')

cont\_biplot(truncated\_array(df.Night\_Charge, [95,5]), distribution=scipy.stats.beta, label='Night\_Charge')

# Квантильный биплот дискретных случайных величин

def discr\_biplot(array, distribution, label, args):

figure, ax = plt.subplots(1, 1, figsize=(10, 6))

print('Распределение', type(distribution).\_\_name\_\_, 'аргументы:', args)

scipy.stats.probplot(array, dist = distribution, sparams=(args), plot=ax)

ax.get\_lines()[1].set\_color('black')

plt.title(f'{label} biplot', fontsize=20)

plt.show()

discr\_biplot(df['CustServ\_Calls'], scipy.stats.poisson, 'CustServ\_Calls', CustServ\_Calls\_poisson\_params\_ls)

# Статистические тесты Колмагорова-Смирнова и Омега-квадрат

def stat\_test\_cont(array, distribution, cdf):

args = distribution.fit(array)

kstest = scipy.stats.kstest(array, cdf=cdf, args=args).pvalue

omega2 = scipy.stats.cramervonmises(array, cdf=cdf, args=args).pvalue

print('Критерий Колмагорова-Смирнова: {:.4f}'.format(kstest))

print('Критерий Омега-квадрат: {:.4f}'.format(omega2))

stat\_test\_cont(truncated\_array(df.Night\_Charge, [95,5]), distribution=scipy.stats.beta, cdf='beta')

stat\_test\_cont(df.Eve\_Charge, distribution=scipy.stats.gamma, cdf='gamma')

# Проверяем нулевую гипотезу о том, что категориальные данные имеют заданные частоты теоретического распределения

def stat\_test\_chi2(array, distribution, args):

val\_sum = array.shape[0]

categories = array.unique()

categories.sort()

pmf = distribution(\*args).pmf(categories)

pmf = (pmf \* val\_sum).round().astype('int')

if abs(freq.sum() - pmf.sum()) > 0:

rest = freq.sum() - pmf.sum()

pmf[pmf.argmax()] += rest

p\_value = scipy.stats.chisquare(f\_obs = freq, f\_exp = pmf)[1]

print('Критерий Хи-квадрат: {:.5f}'.format(p\_value))

stat\_test\_chi2(df['CustServ\_Calls'], scipy.stats.poisson, CustServ\_Calls\_poisson\_params\_mle)

stat\_test\_chi2(df['Intl\_Calls'], scipy.stats.hypergeom, Intl\_Calls\_hypergeom\_params\_mle)

stat\_test\_chi2(Intl\_Calls\_train, scipy.stats.poisson, Intl\_Calls\_train\_poisson\_params\_mle)

stat\_test\_chi2(CustServ\_Calls\_train, scipy.stats.hypergeom, CustServ\_Calls\_train\_hypergeom\_params\_mle)

# Семплирование непрерывной СВ методом обратной функции

def cont\_inverse\_sampling(array, distribution):

args = distribution.fit(array)

x\_array = np.random.rand(2000)

inverse = distribution.ppf(x\_array, \*args)

sns.histplot(array, stat='density', label = 'distribution')

sns.kdeplot(x=inverse, label = 'sample')

plt.legend()

cont\_inverse\_sampling(df['Night\_Charge'], scipy.stats.beta)

cont\_inverse\_sampling(df['Eve\_Charge'], scipy.stats.gamma)

# Семплирование дискретной СВ методом обратной функции

def discr\_inverse\_sampling(array, distribution, args):

x\_array = np.random.rand(2000)

inverse = distribution.ppf(x\_array, \*args).round()

sns.histplot(inverse, stat='density', label = 'sample', color='red')

sns.histplot(array, stat='density', label = 'distribution')

plt.legend()

discr\_inverse\_sampling(df['Intl\_Calls'], scipy.stats.poisson, Intl\_Calls\_poisson\_params\_mle)

discr\_inverse\_sampling(df['CustServ\_Calls'], scipy.stats.poisson, CustServ\_Calls\_poisson\_params\_mle)

# Семплирование непрерывной СВ методом rvs

def cont\_rvs\_sampling(array, distribution):

args = distribution.fit(array)

print(args)

rvs = distribution.rvs(\*args, size=2000)

sns.kdeplot(x=rvs, label = 'sample')

sns.histplot(array, stat='density', label = 'distribution')

plt.legend()

cont\_rvs\_sampling(df['Eve\_Charge'], scipy.stats.gamma)

cont\_rvs\_sampling(df['Night\_Charge'], scipy.stats.beta)

# Подбор теоретического распределния методом accept-reject

def select\_disctr(array, distribution, params, M):

x\_plot = np.linspace(np.min(array), np.max(array), 100)

g\_vals = scipy.stats.norm.pdf(x\_plot, \*params)

args = distribution.fit(array)

y\_plot = distribution.pdf(x\_plot, \*args)

sns.lineplot(x\_plot, y\_plot, label=type(distribution).\_\_name\_\_)

plt.plot(x\_plot, M\*g\_vals, label='norm\_gen')

plt.legend()

select\_disctr(df['Eve\_Charge'], scipy.stats.gamma, [17.09442093 , 6.2], 1.5)

select\_disctr(df['Night\_Charge'], scipy.stats.beta, [9.04105244, 2.4], 1.1)

# Сэмплирование методом accept-reject

def accept\_reject(array, distribution, M, params):

args = distribution.fit(array)

samples = []

N = 10000

for \_ in range(N):

x = np.random.normal(\*params)

prob\_accept = distribution.pdf(x, \*args) / (M \* scipy.stats.norm.pdf(x, \*params))

if np.random.random() < prob\_accept:

samples.append(x)

plt.hist(samples, bins=200, density=True, label = 'sample')

x\_plot = np.linspace(np.min(array), np.max(array), 100)

y\_plot = distribution.pdf(x\_plot, \*args)

sns.lineplot(x\_plot, y\_plot, label=type(distribution).\_\_name\_\_)

accept\_reject(df['Eve\_Charge'], scipy.stats.gamma, 1.5, [17.09442093 , 6.2])

accept\_reject(df['Night\_Charge'], scipy.stats.beta, 1.1, [9.04105244, 2.4])