МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение

высшего образования

**«Северный (Арктический) федеральный университет имени М.В. Ломоносова»**

Высшая школа информационных технологий и автоматизированных систем

(наименование высшей школы / филиала / института)

**КУРСОВОЙ ПРОЕКТ**

|  |  |  |  |
| --- | --- | --- | --- |
| По дисциплине | | | Глубокое обучение. Продвинутый уровень (модуль) |
|  | | | |
| На тему | | Прогнозирование погоды на основе годов-гомологов | |
|  | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ф.И.О.  обучающихся | Наименование направления подготовки / специальности | Курс | Группа | Ф.И.О. руководителя (-ей) должность / уч. степень / звание |
| Сухих Алексей Михайлович | 09.04.02 Информационные системы и технологии | 2 | 151266 | Васендина Ирина Сергеевна, доцент, к.т.н. |
| Терюхин Борис Васильевич | 09.04.02 Информационные системы и технологии | 2 | 151266 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Признать, что проект выполнен и защищён с отметкой |  |  |  |  |
|  | |  | (отметка прописью) |  | (дата) |
| Руководитель | |  |  |  |  |
|  | |  | (подпись руководителя) |  | (инициалы, фамилия) |

Архангельск 2024

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ | | | | | | | | | | | | | | | | | | | | | |
| федеральное государственное автономное образовательное учреждение | | | | | | | | | | | | | | | | | | | | | |
|  | высшего образования  **«Северный (Арктический) федеральный университет имени М.В. Ломоносова»** | | | | | | | | | | | | | | | | | | |  | |
|  |  | | | | | | | | | | | | | | | | | | |  | |
|  | Высшая школа информационных технологий и автоматизированных систем | | | | | | | | | | | | | | | | | | |  | |
|  | (наименование высшей школы / филиала / института / колледжа) | | | | | | | | | | | | | | | | | | |  | |
|  |  | | | | | | | | | | | | | | | | | | |  | |
|  | **ЗАДАНИЕ НА КУРСОВОЙ ПРОЕКТ** | | | | | | | | | | | | | | | | | | |  | |
|  |  | | | | | | | | | | | | | | | | | | |  | |
|  | ппо | Глубокое обучение. Продвинутый уровень (модуль) | | | | | | | | | | | | | | | | | |  | |
|  |  | | | | (наименование дисциплины) | | | | | | | | | | | | | | |  | |
|  | студенту | | | ВШИТиАС | высшей школы | | | | 2 | | | | | курса | 151266 | | | | группы |  | |
|  | Сухих Алексей Михайлович | | | | | | | | | | | | | | | | | | |  | |
|  | (фамилия, имя, отчество студента) | | | | | | | | | | | | | | | | | | |  | |
|  | |  | | --- | | 09.04.02 «Информационные системы и технологии» | |  | | Терюхин Борис Васильевич | | (фамилия, имя, отчество студента) | | |  | | --- | | 09.04.02 «Информационные системы и технологии» | | (код и наименование направления подготовки/специальности) | | | | | | | | | | | | | | | | | | | | |  | |
|  | ТЕМА: | | Прогнозирование погоды на основе годов-гомологов | | | | | | | | | | | | | | | | |  | |
|  |  | | | | | | | | | | | | | | | | | | |  | |
|  |  | | | | | | | | | | | | | | | | | | |  | |
|  | ИСХОДНЫЕ ДАННЫЕ: | | | | | Архивные данные погоды с ресурса «Погодные сервисы» | | | | | | | | | | | | | |  | |
|  | (http://pogoda-service.ru/) | | | | | | | | | | | | | | | | | | |  | |
|  | 1. Поиск и подготовка данных | | | | | | | | | | | | | | | | | | |  | |
|  | 1. Выявление годов-гомологов | | | | | | | | | | | | | | | | | | |  | |
|  | 1. Предсказание погодных условий | | | | | | | | | | | | | | | | | | |  | |
|  | 1. Разработка программы | | | | | | | | | | | | | | | | | | |  | |
|  |  | | | | | | | | | | | | | | | | | | |  | |
|  |  | | | | | | | | | | | | | | | | | | |  | |
|  |  | | | | | | | | | | | | | | | | | | |  | |
|  |  | | | | | | | | | | | | | | | | | | |  | |
|  |  | | | | | | | | | | | | | | | | | | |  | |
|  |  | | | | | | | | | | | | | | | | | | |  | |
|  |  | | | | | | | | | | | | | | | | | | |  | |
|  | Руководитель работы | | | | | | | доцент | | |  | |  | | | |  | И.С. Васендина | |  | |
|  |  | | | |  | | (должность) | | |  | | (подпись) | | | |  | | (инициалы, фамилия) | | |

Архангельск 2024

ЛИСТ ДЛЯ ЗАМЕЧАНИЙ

ОГЛАВЛЕНИЕ

[Введение 5](#_Toc164918349)

[1 Поиск и подготовка данных 6](#_Toc164918350)

[2 Выявление годов-гомологов 12](#_Toc164918351)

[3 Предсказание погодных условий 22](#_Toc164918352)

[4 Разработка программы 31](#_Toc164918353)

[Заключение 35](#_Toc164918354)

[Список использованных источников 36](#_Toc164918355)

[Приложение А (обязательное) Листинг программного кода 37](#_Toc164918356)

## ВВЕДЕНИЕ

Согласно распоряжению от 1 августа 2022 года №2115-р Правительством Российской Федерации утвержден план по развитию Северного морского пути до 2035 года [1]. Он состоит из мероприятий, направленных на освоение ресурсных месторождений, организацию регулярных транспортных рейсов, проведение геологоразведочных работ, развитие инфраструктуры, строительство морских комплексов и терминалов, а также на обеспечение предоставления радиолокационных данных дистанционного зондирования Земли Северного морского пути с помощью создания космической системы в составе нескольких космических аппаратов. Одной из проблем для реализации данного плана и поставленных целей являются замерзающие моря и меняющиеся погодные условия, влияющие на судоходство в регионе. Исходя из этого встает вопрос о необходимости создания и использования современных методов анализа и прогнозирования погоды.

Цель проекта – улучшить предсказание погодных условий посредством комбинирования методов прогнозирования временных рядов и результатов поиска годов-гомологов.

В ходе проекта были выполнены следующие задачи:

* Сбор и подготовка исторических данных;
* Выявление годов-гомологов;
* Реализация предсказания погодных условий;
* Разработка интерфейса программы.

## **1 ПОИСК И ПОДГОТОВКА ДАННЫХ**

Одной из основных составляющих, необходимых для поиска годов-гомологов и предсказания будущих погодных условий, являются данные. Для получения более точных результатов требуется использовать как можно большее количество наблюдений за продолжительный период времени с достаточным количеством параметров.

Данные о погоде для поиска годов-гомологов и обучения предсказательных моделей брались с публичного ресурса с архивными данными «Погодные сервисы», раздел «Архив погоды c 1929 года» [2].

Данный архив включает в себя наблюдения по следующим параметрам:

— максимальная температура воздуха за сутки;

— минимальная температура воздуха за сутки;

— средняя температура воздуха за сутки;

— среднее за сутки атмосферное давление на уровне станции;

— средняя за сутки скорость ветра;

— суточное количество атмосферных осадков;

— эффективная температура (расчётная величина, которая является функцией температуры воздуха, влажности воздуха, скорости ветра и, в меньшей степени, атмосферного давления).

Архив погоды за отрезок 1929–2023 годы дает возможность получить данные по более чем 17000 станциям в более чем 240 странах мира. Данный ресурс также позволяет провзаимодействовать с интерактивной картой, где можно посмотреть примерное расположение всех метеостанций и их названия. Скриншот данной карты представлен на рисунке 1.

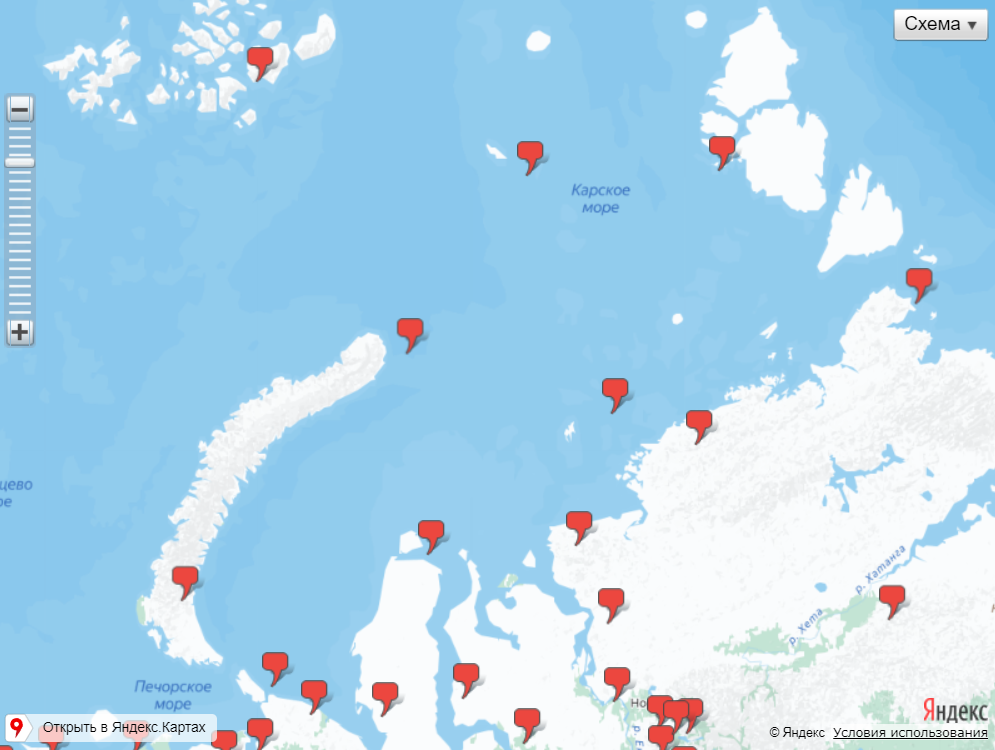


Рисунок 1 — Карта метеостанций

Пример вывода данных одной из станций на сайте представлен на рисунке 2.

Изображение выглядит как текст, число, Параллельный

Автоматически созданное описание

Рисунок 2— Пример данных

Для проекта были отобраны 14 метеорологических станций, находящихся в зоне нашего интереса (Карского моря). За период с 1 января по 31 октября 2023 года из всех морей Северного морского пути Карское море было наиболее востребованным для ледовых прогнозов [3]. Общая информация по данным станциям представлена в таблице 1.

