GeekBrains

Анализ предсказаний курсов валют с помощью модели машинного обучения на разный промежуток времени в зависимости от ключевых индикаторов на фондовом рынке

Программа: Цифровые профессии

Специализация: Искусственный интеллект

Студент: Дрыгин Алексей Васильевич

Рубцовск 2024

Оглавление

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) [4].

Темой проекта является анализ предсказаний курсов валют с помощью модели машинного обучения на разный промежуток времени в зависимости от ключевых индикаторов на фондовом рынке.

Целью будет являться хорошо интерпретируемые данные, показывающие с какой погрешностью модель машинного обучения может предсказывать курс валют по ключевым индикаторам на фондовом рынке на разный промежуток времени, а также показать важность ключевых индикаторов для модели машинного обучения.

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 и тестирования модели;
- Определить признаки (фичи) и ключевые параметры (таргеты);
- Выбрать модель машинного обучения;
- Определить уровень ошибки предсказаний модели на временном ряду, для разных отрезков времени;
- Вывести графики качества предсказаний и важности признаков;
- Автоматизировать процесс вычислений и отрисовки графиков при добавлении новых фичей и таргетов.

Инструменты:

JupyterLab [5; 10], Visual Code [16], Microsoft 365 Word [17], Zotero [23; 2; 22; 7].

Состав команды: Дрыгин Алексей Васильевич

Глава 1 Теоретическая часть

1.1 Выбор инструментов

В качестве инструментов для написания кода будем использовать бесплатные программы. JupiterLab [5] который входит в состав программы Anaconda [10]. JupiterLab это практически то же самое, что и Jupiter Notebook, только с расширенными возможностями. JupiterLab как и Jupiter Notebook позволяют выполнять каждую ячейку кода отдельно, при этом результат выполнения и все переменные сохраняются в ноутбуке. Это позволяет запускать ресурсоёмкие части кода один раз (например обучение модели), а дальше экспериментировать с результатами, без надобности запускать весь код с начала. Это очень удобно, по сколько, например процессы кросс-валидации могут занимать более часа.

Для оформления текстового описания работы будем использовать программу Microsoft 365 Word [17], которая позволяет применять широкие возможности для редактирования документов.

В бесплатной программе Visual Studio Code [16], можно просматривать ноутбук и копировать ячейки кода с сохранением полного форматирования, для последующего переноса этого текста в документ Word.

Бесплатная программа Zotero [23; 2; 22], поможет в оформлении библиографических ссылок.

1.2 Поиск и загрузка датасетов

Датасеты должны содержать информацию о ключевых индексах фондового рынка (фичах), например цены на нефть, драг. металлы, индексы бирж и т.д. и обменных кусах различных валют (таргеты). Эти данные должны быть привязаны ко времени торгов. Желательно, чтобы датасеты были бесплатными и читаемыми в Excel.

1.3 Выбор модели машинного обучения

Выберем модель машинного обучения на основе алгоритма случайного леса RandomForestRegressor из библиотеки Scikit-Learn [20]. Это регрессионная модель, которая может предсказывать числа и есть возможность визуализировать важность признаков для этой модели.

1.4 Оценка качества предсказаний модели

Качество модели будем оценивать по следующей метрике:

$$mape = \left|\frac{\bar{y}_{test} - \bar{y}_{pred}}{\bar{y}_{test}}\right| \cdot 100 \quad (1)$$

Где $\bar{y}_{test}$ – среднее значение курса валют за определённый отрезок времени (неделя, месяц, квартал, год);

$\bar{y}_{pred}$ – предсказанное среднее значение курса валют за определённый отрезок времени (неделя, месяц, квартал, год);

$mape$ – величина ошибки предсказанных средних значений в процентах.

Будем проверять эффективность работы модели на тестовых данных, то есть на тех данных которая модель не видела. Так как последовательность данных для нас важна, будем делать кросс-валидацию на временном ряду [9]. Ниже, рисунок 1, показывает принцип кросс-валидации на временном ряду.

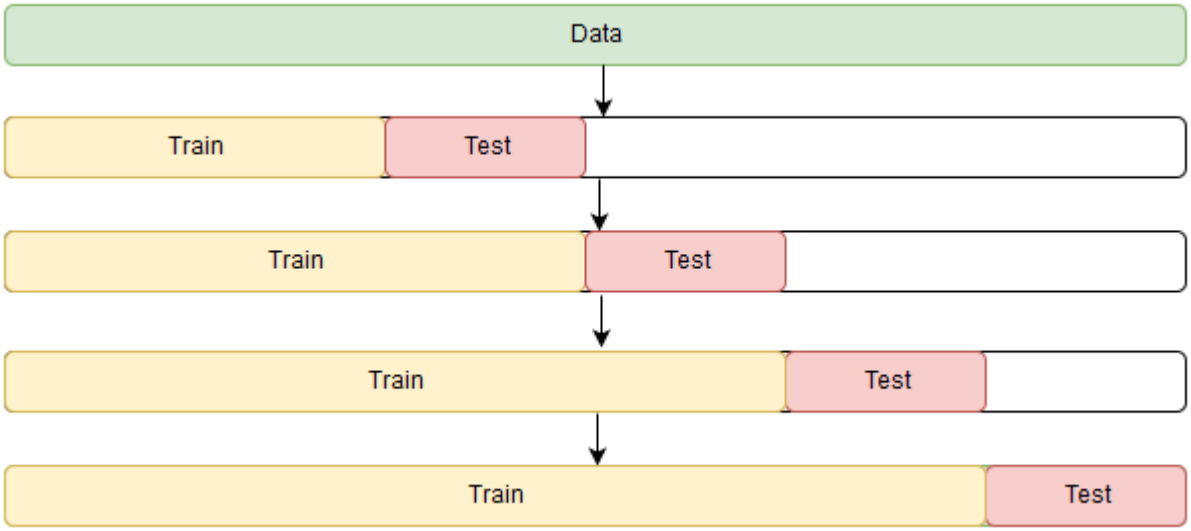

Рисунок 1. Принцип кросс-валидации на временном ряду

То есть берётся набор данных Data и разбивается на Train и Test, на Train модель тренируется, а на Test проверяет свою эффективность предсказания. Внутри Scikit-Learn есть готовая функция TimeSeriesSplit [21] которая разбивает выборку на train и test, но нам она не подойдёт, поскольку она будет разбивать выборку с фиксированным количеством элементов, а нам нужно ориентироваться на даты, а не на количество элементов. Разбивка будет осуществляться по годам, кварталам, месяцам, неделям. Для этого потребуется написать собственную функцию кросс-валидации.

1.5 Концепция ноутбука

Чтобы избавиться от ручного монотонного труда при загрузке, обработке датасетов, отрисовке графиков, нужно продумать программу так, чтобы максимально автоматизировать весь процесс.

Например, датасеты должны автоматически загружаться с какой ни будь директории, в соответствии с правилами загрузки.

Распределение на фичи и таргеты также должно происходить в автоматическом режиме. Например файлы, содержащие символ ‘_’ автоматически будут инициализированы как таргет, все остальные как фичи.

На всех графиках должны в автоматическом режиме добавляться/удалятся фичи и таргеты. Если есть русская расшифровка таргета/фич, она также должна в автоматическом режиме вставляться в график. Возможна некая корректировка размера полотна графика в коде программы для улучшения вида графиков.

Процесс добавления/удаления фич и таргетов должен сводиться к добавлению/удалению датасета в директорию откуда производиться чтение, без вмешательства в код программы.

Кросс-валидация временного ряда должна выполняться с привязкой к датам с определённой длиной периода (год, квартал, месяц, неделя).

Глава 2 Практическая часть

2.1 Среда разработки

В качестве среды разработки был использован JupyterLab [5], там больше возможностей по сравнению с Jupyter Notebook. Можно одновременно работать с несколькими файлами, использовать отладчик [15], если потребуется.

Данная среда разработки, а также другие инструменты входят в бесплатную программу Anaconda [10] 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 2.

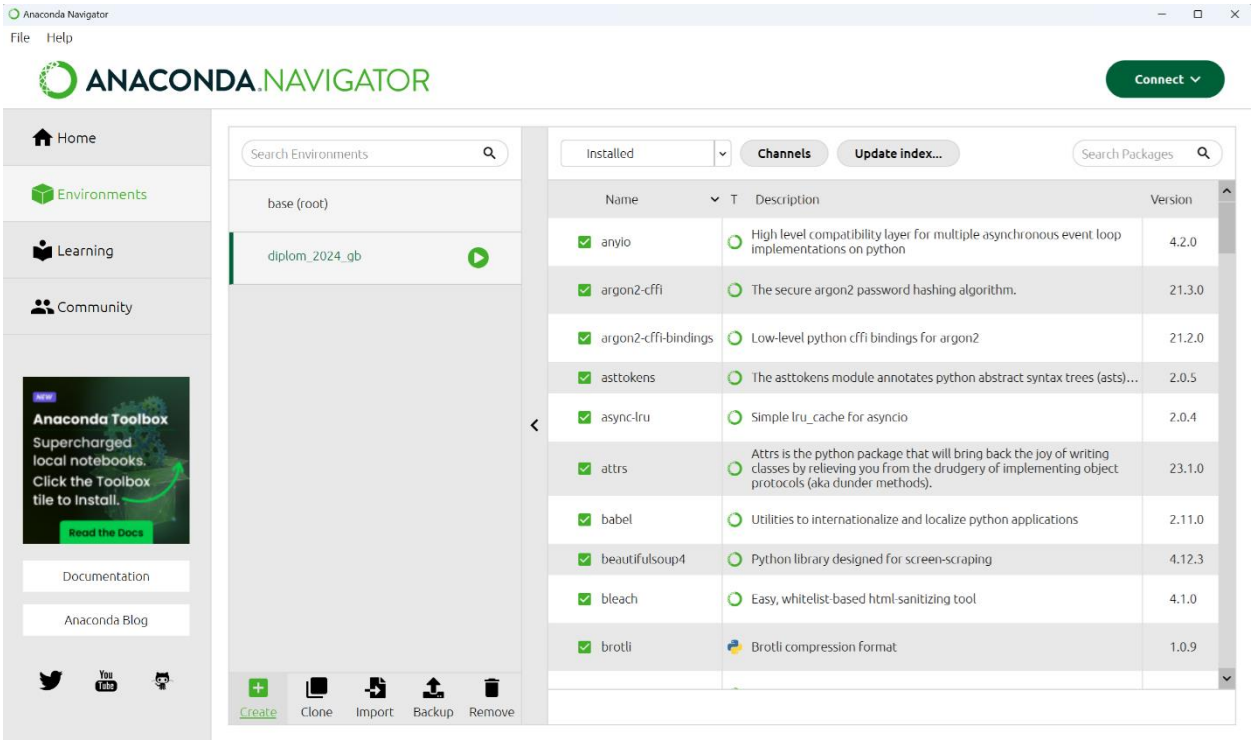

Рисунок 2. Выбор активного окружения diplom_2024_gb

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

2.2 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 [13] 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 [14]:

- 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;

2.3 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.

2.4 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 numpy
!pip install scikit-learn
!pip install openpyxl
```

2.5 Импортируем нужные модули

После установки пакетов нужно импортировать нужные нам для работы модули.

ячейка Jupiter Notebook

```
# Импортируем необходимые библиотеки
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt

# 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'

# 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

# Алгоритм машинного обучения 'Метод случайного леса'
from sklearn.ensemble import RandomForestRegressor

# Пути файловой системы
from pathlib import Path

# Регулярные выражения
import re

# Округление числа в большую сторону
from math import ceil
```

2.6 Загрузка датасетов

Загрузка файлов в дата фрейм происходит в автоматическом режиме. Программа читает все файлы с расширением xlsx в директории заданной константой PATH_DATASET [6]. 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 Значение будет заменено на название 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

	Дата	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)
0	2024-07-09	98.46	11.98	95.66	112.84	0.55	5576.98	88.17	6743.87	3054.07	84.66	2908.39	2899.89	1093.26	86.68
1	2024-07-08	98.11	12.01	95.57	112.48	0.55	5572.85	88.13	6683.46	3132.58	85.64	2904.44	2867.60	1119.25	85.96
2	2024-07-05	97.77	11.99	94.98	112.46	0.55	5567.19	88.12	6690.03	3149.29	86.97	2954.96	2835.97	1125.66	85.36
3	2024-07-03	97.33	11.90	93.69	111.28	0.54	5537.02	87.99	6589.05	3203.07	86.63	2795.06	2786.57	1147.28	82.73
4	2024-07-02	96.98	11.80	93.96	110.36	0.54	5509.01	87.30	6542.06	3217.29	86.24	2728.08	2840.35	1151.82	82.43

Рисунок 3. Вывод первых пяти строк дата фрейма

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.

2.7 Разведочный анализ данных (EDA)

С помощью разведочного анализа (EDA) оценим качество загруженных данных.

2.7.1 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
```

2.7.2 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
```

2.7.3 Посмотри статистику по дата фрейму.

ячейка Jupiter Notebook

```
data_loaded.describe()
```

результат выполнения

	Дата	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)
count	4178	4178.00	4178.00	4178.00	4178.00	4178.00	4178.00	4178.00	4178.00	4178.00	4178.00	4178.00	4178.00	4178.00	4178.00
mean	2015-06-14 12:44:07.008137984	52.65	7.53	60.02	71.82	0.46	2351.96	50.66	2406.81	2017.93	77.91	1889.29	1775.77	1325.14	33.01
min	2006-05-11 00:00:00	20.90	3.28	33.72	40.48	0.21	682.55	23.13	486.82	513.62	19.33	144.32	664.78	498.20	7.81
25%	2010-12-02 06:00:00	31.14	4.59	40.38	49.77	0.34	1362.70	30.46	1296.15	1462.82	59.92	594.88	1504.21	1072.07	20.97
50%	2015-06-18 12:00:00	58.02	8.43	60.63	72.17	0.44	2047.40	56.50	2246.76	1765.40	75.29	1235.60	1714.11	1274.45	31.27
75%	2019-12-11 18:00:00	67.60	9.98	75.22	89.35	0.59	3003.95	66.34	3114.67	2444.39	98.47	3069.00	2051.12	1534.68	37.67
max	2024-07-09 00:00:00	115.22	16.19	113.26	136.04	0.85	5576.98	103.16	7231.12	4287.52	146.08	8507.40	3396.33	2487.92	92.42
std	NaN	24.32	3.10	20.66	23.10	0.15	1184.29	21.26	1536.95	752.57	23.96	1751.72	510.58	361.29	17.51

Рисунок 4. Статистика по загруженному дата фрейму

2.7.4 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)']
```

2.7.5 Создадим словарь признаков и пояснения

Для того, чтобы на графиках отображалось более понятное и презентабельное название. Сделаем словарь, где название столбца будет соответствовать его более подробному описанию. Название столбца соответствует названию файла.

Это действие не обязательно, тогда вместо подробного описания на графике будет выводиться название столбца, которое будет соответствовать названию файла.

