**Министерство науки высшего образования Российской Федерации**

**федеральное государственное автономное образовательное учреждение высшего образования**

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»**

**(Университет ИТМО)**

**ОТЧЕТ**

по лабораторной работе №1

Тема: «Анализ данных с гетерогенных источников»

**Выполнил:**

студент гр. К42112с

Кумашев Э.

**Проверил:**

Мусаев А.А.

Санкт-Петербург

2022 г.

1. **Задачи**
2. 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.).
3. Изучите модуль matplotlib. Для случайного набора данных, определить математическое ожидание, среднеквадратическое отклонение и построить линейную функцию, используя метод наименьших квадратов.
4. 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.
5. **Задача 1**

Был изучен модули random и itertools. За инструмент работы были взяты **Python и Jupyter Notebook.**

Вначале импортируем библиотеку **random и itertools:**

import random as rd

import itertools

Для создания списка из N случайных элементов, которые могут принимать значение 0 или 1, был использован следующий код:

N = 100

numArray = []

counts = []

for i in range(N):

numArray.append(random.randint(0, 1))

print(numArray)

Для подсчета 0 и 1, а также пар цифр:

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

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

print(numArray)

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}%")

N = 100

n = 5

repeating\_list = ['0'\*i for i in range(1, n+1)] + ['1'\*i for i in range(1, n+1)]

print(repeating\_list)

binary\_list = rd.choices(repeating\_list, k = N)

binary\_list

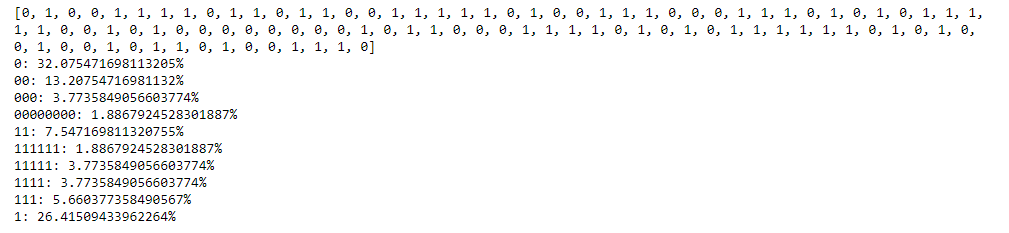


Рисунок 1. Вывод списка элементов 0,1 и повторяющихся элементов в процентном отношении

bin\_1 = (len([i for i in binary\_list if i == '1'])/N)\*100

print("percentage of 1's")

print(round(bin\_1,2))

print('\*'\*10)

bin\_0 = (len([i for i in binary\_list if i == '0'])/N)\*100

print("percentage of 0's")

print(round(bin\_0,2))

1. **Задача 2**

Был изучен модуль matplotlib. За инструмент работы были взяты **Python и Jupyter Notebook**

Вначале импортируем библиотеку **matplotlib:**

import matplotlib.pyplot as plt

Затем создаем случайный набор данных:

N = 100

random\_list = rd.choices(range(1,100), k = N)

random\_list



Рисунок 2. Вывод случайного набора чисел в диапазоне от 1 до 100

Определение математического ожидания:

mean = sum(random\_list)/len(random\_list)

mean

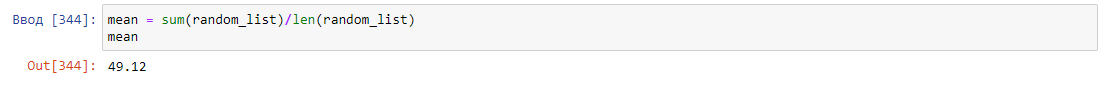


Рисунок 3. Определение математического ожидания с выводом

Определение математического ожидания и среднеквадратического отклонения через **numpy**:

import numpy as np

random\_list = np.array(random\_list)

mean = random\_list.mean()

std = random\_list.std()

print(f'mean of the random dataset is {mean}')

print(f'std of the random dataset is {std}')



Рисунок 4. Вывод математического ожидания и среднеквадратического отклонения

Построение линейной функции используя метод наименьших квадратов:

import matplotlib.pyplot as plt

#create basic scatterplot

x = range(N)

y = random\_list

plt.plot(x, y, 'o')

#obtain m (slope) and b(intercept) of linear regression line

m, b = np.polyfit(x, y, 1)