Таблица 1 — Метеорологические станции

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Станция | Первое наблюдение | Последнее наблюдение | Широта | Долгота | Нет ни одного наблюдения |
| Остров Визе | 10.06.1955 | 31.12.2023 | 79.491 | 76.981 | 1972, 2022 |
| Голомянный | 18.01.1948 | 31.12.2023 | 79.550 | 90.567 | 1950–1954, 1972 |
| Мыс Стерлигова | 26.03.1955 | 31.12.2023 | 75.417 | 88.900 | 1972, 2002–2011 |
| Остров Тройной | 14.07.1955 | 31.12.2023 | 75.952 | 82.946 | 1972, 1995–2011 |
| Остров Диксон | 12.10.1933 | 31.12.2023 | 73.500 | 80.400 | 1937–1947, 1972 |
| Остров Белый | 01.01.1937 | 31.12.2023 | 73.333 | 70.050 | 1938–1947, 1972 |
| Мыс Желания | 16.04.1955 | 29.01.2015 | 76.950 | 68.550 | 1972, 1997–2009 |
| Марресаля | 16.01.1934 | 31.12.2023 | 69.717 | 66.800 | 1937–1947, 1972 |
| Амдерма | 11.01.1948 | 31.12.2023 | 69.783 | 61.783 | 1972 |
| Мыс Болванский Нос | 14.04.1955 | 31.12.2023 | 70.447 | 59.091 | 1972, 2002–2011 |
| Сеяха | 09.07.1957 | 31.12.2023 | 70.170 | 72.514 | 1972, 2002–2011 |
| Мыс Челюскин | 03.02.1934 | 31.12.2023 | 77.717 | 104.300 | 1938–1947, 1972 |
| Мыс Сопочная Карга | 17.04.1955 | 31.12.2023 | 71.875 | 82.706 | 1972, 1999–2000, 2002–2011 |
| Остров Хейса | 02.12.1957 | 31.12.2023 | 80.617 | 58.050 | 1972, 2002–2003 |

Данные со всех выбранных станций были собраны в отдельные excel-файлы, в дальнейшем преобразованные в формат csv.

В общем итоге со всех станций было получено 287104 наблюдения. Графики временных рядов погодных параметров станции Амдерма представлена на рисунке 3.

На графиках видно, что все временные ряды имеют пропуски данных в начале 1970-х годов, а также что наблюдения по атмосферному давлению начинаются только с 1990-х годов.

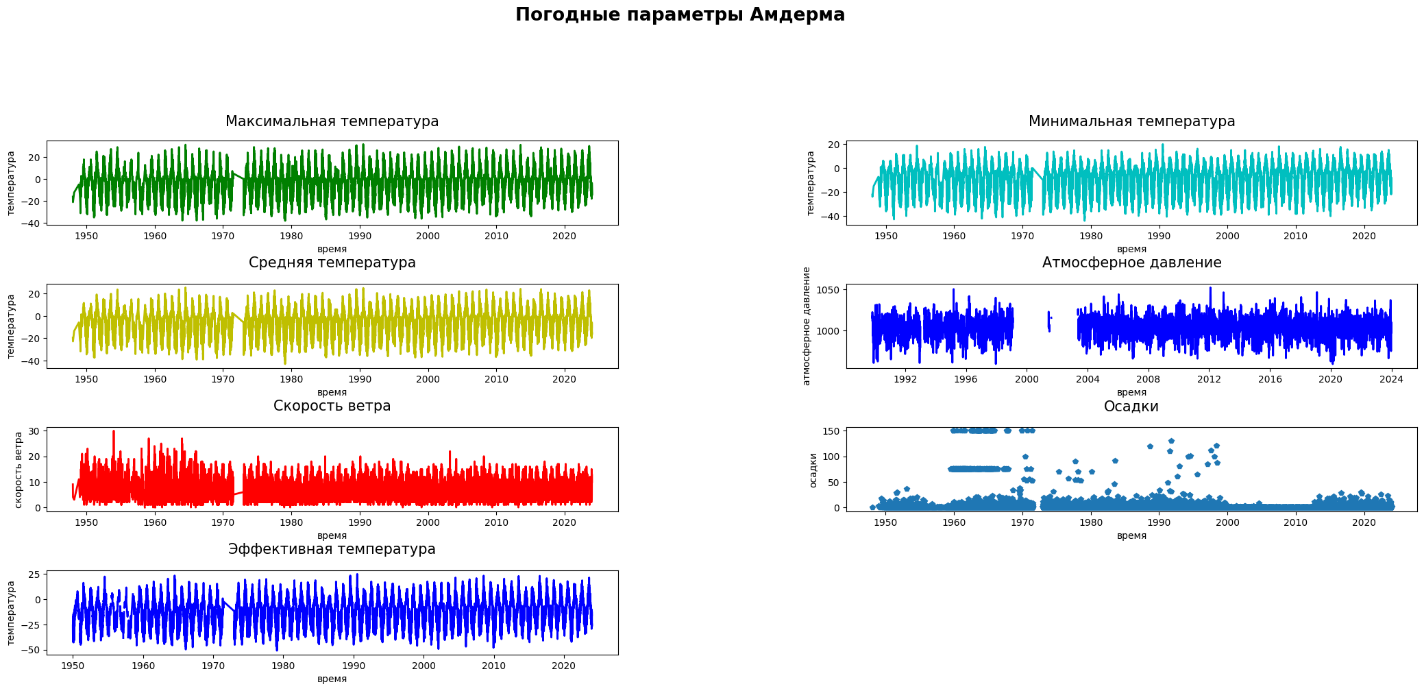


Рисунок 3 — Временные ряды погодных параметров станции Амдерма

Та же самая ситуация прослеживается на тепловой карте, представленной на рисунке 4.

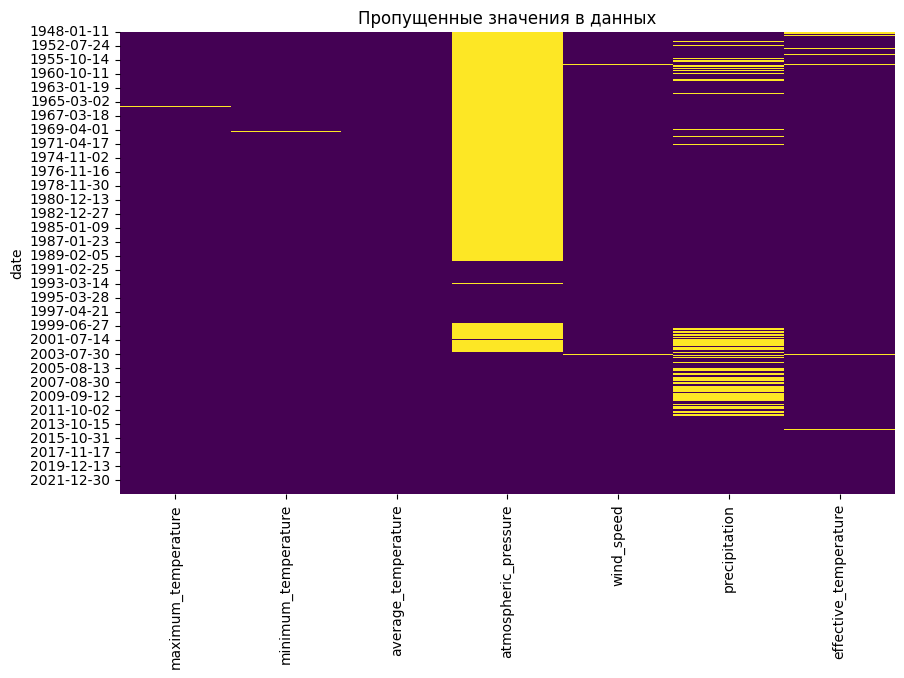


Рисунок 4 — Пропущенные значения в данных станции Амдерма

По тепловой карте также ясно, что осадки тоже имеют достаточно большое количество пропусков.

При дальнейшем изучении данных понятно, что данные заключения характерны для всех станций. Количество пропусков по всем метеостанциям представлено в таблице 2.

Таблица 2 — Пропуски в данных

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Станция | Пропуски | | | | | | |
| Максимальная температура | Минимальная температура | Средняя температура | Атмосферное давление | Скорость ветра | Осадки | Эффективная температура |
| Остров Визе | 129 | 139 | 0 | 12800 | 62 | 1129 | 229 |
| Голомянный | 146 | 139 | 0 | 12876 | 77 | 759 | 336 |
| Мыс Стерлигова | 78 | 98 | 0 | 11204 | 118 | 434 | 281 |
| Остров Тройной | 138 | 152 | 0 | 10128 | 98 | 590 | 282 |
| Остров Диксон | 92 | 88 | 0 | 14480 | 104 | 973 | 747 |
| Остров Белый | 88 | 85 | 0 | 13871 | 123 | 2998 | 771 |
| Мыс Желания | 130 | 144 | 0 | 11521 | 113 | 470 | 249 |
| Марресаля | 59 | 56 | 0 | 13843 | 117 | 2653 | 694 |
| Амдерма | 57 | 55 | 0 | 13911 | 124 | 3686 | 455 |
| Мыс Болванский Нос | 85 | 90 | 0 | 10887 | 61 | 399 | 241 |
| Сеяха | 32 | 25 | 0 | 9970 | 62 | 349 | 158 |
| Мыс Челюскин | 160 | 166 | 0 | 14398 | 144 | 905 | 1043 |
| Мыс Сопочная Карга | 41 | 50 | 0 | 10974 | 103 | 510 | 251 |
| Остров Хейса | 72 | 78 | 0 | 11347 | 48 | 2940 | 151 |
| Все станции | 1307 | 1365 | 0 | 172210 | 1354 | 18795 | 5888 |

По итогам работа с пропусками в параметрах погоды была выполнена следующим образом:

— Атмосферное давление – параметр удален из выборки из-за большого количества пропусков;

— Максимальная температура, минимальная температура, скорость ветра, осадки – пропуски заменены с помощью интерполяции;

— Эффективная температура – замена пропусков по формуле.

Тэф= - 2.7 + 1.04 T + 2.0 P - 0.65 v, где Т - температура воздуха (°С), Р - парциальное давление водяного пара (кПа), v - скорость ветра. Температура воздуха и скорость ветра берутся из набора данных, парциальное давление ввиду его отсутствия в данных заменена на основе этих двух параметров. Таким образом, при сравнении полученные по формуле значения отличаются от реальных данных в рамках 0,5 градуса.

Листинг программного кода замены пропусков представлен в приложении А.

## **2 ВЫЯВЛЕНИЕ ГОДОВ-ГОМОЛОГОВ**

Года-гомологи — это года, которые обладают схожей динамикой изменений исследуемых параметров. Иными словами, это года, которые имеют схожие метеорологические условия. Это означает, что года гомологи характеризуются схожими погодными условиями, например, по средней температуре, осадкам или другим параметрам.

Идея годов гомологов основана на цикличности некоторых метеорологических явлений, которые могут повторяться через определенные интервалы времени. Поэтому года, в которые происходят аналогичные метеорологические явления, могут быть названы годами гомологами.

Использование информации о годах-гомологах может быть актуально для различных сфер деятельности, например метеорологии, сельского хозяйства, строительства, туризма и иных сфер, для которых актуально влияние погодных и других естественных факторов на протекающие процессы.

В перспективе рассматриваемых в наборе данных параметров поиск годов-гомологов можно определить как нахождение таких годов, для которых значение представленных параметров будет наиболее схожим на конкретном отрезке времени.

В рамках проекта предполагается использование годов гомологов для корректировки значений, полученных от прогнозирующей модели. Схожие тенденции развития природных процессов в таких годах должны позволить уменьшить погрешности предсказания путём изменения параметров в направлении тренда в годах-гомологах.

