Geekbrains

Анализ предсказаний курсов валют с помощью

модели машинного обучения на разный промежуток времени в зависимости от ключевых индикаторов на фондовом рынке

Программа:

Специализация: Искусственный интеллект. Цифровые Профессии

Студент: Эгипти Евгений Викторович

Тольятти

2024

Оглавление

[Введение 1](#_Toc172571293)

[Глава 1 Теоретическая часть 3](#_Toc172571294)

[1.1 Формирование гипотезы 3](#_Toc172571295)

[1.2 Пример 3](#_Toc172571296)

[Глава 2 Практическая часть 4](#_Toc172571297)

[2.1 Среда разработки 4](#_Toc172571298)

[2.2 Выбор датасетов 4](#_Toc172571299)

[2.3 Создание файла Jupiter Notebook 5](#_Toc172571300)

[2.4 Установка необходимых python-пакетов 5](#_Toc172571301)

[2.5 Импортируем нужные модули 6](#_Toc172571302)

[2.6 Загрузка датасетов 7](#_Toc172571303)

[2.7 Разведочный анализ данных (EDA) 9](#_Toc172571304)

[2.8 Визуализация данных 11](#_Toc172571305)

[2.8.1 Выделим численные признаки от даты 11](#_Toc172571306)

[2.8.2 Создадим словарь признаков и пояснения 11](#_Toc172571307)

[2.8.3 Визуализируем распределение числовых признаков 12](#_Toc172571308)

[Глава 3 13](#_Toc172571309)

[3.1 Создание новых признаков 13](#_Toc172571310)

[3.2 Проверка состояния данных 14](#_Toc172571311)

[3.3 Метрика, по которой будем оценивать качество работы модели 15](#_Toc172571312)

[3.4 Разбиение временного ряда на train и test 16](#_Toc172571313)

[Вывод 19](#_Toc172571314)

[Литература 20](#_Toc172571315)

Введение

Роль валютного курса в экономике переоценить сложно. Изменения валютного курса предопределяют статус той или иной страны в мировом экономическом пространстве, а вместе с ним ориентиры ее будущего развития, как во внешних, так и во внутренних связях.

Курс национальной валюты является важным ориентиром для принятия решений экономическими агентами, как на уровне населения, так и на уровне правительства. При высокой волатильности валюты возрастают издержки экспортно-импортных операций, растут цены на импортируемые товары. Это ведет к повышению цен на продукцию на внутреннем рынке. Чтобы защититься от роста цен и возможной девальвации, потребители увеличивают долю сбережений в иностранной валюте, что не способствует укреплению и стабилизации национальной. Резкие скачки национальной валюты дают повод населению снизить потребление, а предпринимателем сократить инвестиции в национальную экономику. Высокая волатильность препятствует бизнесу в построении долгосрочных стратегий развития, что приводит к снижению экспорта предприятиями и замедлением производства [1].

На данный момент машинное обучение является одной из наиболее развивающихся областей прикладной математики, позволяющих решать большой спектр задач предсказания и распознавания. Многие экономисты используют методы анализа данных для предсказания валютных курсов. Так, например, Martin Evans и Richard Lyons в своей статье «Micro-Based Exchange-Rate Forecasting» используют метод k ближайших соседей и метод опорных векторов для прогнозирования основных мировых валютных пар (EUR/USD, GBP/USD, USD/JPY) [3].

**Темой проекта** является предсказание курса валют с помощью модели машинного обучения.

**Целью** будет являться создание модели машинного обучения, которая будет предсказывать курс валют по отношению к рублю.

**Будет решена проблема** вероятности недополучить прибыль или понести убытки от финансовых, торговых и кредитных операций из-за изменчивости соотношения валют**.** То есть снизятся валютные риски.

**Задачи:**

* Собрать и обработать данные необходимые для обучения модели;
* Обучить модель на собранных данных;
* Поэкспериментировать с разными типами моделей;
* Проверить эффективность модели на тестовых данных;
* Проверить предсказания модели на актуальных данных.

**Инструменты:**

JupyterLab, Visual Code

**Состав команды:** Эгипти Евгений Викторович

1. Теоретическая часть
   1. Формирование гипотезы
   2. Пример
2. Практическая часть
   1. Среда разработки

В качестве среды разработки был использован JupyterLab [4], там больше возможностей по сравнению с Jupyter Notebook. Можно одновременно работать с несколькими файлами, использовать отладчик [8], если потребуется.

Данная среда разработки, а также другие инструменты входят в бесплатную программу Anaconda [12] которую можно скачать с сайта <https://www.anaconda.com/>.

Разработку проекта желательно вести в новом окружении, чтобы не возникло конфликта между версиями установленных и устанавливаемых пакетов. Для этого, открываем программу Anaconda и выбираем вкладку Environments, в открывшейся вкладке нажимаем кнопку Create, задаём название окружения и версию Python. Делаем активным наше окружение, в данном случае окружение diplom\_2024\_gb, см. рисунок 1.

