**Министерство науки высшего образования Российской Федерации**

**федеральное государственное автономное образовательное учреждение высшего образования**

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»**

**(Университет ИТМО)**

**ОТЧЕТ**

по лабораторной работе №1

**Выполнил:**

студент гр. К42102с

Мирзоев А. Ш.

**Проверил:**

Мусаев А.А.

Санкт-Петербург

2022 г.

**Задание 1**

Изучите модуль random. Создайте список из N случайных элементов, которые могут принимать значение 0 или 1. Для созданного списка определить процентное отношение 0, 1, а также повторяющихся подряд элементов (00, 11, 000, 111 и т.д.)

Ссылка на Git:<https://github.com/dropex7/Data_analysis_labs/tree/main/NewLab1>

**Решение**

Был написан алгоритм для подсчета повторений 0 и 1, а далее высчитано процентное отношение

import itertools  
import random  
  
N = 100  
numArray = []  
counts = []  
  
for i in range(N):  
 numArray.append(random.randint(0, 1))  
  
  
def show\_subsequence(count):  
 if count > 0:  
 return "".join(itertools.repeat("0", count))  
 else:  
 return "".join(itertools.repeat("1", count \* -1))  
  
  
tempValue = numArray[0]  
tempCount = 1  
  
print(numArray)  
  
# отрицательное значение = 1, положительное = 0  
for item in numArray[1:]:  
 if tempValue == item:  
 tempCount += 1  
 else:  
 if tempValue == 0:  
 counts.append(tempCount)  
 tempValue = 1  
 else:  
 counts.append(tempCount \* -1)  
 tempValue = 0  
 tempCount = 1  
  
if tempValue == 0:  
 counts.append(tempCount)  
else:  
 counts.append(tempCount \* -1)  
  
for k in set(counts):  
 print(f"{show\_subsequence(k)}: {(counts.count(k) / len(counts)) \* 100}%")

**Результат** Изображение выглядит как текст

Автоматически созданное описание

**Задание 2**

Изучите модуль matplotlib. Для случайного набора данных, определить математическое ожидание, среднеквадратическое отклонение и построить линейную функцию, используя метод наименьших квадратов.

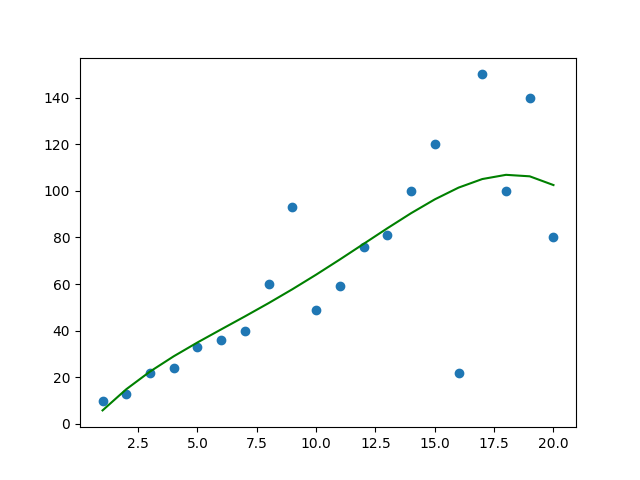
**Решение**

import numpy as np  
from matplotlib import pyplot as plt  
  
x = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20])  
y = np.array([10, 13, 22, 24, 33, 36, 40, 60, 93, 49, 59, 76, 81, 100, 120, 22, 150, 100, 140, 80])  
  
print(f"Математическое ожидание: {np.mean(y)}")  
print(f"Среднеквадратичное отклонение: {np.std(y)}")  
  
t = np.polyfit(x, y, 4)  
f = np.poly1d(t)  
  
plt.scatter(x, y)  
plt.plot(x, f(x), 'g')  
plt.show()