ячейка Jupiter Notebook

```
# Словарь признаков/таргетов 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'
}
```

2.7.6 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

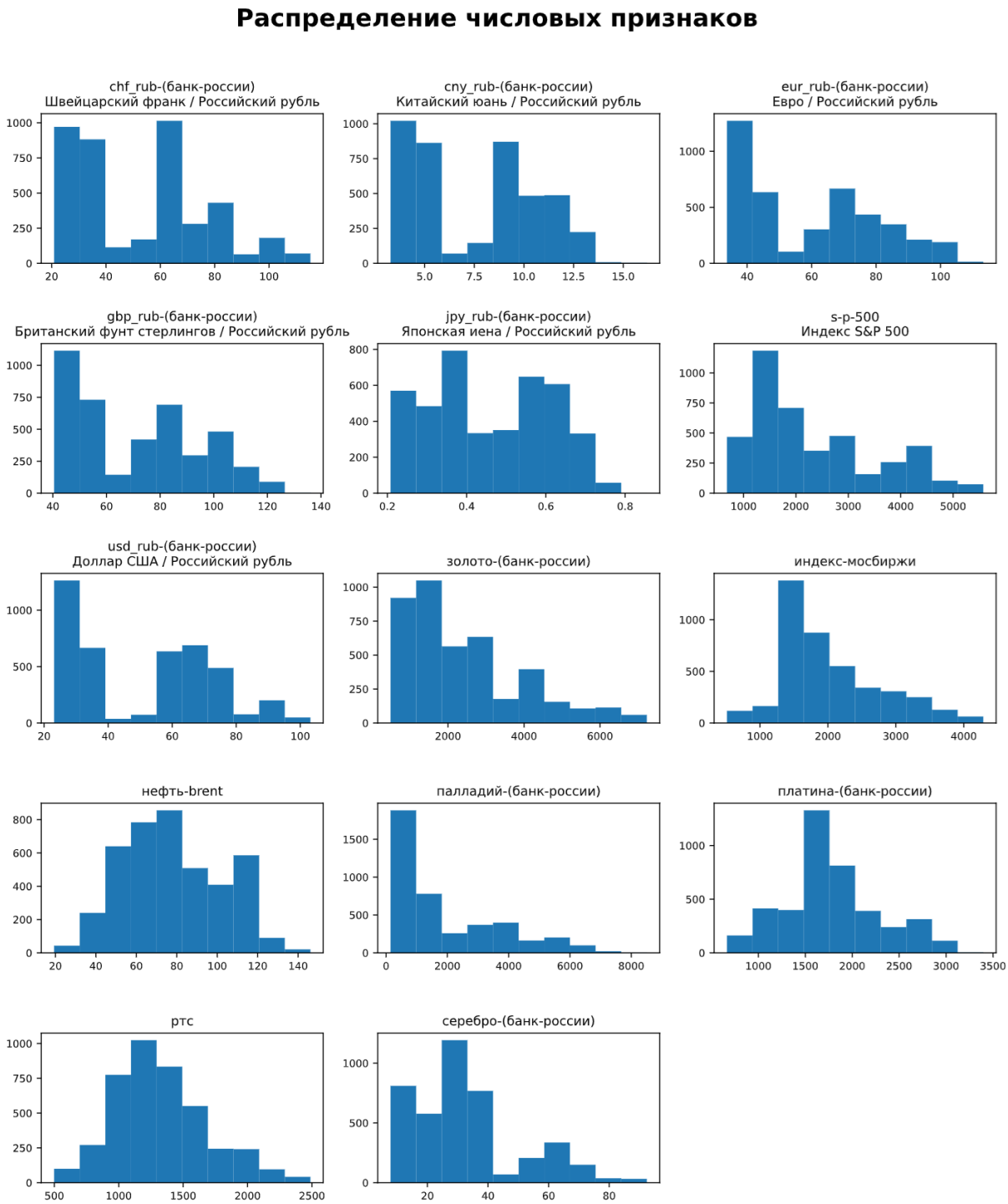

Рисунок 5. Распределение численных показателей дата фрейма

2.8 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
```

ячейка Jupiter Notebook

```
# Выведем признаки которые имеют выбросы
text_outliers, outliers = emission_test(ds=data_loaded, col_test=num_cols)

# Выведем текстовый отчёт по выбросам
print(text_outliers)
```

результат выполнения

```
В столбце золото-(банк-россии),
11 выбросов. Mean: 2406.81, Min: 486.82, Max: 7231.12, Moda: 5993.16, Median: 2246.76

В столбце индекс-мосбиржи,
3 выбросов. Mean: 2017.93, Min: 513.62, Max: 4287.52, Moda: 647.8, Median: 1765.4

В столбце палладий-(банк-россии),
6 выбросов. Mean: 1889.29, Min: 144.32, Max: 8507.4, Moda: 3275.71, Median: 1235.6
```

```
В столбце платина-(банк-россии),
1 выбросов. Mean: 1775.77, Min: 664.78, Max: 3396.33, Moda: 2883.54, Median: 1714.11

В столбце ртс,
11 выбросов. Mean: 1325.14, Min: 498.2, Max: 2487.92, Moda: 982.94, Median: 1274.45

В столбце серебро-(банк-россии),
21 выбросов. Mean: 33.01, Min: 7.81, Max: 92.42, Moda: 32.11, Median: 31.27
```

Каких то аномалий в выбросах не наблюдается, данные корректны. При применении модели RandomForestRegressor сглаживание выбросов с помощью логарифмической функции или другими методами, результат предсказаний не улучшает. Оставим дата фрейм без изменений.

2.9 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()
```

результат выполнения

	Дата	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	...	палладий-(банк-россии)	платина-(банк-россии)	ртс	серебро-(банк-россии)	Year	Month	Weekday	Day	Week	Quarter
0	2024-07-09	98.46	11.98	95.66	112.84	0.55	5576.98	88.17	6743.87	3054.07	...	2908.39	2899.89	1093.26	86.68	2024	7	1	9	28	3
1	2024-07-08	98.11	12.01	95.57	112.48	0.55	5572.85	88.13	6683.46	3132.58	...	2904.44	2867.60	1119.25	85.96	2024	7	0	8	28	3
2	2024-07-05	97.77	11.99	94.98	112.46	0.55	5567.19	88.12	6690.03	3149.29	...	2954.96	2835.97	1125.66	85.36	2024	7	4	5	27	3
3	2024-07-03	97.33	11.90	93.69	111.28	0.54	5537.02	87.99	6589.05	3203.07	...	2795.06	2786.57	1147.28	82.73	2024	7	2	3	27	3
4	2024-07-02	96.98	11.80	93.96	110.36	0.54	5509.01	87.30	6542.06	3217.29	...	2728.08	2840.35	1151.82	82.43	2024	7	1	2	27	3

5 rows × 21 columns

Рисунок 6. Первые пять строк дата фрейма с созданными новыми признаками

2.10 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: 8, кол. дней: 157
```

Видно, что за 2006 и 2024 собрана информация не полностью за год, но удалять эти года не будем просто будем иметь в виду этот момент при кросс-валидации.

2.11 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 ниже [3].

$$mape = \left|\frac{\bar{y}_{test} - \bar{y}_{pred}}{\bar{y}_{test}}\right| \cdot 100 \quad (2)$$

Где $\bar{y}_{test}$ – среднее значение курса валют за определённый отрезок времени (неделя, месяц, квартал, год);

$\bar{y}_{pred}$ – предсказанное среднее значение курса валют за определённый отрезок времени (неделя, месяц, квартал, год);

$mape$ – 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
```

2.12 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 [21]. Поэтому создадим собственную функцию, которая будет разбивать временной ряд привязываясь к датам [19; 18; 12]. 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]).sort_index(ascending=False)
            y_train = pd.concat([y_train, y_test]).sort_index(ascending=False)
        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
```

2.13 Функция кросс-валидации

Функция кросс-валидации обучает модель на train данных и выполняет предсказание на test данных, которая модель не видела. Разбивка на train и test происходит с учетом временного ряда и привязана к датам. Ошибка предсказания оценивается по формуле 2 (пункт 2.11). Данные, на которых будет тестироваться модель, могут быть разного объёма (год, квартал, месяц, неделя), это зависит от параметра type_split. В качестве целевой переменной (таргета) указывается список валют – параметр currency. Чтобы была информация о ходе процесса, данные выводятся с помощью команды print. Также данные о кросс-валидации, сохраняются в дата фрейм Pandas, параметр cv_info.

ячейка Jupiter Notebook

```
def cross_validation_time_series(model, df, year, n_split, type_split, currency, cv_info):
    '''
    Функция кросс-валидирует список валют на временном ряду.

    :param  model: ML модель которую будем использовать для кросс-валидации,
    :param  df: датафрейм с данными,
    :param  year: год с которого нужно начать кросс-валидацию,
    :param  n_split: число кросс-валидаций,
    :param  type_split: тип кросс валидаций (Year, Quarter, Month, Week),
```

```
    :param  currency: список полей валют,
    :param  cv_info: дата фрейм pandas куда будет записываться статистика о кросс-валидациях,
    :return: возвращает дата фрейм статистики кросс-валидаций.
    '''
    for currency_item in currency:
        for  X_train, y_train, X_test, y_test, summary, time_point_test in
split_on_time_series(df, year, n_split, type_split, currency_item):
            # Обучим модель
            model.fit(X_train, y_train)
            # Выполним предикт
            y_pred = model.predict(X_test)
            # Посмотрим эффективность модели
            cv_error = mape(y_test, y_pred)
            # Запишем статистику кросс валидации
            cv_info.loc[len(cv_info)]  = {'date':time_point_test, 'currency':currency_item,
'type_split':type_split, 'error':cv_error}
            # Сделал, чтобы выводилась статистика о ходе процесса
            print(f'{summary}, \nОшибка: {round(cv_error, 2)} %,  Валюта: {re.sub(r'[^a-zA-
z_]', '', currency_item)}\n')

    return cv_info
```

2.14 Создание модели и дата фрейма для хранения результатов кросс-валидации

Создадим модель машинного обучения, которая будет использоваться при кросс-валидации, а также дата фрейм, где будут храниться результаты кросс-валидации.

ячейка Jupiter Notebook

```
# Создадим модель с количеством деревьев в лесу 2000, с максимальной глубиной залегания дерева
21
rf_model = RandomForestRegressor(n_estimators=2000, max_depth=21, random_state=42,
criterion='squared_error')

# Обновим список валют
currency_list = [column for column in data_loaded if '_' in column]

# Датафрейм статистики кросс-валидации
cv_info = pd.DataFrame({'date':[], 'currency':[], 'type_split':[], 'error':[]})
```

2.15 Кросс-валидация данных

Будем производить кросс-валидацию данных на временном ряду с количеством проверок на тестовых данных равным 10. Тестовые данные будут взяты за год, квартал, месяц, неделю.

Оцениваться качество работы модели будет по метрике mape описанной в пункте 2.11, для каждой валюты отдельно. Взяв среднее значение этой метрики по 10 попыткам, получим более точные результаты. Более подробно о кросс-валидации на временном ряду описано в пункте 1.4.

2.15.1 С шагом в год

Запустим кросс-валидацию данных data_loaded с применением модели rf_model, начиная с 2013 года, генерить тестовую и тренировочные выборки будем 10 раз, с шагом тестовой выборки 1 год, по каждой валюте отдельно из списка валют currency_list, сохранять результаты кросс-валидации будем в дата фрейм cv_info.

ячейка Jupiter Notebook

```
%%time
# Будем кросс-валидировать 10 раз по году
cv_info = cross_validation_time_series(rf_model, data_loaded, 2013, 10, 'Year', currency_list,
cv_info)
```

В процессе кросс-валидации выводиться информация для Train и Test - диапазон дат выборки, размерность фичей и таргетов, ошибка предсказания рассчитывается по метрике mape описанной в пункте 2.11, так же выводиться наименование курса валюты.

результат выполнения

```
Train(11.05.2006-28.12.2012, X.shape(1526, 14), y.shape(1526,)), Test(10.01.2013-30.12.2013,
X.shape(232, 14), y.shape(232,)),
Ошибка: 5.29 %,  Валюта: chf_rub

Train(11.05.2006-30.12.2013, X.shape(1758, 14), y.shape(1758,)), Test(10.01.2014-30.12.2014,
X.shape(233, 14), y.shape(233,)),
Ошибка: 18.19 %,  Валюта: chf_rub

Train(11.05.2006-30.12.2014, X.shape(1991, 14), y.shape(1991,)), Test(13.01.2015-30.12.2015,
X.shape(231, 14), y.shape(231,)),
Ошибка: 11.26 %,  Валюта: chf_rub

Train(11.05.2006-30.12.2015, X.shape(2222, 14), y.shape(2222,)), Test(12.01.2016-30.12.2016,
X.shape(232, 14), y.shape(232,)),
Ошибка: 4.85 %,  Валюта: chf_rub

Train(11.05.2006-30.12.2016, X.shape(2454, 14), y.shape(2454,)), Test(10.01.2017-29.12.2017,
X.shape(232, 14), y.shape(232,)),
Ошибка: 7.86 %,  Валюта: chf_rub

Train(11.05.2006-29.12.2017, X.shape(2686, 14), y.shape(2686,)), Test(10.01.2018-28.12.2018,
X.shape(227, 14), y.shape(227,)),
Ошибка: 3.66 %,  Валюта: chf_rub

Train(11.05.2006-28.12.2018, X.shape(2913, 14), y.shape(2913,)), Test(10.01.2019-30.12.2019,
X.shape(232, 14), y.shape(232,)),
Ошибка: 6.55 %,  Валюта: chf_rub

Train(11.05.2006-30.12.2019, X.shape(3145, 14), y.shape(3145,)), Test(10.01.2020-30.12.2020,
X.shape(225, 14), y.shape(225,)),
Ошибка: 13.72 %,  Валюта: chf_rub

Train(11.05.2006-30.12.2020, X.shape(3370, 14), y.shape(3370,)), Test(12.01.2021-30.12.2021,
X.shape(233, 14), y.shape(233,)),
```

```
Ошибка: 0.6 %,  Валюта: chf_rub

Train(11.05.2006-30.12.2021, X.shape(3603, 14), y.shape(3603,)), Test(11.01.2022-30.12.2022,
X.shape(214, 14), y.shape(214,)),
Ошибка: 6.0 %,  Валюта: chf_rub

Train(11.05.2006-28.12.2012, X.shape(1526, 14), y.shape(1526,)), Test(10.01.2013-30.12.2013,
X.shape(232, 14), y.shape(232,)),
Ошибка: 10.15 %,  Валюта: cny_rub

Train(11.05.2006-30.12.2013, X.shape(1758, 14), y.shape(1758,)), Test(10.01.2014-30.12.2014,
X.shape(233, 14), y.shape(233,)),
Ошибка: 16.41 %,  Валюта: cny_rub

Train(11.05.2006-30.12.2014, X.shape(1991, 14), y.shape(1991,)), Test(13.01.2015-30.12.2015,
X.shape(231, 14), y.shape(231,)),
Ошибка: 10.55 %,  Валюта: cny_rub

Train(11.05.2006-30.12.2015, X.shape(2222, 14), y.shape(2222,)), Test(12.01.2016-30.12.2016,
X.shape(232, 14), y.shape(232,)),
Ошибка: 2.82 %,  Валюта: cny_rub

Train(11.05.2006-30.12.2016, X.shape(2454, 14), y.shape(2454,)), Test(10.01.2017-29.12.2017,
X.shape(232, 14), y.shape(232,)),
Ошибка: 8.63 %,  Валюта: cny_rub