Именно для этой цели необходимо определить и реализовать метод для поиска таких годов, который позволит определять гомологичные временные ряды для дальнейшего их использования.

В рамках работ по проекту были рассмотрены два варианта поиска годов-гомологов на основе имеющихся исторических погодных данных:

* Сравнение векторного расстояния между дневными значениями исследуемых параметров;
* Кластеризация временных рядов.

Одним из вариантов реализации поиска годов гомологов был разработан самостоятельно. Он заключается в представлении данных о каждом годе в виде вектора значений. Такой вектор значений формируется из значений параметров для каждого года. В представленном наборе данных имеется 6 параметров, которые мы используем в рамках проекта. Получается, что при использовании данного метода каждый год превращается в 6 векторов из параметров. Длина такого вектора соответствует количеству дней, данные по которым известны. Получается, что максимальная длина такого вектора равняется 365 или 366 дням для високосного года.

Для реализации данного метода был написан код программы, которая загружает данные из соответствующего файла с параметрами для конкретного года, для которого происходит поиск годов-гомологов. Далее происходит загрузка все остальных файлов с годами. В результате мы получаем некоторое множество таблиц, каждая из которых соответствует множеству параметров для конкретного года. Каждый столбец каждой из этих таблиц это вектор значений параметра, каждое из значений данного вектора это значение рассматриваемого параметра в конкретный день. В результате получается, что каждый год это множество из 6 векторов, длина которых соответствует количеству представленных в наборе данных дней для этого года.

Далее необходимо реализовать сравнение этих векторов в получившимся пространстве значений. На данном этапе мы имеем 6 пространств значений, в каждом из которых имеется N векторов, количество которых равно количеству рассматриваемых годов. Для сравнения удаленности векторов друг от друга в пространстве значений было решено использовать два пространственных расстояния: манхэттенское и косинусное. Код для реализации манхэттенского расстояния представлен на рисунке 5.

Изображение выглядит как текст, Шрифт, программное обеспечение, снимок экрана

Автоматически созданное описание

Рисунок 5 — Код для манхэттенского расстояния

Также для добавления дополнительной оценки был реализован поиск косинусного расстояние между векторами, код для которого представлен на рисунке 6.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 6 — Код для косинусного расстояния

В результате мы имеем 6 множеств векторов в 6 пространствах из параметров и функции, которые позволяют искать расстояние между этими векторами. Теперь возможно реализовать поиск годов гомологов путём сравнения этих расстояний относительно целевого года. Для демонстрации целевым был выбран 1958 год.

В результате работы программы до каждого года рассчитывались оба расстояния по каждому из параметров. Ход выполнения программы представлен на рисунке 7.

Изображение выглядит как текст, снимок экрана, Шрифт, информация

Автоматически созданное описание

Рисунок 7 — Ход выполнения поиска расстояний

После нахождения каждого из расстояний возможно отображение ближайших годов по каждому из параметров. Пример такого отображения для манхэттенского расстояния по параметру максимальная температура представлен на рисунке 8.

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

Рисунок 8 — 10 ближайших годов по максимальной температуре

Но каждый год является комплексом из 6 параметров. Близкий по одному параметру год может оказаться удаленным по всем остальным параметрам. Для этого необходимо подсчитать количество вхождений каждого из годов в десятку ближайших по каждому из параметров. В результате работы алгоритма мы получим словарь, где ключом является год, а значением количество вхождений этого года в 10 ближайших по всем параметрам относительно целевого года. Результат работы алгоритма представлен на рисунке 9.

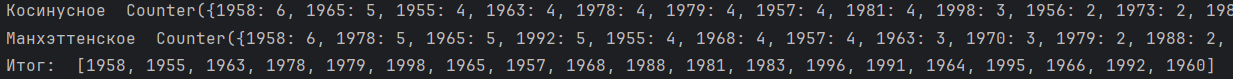


Рисунок 9 — Список ближайших годов

Поле итог включает в себя те года, которые попали в оба списка – по манхэттенскому и косинусным расстояниям. Именно эти года были определены как гомологичные для 1958 года с помощью рассматриваемого метода.

Очевидно, что ближайшим вектором будет сам 1958 год. Также видно, что разница в ближайшей тройке годов не существенна. Таким образом можно считать, что данный метод определил 1981, 1983 и 1988 и другие года как гомологичные для 1958 года.

Вторым способом был рассмотрен вариант с применением кластеризации временных рядов. Для реализации данного метода необходимо представить данные о всех годах в формате, удобном для дальнейшей кластеризации.

Рассмотрим пример такого представления данных, в котором все года определены как столбцы общей таблицы, отражающей все множество имеющихся годов. Для каждого из параметров должна быть создана соответствующая таблица с данными. Пример такой таблицы для параметра средней температуры представлен на рисунке 10.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 10 — Таблица годов для кластеризации

В полученной таблице имеется 49 столбцов, каждый из которых соответствует множеству значений параметра средняя температура для соответствующего года.

Для поиска гомологичного временного ряда для введённых пользователем данных необходимо выполнение двух действий: добавление пользовательских данных в таблицу и усечение всего набора данных до размера, соответствующего введённым данным. В результате в представленную таблицу были добавлены значения параметров, введённые пользователем.

После получения необходимого вида данных можно приступать к реализации кластеризации.

Для реализации кластеризации был выбран метод TimeSeriesKMeans из библиотеки tslearn. Это вариация алгоритма K-Means, специально разработанная для кластеризации временных рядов.

В отличие от классического K-Means, который работает с обычными точечными данными, TimeSeriesKMeans учитывает временную последовательность данных при кластеризации. Основная идея TimeSeriesKMeans заключается в том, что временные ряды имеют специфические характеристики, такие как тренды, сезонность и автокорреляция, которые могут быть использованы для более точной кластеризации. Алгоритм учитывает не только значения точек данных, но и их порядок и временную структуру.

На данном этапе мы имеем подготовленные к кластеризации данные и необходимую модель кластеризации. Для реализации кластеризации с помощью K-Means необходимо определить количество кластеров, в которые будут распределяться данные. Для этого воспользуемся методом плеча и силуэта. Для этого необходимо построить график для метода плеча как показано на рисунке 11.

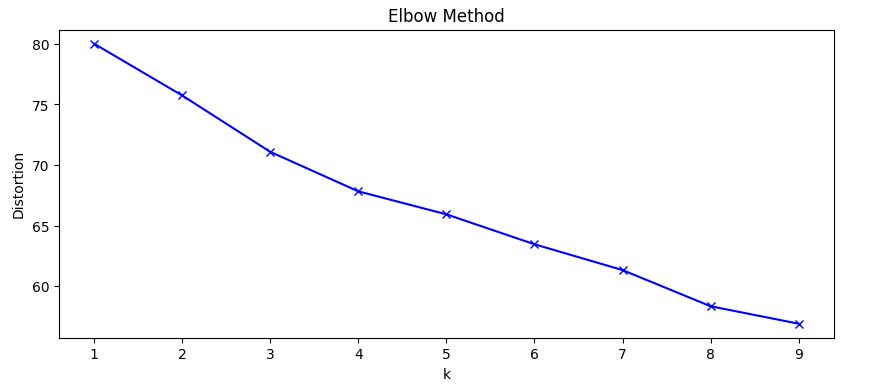


Рисунок 11 — График для метода плеча

Также построим график для метода силуэта как показано на рисунке 12.

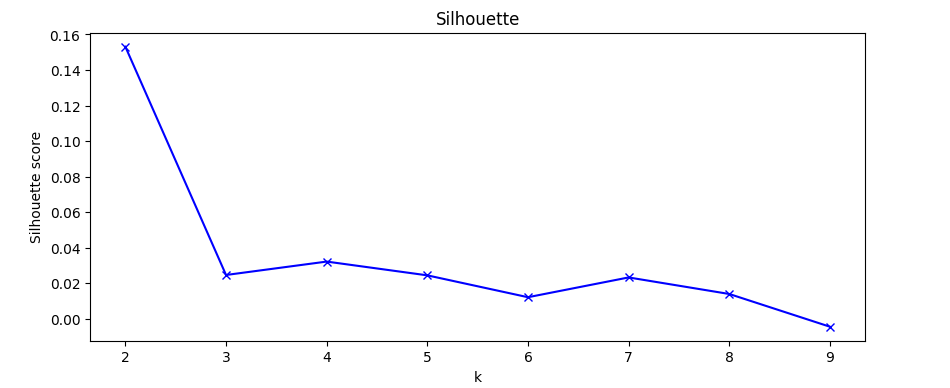


Рисунок 12 — График для метода плеча

В результате анализа графика по методу локтя количество кластеров было определено равным 6. Теперь можно выполнять кластеризацию.

В результате выполнения кластеризации для рассматриваемого примера были получены следующие центройды кластеров, как показано на рисунке 13.

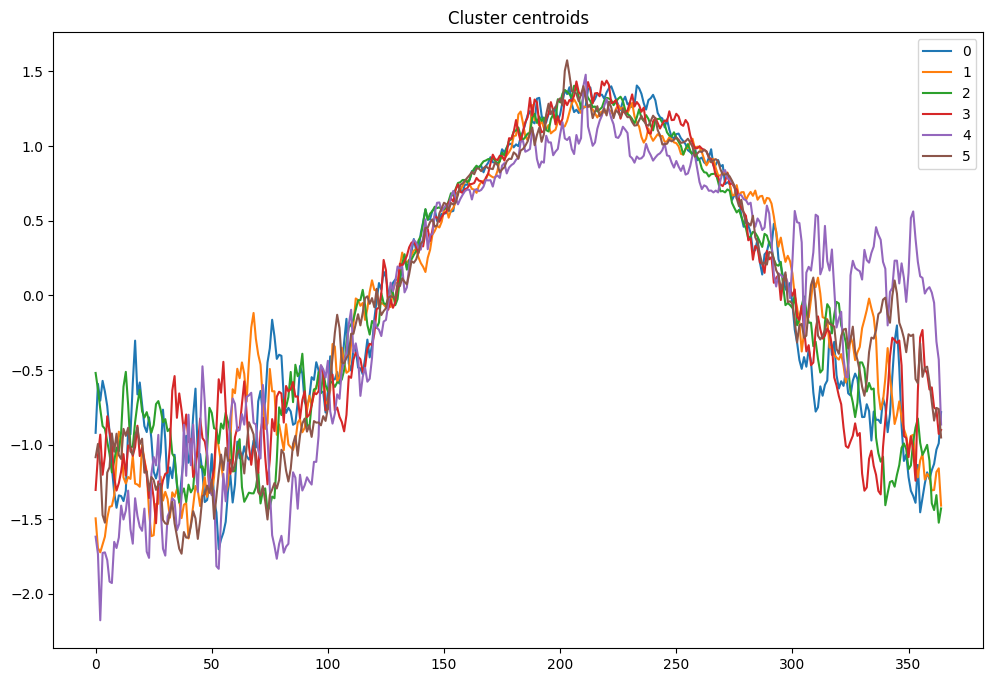


Рисунок 13 — Центройды полученных кластеров

В результате кластеризации получены 6 кластеров. Каждый год отнесён к соответствующему кластеру. Попробуем отобразить года, которые были соотнесены к общим кластерам.