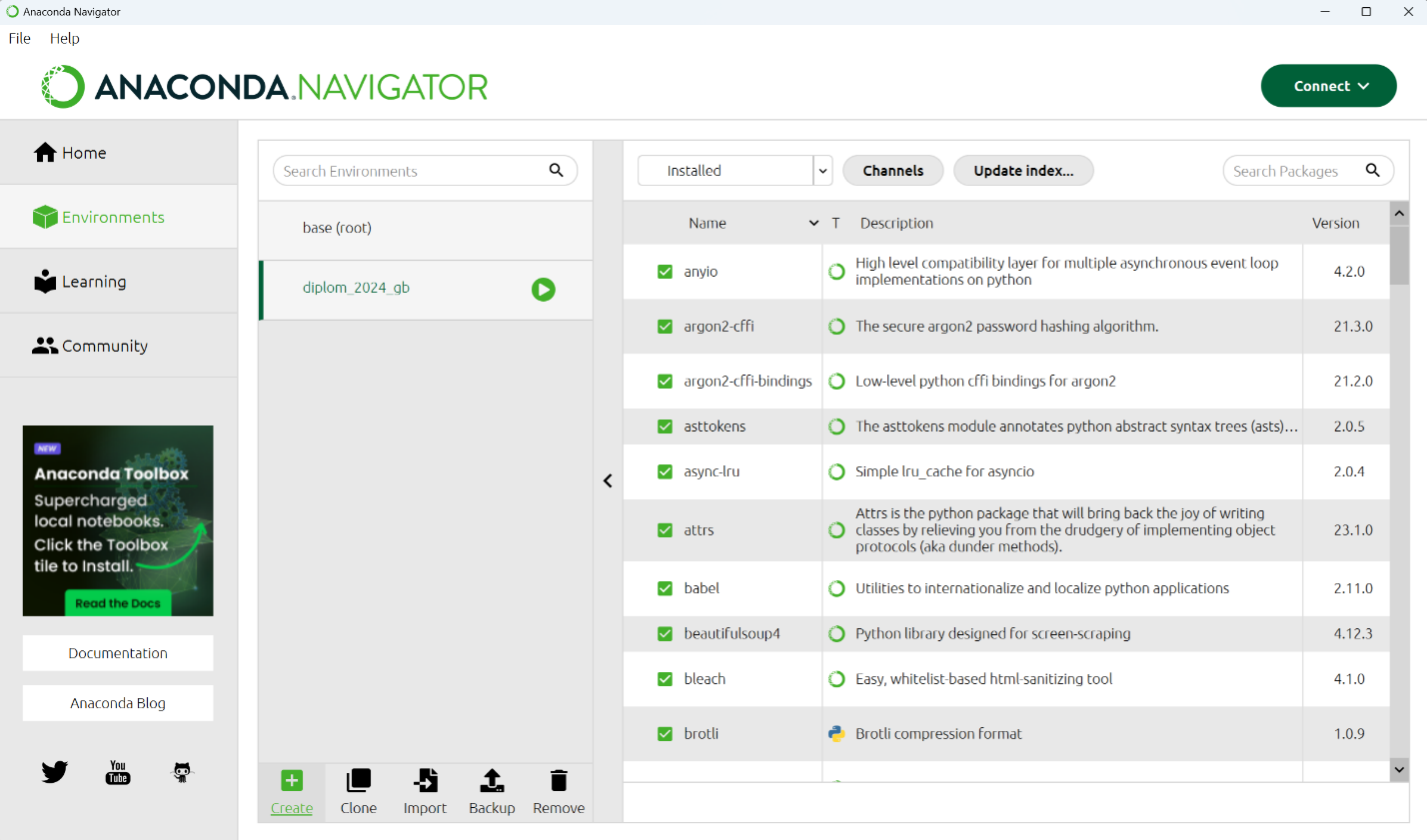


Рисунок 1. Выбор активного окружения diplom\_2024\_gb

После выбора окружения, нажимаем вкладку Home и выбираем среду разработки JupyterLab.

* 1. Выбор датасетов

Датасеты были взяты с сайта InvestFunds (<https://investfunds.ru/>), их можно бесплатно скачать в формате Excel, за выбранный промежуток времени.

В качестве ключевых индикаторов [6] были приняты следующие показатели:

* индекс Мосбиржи;
* индекс РТС;
* индекс S&P 500;
* платина (Банк России);
* нефть Brent;
* золото (Банк России);
* серебро (Банк России);
* палладий (Банк России).

В качестве таргетов (целевой переменной) были взяты следующие курсы валют [7]:

* доллар США / Российский рубль;
* Евро / Российский рубль;
* бивалютная корзина / Российский рубль;
* Британский фунт стерлингов / Российский рубль;
* Швейцарский франк / Российский рубль;
* Китайский юань / Российский рубль;
* Японская иена / Российский рубль;
  1. Создание файла Jupiter Notebook

После того как мы установили JupyterLab, выбрали рабочее окружение и открыли среду разработки (пункт 2.1), создадим файл Jupiter Notebook. Для этого нужно перейти меню File → New → Notebook.

Файл Jupiter Notebook состоит из ячеек, в которых может находиться код программы или пояснительный текст программы, написанный на языке Markdown. Код каждой ячейки может выполняться отдельно, а также есть возможность запустить выполнение всех ячеек. Результат выполнения ячейки может выводиться непосредственно после самой ячейки.