Train(11.05.2006-29.12.2017, X.shape(2686, 14), y.shape(2686,)), Test(10.01.2018-28.12.2018,
X.shape(227, 14), y.shape(227,)),
Ошибка: 6.05 %,  Валюта: cny_rub

Train(11.05.2006-28.12.2018, X.shape(2913, 14), y.shape(2913,)), Test(10.01.2019-30.12.2019,
X.shape(232, 14), y.shape(232,)),
Ошибка: 6.36 %,  Валюта: cny_rub

Train(11.05.2006-30.12.2019, X.shape(3145, 14), y.shape(3145,)), Test(10.01.2020-30.12.2020,
X.shape(225, 14), y.shape(225,)),
Ошибка: 11.95 %,  Валюта: cny_rub

Train(11.05.2006-30.12.2020, X.shape(3370, 14), y.shape(3370,)), Test(12.01.2021-30.12.2021,
X.shape(233, 14), y.shape(233,)),
Ошибка: 3.73 %,  Валюта: cny_rub

Train(11.05.2006-30.12.2021, X.shape(3603, 14), y.shape(3603,)), Test(11.01.2022-30.12.2022,
X.shape(214, 14), y.shape(214,)),
Ошибка: 5.99 %,  Валюта: cny_rub

Train(11.05.2006-28.12.2012, X.shape(1526, 14), y.shape(1526,)), Test(10.01.2013-30.12.2013,
X.shape(232, 14), y.shape(232,)),
Ошибка: 5.05 %,  Валюта: eur_rub

Train(11.05.2006-30.12.2013, X.shape(1758, 14), y.shape(1758,)), Test(10.01.2014-30.12.2014,
X.shape(233, 14), y.shape(233,)),
Ошибка: 16.35 %,  Валюта: eur_rub

Train(11.05.2006-30.12.2014, X.shape(1991, 14), y.shape(1991,)), Test(13.01.2015-30.12.2015,
X.shape(231, 14), y.shape(231,)),
Ошибка: 0.19 %,  Валюта: eur_rub
```

```
Train(11.05.2006-30.12.2015, X.shape(2222, 14), y.shape(2222,)), Test(12.01.2016-30.12.2016,
X.shape(232, 14), y.shape(232,)),
Ошибка: 1.22 %,  Валюта: eur_rub

Train(11.05.2006-30.12.2016, X.shape(2454, 14), y.shape(2454,)), Test(10.01.2017-29.12.2017,
X.shape(232, 14), y.shape(232,)),
Ошибка: 2.16 %,  Валюта: eur_rub

Train(11.05.2006-29.12.2017, X.shape(2686, 14), y.shape(2686,)), Test(10.01.2018-28.12.2018,
X.shape(227, 14), y.shape(227,)),
Ошибка: 7.02 %,  Валюта: eur_rub

Train(11.05.2006-28.12.2018, X.shape(2913, 14), y.shape(2913,)), Test(10.01.2019-30.12.2019,
X.shape(232, 14), y.shape(232,)),
Ошибка: 7.34 %,  Валюта: eur_rub

Train(11.05.2006-30.12.2019, X.shape(3145, 14), y.shape(3145,)), Test(10.01.2020-30.12.2020,
X.shape(225, 14), y.shape(225,)),
Ошибка: 10.42 %,  Валюта: eur_rub

Train(11.05.2006-30.12.2020, X.shape(3370, 14), y.shape(3370,)), Test(12.01.2021-30.12.2021,
X.shape(233, 14), y.shape(233,)),
Ошибка: 1.03 %,  Валюта: eur_rub

Train(11.05.2006-30.12.2021, X.shape(3603, 14), y.shape(3603,)), Test(11.01.2022-30.12.2022,
X.shape(214, 14), y.shape(214,)),
Ошибка: 15.41 %,  Валюта: eur_rub

Train(11.05.2006-28.12.2012, X.shape(1526, 14), y.shape(1526,)), Test(10.01.2013-30.12.2013,
X.shape(232, 14), y.shape(232,)),
Ошибка: 2.1 %,  Валюта: gbp_rub

Train(11.05.2006-30.12.2013, X.shape(1758, 14), y.shape(1758,)), Test(10.01.2014-30.12.2014,
X.shape(233, 14), y.shape(233,)),
Ошибка: 19.79 %,  Валюта: gbp_rub

Train(11.05.2006-30.12.2014, X.shape(1991, 14), y.shape(1991,)), Test(13.01.2015-30.12.2015,
X.shape(231, 14), y.shape(231,)),
Ошибка: 9.43 %,  Валюта: gbp_rub

Train(11.05.2006-30.12.2015, X.shape(2222, 14), y.shape(2222,)), Test(12.01.2016-30.12.2016,
X.shape(232, 14), y.shape(232,)),
Ошибка: 13.05 %,  Валюта: gbp_rub

Train(11.05.2006-30.12.2016, X.shape(2454, 14), y.shape(2454,)), Test(10.01.2017-29.12.2017,
X.shape(232, 14), y.shape(232,)),
Ошибка: 4.42 %,  Валюта: gbp_rub

Train(11.05.2006-29.12.2017, X.shape(2686, 14), y.shape(2686,)), Test(10.01.2018-28.12.2018,
X.shape(227, 14), y.shape(227,)),
Ошибка: 8.71 %,  Валюта: gbp_rub

Train(11.05.2006-28.12.2018, X.shape(2913, 14), y.shape(2913,)), Test(10.01.2019-30.12.2019,
X.shape(232, 14), y.shape(232,)),
Ошибка: 7.7 %,  Валюта: gbp_rub
```

```
Train(11.05.2006-30.12.2019, X.shape(3145, 14), y.shape(3145,)), Test(10.01.2020-30.12.2020,
X.shape(225, 14), y.shape(225,)),
Ошибка: 11.74 %,  Валюта: gbp_rub

Train(11.05.2006-30.12.2020, X.shape(3370, 14), y.shape(3370,)), Test(12.01.2021-30.12.2021,
X.shape(233, 14), y.shape(233,)),
Ошибка: 5.23 %,  Валюта: gbp_rub

Train(11.05.2006-30.12.2021, X.shape(3603, 14), y.shape(3603,)), Test(11.01.2022-30.12.2022,
X.shape(214, 14), y.shape(214,)),
Ошибка: 13.33 %,  Валюта: gbp_rub

Train(11.05.2006-28.12.2012, X.shape(1526, 14), y.shape(1526,)), Test(10.01.2013-30.12.2013,
X.shape(232, 14), y.shape(232,)),
Ошибка: 12.72 %,  Валюта: jpy_rub

Train(11.05.2006-30.12.2013, X.shape(1758, 14), y.shape(1758,)), Test(10.01.2014-30.12.2014,
X.shape(233, 14), y.shape(233,)),
Ошибка: 6.77 %,  Валюта: jpy_rub

Train(11.05.2006-30.12.2014, X.shape(1991, 14), y.shape(1991,)), Test(13.01.2015-30.12.2015,
X.shape(231, 14), y.shape(231,)),
Ошибка: 9.23 %,  Валюта: jpy_rub

Train(11.05.2006-30.12.2015, X.shape(2222, 14), y.shape(2222,)), Test(12.01.2016-30.12.2016,
X.shape(232, 14), y.shape(232,)),
Ошибка: 11.7 %,  Валюта: jpy_rub

Train(11.05.2006-30.12.2016, X.shape(2454, 14), y.shape(2454,)), Test(10.01.2017-29.12.2017,
X.shape(232, 14), y.shape(232,)),
Ошибка: 2.93 %,  Валюта: jpy_rub

Train(11.05.2006-29.12.2017, X.shape(2686, 14), y.shape(2686,)), Test(10.01.2018-28.12.2018,
X.shape(227, 14), y.shape(227,)),
Ошибка: 1.93 %,  Валюта: jpy_rub

Train(11.05.2006-28.12.2018, X.shape(2913, 14), y.shape(2913,)), Test(10.01.2019-30.12.2019,
X.shape(232, 14), y.shape(232,)),
Ошибка: 2.27 %,  Валюта: jpy_rub

Train(11.05.2006-30.12.2019, X.shape(3145, 14), y.shape(3145,)), Test(10.01.2020-30.12.2020,
X.shape(225, 14), y.shape(225,)),
Ошибка: 8.41 %,  Валюта: jpy_rub

Train(11.05.2006-30.12.2020, X.shape(3370, 14), y.shape(3370,)), Test(12.01.2021-30.12.2021,
X.shape(233, 14), y.shape(233,)),
Ошибка: 4.12 %,  Валюта: jpy_rub

Train(11.05.2006-30.12.2021, X.shape(3603, 14), y.shape(3603,)), Test(11.01.2022-30.12.2022,
X.shape(214, 14), y.shape(214,)),
Ошибка: 27.96 %,  Валюта: jpy_rub

Train(11.05.2006-28.12.2012, X.shape(1526, 14), y.shape(1526,)), Test(10.01.2013-30.12.2013,
X.shape(232, 14), y.shape(232,)),
Ошибка: 4.1 %,  Валюта: usd_rub
```

```
Train(11.05.2006-30.12.2013, X.shape(1758, 14), y.shape(1758,)), Test(10.01.2014-30.12.2014,
X.shape(233, 14), y.shape(233,)),
Ошибка: 16.43 %,  Валюта: usd_rub

Train(11.05.2006-30.12.2014, X.shape(1991, 14), y.shape(1991,)), Test(13.01.2015-30.12.2015,
X.shape(231, 14), y.shape(231,)),
Ошибка: 12.09 %,  Валюта: usd_rub

Train(11.05.2006-30.12.2015, X.shape(2222, 14), y.shape(2222,)), Test(12.01.2016-30.12.2016,
X.shape(232, 14), y.shape(232,)),
Ошибка: 1.67 %,  Валюта: usd_rub

Train(11.05.2006-30.12.2016, X.shape(2454, 14), y.shape(2454,)), Test(10.01.2017-29.12.2017,
X.shape(232, 14), y.shape(232,)),
Ошибка: 7.68 %,  Валюта: usd_rub

Train(11.05.2006-29.12.2017, X.shape(2686, 14), y.shape(2686,)), Test(10.01.2018-28.12.2018,
X.shape(227, 14), y.shape(227,)),
Ошибка: 5.14 %,  Валюта: usd_rub

Train(11.05.2006-28.12.2018, X.shape(2913, 14), y.shape(2913,)), Test(10.01.2019-30.12.2019,
X.shape(232, 14), y.shape(232,)),
Ошибка: 2.86 %,  Валюта: usd_rub

Train(11.05.2006-30.12.2019, X.shape(3145, 14), y.shape(3145,)), Test(10.01.2020-30.12.2020,
X.shape(225, 14), y.shape(225,)),
Ошибка: 6.42 %,  Валюта: usd_rub

Train(11.05.2006-30.12.2020, X.shape(3370, 14), y.shape(3370,)), Test(12.01.2021-30.12.2021,
X.shape(233, 14), y.shape(233,)),
Ошибка: 0.98 %,  Валюта: usd_rub

Train(11.05.2006-30.12.2021, X.shape(3603, 14), y.shape(3603,)), Test(11.01.2022-30.12.2022,
X.shape(214, 14), y.shape(214,)),
Ошибка: 6.21 %,  Валюта: usd_rub

CPU times: total: 22min 12s
Wall time: 32min 44s
```

2.15.2 С шагом в квартал

Запустим кросс-валидацию данных data_loaded с применением модели rf_model, начиная с 2022 года, генерить тестовую и тренировочные выборки будем 10 раз, с шагом тестовой выборки 1 квартал, по каждой валюте отдельно из списка валют currency_list, сохранять результаты кросс-валидации будем в дата фрейм cv_info.

ячейка Jupiter Notebook

```
%%time
# Будем кросс-валидировать 10 раз по кварталу
cv_info = cross_validation_time_series(rf_model, data_loaded, 2022, 10, 'Quarter',
currency_list, cv_info)
```

В процессе кросс-валидации выводиться информация для Train и Test - диапазон дат выборки, размерность фичей и таргетов, ошибка предсказания рассчитывается по метрике mape описанной в пункте 2.11, так же выводиться наименование курса валюты.

результат выполнения

```
Train(11.05.2006-30.12.2021, X.shape(3603, 14), y.shape(3603,)), Test(11.01.2022-31.03.2022,
X.shape(35, 14), y.shape(35,)),
Ошибка: 6.52 %,  Валюта: chf_rub

Train(11.05.2006-31.03.2022, X.shape(3638, 14), y.shape(3638,)), Test(01.04.2022-30.06.2022,
X.shape(54, 14), y.shape(54,)),
Ошибка: 12.28 %,  Валюта: chf_rub

Train(11.05.2006-30.06.2022, X.shape(3692, 14), y.shape(3692,)), Test(01.07.2022-30.09.2022,
X.shape(64, 14), y.shape(64,)),
Ошибка: 5.32 %,  Валюта: chf_rub

Train(11.05.2006-30.09.2022, X.shape(3756, 14), y.shape(3756,)), Test(03.10.2022-30.12.2022,
X.shape(61, 14), y.shape(61,)),
Ошибка: 2.75 %,  Валюта: chf_rub

Train(11.05.2006-30.12.2022, X.shape(3817, 14), y.shape(3817,)), Test(10.01.2023-31.03.2023,
X.shape(52, 14), y.shape(52,)),
Ошибка: 1.32 %,  Валюта: chf_rub

Train(11.05.2006-31.03.2023, X.shape(3869, 14), y.shape(3869,)), Test(03.04.2023-30.06.2023,
X.shape(59, 14), y.shape(59,)),
Ошибка: 3.19 %,  Валюта: chf_rub

Train(11.05.2006-30.06.2023, X.shape(3928, 14), y.shape(3928,)), Test(03.07.2023-29.09.2023,
X.shape(63, 14), y.shape(63,)),
Ошибка: 9.4 %,  Валюта: chf_rub

Train(11.05.2006-29.09.2023, X.shape(3991, 14), y.shape(3991,)), Test(02.10.2023-29.12.2023,
X.shape(63, 14), y.shape(63,)),
Ошибка: 2.66 %,  Валюта: chf_rub

Train(11.05.2006-29.12.2023, X.shape(4054, 14), y.shape(4054,)), Test(03.01.2024-28.03.2024,
X.shape(58, 14), y.shape(58,)),
Ошибка: 1.2 %,  Валюта: chf_rub

Train(11.05.2006-28.03.2024, X.shape(4112, 14), y.shape(4112,)), Test(01.04.2024-28.06.2024,
X.shape(60, 14), y.shape(60,)),
Ошибка: 5.04 %,  Валюта: chf_rub

Train(11.05.2006-30.12.2021, X.shape(3603, 14), y.shape(3603,)), Test(11.01.2022-31.03.2022,
X.shape(35, 14), y.shape(35,)),
Ошибка: 7.96 %,  Валюта: cny_rub

Train(11.05.2006-31.03.2022, X.shape(3638, 14), y.shape(3638,)), Test(01.04.2022-30.06.2022,
X.shape(54, 14), y.shape(54,)),
Ошибка: 9.06 %,  Валюта: cny_rub

Train(11.05.2006-30.06.2022, X.shape(3692, 14), y.shape(3692,)), Test(01.07.2022-30.09.2022,
X.shape(64, 14), y.shape(64,)),
```