Два года, добавленных в первый кластер, представлены на рисунке 14.

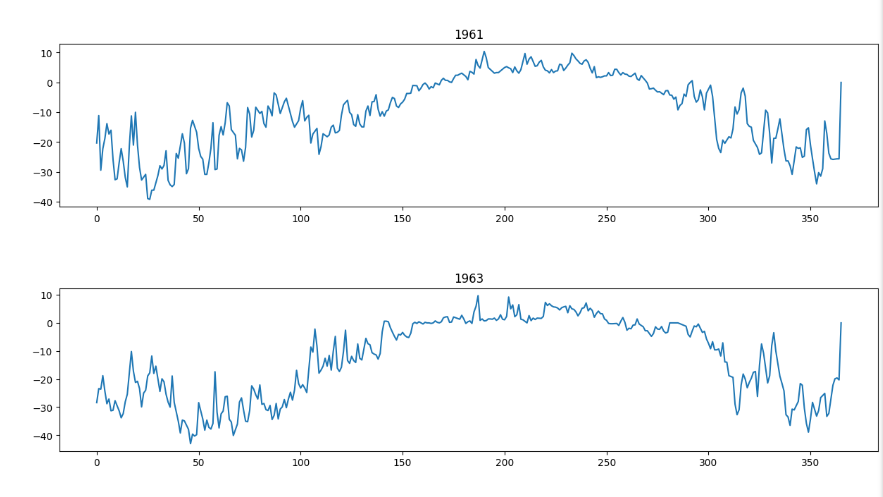


Рисунок 14 — Пример годов из первого кластера

Помимо первого кластера, проверим как сгруппированы года в другом кластере. Пример годов, отнесённых ко второму кластеру представлен на рисунке 15.

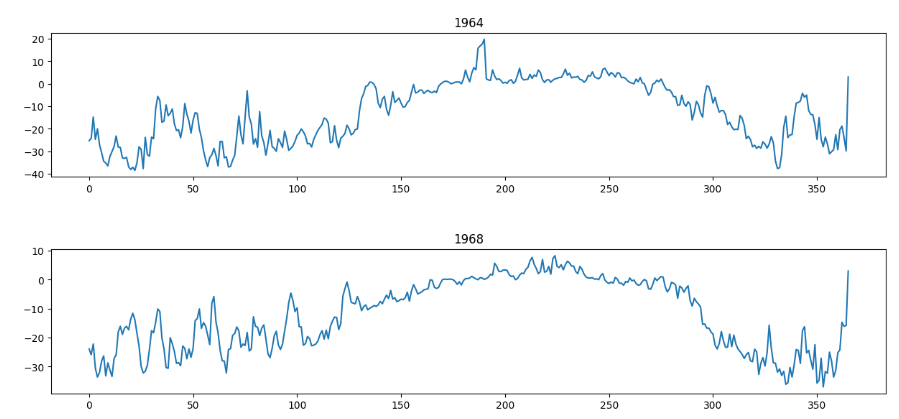


Рисунок 15 — Пример годов из второго кластера

В результате анализа графиков можно определить, что кластеризация отрабатывает верно. В представленных годах можно определить схожую динамику изменения параметра.

При применении кластеризации на пользовательском вводе необходимо добавить в общую таблицу годов введённые пользователем данные. Далее весь набор данных урезается до дней, введённых пользователем. После этого можно выполнять кластеризацию аналогичным образом.

В результате кластеризации каждый год был определен в конкретный кластер, как и введённый пользователем отрезок данных.

Далее необходимо определить гомологичность годов. Для этого необходимо реализовать функцию, которая будет подсчитывать количество вхождений годов в кластер с целевым пользовательским отрезком. Года, которые по наибольшему количеству параметров попали в тот же кластер, что и данные пользователя, должны помечены как гомологичные.

Код, реализующий данный подсчет, представлен на рисунке 16.

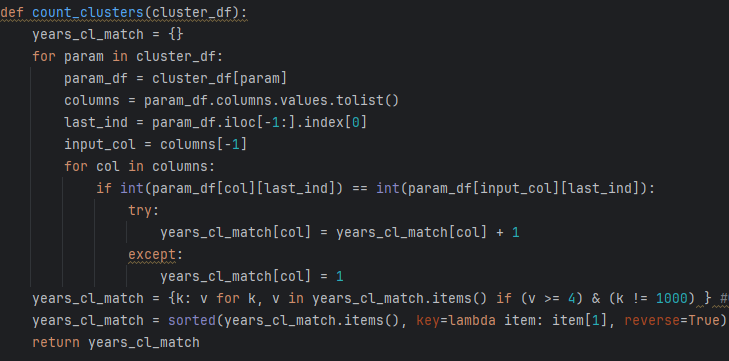


Рисунок 16 — Код для подсчета годов-гомологов

В результате работы данной функции происходит возврат списка годов-гомологов, как показано на рисунке 17.

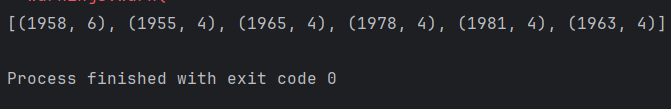


Рисунок 17 – Года гомологи для 1958

Как видно на рисунке, для каждого года представлен кортеж, содержащий пару значений: (год, количество совпавших кластеров). Если год совпал с целевым по 4 и более кластерам, то он помечается как гомологичный.

В результате сравнения работы параметров для 1958 года мы получили совпадения по нескольким годам, определённым как гомологи. Это 1981, 1965 и 1978 год. Наличие совпадений в обоих методах позволяет считать, что данные года можно считать годами-гомологами.

Для работы модуля в его конечном виде был выбран метод поиска годов-гомологов с помощью кластеризации, т.к. данный метод обладает большим потенциалом для тонкой настройки и повышения качества работы. Например, за счёт изменения метода кластеризации в будущем.

В результате была реализована кластеризация по 6 признакам и определение годов-гомологов в том случае, если по 4 из 6 признаков год попал в тот же кластер, что и целевой отрезок.

## **3 ПРЕДСКАЗАНИЕ ПОГОДНЫХ УСЛОВИЙ**

Предсказание погодных условий предполагает прогнозирование параметров, которые представлены в рассматриваемом наборе данных. Каждый из представленных параметров является элементов общего временного ряда.

В рамках данного проекта предлагается принять один год как 6 временных рядов, каждый из которых соответствует отдельному параметру, обладающему своим трендом и абсолютными значениями.

Существуют разные способы прогнозирования временного ряда, основывающиеся на моделях различных уровней сложности реализации и обучения. В рамках рассматриваемого проекта предлагается использование метода SARIMAX для прогнозирования временного ряда параметров.

SARIMAX – это модель сезонного авторегрессивного интегрированного скользящего среднего. Данная модель является расширением классической модели ARIMA (AutoRegressive Integrated Moving Average) для учета сезонности и внешних факторов. SARIMAX является мощным инструментом для анализа и прогнозирования временных рядов с учетом сезонности и внешних переменных.

Модель SARIMAX широко используется для прогнозирования временных рядов в различных областях, таких как финансы, экономика, метеорология, продажи и другие. Она позволяет учитывать сложные закономерности и взаимосвязи в данных, что делает ее эффективным инструментом для анализа и прогнозирования временных рядов.

Выбор модели осуществлен, далее необходимо определиться с реализацией предсказания.

Для предсказания необходим временной ряд, который будет подаваться модели на обучение. Модель SARIMAX, как и многие другие модели, предсказывает значения начиная с последней даты тренировочного временного ряда.

Таким образом необходимо получить некоторый составной временной ряд, который будет содержать как исторические данные, так и введённый пользователем отрезок.

В следствие возможно определить требования к вводимой пользователем информации в модель предсказания:

* Хронология пользовательского ввода должна начинаться на следующий день после тренировочного набора данных;
* Пользовательский ввод должен обладать тем же набором параметром, что и тренировочные данные.

Первое требование обусловлено тем, что при обучении модели набор данных задаётся как набор с ежедневной частотой представления показателей.

После ввода информации от пользователя происходит объединение обучающего исторического и пользовательского набора данных. Именно этот временной ряд будет подаваться модели для обучения.

В результате применения такого подхода обработки данных мы получаем 6 временных рядов длиной равной: «исторические данные» + «пользовательский ввод». Каждый временной ряд соответствует погодному параметру из набора данных. Следовательно, в рамках прогнозирования необходимо обучить и использовать 6 моделей для предсказания.

Следует помнить, что весь наш набор состоит из 14 метеостанций, каждая из которых находится на определенном удалении от остальных. Такой подход позволяет оставить некоторую географическую привязку для нашего набора данных. В виду отличия природных процессов, происходящих на удаленных территориях, невозможно построить одну такую модель, которая сможет с равной долей уверенности предсказывать параметр для каждой из станций.

Отсюда следует, что для реализации предсказания по каждой из станций необходимо обучение отдельных моделей. Как уже было сказано, прогнозирование каждого из параметров также требует обучение отдельной модели прогнозирования.

Анализируя всю описанную информацию, можно сделать вывод, что для реализации системы, которая обучена определять каждый параметр на каждой станции, необходимо обучить 14x6 = 84 прогнозные модели SARIMAX.

Подобный способ реализации прогнозирования имеет существенный минус – скорость выполнения прогнозирования. Если использовать большой временной ряд, содержащий все имеющиеся исторические данные, то скорость обучения модели будет очень низкой.

Очевидно, что обучение такого количества моделей на данном этапе не целесообразно. Предлагается проверить работу разработанных методов на ограниченном наборе станций. Однако, разработанная система должна предоставлять пользователю возможность получить прогноз на любой станции, которую он выберет.

Данный набор условий создает необходимость в обучении модели прямо во время работы программы. В таком случае у нас не будет необходимости предобучать все 84 модели, пользователь сам выберет какая модель ему нужна. Существенным минусом такой реализации является время выполнения предсказания, т.к. обучение может занимать длительное время.

Для решения описанных проблем был сформирован следующий алгоритм предсказательного модуля:

* При первом запуске предсказания на станции происходит обучение прогнозирующей модели;
* После завершения обучения модель сохраняется в корне программы;
* При повторном запуске прогнозирования на данной станции происходит поиск модели в корне программы;
* Если файл модели найден, то происходит перенос параметров обучения на новую модель с новыми данными, в противном случае происходит обучение и возврат к началу алгоритма.

В результате работы описанного алгоритма средняя скорость работы программы была увеличена.

Для реализации данного алгоритма было необходимо при каждом запуске обучения предварительно проверять наличие сохранённой модели в корне программы.

В противном случае необходимо реализовать обучение модели и сохранение.

Следуя описанному алгоритму, можно смоделировать следующее поведение системы прогнозирования. Первые запуски модуля и первые прогнозы на новых станциях будут сопряжены с долгим обучением, но повторный запуск уже будет выполняться кратно быстрее благодаря наличию соответствующих файлов моделей. Рассмотрим код для реализации описанного метода, который представлен на рисунке 18.