#add linear regression line to scatterplot

plt.plot(x, m\*x+b)

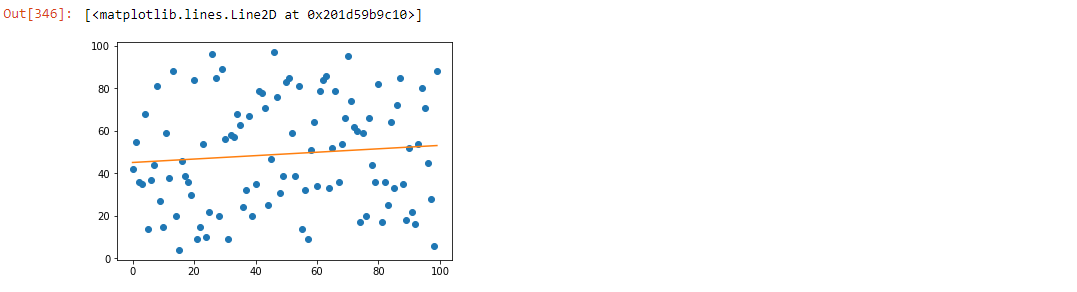


Рисунок 5. Вывод линейной функции

1. **Задача 3**

За инструмент работы были взяты **Python и Jupyter Notebook.**

Для того чтобы скачать котировки акций упомянутых компаний нужно зайти на сайт https://ru.investing.com/markets/ . Выбрать нужную компанию, зайти на вкладку “Прошлые данные” (Рисунок 1), ставим “Временной период” – он же интервал на день, выставляем временной промежуток, в данном случае с 27.11.2021 по 27.11.2022 гг. и скачиваем котировки, нажимая на кнопку “Скачать данные”. И так с остальными акциями.

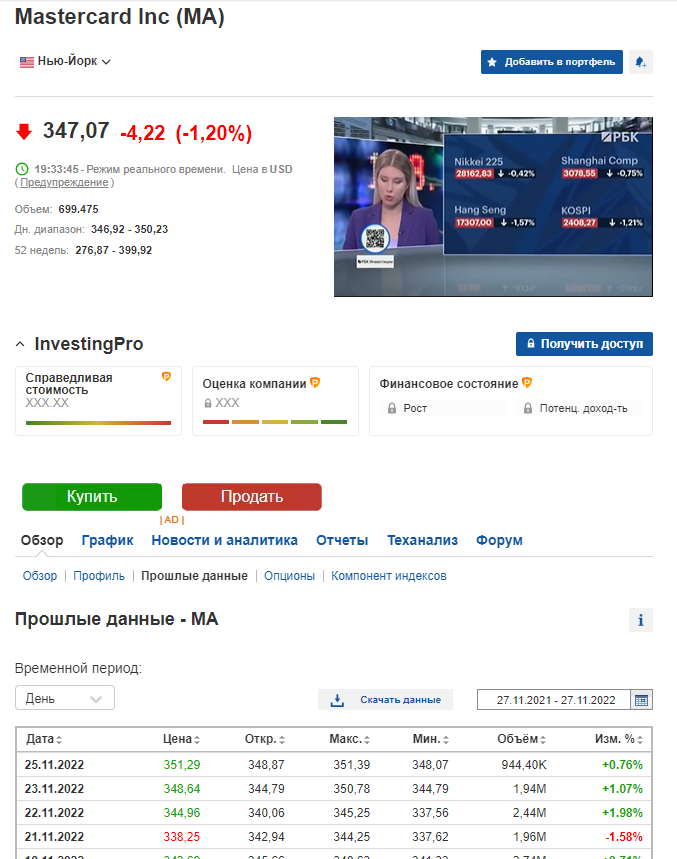


Рисунок 6. Сайт

Скачанные котировки компаний кладем в папку и импортируем модуль **pandas**:

import pandas as pd

Считывание файлов котировок .csv по пути расположения MA - MasterCard, V - Visa:

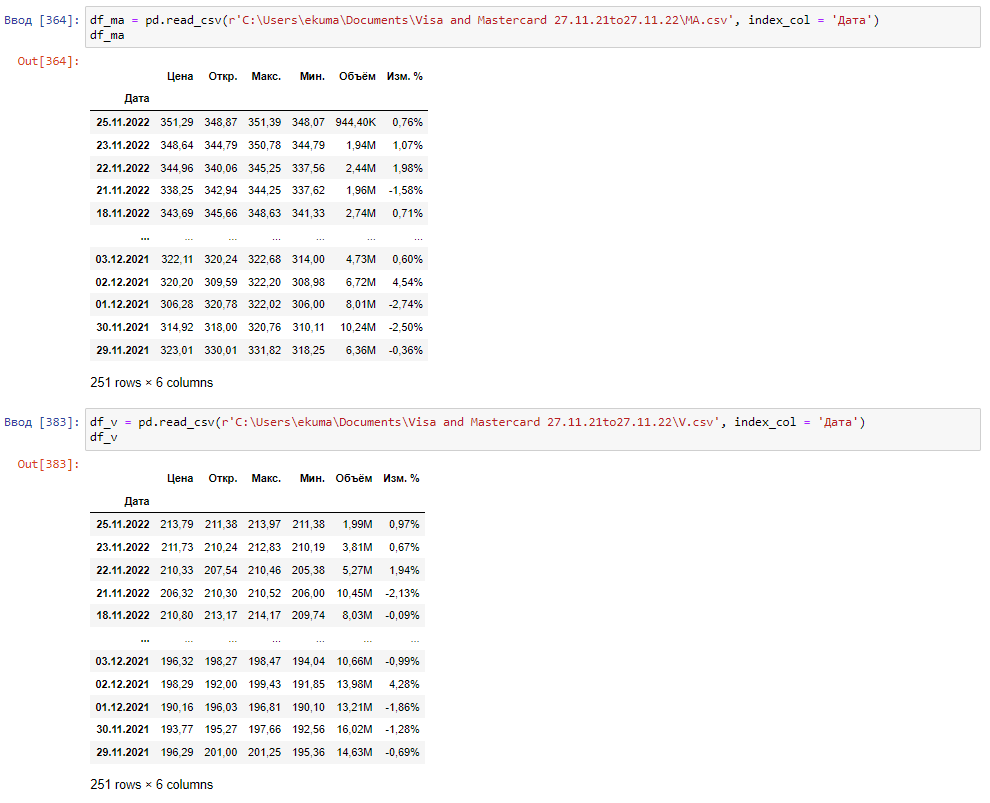


Рисунок 7. Считывание файлов котировок

Так как данные в колонках отличались, где-то это были несколько запятых в числе, где-то две запятые, понадобилось преобразовать все данные в одинаковый вид и создать датафреймы:

for i in range(len(df\_ma)):

for col in df\_ma.columns:

if col == 'Объём' or col == 'Изм. %':

df\_ma[col][i] = df\_ma[col][i][:-1].replace('.','').replace(',','.')

df\_v[col][i] = df\_v[col][i][:-1].replace('.','').replace(',','.')

else:

df\_ma[col][i] = df\_ma[col][i].replace('.','').replace(',','.')

df\_v[col][i] = df\_v[col][i].replace('.','').replace(',','.')

df\_ma = df\_ma.apply(pd.to\_numeric)

df\_v = df\_v.apply(pd.to\_numeric)

Для выведения значений корреляции:

df\_ma.corrwith(df\_v)



Рисунок 8. Корреляция между столбцами двух котировок акций

Необходимо отметить, что корреляция по столбцу «Объем» является отрицательной. С другими столбцами внутри котировок MasterCard низкая корреляция.

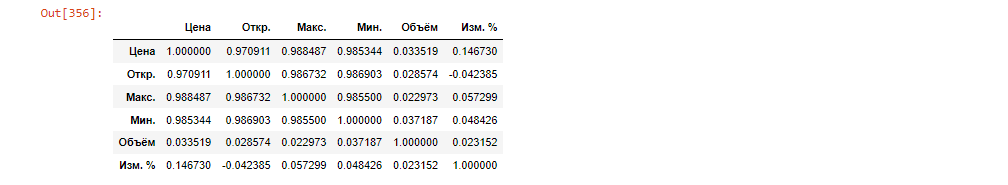


Рисунок 9. Корреляция между другими столбцами MasterCard

Следом были удалены данные котировок Mastercard за 21.11.2022, перед этим был осуществлен вывод значений за этот день:

print(df\_ma.loc['21.11.2022'] )