```
Ошибка: 6.03 %,  Валюта: cny_rub

Train(11.05.2006-30.09.2022, X.shape(3756, 14), y.shape(3756,)), Test(03.10.2022-30.12.2022,
X.shape(61, 14), y.shape(61,)),
Ошибка: 4.86 %,  Валюта: cny_rub

Train(11.05.2006-30.12.2022, X.shape(3817, 14), y.shape(3817,)), Test(10.01.2023-31.03.2023,
X.shape(52, 14), y.shape(52,)),
Ошибка: 6.87 %,  Валюта: cny_rub

Train(11.05.2006-31.03.2023, X.shape(3869, 14), y.shape(3869,)), Test(03.04.2023-30.06.2023,
X.shape(59, 14), y.shape(59,)),
Ошибка: 8.33 %,  Валюта: cny_rub

Train(11.05.2006-30.06.2023, X.shape(3928, 14), y.shape(3928,)), Test(03.07.2023-29.09.2023,
X.shape(63, 14), y.shape(63,)),
Ошибка: 1.53 %,  Валюта: cny_rub

Train(11.05.2006-29.09.2023, X.shape(3991, 14), y.shape(3991,)), Test(02.10.2023-29.12.2023,
X.shape(63, 14), y.shape(63,)),
Ошибка: 1.33 %,  Валюта: cny_rub

Train(11.05.2006-29.12.2023, X.shape(4054, 14), y.shape(4054,)), Test(03.01.2024-28.03.2024,
X.shape(58, 14), y.shape(58,)),
Ошибка: 0.84 %,  Валюта: cny_rub

Train(11.05.2006-28.03.2024, X.shape(4112, 14), y.shape(4112,)), Test(01.04.2024-28.06.2024,
X.shape(60, 14), y.shape(60,)),
Ошибка: 3.27 %,  Валюта: cny_rub

Train(11.05.2006-30.12.2021, X.shape(3603, 14), y.shape(3603,)), Test(11.01.2022-31.03.2022,
X.shape(35, 14), y.shape(35,)),
Ошибка: 2.49 %,  Валюта: eur_rub

Train(11.05.2006-31.03.2022, X.shape(3638, 14), y.shape(3638,)), Test(01.04.2022-30.06.2022,
X.shape(54, 14), y.shape(54,)),
Ошибка: 15.38 %,  Валюта: eur_rub

Train(11.05.2006-30.06.2022, X.shape(3692, 14), y.shape(3692,)), Test(01.07.2022-30.09.2022,
X.shape(64, 14), y.shape(64,)),
Ошибка: 6.11 %,  Валюта: eur_rub

Train(11.05.2006-30.09.2022, X.shape(3756, 14), y.shape(3756,)), Test(03.10.2022-30.12.2022,
X.shape(61, 14), y.shape(61,)),
Ошибка: 0.7 %,  Валюта: eur_rub

Train(11.05.2006-30.12.2022, X.shape(3817, 14), y.shape(3817,)), Test(10.01.2023-31.03.2023,
X.shape(52, 14), y.shape(52,)),
Ошибка: 3.65 %,  Валюта: eur_rub

Train(11.05.2006-31.03.2023, X.shape(3869, 14), y.shape(3869,)), Test(03.04.2023-30.06.2023,
X.shape(59, 14), y.shape(59,)),
Ошибка: 3.95 %,  Валюта: eur_rub

Train(11.05.2006-30.06.2023, X.shape(3928, 14), y.shape(3928,)), Test(03.07.2023-29.09.2023,
X.shape(63, 14), y.shape(63,)),
Ошибка: 3.93 %,  Валюта: eur_rub
```

```
Train(11.05.2006-29.09.2023, X.shape(3991, 14), y.shape(3991,)), Test(02.10.2023-29.12.2023,
X.shape(63, 14), y.shape(63,)),
Ошибка: 2.86 %,  Валюта: eur_rub

Train(11.05.2006-29.12.2023, X.shape(4054, 14), y.shape(4054,)), Test(03.01.2024-28.03.2024,
X.shape(58, 14), y.shape(58,)),
Ошибка: 0.32 %,  Валюта: eur_rub

Train(11.05.2006-28.03.2024, X.shape(4112, 14), y.shape(4112,)), Test(01.04.2024-28.06.2024,
X.shape(60, 14), y.shape(60,)),
Ошибка: 2.27 %,  Валюта: eur_rub

Train(11.05.2006-30.12.2021, X.shape(3603, 14), y.shape(3603,)), Test(11.01.2022-31.03.2022,
X.shape(35, 14), y.shape(35,)),
Ошибка: 5.03 %,  Валюта: gbp_rub

Train(11.05.2006-31.03.2022, X.shape(3638, 14), y.shape(3638,)), Test(01.04.2022-30.06.2022,
X.shape(54, 14), y.shape(54,)),
Ошибка: 13.19 %,  Валюта: gbp_rub

Train(11.05.2006-30.06.2022, X.shape(3692, 14), y.shape(3692,)), Test(01.07.2022-30.09.2022,
X.shape(64, 14), y.shape(64,)),
Ошибка: 9.22 %,  Валюта: gbp_rub

Train(11.05.2006-30.09.2022, X.shape(3756, 14), y.shape(3756,)), Test(03.10.2022-30.12.2022,
X.shape(61, 14), y.shape(61,)),
Ошибка: 5.41 %,  Валюта: gbp_rub

Train(11.05.2006-30.12.2022, X.shape(3817, 14), y.shape(3817,)), Test(10.01.2023-31.03.2023,
X.shape(52, 14), y.shape(52,)),
Ошибка: 3.79 %,  Валюта: gbp_rub

Train(11.05.2006-31.03.2023, X.shape(3869, 14), y.shape(3869,)), Test(03.04.2023-30.06.2023,
X.shape(59, 14), y.shape(59,)),
Ошибка: 4.3 %,  Валюта: gbp_rub

Train(11.05.2006-30.06.2023, X.shape(3928, 14), y.shape(3928,)), Test(03.07.2023-29.09.2023,
X.shape(63, 14), y.shape(63,)),
Ошибка: 4.0 %,  Валюта: gbp_rub

Train(11.05.2006-29.09.2023, X.shape(3991, 14), y.shape(3991,)), Test(02.10.2023-29.12.2023,
X.shape(63, 14), y.shape(63,)),
Ошибка: 3.55 %,  Валюта: gbp_rub

Train(11.05.2006-29.12.2023, X.shape(4054, 14), y.shape(4054,)), Test(03.01.2024-28.03.2024,
X.shape(58, 14), y.shape(58,)),
Ошибка: 0.95 %,  Валюта: gbp_rub

Train(11.05.2006-28.03.2024, X.shape(4112, 14), y.shape(4112,)), Test(01.04.2024-28.06.2024,
X.shape(60, 14), y.shape(60,)),
Ошибка: 2.24 %,  Валюта: gbp_rub

Train(11.05.2006-30.12.2021, X.shape(3603, 14), y.shape(3603,)), Test(11.01.2022-31.03.2022,
X.shape(35, 14), y.shape(35,)),
Ошибка: 2.69 %,  Валюта: jpy_rub
```

```
Train(11.05.2006-31.03.2022, X.shape(3638, 14), y.shape(3638,)), Test(01.04.2022-30.06.2022,
X.shape(54, 14), y.shape(54,)),
Ошибка: 27.38 %,  Валюта: jpy_rub

Train(11.05.2006-30.06.2022, X.shape(3692, 14), y.shape(3692,)), Test(01.07.2022-30.09.2022,
X.shape(64, 14), y.shape(64,)),
Ошибка: 8.99 %,  Валюта: jpy_rub

Train(11.05.2006-30.09.2022, X.shape(3756, 14), y.shape(3756,)), Test(03.10.2022-30.12.2022,
X.shape(61, 14), y.shape(61,)),
Ошибка: 3.28 %,  Валюта: jpy_rub

Train(11.05.2006-30.12.2022, X.shape(3817, 14), y.shape(3817,)), Test(10.01.2023-31.03.2023,
X.shape(52, 14), y.shape(52,)),
Ошибка: 3.38 %,  Валюта: jpy_rub

Train(11.05.2006-31.03.2023, X.shape(3869, 14), y.shape(3869,)), Test(03.04.2023-30.06.2023,
X.shape(59, 14), y.shape(59,)),
Ошибка: 2.49 %,  Валюта: jpy_rub

Train(11.05.2006-30.06.2023, X.shape(3928, 14), y.shape(3928,)), Test(03.07.2023-29.09.2023,
X.shape(63, 14), y.shape(63,)),
Ошибка: 10.2 %,  Валюта: jpy_rub

Train(11.05.2006-29.09.2023, X.shape(3991, 14), y.shape(3991,)), Test(02.10.2023-29.12.2023,
X.shape(63, 14), y.shape(63,)),
Ошибка: 3.11 %,  Валюта: jpy_rub

Train(11.05.2006-29.12.2023, X.shape(4054, 14), y.shape(4054,)), Test(03.01.2024-28.03.2024,
X.shape(58, 14), y.shape(58,)),
Ошибка: 2.04 %,  Валюта: jpy_rub

Train(11.05.2006-28.03.2024, X.shape(4112, 14), y.shape(4112,)), Test(01.04.2024-28.06.2024,
X.shape(60, 14), y.shape(60,)),
Ошибка: 5.83 %,  Валюта: jpy_rub

Train(11.05.2006-30.12.2021, X.shape(3603, 14), y.shape(3603,)), Test(11.01.2022-31.03.2022,
X.shape(35, 14), y.shape(35,)),
Ошибка: 6.08 %,  Валюта: usd_rub

Train(11.05.2006-31.03.2022, X.shape(3638, 14), y.shape(3638,)), Test(01.04.2022-30.06.2022,
X.shape(54, 14), y.shape(54,)),
Ошибка: 15.88 %,  Валюта: usd_rub

Train(11.05.2006-30.06.2022, X.shape(3692, 14), y.shape(3692,)), Test(01.07.2022-30.09.2022,
X.shape(64, 14), y.shape(64,)),
Ошибка: 4.29 %,  Валюта: usd_rub

Train(11.05.2006-30.09.2022, X.shape(3756, 14), y.shape(3756,)), Test(03.10.2022-30.12.2022,
X.shape(61, 14), y.shape(61,)),
Ошибка: 2.06 %,  Валюта: usd_rub

Train(11.05.2006-30.12.2022, X.shape(3817, 14), y.shape(3817,)), Test(10.01.2023-31.03.2023,
X.shape(52, 14), y.shape(52,)),
Ошибка: 2.81 %,  Валюта: usd_rub
```

```
Train(11.05.2006-31.03.2023, X.shape(3869, 14), y.shape(3869,)), Test(03.04.2023-30.06.2023,
X.shape(59, 14), y.shape(59,)),
Ошибка: 0.66 %,  Валюта: usd_rub

Train(11.05.2006-30.06.2023, X.shape(3928, 14), y.shape(3928,)), Test(03.07.2023-29.09.2023,
X.shape(63, 14), y.shape(63,)),
Ошибка: 6.14 %,  Валюта: usd_rub

Train(11.05.2006-29.09.2023, X.shape(3991, 14), y.shape(3991,)), Test(02.10.2023-29.12.2023,
X.shape(63, 14), y.shape(63,)),
Ошибка: 2.32 %,  Валюта: usd_rub

Train(11.05.2006-29.12.2023, X.shape(4054, 14), y.shape(4054,)), Test(03.01.2024-28.03.2024,
X.shape(58, 14), y.shape(58,)),
Ошибка: 1.52 %,  Валюта: usd_rub

Train(11.05.2006-28.03.2024, X.shape(4112, 14), y.shape(4112,)), Test(01.04.2024-28.06.2024,
X.shape(60, 14), y.shape(60,)),
Ошибка: 1.79 %,  Валюта: usd_rub

CPU times: total: 32min 11s
Wall time: 49min 56s
```

2.15.3 С шагом в месяц

Запустим кросс-валидацию данных data_loaded с применением модели rf_model, начиная с 2023 года, генерить тестовую и тренировочные выборки будем 10 раз, с шагом тестовой выборки 1 месяц, по каждой валюте отдельно из списка валют currency_list, сохранять результаты кросс-валидации будем в дата фрейм cv_info.

ячейка Jupiter Notebook

```
%%time
# Будем кросс-валидировать 10 раз по месяцу
cv_info = cross_validation_time_series(rf_model, data_loaded, 2023, 10, 'Month', currency_list,
cv_info)
```

В процессе кросс-валидации выводиться информация для Train и Test - диапазон дат выборки, размерность фичей и таргетов, ошибка предсказания рассчитывается по метрике mape описанной в пункте 2.11, так же выводиться наименование курса валюты.

результат выполнения

```
Train(11.05.2006-30.12.2022, X.shape(3817, 14), y.shape(3817,)), Test(10.01.2023-31.01.2023,
X.shape(15, 14), y.shape(15,)),
Ошибка: 3.38 %,  Валюта: chf_rub

Train(11.05.2006-31.01.2023, X.shape(3832, 14), y.shape(3832,)), Test(01.02.2023-28.02.2023,
X.shape(16, 14), y.shape(16,)),
Ошибка: 4.48 %,  Валюта: chf_rub

Train(11.05.2006-28.02.2023, X.shape(3848, 14), y.shape(3848,)), Test(01.03.2023-31.03.2023,
X.shape(21, 14), y.shape(21,)),
Ошибка: 2.93 %,  Валюта: chf_rub
```

```
Train(11.05.2006-31.03.2023, X.shape(3869, 14), y.shape(3869,)), Test(03.04.2023-28.04.2023,
X.shape(19, 14), y.shape(19,)),
Ошибка: 1.41 %,  Валюта: chf_rub

Train(11.05.2006-28.04.2023, X.shape(3888, 14), y.shape(3888,)), Test(02.05.2023-31.05.2023,
X.shape(20, 14), y.shape(20,)),
Ошибка: 1.89 %,  Валюта: chf_rub

Train(11.05.2006-31.05.2023, X.shape(3908, 14), y.shape(3908,)), Test(01.06.2023-30.06.2023,
X.shape(20, 14), y.shape(20,)),
Ошибка: 3.03 %,  Валюта: chf_rub

Train(11.05.2006-30.06.2023, X.shape(3928, 14), y.shape(3928,)), Test(03.07.2023-31.07.2023,
X.shape(20, 14), y.shape(20,)),
Ошибка: 7.4 %,  Валюта: chf_rub

Train(11.05.2006-31.07.2023, X.shape(3948, 14), y.shape(3948,)), Test(01.08.2023-31.08.2023,
X.shape(23, 14), y.shape(23,)),
Ошибка: 3.53 %,  Валюта: chf_rub

Train(11.05.2006-31.08.2023, X.shape(3971, 14), y.shape(3971,)), Test(01.09.2023-29.09.2023,
X.shape(20, 14), y.shape(20,)),
Ошибка: 2.18 %,  Валюта: chf_rub

Train(11.05.2006-29.09.2023, X.shape(3991, 14), y.shape(3991,)), Test(02.10.2023-31.10.2023,
X.shape(22, 14), y.shape(22,)),
Ошибка: 0.01 %,  Валюта: chf_rub

Train(11.05.2006-30.12.2022, X.shape(3817, 14), y.shape(3817,)), Test(10.01.2023-31.01.2023,
X.shape(15, 14), y.shape(15,)),
Ошибка: 11.96 %,  Валюта: cny_rub