Рисунок 18 – Код для обучения модели

Метод filter на представленном рисунке отвечает за получение параметров, полученных при обучении сохраненной модели.

В случае не выполнении функции .load, отвечающей за загрузку модели, происходить exception, в рамках которого выполняется обучение и сохранение модели.

В результате будет получена модель, которая далее может быть использована для предсказания. Код для выполнения предсказания представлен на рисунке 19.

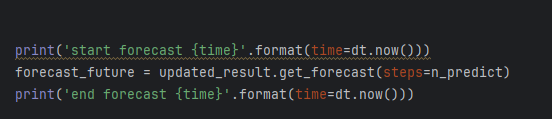


Рисунок 19 – Код для предсказания

Полученные в результате метода .get\_forecast предсказания могут быть отображены на графике предсказания, который представлен на рисунке 20.

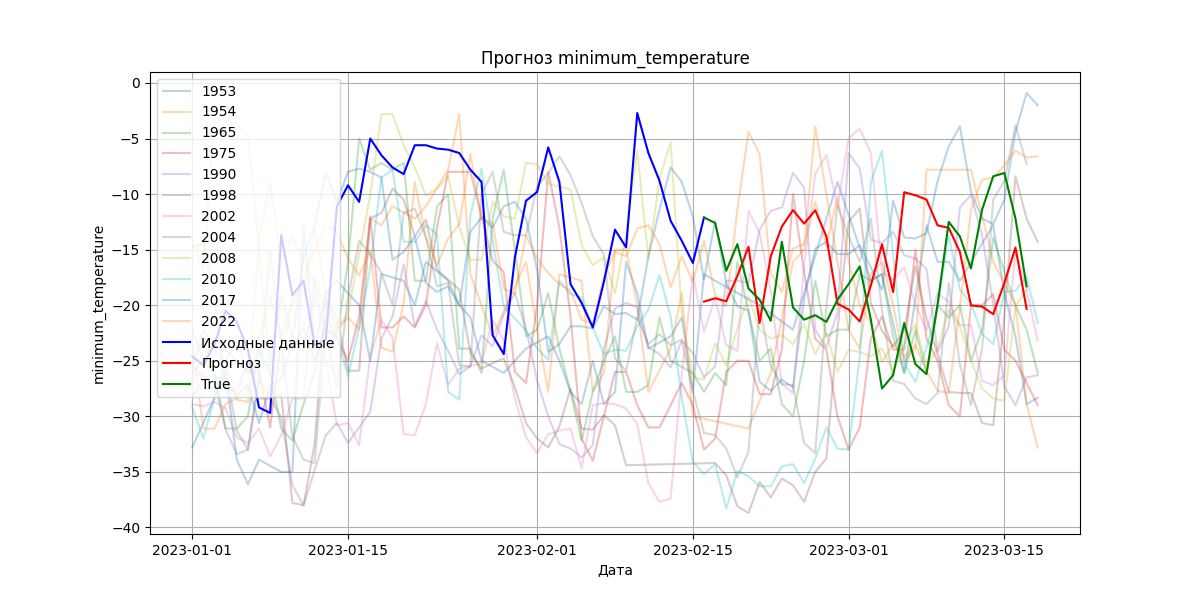


Рисунок 20 – График предсказания максимальной температуры

На представленном графике ярким синим цветом показаны введённые пользователем данные. Ярким зелёным цветом отображены реальные данные за представленные даты из тестовой выборки. Ярким красным цветом показан график предсказанных значений. Тусклые графики – это графики значений параметра температуры для годов-гомологов.

Представленные результаты являются результатами самостоятельно предсказания модели SARIMAX. На данном этапе разработки функции предсказания мы никак не использовали информацию о годах-гомологах, полученную ранее.

Далее предлагается сформировать функцию корректировки предсказанных значений относительно годов-гомологов. Как уже было сказано ранее, предполагается, что года-гомологи обладают схожими тенденциями изменения погодных параметров, в них происходят похожие климатические процессы и изменения.

В таком случае будет уместно предположить, что, анализируя весь набор таких похожих годов возможно выявление некоего тренда, который будет в той или иной степени актуален для каждого из годов-гомологов.

Тогда необходимо сформировать метод, который мы будем использовать для достижения этих целей.

В рамках выполнения проекта было предложено оценивать и корректировать полученные предсказания дискретно по дням. Т.е. мы будем корректировать прогноз на каждый отдельный день относительно такого же отдельного дня в годах гомологах.

Для начала сформируем логический алгоритм работы. Необходимо сравнивать предсказанное значение за день X с такими же реальными значениями дня X в гомологичных годах. Метрикой сравнения выберем функцию разности. В таком случае отличие между показаниями параметра будет измеряться значением такой разности, где итог равный нулю – это идеальное сходство, а стремящаяся к бесконечности разность — это бесконечно далекие друг от друга дни.

Следовательно, необходимо рассчитать такую разность до каждого из дней X в годах гомологах. В результате мы получим некоторый ряд значений, где каждое значение – отличие предсказанного значения от соответствующего значения года-гомолога.

Далее необходимо привести данный ряд к некоторой общей величине, с помощью которой можно будет скорректировать предсказанное значение. Здесь были рассмотрены два варианта нахождения этой общей величины: среднее значение ряда и медианное значение.

Далее это значение необходимо вычесть из спрогнозированного параметра, приблизив итоговое значение к общему тренду в годах-гомологах.

Теперь необходимо реализовать данный алгоритм в коде функции. Описываемый алгоритм представлен на рисунке 21.

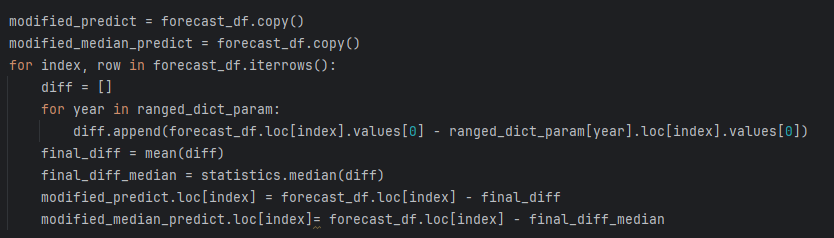


Рисунок 21 – Код для корректировки значений

Представленный код находится внутри цикла для каждого из параметров. Т.е. данный код будет выполнен 6 раз по числу параметров. В представленном коде происходит итерация по каждой строке датафрейма предсказаний. Далее для каждого года в словаре годов-гомологов происходит вычитание значения строки и соответствующего значения в датафрейме года гомолога.

Результат добавляется в список diff, который отражает результат выполнения операции по всем годам. Далее с помощью функций mean и median находятся средние и медианные значения для данной строки соответственно. Строка соответствует одному дню предсказания.

В конце выполнения одной итерации строки происходит вычитание из предсказанного значения полученного значения разности. В итоге происходит сохранение в два новых датафрейма для дальнейшего сравнения результатов.

Полученные результаты можно отразить на графике, представленном на рисунке 22.

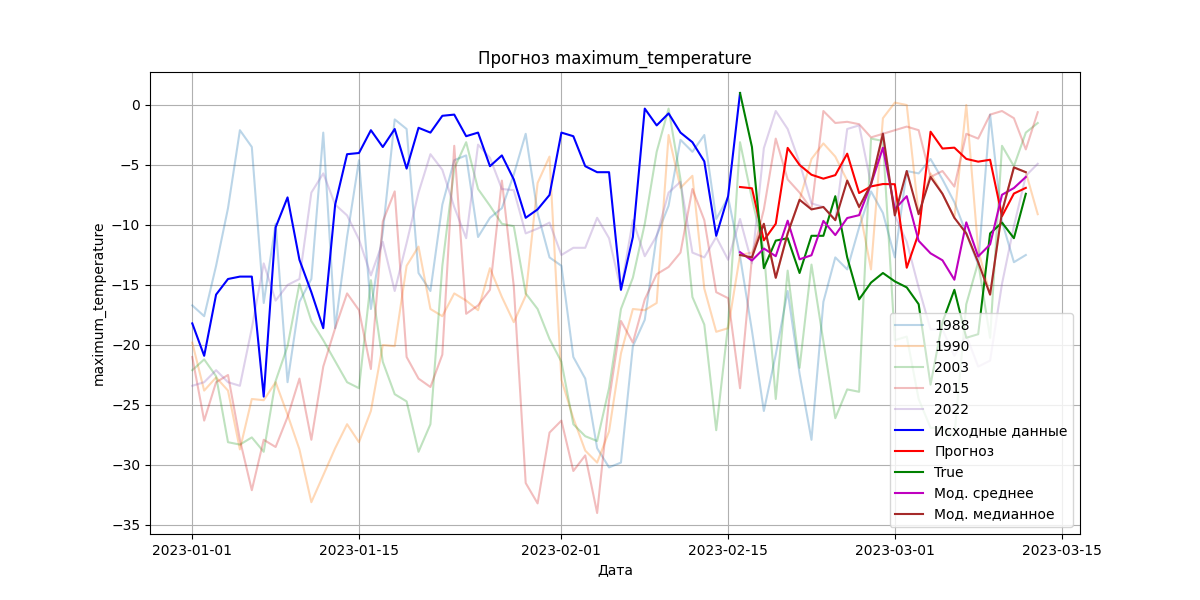


Рисунок 22 – Сводный график для максимальной температуры

На рисунке можно видеть несколько кривых.

Синий цвет, как и ранее, отражает исходные данные, введённые пользователем.

Красная линия показывает прогноз только моделью SARIMAX.

Зелёный цвет — это реальные значения из тестовой выборки.

Розовое значение — это значение предсказания, к которому была применена корректировка средним значением разности.

Коричневым цветом отображен график, который показывает предсказанное значение после корректировки медианной разностью.

Также для определения результатов корректировки возможно использование метрик. Получившиеся значения метрик для каждого из видов результирующего предсказания представлены в таблице 3.

Таблица 3 — Метрики предсказания максимальной температуры

|  |  |  |
| --- | --- | --- |
| Предсказание | MSE | MAE |
| Sarimax | 76.82 | 7.17 |
| Мод. среднее | 38.1 | 4.96 |
| Мод. медианное | 58.05 | 6.46 |

Также необходимо рассмотреть полученные данные по другому параметру, чтобы исключить вероятность правильной работы метода только на одном наборе данных.

Сводный график работы описываемых методов для параметра средней температуры представлен на рисунке 23.