**Результат**

Изображение выглядит как текст

Автоматически созданное описание

****

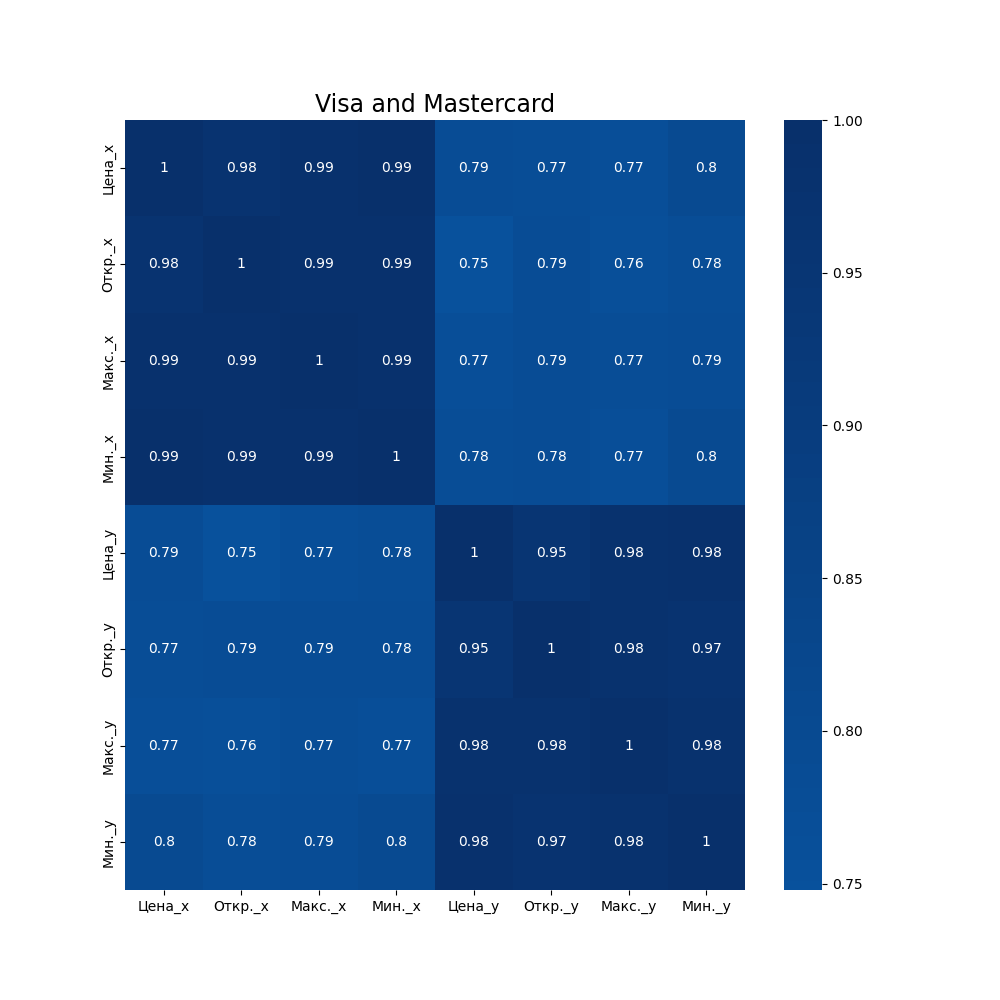
**Задание 3**

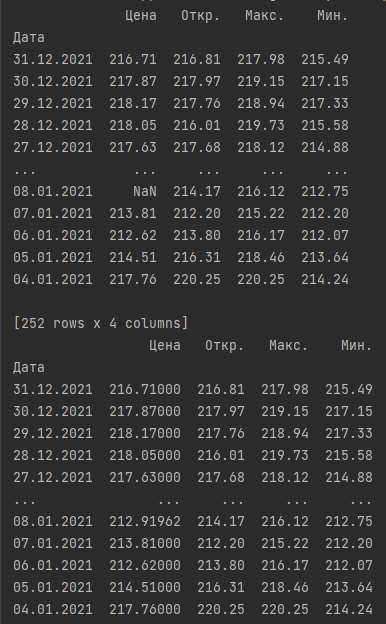
Скачайте данные о котировках акций Visa и Mastercard за любой год с интервалом 1 день. Используя коэффициент корреляции Пирсона, оцените взаимосвязь этих двух компаний. Удалите случайные данные о котировках данных компаний и напишите функции для их восстановления с помощью винзорирования, линейной аппроксимации и корреляционного восстановления. Сделайте выводы о каждом методе восстановления данных.