Train(11.05.2006-31.01.2023, X.shape(3832, 14), y.shape(3832,)), Test(01.02.2023-28.02.2023,
X.shape(16, 14), y.shape(16,)),
Ошибка: 1.86 %,  Валюта: cny_rub

Train(11.05.2006-28.02.2023, X.shape(3848, 14), y.shape(3848,)), Test(01.03.2023-31.03.2023,
X.shape(21, 14), y.shape(21,)),
Ошибка: 1.63 %,  Валюта: cny_rub

Train(11.05.2006-31.03.2023, X.shape(3869, 14), y.shape(3869,)), Test(03.04.2023-28.04.2023,
X.shape(19, 14), y.shape(19,)),
Ошибка: 7.94 %,  Валюта: cny_rub

Train(11.05.2006-28.04.2023, X.shape(3888, 14), y.shape(3888,)), Test(02.05.2023-31.05.2023,
X.shape(20, 14), y.shape(20,)),
Ошибка: 3.18 %,  Валюта: cny_rub

Train(11.05.2006-31.05.2023, X.shape(3908, 14), y.shape(3908,)), Test(01.06.2023-30.06.2023,
X.shape(20, 14), y.shape(20,)),
Ошибка: 1.52 %,  Валюта: cny_rub

Train(11.05.2006-30.06.2023, X.shape(3928, 14), y.shape(3928,)), Test(03.07.2023-31.07.2023,
X.shape(20, 14), y.shape(20,)),
Ошибка: 1.85 %,  Валюта: cny_rub
```

```
Train(11.05.2006-31.07.2023, X.shape(3948, 14), y.shape(3948,)), Test(01.08.2023-31.08.2023,
X.shape(23, 14), y.shape(23,)),
Ошибка: 2.38 %,  Валюта: cny_rub

Train(11.05.2006-31.08.2023, X.shape(3971, 14), y.shape(3971,)), Test(01.09.2023-29.09.2023,
X.shape(20, 14), y.shape(20,)),
Ошибка: 1.28 %,  Валюта: cny_rub

Train(11.05.2006-29.09.2023, X.shape(3991, 14), y.shape(3991,)), Test(02.10.2023-31.10.2023,
X.shape(22, 14), y.shape(22,)),
Ошибка: 1.16 %,  Валюта: cny_rub

Train(11.05.2006-30.12.2022, X.shape(3817, 14), y.shape(3817,)), Test(10.01.2023-31.01.2023,
X.shape(15, 14), y.shape(15,)),
Ошибка: 0.39 %,  Валюта: eur_rub

Train(11.05.2006-31.01.2023, X.shape(3832, 14), y.shape(3832,)), Test(01.02.2023-28.02.2023,
X.shape(16, 14), y.shape(16,)),
Ошибка: 4.09 %,  Валюта: eur_rub

Train(11.05.2006-28.02.2023, X.shape(3848, 14), y.shape(3848,)), Test(01.03.2023-31.03.2023,
X.shape(21, 14), y.shape(21,)),
Ошибка: 1.68 %,  Валюта: eur_rub

Train(11.05.2006-31.03.2023, X.shape(3869, 14), y.shape(3869,)), Test(03.04.2023-28.04.2023,
X.shape(19, 14), y.shape(19,)),
Ошибка: 4.64 %,  Валюта: eur_rub

Train(11.05.2006-28.04.2023, X.shape(3888, 14), y.shape(3888,)), Test(02.05.2023-31.05.2023,
X.shape(20, 14), y.shape(20,)),
Ошибка: 2.15 %,  Валюта: eur_rub

Train(11.05.2006-31.05.2023, X.shape(3908, 14), y.shape(3908,)), Test(01.06.2023-30.06.2023,
X.shape(20, 14), y.shape(20,)),
Ошибка: 3.41 %,  Валюта: eur_rub

Train(11.05.2006-30.06.2023, X.shape(3928, 14), y.shape(3928,)), Test(03.07.2023-31.07.2023,
X.shape(20, 14), y.shape(20,)),
Ошибка: 2.51 %,  Валюта: eur_rub

Train(11.05.2006-31.07.2023, X.shape(3948, 14), y.shape(3948,)), Test(01.08.2023-31.08.2023,
X.shape(23, 14), y.shape(23,)),
Ошибка: 2.64 %,  Валюта: eur_rub

Train(11.05.2006-31.08.2023, X.shape(3971, 14), y.shape(3971,)), Test(01.09.2023-29.09.2023,
X.shape(20, 14), y.shape(20,)),
Ошибка: 2.06 %,  Валюта: eur_rub

Train(11.05.2006-29.09.2023, X.shape(3991, 14), y.shape(3991,)), Test(02.10.2023-31.10.2023,
X.shape(22, 14), y.shape(22,)),
Ошибка: 0.45 %,  Валюта: eur_rub

Train(11.05.2006-30.12.2022, X.shape(3817, 14), y.shape(3817,)), Test(10.01.2023-31.01.2023,
X.shape(15, 14), y.shape(15,)),
Ошибка: 7.33 %,  Валюта: gbp_rub
```

```
Train(11.05.2006-31.01.2023, X.shape(3832, 14), y.shape(3832,)), Test(01.02.2023-28.02.2023,
X.shape(16, 14), y.shape(16,)),
Ошибка: 0.8 %,  Валюта: gbp_rub

Train(11.05.2006-28.02.2023, X.shape(3848, 14), y.shape(3848,)), Test(01.03.2023-31.03.2023,
X.shape(21, 14), y.shape(21,)),
Ошибка: 1.41 %,  Валюта: gbp_rub

Train(11.05.2006-31.03.2023, X.shape(3869, 14), y.shape(3869,)), Test(03.04.2023-28.04.2023,
X.shape(19, 14), y.shape(19,)),
Ошибка: 3.51 %,  Валюта: gbp_rub

Train(11.05.2006-28.04.2023, X.shape(3888, 14), y.shape(3888,)), Test(02.05.2023-31.05.2023,
X.shape(20, 14), y.shape(20,)),
Ошибка: 2.42 %,  Валюта: gbp_rub

Train(11.05.2006-31.05.2023, X.shape(3908, 14), y.shape(3908,)), Test(01.06.2023-30.06.2023,
X.shape(20, 14), y.shape(20,)),
Ошибка: 4.85 %,  Валюта: gbp_rub

Train(11.05.2006-30.06.2023, X.shape(3928, 14), y.shape(3928,)), Test(03.07.2023-31.07.2023,
X.shape(20, 14), y.shape(20,)),
Ошибка: 2.05 %,  Валюта: gbp_rub

Train(11.05.2006-31.07.2023, X.shape(3948, 14), y.shape(3948,)), Test(01.08.2023-31.08.2023,
X.shape(23, 14), y.shape(23,)),
Ошибка: 2.61 %,  Валюта: gbp_rub

Train(11.05.2006-31.08.2023, X.shape(3971, 14), y.shape(3971,)), Test(01.09.2023-29.09.2023,
X.shape(20, 14), y.shape(20,)),
Ошибка: 2.16 %,  Валюта: gbp_rub

Train(11.05.2006-29.09.2023, X.shape(3991, 14), y.shape(3991,)), Test(02.10.2023-31.10.2023,
X.shape(22, 14), y.shape(22,)),
Ошибка: 1.17 %,  Валюта: gbp_rub

Train(11.05.2006-30.12.2022, X.shape(3817, 14), y.shape(3817,)), Test(10.01.2023-31.01.2023,
X.shape(15, 14), y.shape(15,)),
Ошибка: 5.82 %,  Валюта: jpy_rub

Train(11.05.2006-31.01.2023, X.shape(3832, 14), y.shape(3832,)), Test(01.02.2023-28.02.2023,
X.shape(16, 14), y.shape(16,)),
Ошибка: 0.34 %,  Валюта: jpy_rub

Train(11.05.2006-28.02.2023, X.shape(3848, 14), y.shape(3848,)), Test(01.03.2023-31.03.2023,
X.shape(21, 14), y.shape(21,)),
Ошибка: 2.39 %,  Валюта: jpy_rub

Train(11.05.2006-31.03.2023, X.shape(3869, 14), y.shape(3869,)), Test(03.04.2023-28.04.2023,
X.shape(19, 14), y.shape(19,)),
Ошибка: 0.65 %,  Валюта: jpy_rub

Train(11.05.2006-28.04.2023, X.shape(3888, 14), y.shape(3888,)), Test(02.05.2023-31.05.2023,
X.shape(20, 14), y.shape(20,)),
Ошибка: 5.31 %,  Валюта: jpy_rub
```

```
Train(11.05.2006-31.05.2023, X.shape(3908, 14), y.shape(3908,)), Test(01.06.2023-30.06.2023,
X.shape(20, 14), y.shape(20,)),
Ошибка: 0.16 %,  Валюта: jpy_rub

Train(11.05.2006-30.06.2023, X.shape(3928, 14), y.shape(3928,)), Test(03.07.2023-31.07.2023,
X.shape(20, 14), y.shape(20,)),
Ошибка: 4.88 %,  Валюта: jpy_rub

Train(11.05.2006-31.07.2023, X.shape(3948, 14), y.shape(3948,)), Test(01.08.2023-31.08.2023,
X.shape(23, 14), y.shape(23,)),
Ошибка: 3.13 %,  Валюта: jpy_rub

Train(11.05.2006-31.08.2023, X.shape(3971, 14), y.shape(3971,)), Test(01.09.2023-29.09.2023,
X.shape(20, 14), y.shape(20,)),
Ошибка: 0.74 %,  Валюта: jpy_rub

Train(11.05.2006-29.09.2023, X.shape(3991, 14), y.shape(3991,)), Test(02.10.2023-31.10.2023,
X.shape(22, 14), y.shape(22,)),
Ошибка: 0.41 %,  Валюта: jpy_rub

Train(11.05.2006-30.12.2022, X.shape(3817, 14), y.shape(3817,)), Test(10.01.2023-31.01.2023,
X.shape(15, 14), y.shape(15,)),
Ошибка: 7.9 %,  Валюта: usd_rub

Train(11.05.2006-31.01.2023, X.shape(3832, 14), y.shape(3832,)), Test(01.02.2023-28.02.2023,
X.shape(16, 14), y.shape(16,)),
Ошибка: 3.89 %,  Валюта: usd_rub

Train(11.05.2006-28.02.2023, X.shape(3848, 14), y.shape(3848,)), Test(01.03.2023-31.03.2023,
X.shape(21, 14), y.shape(21,)),
Ошибка: 2.22 %,  Валюта: usd_rub

Train(11.05.2006-31.03.2023, X.shape(3869, 14), y.shape(3869,)), Test(03.04.2023-28.04.2023,
X.shape(19, 14), y.shape(19,)),
Ошибка: 2.29 %,  Валюта: usd_rub

Train(11.05.2006-28.04.2023, X.shape(3888, 14), y.shape(3888,)), Test(02.05.2023-31.05.2023,
X.shape(20, 14), y.shape(20,)),
Ошибка: 1.73 %,  Валюта: usd_rub

Train(11.05.2006-31.05.2023, X.shape(3908, 14), y.shape(3908,)), Test(01.06.2023-30.06.2023,
X.shape(20, 14), y.shape(20,)),
Ошибка: 3.34 %,  Валюта: usd_rub

Train(11.05.2006-30.06.2023, X.shape(3928, 14), y.shape(3928,)), Test(03.07.2023-31.07.2023,
X.shape(20, 14), y.shape(20,)),
Ошибка: 3.42 %,  Валюта: usd_rub

Train(11.05.2006-31.07.2023, X.shape(3948, 14), y.shape(3948,)), Test(01.08.2023-31.08.2023,
X.shape(23, 14), y.shape(23,)),
Ошибка: 4.0 %,  Валюта: usd_rub

Train(11.05.2006-31.08.2023, X.shape(3971, 14), y.shape(3971,)), Test(01.09.2023-29.09.2023,
X.shape(20, 14), y.shape(20,)),
Ошибка: 0.24 %,  Валюта: usd_rub
```

```
Train(11.05.2006-29.09.2023, X.shape(3991, 14), y.shape(3991,)), Test(02.10.2023-31.10.2023,
X.shape(22, 14), y.shape(22,)),
Ошибка: 1.08 %,  Валюта: usd_rub

CPU times: total: 35min 8s
Wall time: 50min 43s
```

2.15.4 С шагом в неделю

Запустим кросс-валидацию данных data_loaded с применением модели rf_model, начиная с 2024 года, генерить тестовую и тренировочные выборки будем 10 раз, с шагом тестовой выборки 1 неделя, по каждой валюте отдельно из списка валют currency_list, сохранять результаты кросс-валидации будем в дата фрейм cv_info.

ячейка Jupiter Notebook

```
%%time
# Будем кросс-валидировать 10 раз по недели
cv_info = cross_validation_time_series(rf_model, data_loaded, 2024, 10, 'Week', currency_list,
cv_info)
```

В процессе кросс-валидации выводиться информация для Train и Test - диапазон дат выборки, размерность фичей и таргетов, ошибка предсказания рассчитывается по метрике mape описанной в пункте 2.11, так же выводиться наименование курса валюты.

результат выполнения

```
Train(11.05.2006-29.12.2023, X.shape(4054, 14), y.shape(4054,)), Test(03.01.2024-05.01.2024,
X.shape(3, 14), y.shape(3,)),
Ошибка: 1.61 %,  Валюта: chf_rub

Train(11.05.2006-05.01.2024, X.shape(4057, 14), y.shape(4057,)), Test(08.01.2024-12.01.2024,
X.shape(5, 14), y.shape(5,)),
Ошибка: 1.5 %,  Валюта: chf_rub

Train(11.05.2006-12.01.2024, X.shape(4062, 14), y.shape(4062,)), Test(16.01.2024-19.01.2024,
X.shape(4, 14), y.shape(4,)),
Ошибка: 1.33 %,  Валюта: chf_rub

Train(11.05.2006-19.01.2024, X.shape(4066, 14), y.shape(4066,)), Test(22.01.2024-26.01.2024,
X.shape(5, 14), y.shape(5,)),
Ошибка: 0.07 %,  Валюта: chf_rub

Train(11.05.2006-26.01.2024, X.shape(4071, 14), y.shape(4071,)), Test(29.01.2024-02.02.2024,
X.shape(5, 14), y.shape(5,)),
Ошибка: 0.99 %,  Валюта: chf_rub

Train(11.05.2006-02.02.2024, X.shape(4076, 14), y.shape(4076,)), Test(05.02.2024-09.02.2024,
X.shape(5, 14), y.shape(5,)),
Ошибка: 1.16 %,  Валюта: chf_rub

Train(11.05.2006-09.02.2024, X.shape(4081, 14), y.shape(4081,)), Test(12.02.2024-16.02.2024,
X.shape(5, 14), y.shape(5,)),
Ошибка: 0.11 %,  Валюта: chf_rub
```

```
Train(11.05.2006-16.02.2024, X.shape(4086, 14), y.shape(4086,)), Test(20.02.2024-22.02.2024,
X.shape(3, 14), y.shape(3,)),
Ошибка: 0.21 %,  Валюта: chf_rub

Train(11.05.2006-22.02.2024, X.shape(4089, 14), y.shape(4089,)), Test(26.02.2024-01.03.2024,
X.shape(5, 14), y.shape(5,)),
Ошибка: 0.17 %,  Валюта: chf_rub