* 1. Установка необходимых python-пакетов

Для того чтобы использовать необходимые нам библиотеки (Pandas, NumPy, Matplotlib и т.д.) нужно установить соответствующие пакеты с помощью системы управления пакетами pip. Для этого в ячейки ноутбука запустим следующий код:

ячейка Jupiter Notebook

# Для каждого проекта я использую новое окружение,

# думаю это помогает избежать ошибок связанных с версиями пакетов

!pip install pandas

!pip install matplotlib

!pip install seaborn

!pip install numpy

!pip install scikit-learn

!pip install openpyxl

* 1. Импортируем нужные модули

ячейка Jupiter Notebook

# Импортируем необходимые библиотеки

import numpy as np

import pandas as pd

from matplotlib import pyplot as plt

import seaborn as sns

# Следующая магическая команда Jupyter Notebook нужна для того, чтобы графики

# отображались прямо в ноутбуке, а не в отдельном окне

%matplotlib inline

# Настройка более четкого отображения графиков

%config InlineBackend.figure\_format = 'svg'

# Модуль для зазбиения выборки на тренировочнию и тестовую

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

# Уберем warnings

import warnings

warnings.filterwarnings('ignore')

# Настройка формата вывода чисел float

pd.set\_option('display.float\_format', '{:.2f}'.format)

# Вычисление Z-score

from scipy import stats

from sklearn.preprocessing import LabelEncoder

from scipy.stats.mstats import winsorize

# Дата и время

from datetime import datetime

# Случайные числа

import random as rnd

# Алгоритм машинного обучения 'Метод случайного леса'

from sklearn.ensemble import RandomForestRegressor

# Кросс-валидация

from sklearn.model\_selection import cross\_validate

# Разбиение

from sklearn.model\_selection import KFold

# Метрика r2

from sklearn.metrics import r2\_score

# Пути файловой системы

from pathlib import Path

# Регулярные выражения

import re

# Округление числа в большую сторону

from math import ceil

* 1. Загрузка датасетов

Загрузка файлов в дата фрейм происходит в автоматическом режиме. Программа читает все файлы с расширением xlsx в директории заданной константой PATH\_DATASET [5]. Все файлы соединяются по полю Дата. Если число строк в файле меньше, чем MIN\_ROW, то такие данные не будут включены в итоговый дата фрейм. В дата фрейме название столбца будет соответствовать названию файла. Информация о ходе процесса выводится под ячейкой.

ячейка Jupiter Notebook

# Расположение данных

PATH\_DATASET = './Dataset'

# Минимальное разрешённое количество строк для загрузки

MIN\_ROW = 5000

# Итоговый датафрейм со всеми данными

data\_loaded = pd.DataFrame()

# Расположение файлов датасета

p = Path(Path.cwd() / PATH\_DATASET)

for obj in p.iterdir():

    if obj.is\_file():

        \*\_, name\_file\_all = str(obj).split('\\')

        name\_file, type\_file = name\_file\_all.split('.')

        if type\_file == 'xlsx':

            df\_temp = pd.read\_excel(f'{PATH\_DATASET}/{name\_file\_all}')

            # Переименуем стобцы

            df\_temp.rename(columns={'Значение': f'{name\_file}'}, inplace=True)

            # Нормализуем дату

            df\_temp['Дата'] = df\_temp['Дата'].dt.normalize()

            # Проверяем условие первой загрузки

            if data\_loaded.shape[1]:

                if df\_temp.shape[0] >= MIN\_ROW:

                    print(f'Обработан: {name\_file\_all} {df\_temp.shape}')

                    data\_loaded = pd.merge(data\_loaded, df\_temp, on='Дата', how='inner')

                    print(f'Размерность после merge {data\_loaded.shape}\n')

                else:

                    print(f'Отклонён, мало строк: {name\_file\_all}\n')

            else:

                data\_loaded = df\_temp

                print(f'Обработан: {name\_file\_all} {df\_temp.shape}\n')

результат выполнения

Обработан: chf\_rub-(банк-россии).xlsx (7303, 2)

Обработан: cny\_rub-(банк-россии).xlsx (5419, 2)

Размерность после merge (5419, 3)

Обработан: eur\_rub-(банк-россии).xlsx (7303, 2)

Размерность после merge (5419, 4)

Обработан: gbp\_rub-(банк-россии).xlsx (7303, 2)

Размерность после merge (5419, 5)

Обработан: jpy\_rub-(банк-россии).xlsx (7303, 2)

Размерность после merge (5419, 6)

Обработан: s-p-500.xlsx (6178, 2)

Размерность после merge (4303, 7)

Обработан: usd\_rub-(банк-россии).xlsx (7303, 2)

Размерность после merge (4303, 8)

Отклонён, мало строк: бивалютная-корзина\_rub.xlsx

Обработан: золото-(банк-россии).xlsx (6034, 2)

Размерность после merge (4285, 9)

Обработан: индекс-мосбиржи.xlsx (6130, 2)

Размерность после merge (4185, 10)

Обработан: нефть-brent.xlsx (6514, 2)

Размерность после merge (4185, 11)

Обработан: палладий-(банк-россии).xlsx (6838, 2)

Размерность после merge (4185, 12)