df\_ma.loc['21.11.2022'] = None

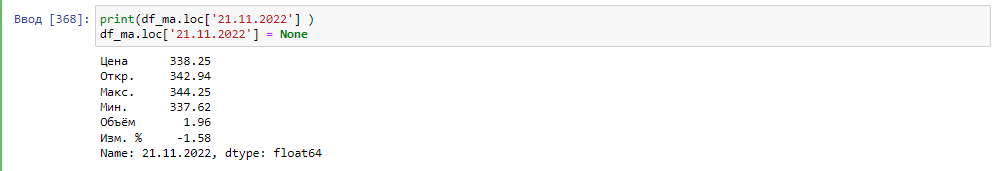


Рисунок 10. Вывод удаляемых значений

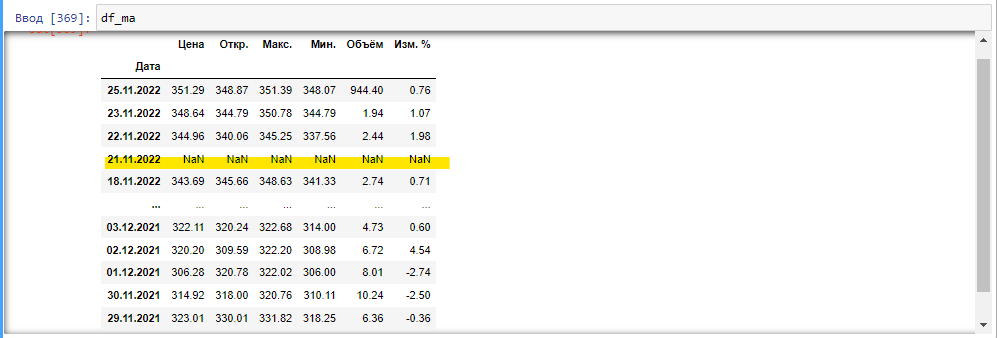


Рисунок 11. Вывод таблицы после удаления

Восстановление данных с помощью винзорирования:

from scipy.stats.mstats import winsorize

for col in df\_ma.columns:

missing\_value = winsorize(np.ma.masked\_invalid(df\_ma[col]), limits=(0.25, 0.25))[3]

print(f'{col} : {missing\_value}')



Рисунок 12. Вывод восстановленных данных используя виндзорирование

Исходя из данных полученных через виндзорирование можно сделать вывод, что в отличие от данных столбцов «Объем» и «Изм.» данные имеют небольшой разброс, однако вышеупомянутые столбцы имеют сильное различие (в столбце «Объем» более чем в 2 раза, а в столбце «Изм.» положительные значения вместо реальных отрицательных.

Восстановление данных с помощью линейной аппроксимации:

for col in df\_ma.columns:

missing\_value = df\_ma[col].interpolate()[3]

print(f'{col} : {missing\_value}')



Рисунок 13. Вывод восстановленных данных используя линейную аппроксимацию

Исходя из данных, полученных используя линейную аппроксимацию можно сделать вывод, что в отличие от метода винзорирования, данные столбцов «Цена», «Откр», «Макс», «Мин», «Объем» более приближены к реальным значениям, что нельзя сказать про «Изм.», которые также имеют положительные значения вместо реальных отрицательных.

Восстановление данных с помощью корреляционного восстановления:

for col in df\_ma.columns:

missing\_value = (df\_v.loc['21.11.2022'][col]/df\_v.loc['22.11.2022'][col])\*df\_ma.loc['22.11.2022'][col]

print(f'{col} : {missing\_value}')



Рисунок 14. Вывод восстановленных данных используя корреляционное восстановление

Как мы можем видеть, восстановленные данные с использованием корреляционного восстановления являются наиболее приближенными к реальным данным в отличие от других способов восстановления, кроме столбца «Объем».

**Вывод:**

На основании полученных данных с использованием методов восстановления данных с помощью винзорирования, линейной аппроксимации и корреляционного восстановления можно сделать вывод, что при положительном значении корреляции, восстановление данных используя метод корреляционного восстановления является наиболее оптимальным, однако, если же корреляция низкая или же отрицательная, есть смысл использовать другие методы.

Ссылка на Git: https://github.com/Aik101/Data-Analysis-LabWork1.git