Train(11.05.2006-01.03.2024, X.shape(4094, 14), y.shape(4094,)), Test(04.03.2024-07.03.2024,
X.shape(4, 14), y.shape(4,)),
Ошибка: 1.74 %,  Валюта: chf_rub

Train(11.05.2006-29.12.2023, X.shape(4054, 14), y.shape(4054,)), Test(03.01.2024-05.01.2024,
X.shape(3, 14), y.shape(3,)),
Ошибка: 1.48 %,  Валюта: cny_rub

Train(11.05.2006-05.01.2024, X.shape(4057, 14), y.shape(4057,)), Test(08.01.2024-12.01.2024,
X.shape(5, 14), y.shape(5,)),
Ошибка: 0.56 %,  Валюта: cny_rub

Train(11.05.2006-12.01.2024, X.shape(4062, 14), y.shape(4062,)), Test(16.01.2024-19.01.2024,
X.shape(4, 14), y.shape(4,)),
Ошибка: 1.75 %,  Валюта: cny_rub

Train(11.05.2006-19.01.2024, X.shape(4066, 14), y.shape(4066,)), Test(22.01.2024-26.01.2024,
X.shape(5, 14), y.shape(5,)),
Ошибка: 0.04 %,  Валюта: cny_rub

Train(11.05.2006-26.01.2024, X.shape(4071, 14), y.shape(4071,)), Test(29.01.2024-02.02.2024,
X.shape(5, 14), y.shape(5,)),
Ошибка: 0.72 %,  Валюта: cny_rub

Train(11.05.2006-02.02.2024, X.shape(4076, 14), y.shape(4076,)), Test(05.02.2024-09.02.2024,
X.shape(5, 14), y.shape(5,)),
Ошибка: 1.55 %,  Валюта: cny_rub

Train(11.05.2006-09.02.2024, X.shape(4081, 14), y.shape(4081,)), Test(12.02.2024-16.02.2024,
X.shape(5, 14), y.shape(5,)),
Ошибка: 0.26 %,  Валюта: cny_rub

Train(11.05.2006-16.02.2024, X.shape(4086, 14), y.shape(4086,)), Test(20.02.2024-22.02.2024,
X.shape(3, 14), y.shape(3,)),
Ошибка: 1.26 %,  Валюта: cny_rub

Train(11.05.2006-22.02.2024, X.shape(4089, 14), y.shape(4089,)), Test(26.02.2024-01.03.2024,
X.shape(5, 14), y.shape(5,)),
Ошибка: 0.53 %,  Валюта: cny_rub

Train(11.05.2006-01.03.2024, X.shape(4094, 14), y.shape(4094,)), Test(04.03.2024-07.03.2024,
X.shape(4, 14), y.shape(4,)),
Ошибка: 0.21 %,  Валюта: cny_rub

Train(11.05.2006-29.12.2023, X.shape(4054, 14), y.shape(4054,)), Test(03.01.2024-05.01.2024,
X.shape(3, 14), y.shape(3,)),
Ошибка: 0.61 %,  Валюта: eur_rub
```

```
Train(11.05.2006-05.01.2024, X.shape(4057, 14), y.shape(4057,)), Test(08.01.2024-12.01.2024,
X.shape(5, 14), y.shape(5,)),
Ошибка: 0.01 %,  Валюта: eur_rub

Train(11.05.2006-12.01.2024, X.shape(4062, 14), y.shape(4062,)), Test(16.01.2024-19.01.2024,
X.shape(4, 14), y.shape(4,)),
Ошибка: 1.43 %,  Валюта: eur_rub

Train(11.05.2006-19.01.2024, X.shape(4066, 14), y.shape(4066,)), Test(22.01.2024-26.01.2024,
X.shape(5, 14), y.shape(5,)),
Ошибка: 0.35 %,  Валюта: eur_rub

Train(11.05.2006-26.01.2024, X.shape(4071, 14), y.shape(4071,)), Test(29.01.2024-02.02.2024,
X.shape(5, 14), y.shape(5,)),
Ошибка: 0.28 %,  Валюта: eur_rub

Train(11.05.2006-02.02.2024, X.shape(4076, 14), y.shape(4076,)), Test(05.02.2024-09.02.2024,
X.shape(5, 14), y.shape(5,)),
Ошибка: 0.66 %,  Валюта: eur_rub

Train(11.05.2006-09.02.2024, X.shape(4081, 14), y.shape(4081,)), Test(12.02.2024-16.02.2024,
X.shape(5, 14), y.shape(5,)),
Ошибка: 0.23 %,  Валюта: eur_rub

Train(11.05.2006-16.02.2024, X.shape(4086, 14), y.shape(4086,)), Test(20.02.2024-22.02.2024,
X.shape(3, 14), y.shape(3,)),
Ошибка: 0.94 %,  Валюта: eur_rub

Train(11.05.2006-22.02.2024, X.shape(4089, 14), y.shape(4089,)), Test(26.02.2024-01.03.2024,
X.shape(5, 14), y.shape(5,)),
Ошибка: 0.63 %,  Валюта: eur_rub

Train(11.05.2006-01.03.2024, X.shape(4094, 14), y.shape(4094,)), Test(04.03.2024-07.03.2024,
X.shape(4, 14), y.shape(4,)),
Ошибка: 0.85 %,  Валюта: eur_rub

Train(11.05.2006-29.12.2023, X.shape(4054, 14), y.shape(4054,)), Test(03.01.2024-05.01.2024,
X.shape(3, 14), y.shape(3,)),
Ошибка: 1.24 %,  Валюта: gbp_rub

Train(11.05.2006-05.01.2024, X.shape(4057, 14), y.shape(4057,)), Test(08.01.2024-12.01.2024,
X.shape(5, 14), y.shape(5,)),
Ошибка: 0.56 %,  Валюта: gbp_rub

Train(11.05.2006-12.01.2024, X.shape(4062, 14), y.shape(4062,)), Test(16.01.2024-19.01.2024,
X.shape(4, 14), y.shape(4,)),
Ошибка: 0.08 %,  Валюта: gbp_rub

Train(11.05.2006-19.01.2024, X.shape(4066, 14), y.shape(4066,)), Test(22.01.2024-26.01.2024,
X.shape(5, 14), y.shape(5,)),
Ошибка: 0.09 %,  Валюта: gbp_rub

Train(11.05.2006-26.01.2024, X.shape(4071, 14), y.shape(4071,)), Test(29.01.2024-02.02.2024,
X.shape(5, 14), y.shape(5,)),
Ошибка: 1.08 %,  Валюта: gbp_rub
```

```
Train(11.05.2006-02.02.2024, X.shape(4076, 14), y.shape(4076,)), Test(05.02.2024-09.02.2024,
X.shape(5, 14), y.shape(5,)),
Ошибка: 0.96 %,  Валюта: gbp_rub

Train(11.05.2006-09.02.2024, X.shape(4081, 14), y.shape(4081,)), Test(12.02.2024-16.02.2024,
X.shape(5, 14), y.shape(5,)),
Ошибка: 1.07 %,  Валюта: gbp_rub

Train(11.05.2006-16.02.2024, X.shape(4086, 14), y.shape(4086,)), Test(20.02.2024-22.02.2024,
X.shape(3, 14), y.shape(3,)),
Ошибка: 1.05 %,  Валюта: gbp_rub

Train(11.05.2006-22.02.2024, X.shape(4089, 14), y.shape(4089,)), Test(26.02.2024-01.03.2024,
X.shape(5, 14), y.shape(5,)),
Ошибка: 0.74 %,  Валюта: gbp_rub

Train(11.05.2006-01.03.2024, X.shape(4094, 14), y.shape(4094,)), Test(04.03.2024-07.03.2024,
X.shape(4, 14), y.shape(4,)),
Ошибка: 0.52 %,  Валюта: gbp_rub

Train(11.05.2006-29.12.2023, X.shape(4054, 14), y.shape(4054,)), Test(03.01.2024-05.01.2024,
X.shape(3, 14), y.shape(3,)),
Ошибка: 0.22 %,  Валюта: jpy_rub

Train(11.05.2006-05.01.2024, X.shape(4057, 14), y.shape(4057,)), Test(08.01.2024-12.01.2024,
X.shape(5, 14), y.shape(5,)),
Ошибка: 0.17 %,  Валюта: jpy_rub

Train(11.05.2006-12.01.2024, X.shape(4062, 14), y.shape(4062,)), Test(16.01.2024-19.01.2024,
X.shape(4, 14), y.shape(4,)),
Ошибка: 0.95 %,  Валюта: jpy_rub

Train(11.05.2006-19.01.2024, X.shape(4066, 14), y.shape(4066,)), Test(22.01.2024-26.01.2024,
X.shape(5, 14), y.shape(5,)),
Ошибка: 0.53 %,  Валюта: jpy_rub

Train(11.05.2006-26.01.2024, X.shape(4071, 14), y.shape(4071,)), Test(29.01.2024-02.02.2024,
X.shape(5, 14), y.shape(5,)),
Ошибка: 0.29 %,  Валюта: jpy_rub

Train(11.05.2006-02.02.2024, X.shape(4076, 14), y.shape(4076,)), Test(05.02.2024-09.02.2024,
X.shape(5, 14), y.shape(5,)),
Ошибка: 0.83 %,  Валюта: jpy_rub

Train(11.05.2006-09.02.2024, X.shape(4081, 14), y.shape(4081,)), Test(12.02.2024-16.02.2024,
X.shape(5, 14), y.shape(5,)),
Ошибка: 0.13 %,  Валюта: jpy_rub

Train(11.05.2006-16.02.2024, X.shape(4086, 14), y.shape(4086,)), Test(20.02.2024-22.02.2024,
X.shape(3, 14), y.shape(3,)),
Ошибка: 0.95 %,  Валюта: jpy_rub

Train(11.05.2006-22.02.2024, X.shape(4089, 14), y.shape(4089,)), Test(26.02.2024-01.03.2024,
X.shape(5, 14), y.shape(5,)),
Ошибка: 0.34 %,  Валюта: jpy_rub
```

```
Train(11.05.2006-01.03.2024, X.shape(4094, 14), y.shape(4094,)), Test(04.03.2024-07.03.2024,
X.shape(4, 14), y.shape(4,)),
Ошибка: 0.89 %,  Валюта: jpy_rub

Train(11.05.2006-29.12.2023, X.shape(4054, 14), y.shape(4054,)), Test(03.01.2024-05.01.2024,
X.shape(3, 14), y.shape(3,)),
Ошибка: 2.5 %,  Валюта: usd_rub

Train(11.05.2006-05.01.2024, X.shape(4057, 14), y.shape(4057,)), Test(08.01.2024-12.01.2024,
X.shape(5, 14), y.shape(5,)),
Ошибка: 0.41 %,  Валюта: usd_rub

Train(11.05.2006-12.01.2024, X.shape(4062, 14), y.shape(4062,)), Test(16.01.2024-19.01.2024,
X.shape(4, 14), y.shape(4,)),
Ошибка: 1.25 %,  Валюта: usd_rub

Train(11.05.2006-19.01.2024, X.shape(4066, 14), y.shape(4066,)), Test(22.01.2024-26.01.2024,
X.shape(5, 14), y.shape(5,)),
Ошибка: 0.49 %,  Валюта: usd_rub

Train(11.05.2006-26.01.2024, X.shape(4071, 14), y.shape(4071,)), Test(29.01.2024-02.02.2024,
X.shape(5, 14), y.shape(5,)),
Ошибка: 0.52 %,  Валюта: usd_rub

Train(11.05.2006-02.02.2024, X.shape(4076, 14), y.shape(4076,)), Test(05.02.2024-09.02.2024,
X.shape(5, 14), y.shape(5,)),
Ошибка: 1.46 %,  Валюта: usd_rub

Train(11.05.2006-09.02.2024, X.shape(4081, 14), y.shape(4081,)), Test(12.02.2024-16.02.2024,
X.shape(5, 14), y.shape(5,)),
Ошибка: 0.29 %,  Валюта: usd_rub

Train(11.05.2006-16.02.2024, X.shape(4086, 14), y.shape(4086,)), Test(20.02.2024-22.02.2024,
X.shape(3, 14), y.shape(3,)),
Ошибка: 1.27 %,  Валюта: usd_rub

Train(11.05.2006-22.02.2024, X.shape(4089, 14), y.shape(4089,)), Test(26.02.2024-01.03.2024,
X.shape(5, 14), y.shape(5,)),
Ошибка: 0.57 %,  Валюта: usd_rub

Train(11.05.2006-01.03.2024, X.shape(4094, 14), y.shape(4094,)), Test(04.03.2024-07.03.2024,
X.shape(4, 14), y.shape(4,)),
Ошибка: 0.15 %,  Валюта: usd_rub

CPU times: total: 39min 36s
Wall time: 53min
```

2.16 Визуализация результатов кросс-валидации

Посмотрим на сколько ошибается модель машинного обучения в зависимости от длительности предсказания и валюты.

ячейка Jupiter Notebook

```
# Стиль легенды
legend_font = {
```

```
    "size": 7,
    "family": "serif",
}
# Стиль заголовков
title_font = {
    "fontsize": 12,
}

# Обновим список валют (на всякий случай)
currency_list = [column for column in data_loaded if '_' in column]

# Задаём размер полотна
plt.figure(figsize=[12, 9])

# Общий заголовок для всех графиков
plt.suptitle('Ошибка предсказаний при кросс-валидации',
               y=0.97,
               fontsize=19,
               fontweight='bold')

# Строим 4-е графика 2x2
for i, type_split_item in enumerate(cv_info['type_split'].unique()):
    plt.subplot(2, 2, i+1)
    # Заголовок для графика
    split_type_dict = {
        'Year': 'годам',
        'Quarter':'кварталам',
        'Month':'месяцам',
        'Week':'неделям'}
    plt.title(f'кросс-валидация по {split_type_dict[type_split_item]}', fontdict=title_font)
    # Задаём размер шривта и угол поворота текста для осей X и Y
    plt.xticks(fontsize=8, rotation=45)
    plt.yticks(fontsize=8, rotation=0)
    # Делаем размер шрифта по Y=8
    plt.ylabel('Ошибка в %', fontsize=8)
    # Задаём границы графика
    plt.ylim(-1.2, 30)
    # Задаём расстояние меду графиками
    plt.subplots_adjust(wspace=0.3, hspace=0.5)
    # Отрисовываем графики
    for currency_item in currency_list:
        # Создаём датафрем по условию валюты и типа разбиения (год, месяц и тд.)
        df_temp = cv_info.query(f'(currency == "{currency_item}") & (type_split ==
"{type_split_item}")')
        # Запишем переменные в x и y
        x = df_temp['date']
        y = df_temp['error']
        # Вставляем русский перевод если он есть, или оригинал, если перевода нет
        legend_text = explanations_col[currency_item] if explanations_col.get(currency_item)
else currency_item
        # Отрисовываем график и легенду
        plt.plot(x, y, label=legend_text)
        # Выставляем стиль и положение легенды
        plt.legend(loc="upper right", prop=legend_font)