Обработан: платина-(банк-россии).xlsx (6912, 2)

Размерность после merge (4185, 13)

Обработан: ртс.xlsx (6128, 2)

Размерность после merge (4178, 14)

Обработан: серебро-(банк-россии).xlsx (6034, 2)

Размерность после merge (4178, 15)

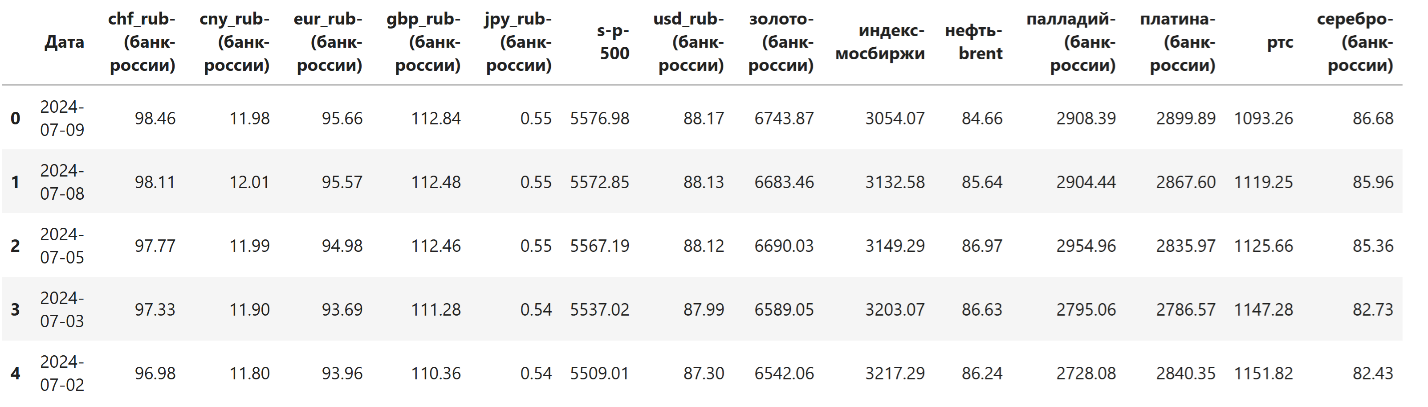
Выведем первые пять строк дата фрейма.

ячейка Jupiter Notebook

# Посмотрим, что загрузилось

data\_loaded.head()

результат выполнения



Проверим размер загруженных данных.

ячейка Jupiter Notebook

# Посмотрим размер

data\_loaded.shape

результат выполнения

(4178, 15)

В итоговый дата фрейм data\_loaded не был включен датасет бивалютная-корзина\_rub, по сколько он не удовлетворяет критерию минимального количества строк MIN\_ROW.

* 1. Разведочный анализ данных (EDA)

Посмотрим типы данных.

ячейка Jupiter Notebook

data\_loaded.dtypes

результат выполнения

Дата datetime64[ns]

chf\_rub-(банк-россии) float64

cny\_rub-(банк-россии) float64

eur\_rub-(банк-россии) float64

gbp\_rub-(банк-россии) float64

jpy\_rub-(банк-россии) float64

s-p-500 float64

usd\_rub-(банк-россии) float64

золото-(банк-россии) float64

индекс-мосбиржи float64

нефть-brent float64

палладий-(банк-россии) float64

платина-(банк-россии) float64

ртс float64

серебро-(банк-россии) float64

dtype: object

Посмотрим пропуски данных.

ячейка Jupiter Notebook

# Проверка того, в каких столбцах отсутствуют значения

print(data\_loaded.isnull().sum(axis=0))

результат выполнения

Дата 0

chf\_rub-(банк-россии) 0

cny\_rub-(банк-россии) 0

eur\_rub-(банк-россии) 0

gbp\_rub-(банк-россии) 0

jpy\_rub-(банк-россии) 0

s-p-500 0

usd\_rub-(банк-россии) 0

золото-(банк-россии) 0

индекс-мосбиржи 0

нефть-brent 0

палладий-(банк-россии) 0

платина-(банк-россии) 0

ртс 0

серебро-(банк-россии) 0

dtype: int64

Посмотри статистику по дата фрейму.

ячейка Jupiter Notebook

data\_loaded.describe()

результат выполнения

Изображение выглядит как текст, снимок экрана, Шрифт, документ

Автоматически созданное описание

* 1. Визуализация данных
     1. Выделим численные признаки от даты

ячейка Jupiter Notebook

# Создадим список численных признаков

num\_cols = [column for column in data\_loaded if not column=='Дата']

num\_cols

результат выполнения

['chf\_rub-(банк-россии)',

'cny\_rub-(банк-россии)',

'eur\_rub-(банк-россии)',

'gbp\_rub-(банк-россии)',

'jpy\_rub-(банк-россии)',

's-p-500',

'usd\_rub-(банк-россии)',

'золото-(банк-россии)',

'индекс-мосбиржи',

'нефть-brent',

'палладий-(банк-россии)',

'платина-(банк-россии)',

'ртс',

'серебро-(банк-россии)']

