Оглавление

[1. Введение 2](#_Toc167407268)

[Обзор приложения прогноза погоды 2](#_Toc167407269)

[Цель и область применения документации 2](#_Toc167407270)

[2. Установка и настройка 2](#_Toc167407271)

[2.1 Предварительные требования 2](#_Toc167407272)

[2.2 Клонирование репозитория проекта с GitHub 2](#_Toc167407273)

[2.3 Шаги по установке необходимых пакетов с использованием pip 2](#_Toc167407274)

[2.4 Инструкции по настройке окружения 3](#_Toc167407275)

[3. Архитектура приложения 4](#_Toc167407276)

[3.1 Обзор структуры приложения 4](#_Toc167407277)

[3.2 Описание каждого модуля 4](#_Toc167407278)

[3.2.1 model.py 4](#_Toc167407279)

[3.2.2 Подробное описание кода в model.py 4](#_Toc167407280)

[3.2.3 app.py 5](#_Toc167407281)

[4. Оценка модели 6](#_Toc167407282)

[4.1 Подробная информация о производительности модели 6](#_Toc167407283)

[4.2 Интерпретация метрик оценки 6](#_Toc167407284)

[5. Сохранение и загрузка модели 7](#_Toc167407285)

[5.1 Инструкции по сохранению модели с использованием joblib 7](#_Toc167407286)

[5.2 Шаги по загрузке модели для дальнейшего использования 7](#_Toc167407287)

[6. Приложение прогноза погоды 7](#_Toc167407288)

[6.1 Обзор функциональности скрипта app.py 7](#_Toc167407289)

[6.2 Объяснение пользовательского интерфейса на основе Streamlit 7](#_Toc167407290)

[6.3 Генерация прогноза на следующие 7 дней 9](#_Toc167407291)

[7. Запуск приложения 11](#_Toc167407292)

[7.1 Инструкции по запуску приложения Streamlit 11](#_Toc167407293)

[7.2 Примеры использования и ожидаемые результаты 11](#_Toc167407294)

[8. Устранение неполадок и часто задаваемые вопросы 12](#_Toc167407295)

[8.1Общие проблемы и их решения 12](#_Toc167407296)

[8.2 Часто задаваемые вопросы 13](#_Toc167407297)

[9. Будущие улучшения 13](#_Toc167407298)

[9.1 Потенциальные улучшения и функции для будущих версий 13](#_Toc167407299)

# 1. Введение

## Обзор приложения прогноза погоды

Это приложение предназначено для предсказания погоды на основе исторических данных. Оно использует методы машинного обучения для построения моделей, которые могут предсказывать температуру на заданную дату и в заданном городе.

## Цель и область применения документации

Цель данной документации - предоставить подробное руководство по установке, настройке, использованию и модификации приложения. Она предназначена для разработчиков, которые хотят понять, как работает приложение, и для пользователей, желающих использовать его для предсказания погоды.

# 2. Установка и настройка

## 2.1 Предварительные требования

• **Python 3.8 или выше**: Для корректной работы приложения необходима версия Python 3.8 или выше.

• **Необходимые библиотеки**: pandas, numpy, scikit-learn, joblib, streamlit, plotly. Эти библиотеки обеспечивают функциональность приложения, включая обработку данных, обучение модели и построение пользовательского интерфейса.

## 2.2 Клонирование репозитория проекта с GitHub

Для начала работы с проектом необходимо клонировать его репозиторий.

1)git clone https://github.com/your-repository/weather-forecast-app.git

2) cd weather-forecast-app

Блок кода 1. Клонирование репозитории

Этот шаг позволяет скачать проект на ваш локальный компьютер и перейти в директорию проекта.

## 2.3 Шаги по установке необходимых пакетов с использованием pip

Для установки всех необходимых пакетов используйте команду:

pip install -r requirements.txt

Блок кода 2. Установка пакетов

Эта команда установит все зависимости, перечисленные в файле requirements.txt.

Файл **requirements.txt** содержит:

* pandas
* scikit-learn
* joblib
* matplotlib
* streamlit
* plotly
* pydeck
* geopy

## 2.4 Инструкции по настройке окружения

Убедитесь, что среда настроена правильно, проверив установку необходимых пакетов. Для этого выполните следующие действия:

- Загрузите скрипт проверки:

"**check\_packages.py**".

Запустите сценарий проверки, чтобы убедиться, что все необходимые пакеты установлены:

python check\_packages.py

Блок кода 3. Запуск сценарий проверки

Сценарий выведет статус установки каждого требуемого пакета. Убедитесь, что все пакеты считаются установленными. Если какой-либо пакет отсутствует, вам может потребоваться установить его вручную, используя:

**Объяснение сценария проверки**

Скрипт проверки check\_packages.py проверяет наличие каждого требуемого пакета и сообщает, установлен он или нет. Здесь приведено подробное объяснение этого скрипта:

import subprocess

def check\_installed\_packages():

required\_packages = [

"pandas",

"scikit-learn",

"joblib",

"matplotlib",

"streamlit",

"plotly",

"pydeck",

"geopy"

for package in required\_packages:

try:

\_\_import\_\_(package)

print(f"{package} is installed.")

except ImportError:

print(f"{package} is NOT installed.")

if \_\_name\_\_ == "\_\_main\_\_":

check\_installed\_packages()

Блок кода 4. **Объяснение сценария проверки**

# 3. Архитектура приложения

## 3.1 Обзор структуры приложения

Приложение состоит из двух основных файлов: app.py и model.py. Файл model.py отвечает за подготовку данных и обучение модели, а файл app.py содержит код для веб-интерфейса приложения.