# Вывести графики на экран
plt.show()
```

результат выполнения

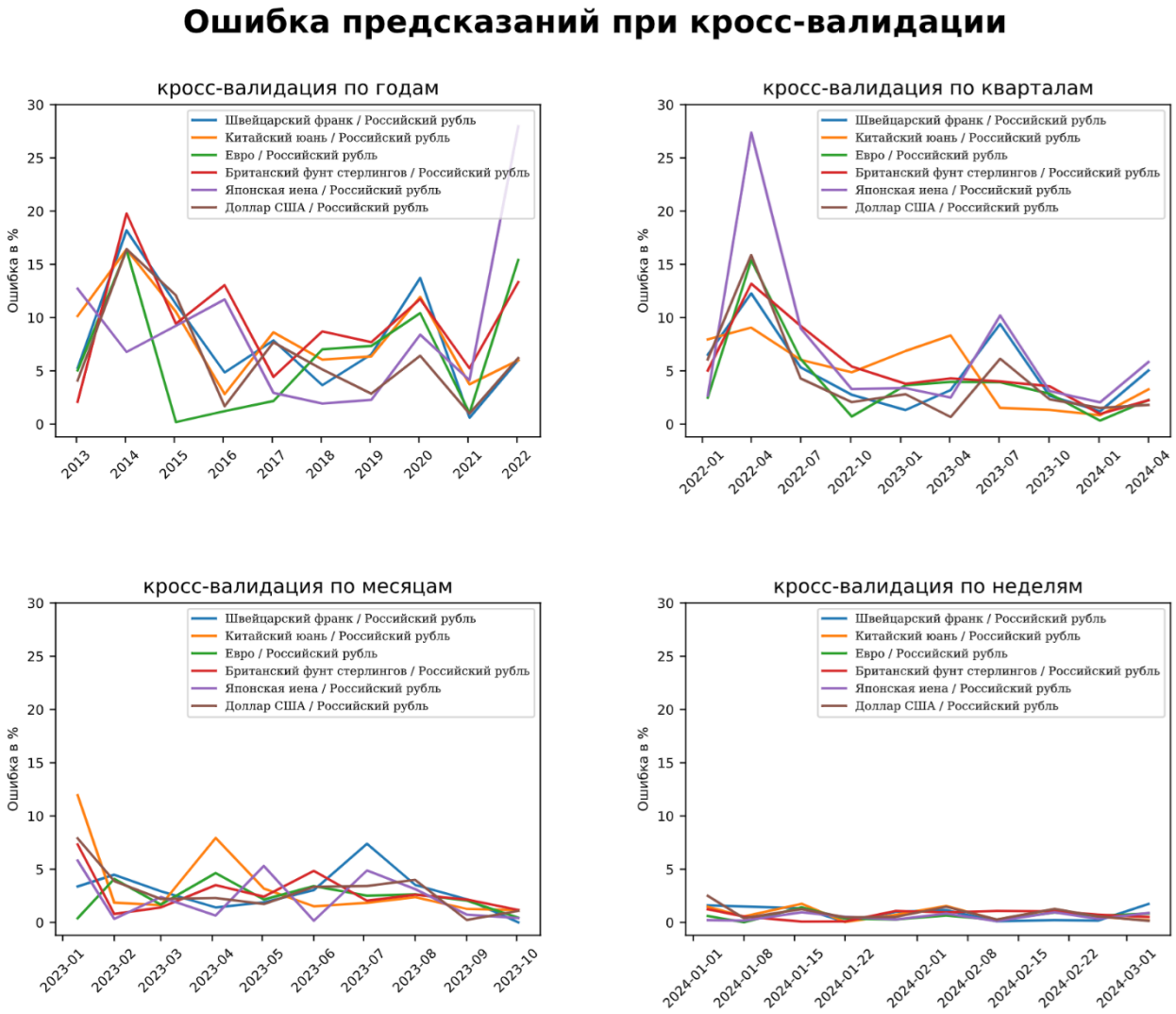

Рисунок 7. График результатов кросс-валидации

Из графика видно, что ошибка предсказания курса валют резко возрастает, когда наблюдается сложная политическая обстановка (2014 и 2022 года). Если добавить признаки, которые отражают политическую обстановку, качество прогноза должно улучшиться. Чем короче период предсказания, тем меньше ошибка предсказания. Лучшие результаты предсказания курса валют получается если предсказывать на неделю.

2.17 Важность признаков модели

Вычислим важность признаков для всех валют

ячейка Jupiter Notebook

```
%%time
# Словарь важности признаков
importance_of_signs = {}

# Пробежимся по всем валютам
for currency_item in currency_list:
    # Проитерируем функцию-генератор
    iterator = iter(split_on_time_series(data_loaded, 2024, 1, 'Year', currency_item))
    X_train, y_train, X_test, y_test, _, _ = next(iterator)
    # Объединим все данные в общий дата фрейм и отсортируем по индексу (индекс - это дата)
    X_train = pd.concat([X_train, X_test]).sort_index(ascending=False)
    y_train = pd.concat([y_train, y_test]).sort_index(ascending=False)
    # Обучим модель на всех данных за весь период
    rf_model.fit(X_train, y_train)
    # Добавим в словарь важность признаков (ключ-валюта, значение-важность признаков)
    importance_of_signs[currency_item] = rf_model.feature_importances_
```

результат выполнения

```
CPU times: total: 4min 8s
Wall time: 5min 28s
```

ячейка Jupiter Notebook

```
# Посмотрим какие признаки в X_train
X_train
```

результат выполнения

Дата	s-p-500	золото-(банк-россии)	индекс-мосбиржи	нефть-brent	палладий-(банк-россии)	платина-(банк-россии)	ртс	серебро-(банк-россии)	Year	Month	Weekday	Day	Week	Quarter
2024-07-09	5576.98	6743.87	3054.07	84.66	2908.39	2899.89	1093.26	86.68	2024	7	1	9	28	3
2024-07-08	5572.85	6683.46	3132.58	85.64	2904.44	2867.60	1119.25	85.96	2024	7	0	8	28	3
2024-07-05	5567.19	6690.03	3149.29	86.97	2954.96	2835.97	1125.66	85.36	2024	7	4	5	27	3
2024-07-03	5537.02	6589.05	3203.07	86.63	2795.06	2786.57	1147.28	82.73	2024	7	2	3	27	3
2024-07-02	5509.01	6542.06	3217.29	86.24	2728.08	2840.35	1151.82	82.43	2024	7	1	2	27	3
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2006-05-17	1270.32	614.86	1343.86	69.04	307.85	1070.25	1543.26	11.23	2006	5	2	17	20	2
2006-05-16	1292.08	588.65	1403.95	70.34	290.90	1024.57	1585.35	11.28	2006	5	1	16	20	2
2006-05-15	1294.50	596.76	1398.20	69.67	301.81	1035.11	1589.46	12.67	2006	5	0	15	20	2
2006-05-12	1291.24	625.53	1483.62	72.32	322.11	1069.70	1681.07	12.25	2006	5	4	12	19	2
2006-05-11	1305.92	612.36	1534.51	73.43	314.03	1025.84	1723.97	12.23	2006	5	3	11	19	2

4178 rows × 14 columns

Рисунок 8. Дата фрейм с тренировочными данными

Отобразим важность признаков на графике

ячейка Jupiter Notebook

```
# Отрисуем графики важности для всех валют
# Стиль заголовков
title_font = {
    "fontsize": 10,
}

# Задаём размер полотна
plt.figure(figsize=[13, 10])

# Рассчитаем количество строк в графике
n_row = ceil(len(currency_list)/2)

# Общий заголовок для всех графиков
plt.suptitle('Важность признаков для курса валюты',
               y=0.97,
               fontsize=19,
               fontweight='bold')

# Получим список всех фичей
feature_names = X_train.columns.values.tolist()

# Строим графики nx2
for i, key in enumerate(importance_of_signs):
    plt.subplot(n_row, 2, i+1)
    # Вставляем русский перевод если он есть, или оригинал, если перевода нет
    title_text = explanations_col[key] if explanations_col.get(key) else key
    plt.title(title_text, fontdict=title_font)
    # 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 X=8 и задаём название оси
    plt.xlabel('Важность признаков', fontsize=8)
    # Задаём расстояние меду графиками
    plt.subplots_adjust(wspace=0.6, hspace=0.45)
    # Отрисовываем графики
    plt.barh(feature_names, importance_of_signs[key])

# Вывести графики на экран
plt.show()
```

результат выполнения

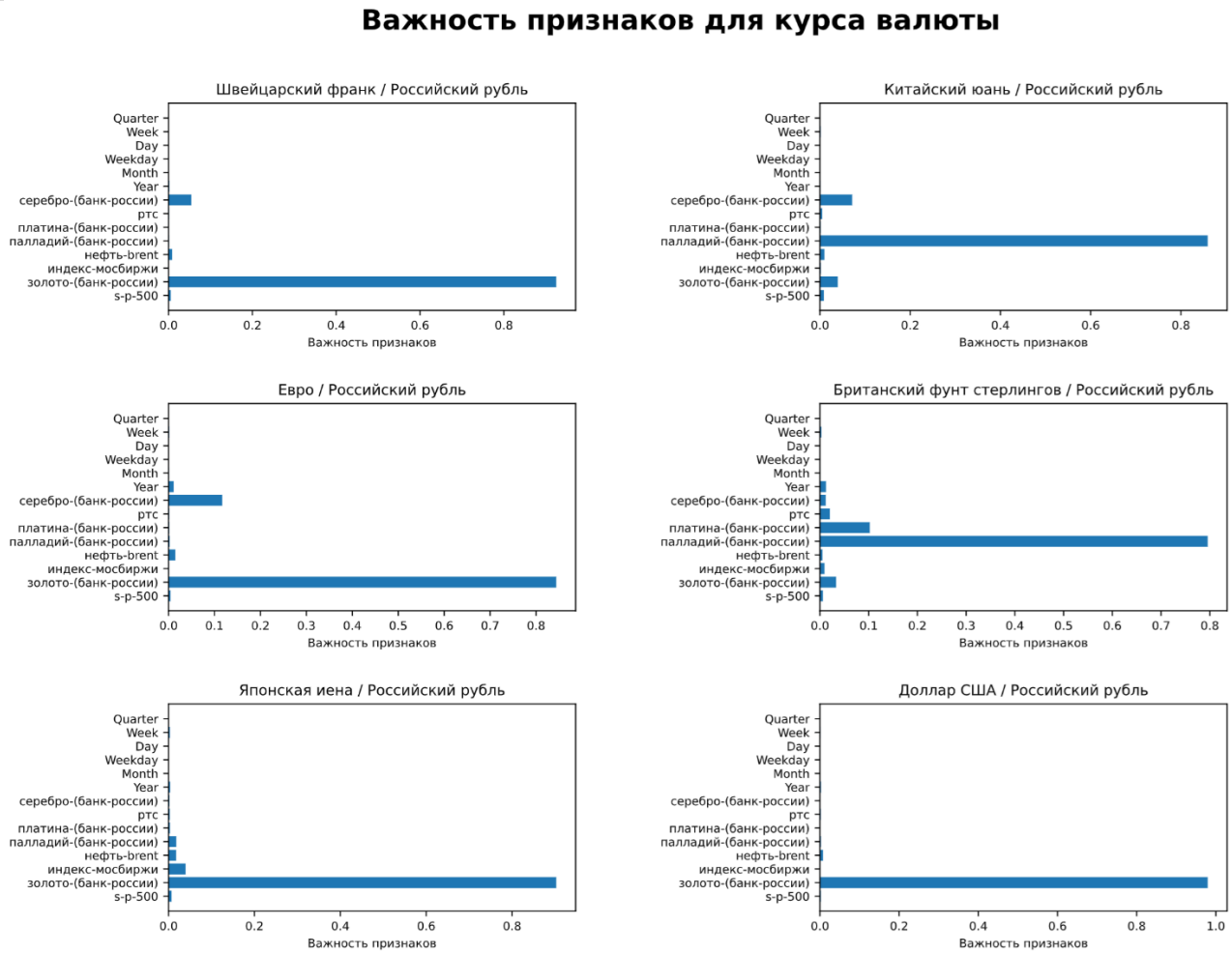

Рисунок 9. Важность признаков, влияющих на курс валюты

На курс каждой валюты влияют свои признаки, но основными признаками влияющие на курс всех валют, является стоимость на драгоценные металлы.

2.18 Средняя ошибка предсказаний модели в зависимости от длительности периода предсказаний

ячейка Jupiter Notebook

```
# Посмотрим, что записалось при кросс-валидации
cv_info
```

результат выполнения

	date	currency	type_split	error
0	2013-01-10	chf_rub-(банк-россии)	Year	5.29
1	2014-01-10	chf_rub-(банк-россии)	Year	18.19
2	2015-01-13	chf_rub-(банк-россии)	Year	11.26
3	2016-01-12	chf_rub-(банк-россии)	Year	4.85
4	2017-01-10	chf_rub-(банк-россии)	Year	7.86
...	...	...	...	...
235	2024-02-05	usd_rub-(банк-россии)	Week	1.46
236	2024-02-12	usd_rub-(банк-россии)	Week	0.29
237	2024-02-20	usd_rub-(банк-россии)	Week	1.27
238	2024-02-26	usd_rub-(банк-россии)	Week	0.57
239	2024-03-04	usd_rub-(банк-россии)	Week	0.15

240 rows × 4 columns

Рисунок 10. Дата фрейм с результатами кросс-валидации

Посмотрим среднее значение из 10 предсказаний в зависимости от валюты и длительности предсказаний.

ячейка Jupiter Notebook

```
# Сгруппируем средние значения в зависимости от типа разбиения и валюты
cv_info_mean = cv_info.groupby(['type_split', 'currency'],
as_index=False).agg({'error':'mean'}).rename(columns={'error':'error_mean'}).sort_values(by='er
ror_mean')
cv_info_mean
```

результат выполнения

	type_split	currency	error_mean
16	Week	jpy_rub-(банк-россии)	0.53
14	Week	eur_rub-(банк-россии)	0.60
15	Week	gbp_rub-(банк-россии)	0.74
13	Week	cny_rub-(банк-россии)	0.84
12	Week	chf_rub-(банк-россии)	0.89
17	Week	usd_rub-(банк-россии)	0.89
4	Month	jpy_rub-(банк-россии)	2.38
2	Month	eur_rub-(банк-россии)	2.40
3	Month	gbp_rub-(банк-россии)	2.83
5	Month	usd_rub-(банк-россии)	3.01
0	Month	chf_rub-(банк-россии)	3.02
1	Month	cny_rub-(банк-россии)	3.48
8	Quarter	eur_rub-(банк-россии)	4.17
11	Quarter	usd_rub-(банк-россии)	4.35
6	Quarter	chf_rub-(банк-россии)	4.97
7	Quarter	cny_rub-(банк-россии)	5.01
9	Quarter	gbp_rub-(банк-россии)	5.17
23	Year	usd_rub-(банк-россии)	6.36
20	Year	eur_rub-(банк-россии)	6.62
10	Quarter	jpy_rub-(банк-россии)	6.94
18	Year	chf_rub-(банк-россии)	7.80
19	Year	cny_rub-(банк-россии)	8.27
22	Year	jpy_rub-(банк-россии)	8.80
21	Year	gbp_rub-(банк-россии)	9.55

Рисунок 11. Дата фрейм со средними значениями ошибки предсказания, сгруппированный по длительности предсказания и валюты

ячейка Jupiter Notebook