* + 1. Создадим словарь признаков и пояснения

ячейка Jupiter Notebook

# Словарь признаков и их пояснения

explanations\_col = {

    'chf\_rub-(банк-россии)': 'Швейцарский франк / Российский рубль',

    'eur\_rub-(банк-россии)': 'Евро / Российский рубль',

    'gbp\_rub-(банк-россии)': 'Британский фунт стерлингов / Российский рубль',

    'jpy\_rub-(банк-россии)': 'Японская иена / Российский рубль',

    's-p-500': 'Индекс S&P 500',

    'usd\_rub-(банк-россии)': 'Доллар США / Российский рубль',

    'cny\_rub-(банк-россии)': 'Китайский юань / Российский рубль'

}

* + 1. Визуализируем распределение числовых признаков

ячейка Jupiter Notebook

# Создаём полотно

plt.figure(figsize=[11, 13])

# Общий заголовок для всех графиков

plt.suptitle('Распределение числовых признаков ',

                y=1.005,

                fontsize=19,

                fontweight='bold')

# Рассчитаем количество строк в графике

n\_row = ceil(len(num\_cols)/3)

for i, col in enumerate(num\_cols):

    plt.subplot(n\_row, 3, i+1)

    # Заголовок для графика

    # Вставляем русский перевод если он есть, или оригинальное название

    explanations = col + '\n' + explanations\_col[col] if explanations\_col.get(col) else '\n' + col

    plt.title(f'\n{explanations}', fontsize=10)

    # Задаём размер шривта и угол поворота текста для осей X и Y

    plt.xticks(fontsize=8, rotation=0)

    plt.yticks(fontsize=8, rotation=0)

    # Делаем размер шрифта по Y=5, не убирая название оси

    plt.ylabel('', fontsize=5)

    # Отрисовываем гистограмму

    plt.hist(data\_loaded[col])

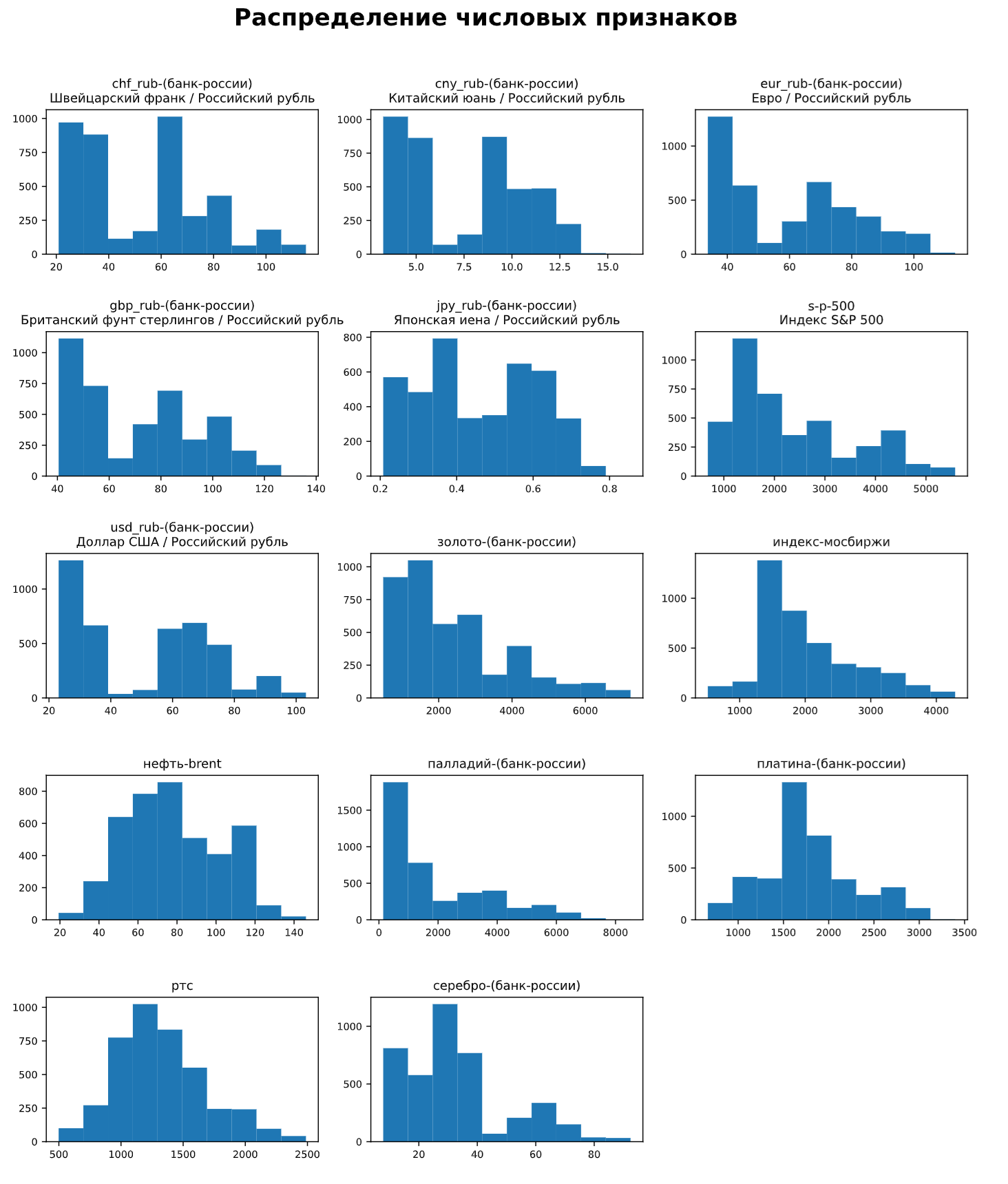
# Автоматически уместить все элементы на полотне

plt.tight\_layout()