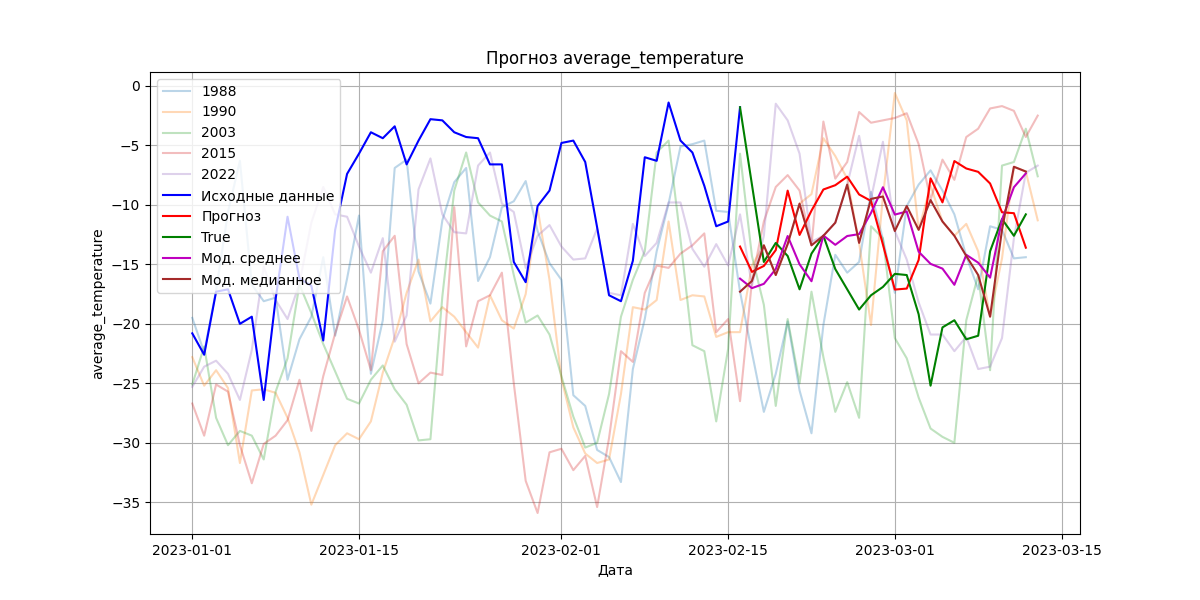


Рисунок 23 – Сводный график для средней температуры

Легенда данного графика соответствует описываемой ранее легенде для графика максимальной температуры.

Также необходимо отразить метрики предсказаний. Метрики представлены в таблице 4.

Таблица 4 — Метрики предсказания средней температуры

|  |  |  |
| --- | --- | --- |
| Предсказание | MSE | MAE |
| Sarimax | 65.45 | 6.50 |
| Мод. среднее | 33.08 | 4.72 |
| Мод. медианное | 49.53 | 5.86 |

В результате анализа графиков и метрик для итоговой реализации системы была выбрана корректировка средним значением.

В результате разработки метода прогнозирования мы смогли спроектировать и реализовать прогнозирование с помощью модели SARIMAX и корректировки на основе данных о годах гомологах.

В результате применения описываемой корректировки нам удалось достичь понижения ошибки MSE и MAE, что отражено в результирующих метриках.

## **4 РАЗРАБОТКА ПРОГРАММЫ**

В ходе работы над проектом также был реализован интерфейс программы, который был создан с помощью CustomTkinter, библиотеки пользовательского интерфейса для компьютеров на основе Tkinter [4].

Вся программа состоит из 6 вкладок, в рамках которых реализован определенный функционал.

Первая вкладка состоит из интерактивной карты выбранных метеостанций, с помощью которой пользователь выбирает данные для дальнейшей работы программы (рисунок 24). Карта реализована на основе виджета TkinterMapView [5].

Изображение выглядит как текст, карта, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 24— Вкладка «Select a station»

Во второй вкладке пользователь загружает свой csv файл и видит его представление на экране приложения (рисунок 25). На его основе делаются дальнейший поиск годов-гомологов и предсказание погодных условий.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 25 — Вкладка «Import»

На третьей вкладке после нажатия кнопки поиска годов гомологов происходит их поиск и отображение результата с помощью графика (рисунок 26).

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 26 — Вкладка «Search homologues»

Четвертая вкладка предназначена для выбора даты, до которой будут делаться предсказания (рисунок 27). Также пользователь может выбрать параметр, для которого будет отображен график предсказания в следующей вкладке (рисунок 28).

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 27 — Вкладка «Select date»

Изображение выглядит как текст, снимок экрана, График, линия

Автоматически созданное описание

Рисунок 28 — Вкладка «Forecast plots»

В последней вкладке, представленной на рисунке 29, пользователь после нажатия соответствующей кнопки получает таблицу, состоящею из внесенных данных и предсказанных значений, а также может загрузить полученные значения в виде csv файла

Изображение выглядит как снимок экрана, текст, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 29 — Вкладка «Download»

Программный код интерфейса разработанной программы представлен в приложении А.

## ЗАКЛЮЧЕНИЕ

В рамках выполнения проекта были достигнуты поставленные цели. Исследован вопрос предсказания погодных параметров и выбран вариант реализации собственной системы с применением данных о годах-гомологах.

Был найден и обработан набор данных, состоящий из исторических погодных данных основанных на показаниях метеостанций.

Были определены два метода поиска годов-гомологов, реализована их работа и выбран наиболее актуальный метод для работы системы.

Также было реализовано предсказание погодных параметров с помощью модели прогнозирования.

Определен и реализован метод корректировки полученных предсказанных значений с применением информации о годах-гомологах.

В результате был получен метод, который позволяет определить набор годов-гомологов на основе исторических данных, осуществить прогнозирование будущих данных и понизить ошибку при предсказании временного ряда с помощью данных о годах-гомологах.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Российская Федерация. Правительство Российской Федерации. План развития Северного морского пути на период до 2035 года [Текст]: распоряжение от 1 августа 2022 года №2115-р – 2022. – Электрон. текстовые дан. – Режим доступа: <http://static.government.ru/media/files/StA6ySKbBceANLRA6V2sF6wbOKSyxNzw.pdf>, свободный (дата обращения: 22.04.2024). – Загл. с экрана.
2. Архив погоды c 1929 года [Электронный ресурс] / Погодные сервисы [офиц. сайт] – Электрон. дан. –– Погодные сервисы – Режим доступа: <http://pogoda-service.ru/archive_gsod.php>, свободный (дата обращения: 22.04.2024). – Загл. с экрана.
3. Клячкин, С.В. Достоверность краткосрочных и долгосрочных ледовых прогнозов для обеспечения мореплавания и других видов хозяйственной деятельности на СМП в 2023 году [Электронный ресурс] / С.В. Клячкин, И.А. Сергеева // Российские полярные исследования. –2024. – № 1 – С. 28–29. – Электрон. текстовые дан. – Режим доступа: <http://old.aari.ru/misc/publicat/sources/55/RPR-55el_l_28-29.pdf>, свободный (дата обращения: 22.04.2024). – Загл. с экрана.
4. CustomTkinter [Electronic resource] / CustomTkinter [official website] – Electronic data – Tom Schimansky, 2024 – Mode of access: <https://customtkinter.tomschimansky.com/>, free access (22.04.2024). – Title from screen.
5. TkinterMapView [Electronic resource] / GitHub [official website] – Electronic data – GitHub, Inc, 2024 – Mode of access: <https://github.com/TomSchimansky/TkinterMapView>, free access (22.04.2024). – Title from screen.

## 

## ПРИЛОЖЕНИЕ А

(обязательное)

Листинг программного кода

# обработка данных

import pandas as pd

list\_of\_stations = ['Amderma', 'Golomyannyy', 'Marresalya', 'Mys\_Bolvanskiy\_Nos',

'Mys\_Chelyuskin', 'Mys\_Sopochnaya\_Karga', 'Mys\_Sterligova',

'Mys\_Zhelaniya', 'Ostrov\_Bely', 'Ostrov\_Dikson',

'Ostrov\_Kheysa', 'Ostrov\_Troynoy', 'Ostrov\_Vize',

'Seyakha']

for i in list\_of\_stations:

df = pd.read\_csv(f'/kaggle/input/stations/{i}.csv')

df.drop('atmospheric\_pressure', axis= 1 , inplace= True )

df['precipitation'] = round(df['precipitation'].interpolate(limit\_direction='both'))

df['maximum\_temperature'] = round(df['maximum\_temperature'].interpolate(limit\_direction='both'), 1)

df['minimum\_temperature'] = round(df['minimum\_temperature'].interpolate(limit\_direction='both'), 1)

df['wind\_speed'] = round(df['wind\_speed'].interpolate(method='nearest',limit\_direction='both', order=2))

# fill NaN in 'wind\_speed' in the ending of dataset'Mys\_Zhelaniy'

import random

import math

random.seed(10)

def random\_fill(row):

if row['wind\_speed'] == -1:

pass

else:

return random.randint(1, 16)

if len(df.loc[(df['wind\_speed'].isnull())]) > 0:

df['wind\_speed'] = df.apply(random\_fill, axis=1)

def assign\_partial\_pressure(row):

if row['average\_temperature'] < -40:

if row['wind\_speed'] in [x / 10.0 for x in range(0, 60, 1)]:

return 0.02

elif row['wind\_speed'] in [x / 10.0 for x in range(60, 110, 1)]:

return 0.01

elif row['wind\_speed'] in [x / 10.0 for x in range(110, 160, 1)]:

return 0.008

elif row['wind\_speed'] in [x / 10.0 for x in range(160, 210, 1)]:

return 0.006

elif row['wind\_speed'] in [x / 10.0 for x in range(210, 260, 1)]:

return 0.004

elif row['wind\_speed'] > 26:

return 0.002

elif row['average\_temperature'] in [x / 10.0 for x in range(-400, -300, 1)]:

if row['wind\_speed'] in [x / 10.0 for x in range(0, 60, 1)]:

return 0.05

elif row['wind\_speed'] in [x / 10.0 for x in range(60, 110, 1)]:

return 0.03

elif row['wind\_speed'] in [x / 10.0 for x in range(110, 160, 1)]:

return 0.01

elif row['wind\_speed'] in [x / 10.0 for x in range(160, 210, 1)]:

return 0.008

elif row['wind\_speed'] in [x / 10.0 for x in range(210, 260, 1)]:

return 0.006

elif row['wind\_speed'] > 26:

return 0.004

elif row['average\_temperature'] in [x / 10.0 for x in range(-300, -200, 1)]:

if row['wind\_speed'] in [x / 10.0 for x in range(0, 60, 1)]:

return 0.07

elif row['wind\_speed'] in [x / 10.0 for x in range(60, 110, 1)]:

return 0.05

elif row['wind\_speed'] in [x / 10.0 for x in range(110, 160, 1)]:

return 0.03

elif row['wind\_speed'] in [x / 10.0 for x in range(160, 210, 1)]:

return 0.01

elif row['wind\_speed'] in [x / 10.0 for x in range(210, 260, 1)]:

return 0.008

elif row['wind\_speed'] > 26:

return 0.006

elif row['average\_temperature'] in [x / 10.0 for x in range(-200, -100, 1)]:

if row['wind\_speed'] in [x / 10.0 for x in range(0, 60, 1)]:

return 0.11

elif row['wind\_speed'] in [x / 10.0 for x in range(60, 110, 1)]:

return 0.1

elif row['wind\_speed'] in [x / 10.0 for x in range(110, 160, 1)]:

return 0.08

elif row['wind\_speed'] in [x / 10.0 for x in range(160, 210, 1)]:

return 0.06

elif row['wind\_speed'] in [x / 10.0 for x in range(210, 260, 1)]:

return 0.03

elif row['wind\_speed'] > 26:

return 0.01

elif row['average\_temperature'] in [x / 10.0 for x in range(-100, 0, 1)]:

if row['wind\_speed'] in [x / 10.0 for x in range(0, 60, 1)]:

return 0.6

elif row['wind\_speed'] in [x / 10.0 for x in range(60, 110, 1)]:

return 0.5

elif row['wind\_speed'] in [x / 10.0 for x in range(110, 160, 1)]:

return 0.3

elif row['wind\_speed'] in [x / 10.0 for x in range(160, 210, 1)]:

return 0.2

elif row['wind\_speed'] in [x / 10.0 for x in range(210, 260, 1)]:

return 0.1

elif row['wind\_speed'] > 26:

return 0.05

elif row['average\_temperature'] in [x / 10.0 for x in range(0, 100, 1)]:

if row['wind\_speed'] in [x / 10.0 for x in range(0, 60, 1)]:

return 0.7

elif row['wind\_speed'] in [x / 10.0 for x in range(60, 110, 1)]:

return 0.5

elif row['wind\_speed'] in [x / 10.0 for x in range(110, 160, 1)]:

return 0.4

elif row['wind\_speed'] in [x / 10.0 for x in range(160, 210, 1)]:

return 0.3

elif row['wind\_speed'] in [x / 10.0 for x in range(210, 260, 1)]:

return 0.2

elif row['wind\_speed'] > 26:

return 0.1

elif row['average\_temperature'] in [x / 10.0 for x in range(100, 210, 1)]:

if row['wind\_speed'] in [x / 10.0 for x in range(0, 60, 1)]:

return 1.3

elif row['wind\_speed'] in [x / 10.0 for x in range(60, 110, 1)]:

return 1.2

elif row['wind\_speed'] in [x / 10.0 for x in range(110, 160, 1)]:

return 1.1

elif row['wind\_speed'] in [x / 10.0 for x in range(160, 210, 1)]:

return 1

elif row['wind\_speed'] in range(21, 26, 0.1):

return 0.9

elif row['wind\_speed'] > 26:

return 0.8

elif row['average\_temperature'] > 21:

if row['wind\_speed'] in [x / 10.0 for x in range(0, 60, 1)]:

return 1.8

elif row['wind\_speed'] in [x / 10.0 for x in range(60, 110, 1)]:

return 1.6

elif row['wind\_speed'] in [x / 10.0 for x in range(110, 160, 1)]:

return 1.4

elif row['wind\_speed'] in [x / 10.0 for x in range(160, 210, 1)]:

return 1.2

elif row['wind\_speed'] in [x / 10.0 for x in range(210, 260, 1)]:

return 1.1

elif row['wind\_speed'] > 26:

return 1

df['partial\_pressure'] = df.apply(assign\_partial\_pressure, axis=1)

effect\_temp = round((- 2.7 + 1.04\*(df["average\_temperature"]) + 2\*df['partial\_pressure'] - 0.65\*(df["wind\_speed"])), 1)

df['effective\_temperature'] = df['effective\_temperature'].fillna(effect\_temp)

df.drop('partial\_pressure', axis= 1 , inplace= True )

print(i)

print(df.isnull().sum())

print()

df.to\_csv(f'{i}\_after\_preprocessing.csv', index=False)

# разработка интерфейса программы

import tkinter

import tkintermapview

import tkinter.messagebox

import customtkinter

from tkinter import \*

import csv

from PIL import Image

import pandas as pd

import os

import datetime

from tkcalendar import Calendar

from pathlib import Path

import main

import homofind\_cluster

customtkinter.set\_appearance\_mode("Dark") # Modes: "System" (standard), "Dark", "Light"

customtkinter.set\_default\_color\_theme("blue") # Themes: "blue" (standard), "green", "dark-blue"

station\_dict = {'Амдерма':'Amderma', 'Голомянный': 'Golomyannyy',

'Марресаля': 'Marresalya', 'Мыс Болванский Нос': 'Mys\_Bolvanskiy\_Nos',

'Мыс Челюскин': 'Mys\_Chelyuskin', 'Мыс Сопочная Карга': 'Mys\_Sopochnaya\_Karga',

'Мыс Стерлигова': 'Mys\_Sterligova', 'Мыс Желания': 'Mys\_Zhelaniya',

'Остров Белый': 'Ostrov\_Bely', 'Остров Диксон': 'Ostrov\_Dikson',

'Остров Хейса': 'Ostrov\_Kheysa', 'Остров Тройной': 'Ostrov\_Troynoy',

'Остров Визе': 'Ostrov\_Vize', 'Сеяха': 'Seyakha', }

class App(customtkinter.CTk):

def \_\_init\_\_(self):

super().\_\_init\_\_()

# configure window

self.title("weather forecasting by year-homologue")

self.geometry(f"{1100}x{600}")

self.grid\_columnconfigure(0, weight=1)

self.grid\_columnconfigure(1, weight=1)

self.grid\_rowconfigure(1, weight=1)

self.logo\_label = customtkinter.CTkLabel(self, text="Homologues", font=customtkinter.CTkFont(size=20, weight="bold"))

self.logo\_label.grid(row=0, column=0, padx=0, pady=(0, 0))

# create tabview

self.tabview = customtkinter.CTkTabview(self, width=1050)

self.tabview.grid(row=1, column=0, padx=(20, 20), pady=(0, 20), sticky="nsew")

self.tabview.add("Select a station")

self.tabview.add("Import")

self.tabview.add("Search homologues")

self.tabview.add("Select date")

self.tabview.add("Forecast plots")

self.tabview.add("Download")

self.tabview.tab("Select a station").grid\_columnconfigure(0, weight=1) # configure grid of individual tabs

self.tabview.tab("Select a station").grid\_rowconfigure(0, weight=1)

self.tabview.tab("Import").grid\_columnconfigure(0, weight=1)

self.tabview.tab("Search homologues").grid\_columnconfigure(0, weight=1)

self.tabview.tab("Select date").grid\_columnconfigure(1, weight=1)

self.tabview.tab("Forecast plots").grid\_columnconfigure(1, weight=1)

self.tabview.tab("Download").grid\_columnconfigure(0, weight=1)

# Select a station tab

self.map\_widget = tkintermapview.TkinterMapView(self.tabview.tab("Select a station"), corner\_radius=0)

self.map\_widget.grid(row=0, column=0, rowspan=2, pady=0, padx=0, sticky="nsew")

self.map\_widget.set\_tile\_server("https://mt0.google.com/vt/lyrs=m&hl=ru&x={x}&y={y}&z={z}&s=Ga", max\_zoom=22)

self.map\_widget.set\_position(76.824153, 76.211605)

self.map\_widget.set\_zoom(4)

marker\_1 = self.map\_widget.set\_marker(79.491, 76.981, text="Остров Визе", text\_color="black",

marker\_color\_circle="black", marker\_color\_outside="gray40",

font=("Helvetica Bold", 8), command=self.click\_marker)

marker\_2 = self.map\_widget.set\_marker(79.550, 90.567, text="Голомянный", text\_color="black",

marker\_color\_circle="black", marker\_color\_outside="gray40",

font=("Helvetica Bold", 8), command=self.click\_marker)

marker\_3 = self.map\_widget.set\_marker(75.417, 88.900, text="Мыс Стерлигова", text\_color="black",

marker\_color\_circle="black", marker\_color\_outside="gray40",

font=("Helvetica Bold", 8), command=self.click\_marker)

marker\_4 = self.map\_widget.set\_marker(75.952, 82.946, text="Остров Тройной", text\_color="black",

marker\_color\_circle="black", marker\_color\_outside="gray40",

font=("Helvetica Bold", 8), command=self.click\_marker)

marker\_5 = self.map\_widget.set\_marker(73.500, 80.400, text="Остров Диксон", text\_color="black",

marker\_color\_circle="black", marker\_color\_outside="gray40",

font=("Helvetica Bold", 8), command=self.click\_marker)

marker\_6 = self.map\_widget.set\_marker(73.333, 70.050, text="Остров Белый", text\_color="black",

marker\_color\_circle="black", marker\_color\_outside="gray40",

font=("Helvetica Bold", 8), command=self.click\_marker)

marker\_7 = self.map\_widget.set\_marker(76.950, 68.550, text="Мыс Желания", text\_color="black",

marker\_color\_circle="black", marker\_color\_outside="gray40",

font=("Helvetica Bold", 8), command=self.click\_marker)

marker\_8 = self.map\_widget.set\_marker(69.717, 66.800, text="Марресаля", text\_color="black",

marker\_color\_circle="black", marker\_color\_outside="gray40",

font=("Helvetica Bold", 8), command=self.click\_marker)

marker\_9 = self.map\_widget.set\_marker(70.447, 59.091, text="Мыс Болванский Нос", text\_color="black",

marker\_color\_circle="black", marker\_color\_outside="gray40",

font=("Helvetica Bold", 8), command=self.click\_marker)

marker\_10 = self.map\_widget.set\_marker(70.170, 72.514, text="Сеяха", text\_color="black",

marker\_color\_circle="black", marker\_color\_outside="gray40",

font=("Helvetica Bold", 8), command=self.click\_marker)

marker\_11 = self.map\_widget.set\_marker(77.717, 104.300, text="Мыс Челюскин", text\_color="black",

marker\_color\_circle="black", marker\_color\_outside="gray40",

font=("Helvetica Bold", 8), command=self.click\_marker)

marker\_12 = self.map\_widget.set\_marker(71.875, 82.706, text="Мыс Сопочная Карга", text\_color="black",

marker\_color\_circle="black", marker\_color\_outside="gray40",

font=("Helvetica Bold", 8), command=self.click\_marker)

marker\_13 = self.map\_widget.set\_marker(80.617, 58.050, text="Остров Хейса", text\_color="black",

marker\_color\_circle="black", marker\_color\_outside="gray40",

font=("Helvetica Bold", 8), command=self.click\_marker)

marker\_14 = self.map\_widget.set\_marker(69.783, 61.783, text="Амдерма", text\_color="black",

marker\_color\_circle="black", marker\_color\_outside="gray40",

font=("Helvetica Bold", 8), command=self.click\_marker)

self.station\_title = customtkinter.CTkLabel(self.tabview.tab("Select a station"),

text="Selected station",

font=customtkinter.CTkFont(size=14, weight="bold"))

self.station\_title.grid(row=0, column=1, columnspan=1, padx=(10, 10), pady=(5, 0), sticky='s')

self.station\_entry = customtkinter.CTkEntry(self.tabview.tab("Select a station"),

placeholder\_text="station")

self.station\_entry.grid(row=1, column=1, columnspan=1, padx=(10, 10), pady=(5, 0), sticky="s")

# Import tab

self.file\_name = customtkinter.CTkLabel(self.tabview.tab("Import"),

text=f'Required data structure: date, maximum temperature,'

f' minimum temperature, average temperature, wind speed,'

f' precipitation, effective temperature',

font=customtkinter.CTkFont(size=14, weight="bold"))

self.file\_name.grid(row=0, column=0, padx=(20, 20), pady=(10, 10))

self.import\_buttom = customtkinter.CTkButton(self.tabview.tab("Import"), text="Select file", width=1000,

font=customtkinter.CTkFont(size=16, weight="bold"),

command=self.import\_buttom\_event)

self.import\_buttom.grid(row=1, column=0, padx=(20, 20), pady=(10, 10), sticky="ew")