```
# Отрисуем графики важности для всех валют
# Стиль заголовков
title_font = {
    "fontsize": 10,
}

# Задаём размер полотна
plt.figure(figsize=[13, 10])

# Общий заголовок для всех графиков
plt.suptitle('Средняя ошибка предсказаний модели \n в зависимости от длительности
предсказаний',
               y=1,
               fontsize=19,
               fontweight='bold')

# Рассчитаем количество строк в графике
n_row = ceil(len(currency_list)/2)

# Строим графики nx2
for i, currency in enumerate(currency_list):
    plt.subplot(n_row, 2, i+1)
    # Вставляем русский перевод если он есть, или оригинал, если перевода нет
    title_text = explanations_col[currency] if explanations_col.get(currency) else currency
    plt.title(title_text, fontdict=title_font)
    # 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 X=8 и задаём название оси
    plt.xlabel('Средняя ошибка предсказаний в %', fontsize=8)
    # Задаём расстояние меду графиками
    plt.subplots_adjust(wspace=0.6, hspace=0.6)

    # Отрисовываем графики
    container = plt.barh(cv_info_mean.query(f'currency == "{currency}"')['type_split'],
round(cv_info_mean.query(f'currency == "{currency}"')['error_mean'], 2))

    plt.bar_label(container, padding=-10, color='black')

# Вывести графики на экран
plt.show()
```

результат выполнения

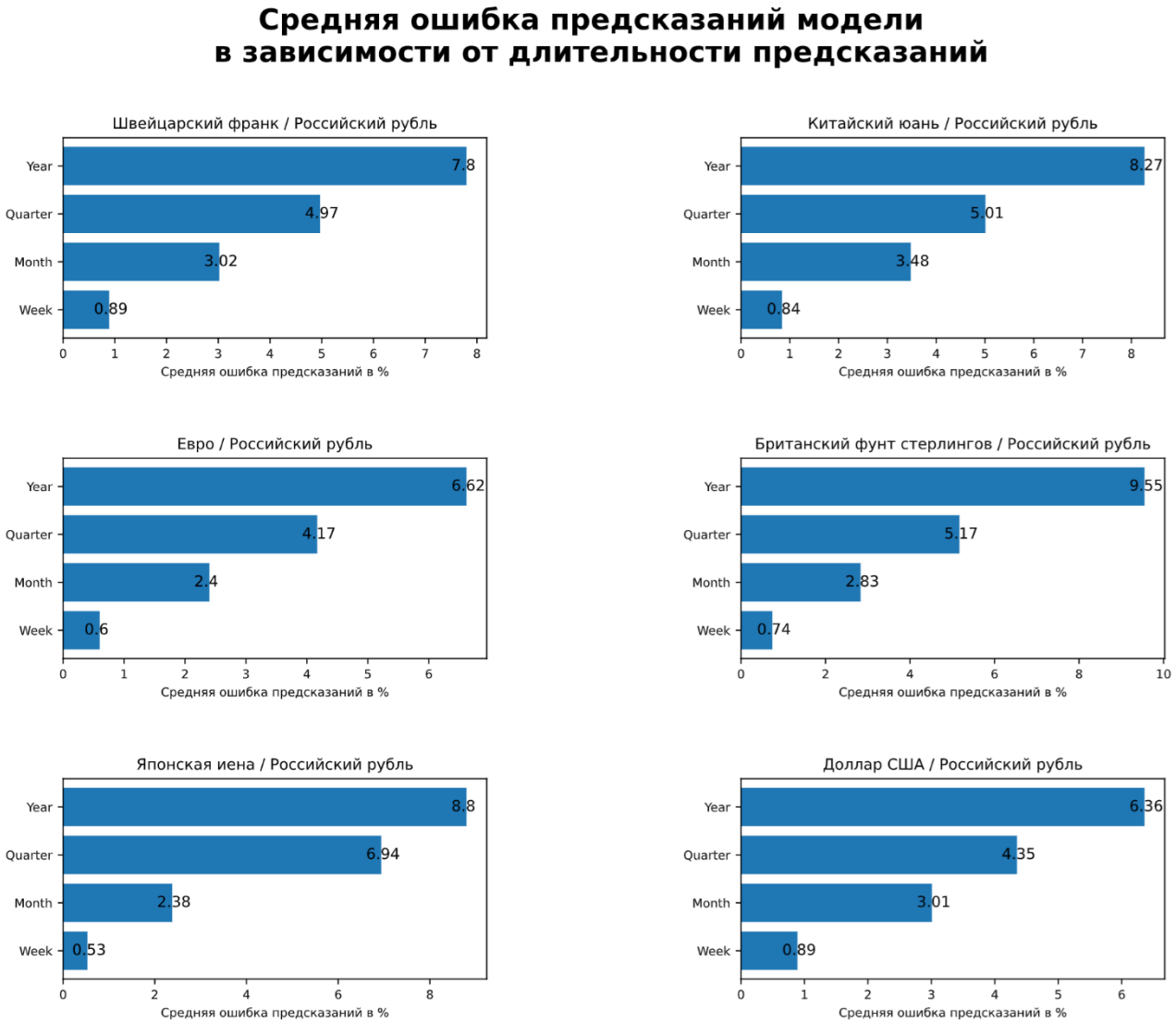

Рисунок 12. Среднее значение ошибки предсказания курса валюты в зависимости от длительности предсказания и валюты

Ошибка предсказания курса валюты меняется, в зависимости от продолжительности предсказаний и самой валюты.

Посмотрим среднею ошибку предсказаний по всем валютам.

ячейка Jupiter Notebook

```
# Сгруппируем средние значения в зависимости от типа разбиения
cv_general_info_mean = cv_info.groupby(['type_split'],
as_index=False).agg({'error':'mean'}).rename(columns={'error':'error_mean'}).sort_values(by='er
ror_mean')

# Отрисуем график средней ошибки в зависимости от продолжительности предсказаний
# Стиль заголовков
title_font = {
    "fontsize": 14,
}

# Задаём размер полотна
plt.figure(figsize=[7, 5])
```

```
# Заголовок графика
plt.title('Средняя ошибка предсказаний курса валют', y=1, fontdict=title_font)
# 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 X=8 и задаём название оси
plt.xlabel('Средняя ошибка предсказаний в %', fontsize=8)

# Отрисовываем графики
container = plt.barh(cv_general_info_mean['type_split'],
round(cv_general_info_mean['error_mean'], 2))

plt.bar_label(container, padding=-10, color='black')

# Вывести график на экран
plt.show()
```

результат выполнения

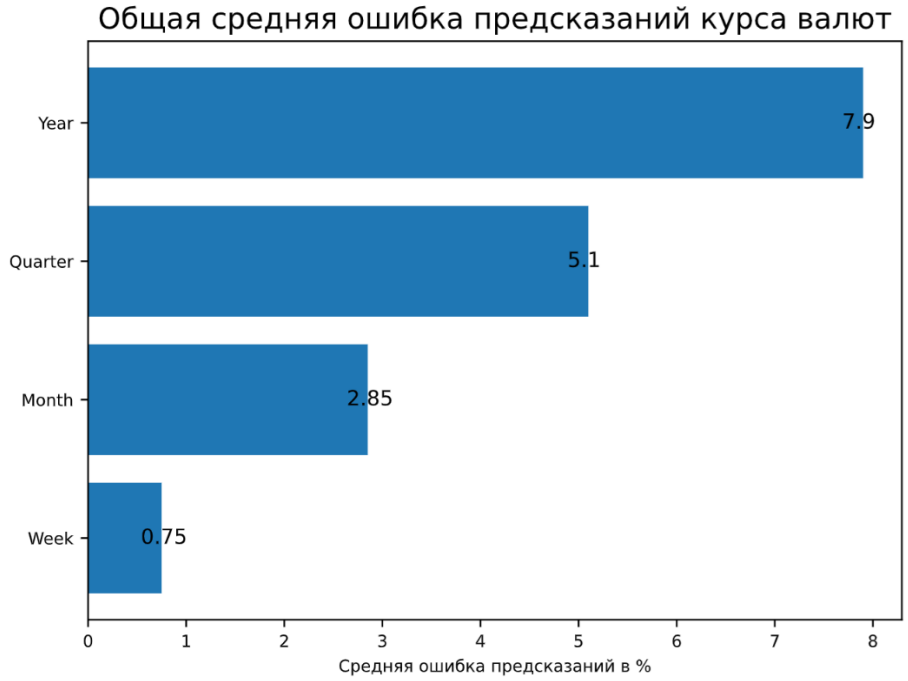

Рисунок 13. Общая средняя ошибка предсказаний курса всех валют

В среднем курсы валют хорошо предсказываются на короткий период и с увеличением периода предсказания, точность предсказаний снижается.

2.19 Автоматизация процесса

Для добавления/удаления признаков (фич) или целевых переменных (таргетов) не потребуется изменять код ноутбука, для этого нужно добавить/удалить датасет в директории откуда производится чтение датасетов (пункт 2.6).

На графиках возможно потребуется скорректировать размер полотна, для лучшего отображения. Добавление новых графиков для каждого таргета и фичи происходит в автоматическом режиме (пункты: 2.7.6, 2.16, 2.17, 2.18).

В автоматическом режиме на графиках также будут добавляться более понятные название таргетов (пункт 2.7.5)

Заключение

Модель машинного обучения RandomForestRegressor обученная на индикаторах фондового рынка:

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

И дополнительных признаках:

- год;
- квартал;
- месяц;
- неделя;
- день недели (пн.-вс.);
- день;

Предсказывая курсы валют:

- доллар США / Российский рубль;
- Евро / Российский рубль;
- Британский фунт стерлингов / Российский рубль;
- Швейцарский франк / Российский рубль;
- Китайский юань / Российский рубль;
- Японская иена / Российский рубль;

В среднем за 10 предсказаний при разной продолжительности прогноза, ошибается:

- ошибка прогноза среднего курса за неделю – 0.75%
- ошибка прогноза среднего курса за месяц – 2.85%
- ошибка прогноза среднего курса за квартал – 5.1%

- ошибка прогноза среднего курса за год – 7.9%

Рисунок 13 демонстрирует среднее значение ошибки предсказаний курса для всех валют.

Если рассматривать среднее значение ошибки предсказаний курса для каждой валюты отдельно, то данная информация показана на рисунок 12.

Признаки, оказывающие влияние на курс каждой валюты показаны на рисунок 9. Важность признаков для каждой валюты различается. По мнению модели особую важность влияющею на курс валюты оказывают цены на драгоценные металлы, цены на нефть на курс валюты влияют не так сильно.

Динамику ошибки предсказаний в зависимости от даты можно посмотреть на рисунок 7. Из этого рисунка видно, что ошибка предсказания курса валют зависит и от других факторов, например, 2014 и 2022 года возрастает ошибка прогноза, когда наблюдается сложная политическая обстановка. Если добавить признаки, которые отражают политическую обстановку, качество прогноза должно улучшиться.

Так же нужно отметить, чем короче период предсказания, тем меньше ошибка. Лучшие результаты предсказания курса валют получается если предсказывать на неделю.

Практическая значимость работы состоит в том, что, эмулируя нужные нам признаки мы можем спрогнозировать курс интересующей нас валюты на заданный промежуток времени. Например, изменим цену на палладий за неделю мы спрогнозируем какая, может быть, средняя цена разных валют за этот промежуток времени при изменении цены на этот драгоценный металл.

С данным ноутбуком можно проверять влияние на курс валюты различных признаков (фич), а также экспериментировать с другими валютами в качестве таргета, для этого не нужно изменять код программы, достаточно просто скопировать выбранные датасеты в директорию откуда производится чтение (пункт 2.19). Добавление новых фич и таргетов на графики также будет происходить в автоматическом режиме (пункт 2.19).

Следует заметить, что время выполнения этого ноутбука на обычном ПК занимает несколько часов. Для ускорения процесса вычисления можно воспользоваться облачными вычислениями [11].

Данный ноутбук и применённые в нём датасеты можно скачать с GitHub [8] (https://github.com/alexeydrygin/Diplom_Gb_2024)

Список используемой литературы

1. Е. Я. Волков // Факторы, влияющие на формирование курса рубля. – 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: 21.09.2024).

2. А. Мареев // Zotero-style-for-Gost-R-7.0.100-2018. – URL: https://github.com/ArtemMareev/Zotero-style-for-Gost-R-7.0.100-2018?tab=readme-ov-file (дата обращения: 20.09.2024).

3. Прикладная статистика // Кросс валидация для временных рядов в python. – URL: https://www.youtube.com/watch?v=l2vhtyWp_ck (дата обращения: 21.09.2024).

4. Е. В. Сангаджиева, Г. А. Оргдаева // Особенности методов машинного обучения для прогнозирования курса валют. – URL: https://files.scienceforum.ru/pdf/2020/5e04936d95941.pdf (дата обращения: 21.09.2024).

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

6. А. Дрыгин // ДЗ «Погружение в Python». Семинар 7. Файлы и файловая система. – URL: https://github.com/alexeydrygin/python_2/blob/main/seminar_7/file_func/task_home_work.py (дата обращения: 21.09.2024).

7. А. Дрыгин // Стиль для программы Zotero. – URL: https://github.com/alexeydrygin/tree/main/Zotero%20csl (дата обращения: 21.09.2024).

8. А. Дрыгин // Diplom-2024. – URL: https://github.com/alexeydrygin/ (дата обращения: 21.09.2024).

9. ALEKSANDR // Cross-Validation на временном ряду. – URL: https://otus.ru/nest/post/483/ (дата обращения: 21.09.2024).

10. Anaconda. – URL: https://www.anaconda.com/ (дата обращения: 19.09.2024).

11. Cloud.ru // Платформа Cloud.ru ML Space. – URL: https://cloud.ru/mlspace (дата обращения: 21.09.2024).

12. dfedorov.spb.ru // 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: 21.09.2024).

13. InvestFunds // Ключевые индикаторы. – URL: https://investfunds.ru/indicators/key-indexes/ (дата обращения: 20.09.2024).

14. InvestFunds // Курсы валют. – URL: https://investfunds.ru/indicators/currency/ (дата обращения: 20.09.2024).

15. JupiterLab // Debugger. – URL: https://jupyterlab.readthedocs.io/en/4.1.x/user/debugger.html (дата обращения: 19.09.2024).

16. Microsoft // Visual Studio Code. – URL: https://code.visualstudio.com/ (дата обращения: 20.09.2024).

17. Microsoft // Word. – URL: https://www.microsoft.com/ru-ru/microsoft-365/word?market=ru (дата обращения: 20.09.2024).

18. Pandas // pandas.DataFrame.resample. – URL: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.resample.html (дата обращения: 21.09.2024).

19. Pandas // Time series / date functionality. – URL: https://pandas.pydata.org/docs/user_guide/timeseries.html (дата обращения: 21.09.2024).

20. Scikit-Learn // RandomForestRegressor. – URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html (дата обращения: 24.09.2024).

21. Scikit-Learn // TimeSeriesSplit. – URL: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.TimeSeriesSplit.html (дата обращения: 21.09.2024).

22. Space G // Оформление списка литературы по ГОСТ Р 7.0.100-2018 при помощи Зотеро. – URL: https://www.youtube.com/watch?v=oT0ZpKnhUtI (дата обращения: 21.09.2024).

23. Zotero. – URL: https://www.zotero.org/ (дата обращения: 21.09.2024).