# Вывести графики на экран

plt.show()

результат выполнения



* 1. Посмотрим какие признаки имеют выбросы

ячейка Jupiter Notebook

def emission\_test(ds, col\_test, threshold\_val=3, explanations\_col=explanations\_col):

    '''

    Функция проводит тестирование столбцов на выбросы методом Z-score.

    :param ds: исследуемый датасет,

    :param col\_test: список колонок,

    :param threshold\_val: пороговое значение Z-score,

    :param explanations\_col: словарь признаков и их пояснения,

    :return: текстовый отчет и список выбросов.

    '''

    result = ''

    NUM = 2  # количество цифр после запятой

    outliers\_list = [] # лист выбросов

    for col in col\_test:

        # Вычисление Z-score

        z = np.abs(stats.zscore(ds[col]))

        # Установка порогового значения Z-score

        threshold = 3

        # Выявление выбросов на основе Z-score

        outliers = ds[col][z > threshold]

        if len(outliers) > 0:

            outliers\_list.append(outliers)

            # Вставляем русский перевод если он есть, или оригинальное название

            explanations = f' ({explanations\_col[col]})' if explanations\_col.get(col) else ''

            result += f'В столбце {col}{explanations},\n{len(outliers)} выбросов. Mean: {round(ds[col].mean(), NUM)}, ' \

            f'Min: {round(ds[col].min(), NUM)}, Max: {round(ds[col].max(), NUM)}, ' \

            f'Moda: {round(ds[col].mode()[0], NUM)}, Median: {round(ds[col].median(), NUM)} \n\n'

    return f'Выбросов нет. ' if result == '' else result, outliers\_list

ячейка Jupiter Notebook

def viev\_outliers(outliers\_list, col\_name):

    '''

    Функция выводит строки датафрейма, в которых есть выбросы.

    :param outliers\_list: список выбросов,

    :param col\_name: название признака (фичи),

    :return: индексы выбросов.

    '''

    for item in outliers:

        if item.name == col\_name:

            return item.index

* 1. Создание новых признаков

Создадим дополнительные признаки год, квартал, месяц, неделя, день, которые в дальнейшем помогут нам при выполнении запросов.

ячейка Jupiter Notebook

def choice\_quarter(month):

    '''

    Функция определения квартала.

    :param month: номер месяца 1-12,

    :return: номер квартала 1-4, или -1, если ошибка.

    '''

    if month <= 3:

        return 1

    elif 4 <= month <= 6:

        return 2

    elif 7 <= month <= 9:

        return 3

    elif 10 <= month <= 12:

        return 4

    else:

        return -1

# Создадим дополнительные признаки (год, месяц, день, день недели (Пн-Вс), номер недели, квартал)

data\_loaded['Year'] = data\_loaded['Дата'].dt.year

data\_loaded['Month'] = data\_loaded['Дата'].dt.month

data\_loaded['Weekday'] = data\_loaded['Дата'].dt.weekday

data\_loaded['Day'] = data\_loaded['Дата'].dt.day

data\_loaded['Week'] = data\_loaded['Дата'].dt.isocalendar().week

data\_loaded['Quarter'] = data\_loaded['Month'].apply(lambda x: choice\_quarter(x))

data\_loaded.head()

* 1. Проверка состояния данных

Проверим за какие временные периоды собраны данные и если понадобиться удалим выбранные периоды.

ячейка Jupiter Notebook

def not\_correct\_year(df=data\_loaded, min\_month=12, min\_day=200):

    '''

    Функция выдает список не корректных годов, где min\_month и min\_day меньше заданного.

    :param  df: исследуемый датафрейм,

    :param  min\_month: минимальное количество месяцев в году,

    :param  min\_day: минимальное количество дней в году,

    :return: возвращает список не корректных годов и текстовый отчёт.

    '''

    not\_corr\_year = []

    txt\_report = ''

    for year in df['Year'].unique():

        month\_quantity = len(df[df['Year']==year]['Month'].unique())

        day\_quantity = len(df[df['Year']==year]['Weekday'])

        if (month\_quantity < min\_month) or (day\_quantity < min\_day):

            not\_corr\_year.append(int(year))

        txt\_report += f'Год: {year}, кол. месяцев: {month\_quantity}, кол. дней: {day\_quantity}\n'

    return not\_corr\_year, txt\_report

# Протестируем датасет и запишем список не корректных годов и отчёт

not\_correct\_year\_list, report = not\_correct\_year()

print(report)