**Решение**

import math  
import random  
  
import numpy as np  
import pandas as pd  
from matplotlib import pyplot as plt  
import seaborn as sns  
  
  
def endorsement(data, column\_index):  
 for index in range(data[column\_index].size):  
 if math.isnan(data[column\_index][index]):  
 if math.isfinite(data[column\_index][index - 1]):  
 data[column\_index][index] = data[column\_index][index - 1]  
 continue  
 elif math.isfinite(data[column\_index][index + 1]):  
 data[column\_index][index] = data[column\_index][index + 1]  
 continue  
  
  
def linear\_recovery(data, column\_index):  
 for index in range(data[column\_index].size):  
 if math.isnan(data[column\_index][index]):  
 missing\_value = data[column\_index].interpolate()[index]  
 data[column\_index][index] = missing\_value  
  
  
def correlation\_recovery(data, column\_index):  
 for item in range(data[column\_index].size):  
 if math.isnan(data[column\_index][item]):  
 if item + 1 < data[column\_index].size:  
 if math.isfinite(data[column\_index][item + 1]):  
 p1 = data["Макс."][item]  
 p2 = data["Макс."][item + 1]  
 v2 = data[column\_index][item + 1]  
 missing\_value = (v2 / p1) \* p2  
 data[column\_index][item] = missing\_value  
 elif math.isfinite(data[column\_index][item - 1]):  
 p1 = data["Макс."][item - 1]  
 v1 = data[column\_index][item - 1]  
 p2 = data["Макс."][item]  
 missing\_value = (p1 / p2) \* v1  
 data[column\_index][item] = missing\_value  
  
  
  
visa = pd.read\_csv('visa.csv', sep=';', index\_col='Дата')  
mastercard = pd.read\_csv('mastercard.csv', sep=';', index\_col='Дата')  
visa\_and\_mastercard = visa.merge(mastercard, on='Дата')  
  
plt.figure(figsize=(10, 10))  
sns.heatmap(visa\_and\_mastercard.corr(),  
 center=0, cmap='Blues', annot=True)  
plt.title('Visa and Mastercard', fontsize=17)  
plt.show()  
  
  
missing\_pct = int(visa['Цена'].size \* 0.1)  
i = [random.choice(range(visa['Цена'].shape[0])) for \_ in range(missing\_pct)]  
visa['Цена'][i] = np.NaN  
visa['Цена'][visa['Цена'].shape[0] - 1] = np.NaN  
visa['Цена'][visa['Цена'].shape[0] - 2] = np.NaN  
  
visa\_1 = visa.copy()  
visa\_2 = visa.copy()  
visa\_3 = visa.copy()  
  
print(visa\_3)  
  
endorsement(visa\_1, "Цена")  
linear\_recovery(visa\_2, "Цена")  
correlation\_recovery(visa\_3, "Цена")  
  
print(visa\_3)

**Результат**

****



**Вывод**

По результатам видно, что все показатели зависят друг от друга. Есть прямая зависимость между visa и mastercard, которая прослеживается на тепловой карте. При замене пустых значений линейная аппроксимация отрабатывает хорошо, есть много альтернатив винзонированию, как замена на среднее значение или медиану, а также константу, что на мой взгляд более точно бы заменяло данные. При коррелиционном восстановление данные тоже восстанавливались довольно точно, но есть кейсы при которых, этот вариант не очень хорошо подходит, когда у нас много пустых значений.