# Search homologues tab

self.search\_homologues\_buttom = customtkinter.CTkButton(self.tabview.tab("Search homologues"),

text="Search homologues", width=1000,

font=customtkinter.CTkFont(size=16, weight="bold"),

command=self.search\_homologues\_buttom\_event)

self.search\_homologues\_buttom.grid(row=0, column=0, padx=(20, 20), pady=(10, 10), sticky="ew")

# Select Date tab

# start date title

self.start\_date\_title = customtkinter.CTkLabel(self.tabview.tab("Select date"),

text="Start Date",

font=customtkinter.CTkFont(size=14, weight="bold"))

self.start\_date\_title.grid(row=0, column=0, columnspan=1, padx=(140, 20), pady=10, sticky='w')

# start date calander

self.start\_cal = Calendar(self.tabview.tab("Select date"), selectmode='day',

showweeknumbers=False, cursor="hand2", date\_pattern='y-mm-dd',

borderwidth=0, bordercolor='white')

self.start\_cal.grid(row=1, column=0, padx=(100, 20), pady=10, sticky='w')

# end date title

self.end\_date\_title = customtkinter.CTkLabel(self.tabview.tab("Select date"),

text="End Date", font=customtkinter.CTkFont(size=14, weight="bold"))

self.end\_date\_title.grid(row=0, column=1, columnspan=1, padx=(20, 140), pady=10, sticky='e')

# end date calander

self.end\_cal = Calendar(self.tabview.tab("Select date"), selectmode='day',

showweeknumbers=False, cursor="hand2", date\_pattern='y-mm-dd',

borderwidth=0, bordercolor='white')

self.end\_cal.grid(row=1, column=1, padx=(20, 100), pady=10, sticky='e')

# start date entry

self.start\_date\_entry = customtkinter.CTkEntry(self.tabview.tab("Select date"),

placeholder\_text="Start Date")

self.start\_date\_entry.grid(row=3, column=0, padx=(100, 20), pady=10, sticky="w")

# end date entry

self.end\_date\_entry = customtkinter.CTkEntry(self.tabview.tab("Select date"),

placeholder\_text="End Date")

self.end\_date\_entry.grid(row=3, column=1, padx=(20, 100), pady=10, sticky="e")

# date confirm button

self.confirm\_date\_buttom = customtkinter.CTkButton(self.tabview.tab("Select date"),

text="Confirm Dates", hover=True,

font=customtkinter.CTkFont(size=16, weight="bold"),

command=self.fetch\_dates\_overall)

self.confirm\_date\_buttom.grid(row=2, column=0, columnspan=2, padx=10, pady=30)

# for Forecast tab

self.forecast\_buttom = customtkinter.CTkButton(self.tabview.tab("Select date"), text="Predict values", width=200,

font=customtkinter.CTkFont(size=16, weight="bold"),

command=self.forecast\_buttom\_event)

self.forecast\_buttom.grid(row=4, column=0, columnspan=2, padx=20, pady=(20, 10), sticky="nsew")

self.optionmenu\_plots = customtkinter.CTkOptionMenu(self.tabview.tab("Select date"), dynamic\_resizing=False,

values=["maximum temperature", "minimum temperature", "average temperature",

"wind speed", "precipitation", "effective temperature"],

width=200)

self.optionmenu\_plots.grid(row=5, column=0, padx=20, pady=(10, 10), sticky="nsew")

self.plots\_buttom = customtkinter.CTkButton(self.tabview.tab("Select date"), text="Show forecast plot", width=200,

font=customtkinter.CTkFont(size=16, weight="bold"),

command=self.plots\_buttom\_event)

self.plots\_buttom.grid(row=5, column=1, padx=20, pady=(10, 10), sticky="nsew")

# Download tab

self.show\_forecast\_data\_buttom = customtkinter.CTkButton(self.tabview.tab("Download"), text="Show forecast data", width=470,

font=customtkinter.CTkFont(size=16, weight="bold"),

command=self.show\_forecast\_data\_event)

self.show\_forecast\_data\_buttom.grid(row=0, column=0, padx=20, pady=(10, 10), sticky="ew")

self.download\_buttom = customtkinter.CTkButton(self.tabview.tab("Download"), text="Download", width=470,

font=customtkinter.CTkFont(size=16, weight="bold"),

command=self.download\_buttom\_event)

self.download\_buttom.grid(row=0, column=1, padx=20, pady=(10, 10), sticky="ew")

# buttoms

# choose station

def click\_marker(self, marker):

station = marker.text

self.station\_entry.configure(placeholder\_text=station)

if station in station\_dict.keys():

name = station\_dict.get(station)

path = str(Path(\_\_file\_\_).parent) + f'/stations/{name}\_after\_preprocessing.csv'

global abs\_path

abs\_path = os.path.abspath(path)

global station\_name

station\_name = name

print("marker clicked")

# import file

def import\_buttom\_event(self):

global filename

filename = customtkinter.filedialog.askopenfilename()

# first date value

f\_read = open(filename, "r")

last\_date = f\_read.readlines()[-1].split(',')[0]

global first\_date

first\_date = datetime.datetime.strptime(last\_date, '%Y-%m-%d').date() + datetime.timedelta(days=1)

self.start\_cal.selection\_set(first\_date)

self.start\_cal.configure(state='disabled', disabledselectbackground='steelblue')

self.end\_cal.selection\_set(first\_date)

# Table style

style = tkinter.ttk.Style()

style.theme\_use("default")

style.configure("Treeview",

background="#2a2d2e",

foreground="white",

rowheight=25,

fieldbackground="#343638",

bordercolor="#343638",

borderwidth=0)

style.map('Treeview', background=[('selected', '#22559b')])

style.configure("Treeview.Heading",

background="#565b5e",

foreground="white",

relief="flat")

style.map("Treeview.Heading",

background=[('active', '#3484F0')])

# Table

tree = tkinter.ttk.Treeview(self.tabview.tab("Import"), show="headings", height=100)

tree.grid(row=3, column=(0), padx=(20, 20), pady=(10, 20))

status\_label = tkinter.Label(self.tabview, text="", padx=20, pady=10)

with open(filename, 'r', newline='') as file:

csv\_reader = csv.reader(file)

header = next(csv\_reader) # Read the header row

tree.delete(\*tree.get\_children()) # Clear the current data

tree["columns"] = header

for col in header:

tree.heading(col, text=col)

tree.column(col, width=150)

for row in csv\_reader:

tree.insert("", "end", values=row)

status\_label.config(text=f"CSV file loaded: {filename}")

print("import\_buttom clicked")

#

def search\_homologues\_buttom\_event(self):

global homologues

homologues, years\_dict = main.homofind(abs\_path, filename)

cluster\_figs = homofind\_cluster.visualize\_clusters(years\_dict, homologues)

print(homologues)

# cluster\_figs - list с названиями файлов - графиков кластеров для каждого параметра

path = str(Path(\_\_file\_\_).parent) + '/' +cluster\_figs[0]

my\_image = customtkinter.CTkImage(light\_image=Image.open(path),

dark\_image=Image.open(path),

size=(600, 450))

image\_label = customtkinter.CTkLabel(self.tabview.tab("Search homologues"), image=my\_image, text="")

image\_label.grid(row=1, column=0, padx=(20, 20), pady=(10, 20))

# select date

def fetch\_dates\_overall(self):

global date1

date1 = self.start\_cal.get\_date()

self.start\_date\_entry.configure(placeholder\_text=date1)

self.start\_date\_entry.configure(state='readonly')

global date2

date2 = self.end\_cal.get\_date()

self.end\_date\_entry.configure(placeholder\_text=date2)

self.end\_date\_entry.configure(state='readonly')

global n\_days

n\_days = pd.to\_datetime(date2) - pd.to\_datetime(date1)

n\_days = n\_days.days

print(n\_days)

print('confirm\_date\_buttom clicked')

# forecast plots

def forecast\_buttom\_event(self):

global preds

global pred\_figs

preds, pred\_figs = main.get\_predicts(abs\_path, filename, n\_days, homologues, station\_name)

print("forecast\_buttom clicked")

def plots\_buttom\_event(self):

parameter = self.optionmenu\_plots.get()

if parameter == "maximum temperature":

parameter\_col = 0

label = 'temperature'

title = "maximum\_temperature"

elif parameter == "minimum temperature":

parameter\_col = 1

label = 'temperature'

title = "minimum\_temperature"

elif parameter == "average temperature":

parameter\_col = 1

label = 'temperature'

title = "average\_temperature"

elif parameter == "wind speed":

parameter\_col = 3

label = 'wind speed'

title = "wind\_speed"

elif parameter == "precipitation":

parameter\_col = 4

label = 'precipitation'

title = "precipitation"

elif parameter == "effective temperature":

parameter\_col = 5

label = 'temperature'

title = "effective\_temperature"

path\_pred = rf'{pred\_figs[title]}'

path\_pred = os.path.abspath(path\_pred)

my\_image = customtkinter.CTkImage(light\_image=Image.open(path\_pred),

dark\_image=Image.open(path\_pred),

size=(800, 500))

image\_label = customtkinter.CTkLabel(self.tabview.tab("Forecast plots"), image=my\_image, text="")

image\_label.grid(row=1, column=0, columnspan=2, padx=(20, 20), pady=(10, 20))

print("plots\_buttom clicked")

# show\_forecast\_data

def show\_forecast\_data\_event(self):

# Table style

style = tkinter.ttk.Style()

style.theme\_use("default")

style.configure("Treeview",

background="#2a2d2e",

foreground="white",

rowheight=25,

fieldbackground="#343638",

bordercolor="#343638",

borderwidth=0)

style.map('Treeview', background=[('selected', '#22559b')])

style.configure("Treeview.Heading",

background="#565b5e",

foreground="white",

relief="flat")

style.map("Treeview.Heading",

background=[('active', '#3484F0')])

# Table

tablefile = main.dict\_to\_df(preds)

tablefile.reset\_index(inplace= True)

tablefile = tablefile.rename(columns={'index': 'date'})

tablefile['date'] = pd.to\_datetime(tablefile['date']).dt.date

columns = tablefile.columns.values.tolist()

print(tablefile)

tree = tkinter.ttk.Treeview(self.tabview.tab("Download"), columns=columns, show="headings", height=100)

tree.grid(row=1, column=0, columnspan=2, padx=(20, 20), pady=(10, 20))

status\_label = tkinter.Label(self.tabview, text="", padx=20, pady=20)

for col in columns:

tree.heading(col,text=col)

tree.column(col, width=100)

for index, row in tablefile.iterrows():

tree.insert("", END, values=row.tolist())

status\_label.config(text=f"CSV file loaded: {filename}")

print("show\_forecast\_data clicked")

# download forecast data

def download\_buttom\_event(self):

types = [("CSV files", ".csv")]

dataFile = main.dict\_to\_df(preds)

SAVING\_PATH = tkinter.filedialog.asksaveasfile(defaultextension=".csv", filetypes=types)

dataFile.to\_csv(SAVING\_PATH, lineterminator='\n')