результат выполнения

Год: 2024, кол. месяцев: 7, кол. дней: 124

Год: 2023, кол. месяцев: 12, кол. дней: 237

Год: 2022, кол. месяцев: 12, кол. дней: 214

Год: 2021, кол. месяцев: 12, кол. дней: 233

Год: 2020, кол. месяцев: 12, кол. дней: 225

Год: 2019, кол. месяцев: 12, кол. дней: 232

Год: 2018, кол. месяцев: 12, кол. дней: 227

Год: 2017, кол. месяцев: 12, кол. дней: 232

Год: 2016, кол. месяцев: 12, кол. дней: 232

Год: 2015, кол. месяцев: 12, кол. дней: 231

Год: 2014, кол. месяцев: 12, кол. дней: 233

Год: 2013, кол. месяцев: 12, кол. дней: 232

Год: 2012, кол. месяцев: 12, кол. дней: 230

Год: 2011, кол. месяцев: 12, кол. дней: 232

Год: 2010, кол. месяцев: 12, кол. дней: 224

Год: 2009, кол. месяцев: 12, кол. дней: 230

Год: 2008, кол. месяцев: 12, кол. дней: 226

Год: 2007, кол. месяцев: 12, кол. дней: 227

Год: 2006, кол. месяцев: 12, кол. дней: 232

Год: 2005, кол. месяцев: 12, кол. дней: 231

Год: 2004, кол. месяцев: 12, кол. дней: 235

Год: 2003, кол. месяцев: 6, кол. дней: 123

Видно, что за 2003 и 2024 собрана информация не полностью за год, но удалять эти года не будем просто будем иметь в виду этот момент при кросс-валидации.

* 1. Метрика, по которой будем оценивать качество работы модели

Метрика будет вычислять какова величина ошибки в процентах между тестовыми средними значениями за определённой период и предсказанными средними значениями (формула 1) [2].

Где – среднее значение курса валют за определённый отрезок времени (неделя, месяц, квартал, год);

– предсказанное среднее значение курса валют за определённый отрезок времени (неделя, месяц, квартал, год);

– величина ошибки предсказанных средних значений в процентах.

ячейка Jupiter Notebook

# Метрика

def mape(y\_test, y\_pred):

    '''

    Функция вычисляет ошибку в процентах от средних значений предсказанных и тестовых.

    :param  y\_test: тестовые значения таргета,

    :param  y\_pred: предсказанные значения таргета,

    :return: возвращает ошибку предсказания в процентах.

    '''

    return abs((y\_test.mean() - y\_pred.mean()) / y\_test.mean()) \* 100

* 1. Разбиение временного ряда на train и test

Функция-генератор split\_on\_time\_series разбивает временной ряд на train и test. Класс TimeSeriesSplit из библиотеки sklearn не подошел так ка он разбивает временной ряд по числу элементов [10]. Поэтому создадим собственную функцию, которая будет разбивать временной ряд привязываясь к датам [13; 9; 11]. Функция также исправляет мелкие орфографические ошибки в указании типа разбиений (Year, Quarter, Month, Week) и выводит ошибку если количество разбиений задано не корректно.

ячейка Jupiter Notebook

def split\_on\_time\_series(data, start\_year, number\_splits, type\_split, target\_name):

    '''

    Функция-генератор, разбивает временной ряд на train и test.

    :param  data: датафрем с данными,

    :param  start\_year: год с которого будем разбивать,

    :param  number\_splits: количество разбиений,

    :param  type\_split: тип разбиений (Year, Quarter, Month, Week),

    :param  target\_name: имя таргета (целевой переменной),

    :return: возвращает X\_train, y\_train, X\_test, y\_test, summary, time\_point\_test.

    '''

    # Удаляем все не латинские знаки, делаем первую букву заглавную, остальные обычные

    resample\_param = re.sub(r'[^a-zA-z]', '', type\_split).capitalize()

    resample\_code = ''

    # Создадим список валют

    currency\_list = [column for column in data if '\_' in column]

    # Тригер первого включения

    triger = False

    # Выставляем параметр для сводной таблицы

    if resample\_param == 'Year':

        resample\_code = 'YE'

    elif resample\_param == 'Quarter':

        resample\_code = 'BQE'

    elif resample\_param == 'Month':

        resample\_code = 'ME'

    elif resample\_param == 'Week':

        resample\_code = 'W'

    else:

        raise ValueError(f'Ошибка параметра "type\_split={type\_split}". Корректно: Year, Quarter, Month, Week')

    # Делаем датафрем независимым

    df = data.copy(deep=True)

    # Сделаем резервную копию даты

    df['Дата-copy'] = df['Дата']

    # Установим столбец даты в качестве индекса

    df = df.set\_index('Дата-copy')

    # Выберем данные после стартового года

    df = df.query(f'(Year >= {start\_year})')

    # Создадим X\_train и y\_train

    # Делаем датафрем независимым

    X\_train = data.copy(deep=True)

    X\_train = X\_train.query(f'(Year < {start\_year})')

    # Установим столбец даты в качестве индекса

    X\_train = X\_train.set\_index('Дата')

    # Создадим таргет

    y\_train = X\_train[target\_name]

    # Удалим все ненужные столбцы из X\_train

    X\_train.drop(columns=currency\_list, axis=1, inplace=True)

    # Проверка

    n\_row = df.resample(resample\_code).count().shape[0]

    if n\_row < number\_splits:

        raise ValueError(f'Столько данных с параметрами "start\_year={start\_year}", не найдём ((')

    # Если всё ок разбиваем на train и test

    for row in range(number\_splits):

        # Добавляем данные из предыдущего цикла, если он не первый

        if triger:

            X\_train = pd.concat([X\_train, X\_test])

            y\_train = pd.concat([y\_train, y\_test])

        else:

            triger = True

        # Читаем строчку сводной таблицы

        resample\_param\_list = ['Year']

        if not resample\_param in resample\_param\_list:

            resample\_param\_list.append(resample\_param)

        pivot\_table\_row = pd.DataFrame(df.resample(resample\_code).mean().iloc[row]).T[resample\_param\_list]

        # Соединяем таблицу и строку сводной таблицы

        result = df.merge(pivot\_table\_row, on=resample\_param\_list, how='right')

        result = result.set\_index('Дата')

        # Запишем таргет в отдельную переменную

        y\_test = result[target\_name]

        # Удалим все таргеты из датасета

        result.drop(columns=currency\_list, axis=1, inplace=True)

        X\_test = result

        # Инфа по данным

        summary = f'Train({X\_train.index.min().strftime('%d.%m.%Y')}-{X\_train.index.max().strftime('%d.%m.%Y')}, X.shape{X\_train.shape}, y.shape{y\_train.shape}), ' \

                  f'Test({X\_test.index.min().strftime('%d.%m.%Y')}-{X\_test.index.max().strftime('%d.%m.%Y')}, X.shape{X\_test.shape}, y.shape{y\_test.shape})'

        # Запишем переменные времени, для графика

        time\_point\_test = X\_test.index.min()

        yield X\_train, y\_train, X\_test, y\_test, summary, time\_point\_test

1. Вывод

Выполнив данное ДЗ, я научился писать скрипты для Linux, а также прокачал свои навыки в Word при оформлении данного ДЗ.

1. Литература

1. Факторы, влияющие на формирование курса рубля Е. Я. Волков. – URL: https://view.officeapps.live.com/op/view.aspx?src=https%3A%2F%2Fwww.kubsu.ru%2Fsites%2Fdefault%2Ffiles%2Fusers%2F21365%2Fportfolio%2Fmvko\_i\_fo\_kursovaya.docx&wdOrigin=BROWSELINK (дата обращения: 11.07.2024) – Текст: электронный.

2. Кросс валидация для временных рядов в python Прикладная статистика. – URL: https://www.youtube.com/watch?v=l2vhtyWp\_ck (дата обращения: 21.07.2024) – Текст: электронный.

3. Особенности методов машинного обучения для прогнозирования курса валют Е. В. Сангаджиева, Г. А. Оргдаева. – URL: https://files.scienceforum.ru/pdf/2020/5e04936d95941.pdf (дата обращения: 11.07.2024) – Текст: электронный.

4. Основы работы в Jupyter/Jupiter Notebook и JupyterLab — Python Tutorial И. Шамаев. – URL: https://python.ivan-shamaev.ru/jupiterlab-jupyter-notebook-install-python-tutorial/ (дата обращения: 19.07.2024) – Текст: электронный.

5. ДЗ «Погружение в Python». Семинар 7. Файлы и файловая система Е. В. Эгипти. – URL: https://github.com/uc20100/python\_2/blob/main/seminar\_7/file\_func/task\_home\_work.py (дата обращения: 21.07.2024) – Текст: электронный.

6. Ключевые индикаторы InvestFunds. – URL: https://investfunds.ru/indicators/key-indexes/ (дата обращения: 20.07.2024) – Текст: электронный.

7. Курсы валют InvestFunds. – URL: https://investfunds.ru/indicators/currency/ (дата обращения: 20.07.2024) – Текст: электронный.

8. Debugger JupiterLab. – URL: https://jupyterlab.readthedocs.io/en/4.1.x/user/debugger.html (дата обращения: 19.07.2024) – Текст: электронный.

9. pandas.DataFrame.resample Pandas. – URL: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.resample.html (дата обращения: 21.07.2024) – Текст: электронный.

10. TimeSeriesSplit sklearn. – URL: https://scikit-learn.org/stable/modules/generated/sklearn.model\_selection.TimeSeriesSplit.html (дата обращения: 21.07.2024) – Текст: электронный.

11. Как легко обрабатывать данные временных рядов. – URL: https://dfedorov.spb.ru/pandas/09.%20%D0%9A%D0%B0%D0%BA%20%D0%BB%D0%B5%D0%B3%D0%BA%D0%BE%20%D0%BE%D0%B1%D1%80%D0%B0%D0%B1%D0%B0%D1%82%D1%8B%D0%B2%D0%B0%D1%82%D1%8C%20%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D0%B5%20%D0%B2%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%BD%D1%8B%D1%85%20%D1%80%D1%8F%D0%B4%D0%BE%D0%B2\_.html (дата обращения: 21.07.2024) – Текст: электронный.

12. Anaconda. – URL: https://www.anaconda.com/ (дата обращения: 19.07.2024) – Текст: электронный.

13. Time series / date functionality Pandas. – URL: https://pandas.pydata.org/docs/user\_guide/timeseries.html (дата обращения: 21.07.2024) – Текст: электронный.