## 3.2 Описание каждого модуля

### 3.2.1 model.py

Содержит код для подготовки данных и построения модели машинного обучения.

### 3.2.2 Подробное описание кода в model.py

**Подробное описание кода:**

1. **Загрузка необходимых библиотек**: Импортируются библиотеки для работы с данными (pandas), разделения данных на тренировочные и тестовые выборки (train\_test\_split), построения модели линейной регрессии (LinearRegression), оценки модели (mean\_squared\_error, r2\_score), обработки категориальных признаков (OneHotEncoder, ColumnTransformer), создания конвейера (Pipeline), сохранения модели (joblib) и визуализации (matplotlib.pyplot).

2. **Загрузка данных**: Данные загружаются из файлов daily\_weather.parquet и cities.csv.

3. **Преобразование столбца даты в тип datetime**: Дата преобразуется в формат datetime.

4. **Фильтрация данных за период с 2013 по 2023 год**: Отбираются данные за указанный период.

5. **Извлечение признаков из даты**: Извлекаются год, месяц и день из даты.

6. **Заполнение пропущенных значений**: Пропущенные значения в столбцах температуры заполняются средними или медианными значениями.

7. **Определение признаков и целевой переменной**: Признаки (независимые переменные) определяются в X, а целевая переменная (зависимая переменная) - в y.

8. **Преобразование категориальных признаков**: Категориальные признаки преобразуются в числовые с помощью OneHotEncoder.

9. **Создание конвейера для подготовки данных и обучения модели**: Конвейер включает шаги предобработки данных и обучение модели линейной регрессии.

10. **Разделение данных на тренировочные и тестовые выборки**: Данные разделяются на тренировочные и тестовые выборки с соотношением 80:20.

11. **Обучение модели**: Модель обучается на тренировочных данных.

12. **Предсказание на тестовых данных**: Модель делает предсказания на тестовых данных.

13. **Оценка модели с использованием MSE и R-squared**: Оцениваются метрики MSE и R².

14. **Сохранение модели и данных**: Модель и данные сохраняются в файлы с использованием joblib.

### 3.2.3 app.py

Содержит код для веб-интерфейса приложения, реализованного с использованием Streamlit.

1. **Настройка конфигурации страницы**: Устанавливаются параметры страницы приложения с помощью st.set\_page\_config.

2. **Загрузка модели и данных**: Загружаются обученная модель и данные из файлов temperature\_predictor.pkl, weather\_data.pkl, cities\_data.pkl и aggregated\_predictions.pkl.

3. **Функция для получения флага страны**: Функция get\_flag принимает код страны и возвращает соответствующий флаг.

4. **Функция для преобразования температуры из Цельсия в Фаренгейты**: Функция celsius\_to\_fahrenheit преобразует температуру из градусов Цельсия в градусы Фаренгейта.

5. **Заголовок приложения**: Устанавливается заголовок приложения с помощью st.title.

6. **Создание колонок для расположения элементов интерфейса**: Используются st.columns для создания двух колонок.

7. **Ввод параметров пользователем**:

• В первой колонке пользователь выбирает страну и город, а также дату.

• Во второй колонке пользователь выбирает единицы измерения температуры (Цельсий или Фаренгейт).

8. **Получение кода страны и флага**: Код страны используется для получения флага с помощью функции get\_flag.

9. **Отображение заголовка с выбранными параметрами**: Заголовок отображает выбранную страну, город и дату.

10. **Проверка наличия данных о погоде**: Если данные о погоде для выбранной даты и города имеются в weather\_data, отображается средняя температура.

11. **Если данных нет, выполняется предсказание**: Если данные отсутствуют, используется aggregated\_predictions для получения предсказанной температуры.

12. **Заголовок для прогноза на 7 дней**: Отображается заголовок для прогноза на следующую неделю.

13. **Генерация прогноза на следующие 7 дней**: Прогнозируются температуры на следующие 7 дней с использованием aggregated\_predictions.

14. **Создание и отображение DataFrame для прогноза**: Прогнозы отображаются в виде таблицы с помощью st.write.

15. **Построение графика изменения температуры**: Если данных достаточно, строится график изменения температуры на неделю с помощью Plotly.

# 4. Оценка модели

## 4.1 Подробная информация о производительности модели

Модель оценивается на тестовых данных с использованием метрик MSE и R². Эти метрики помогают понять, насколько хорошо модель справляется с предсказанием температуры.

predictions = model.predict(X\_test)

mse = mean\_squared\_error(y\_test, predictions)

r2 = r2\_score(y\_test, predictions)

print(f"Mean Squared Error: {mse}")

print(f"R-squared: {r2}")

## 4.2 Интерпретация метрик оценки

• **Mean Squared Error (MSE)**: Чем меньше значение MSE, тем лучше модель предсказывает значения.

• **R-squared (R²)**: Значение R² ближе к 1 указывает на хорошую модель, в то время как значение, близкое к 0, указывает на то, что модель плохо объясняет дисперсию данных.

# 5. Сохранение и загрузка модели

## 5.1 Инструкции по сохранению модели с использованием joblib

Модель сохраняется в файл temperature\_predictor.pkl для дальнейшего использования.

## 5.2 Шаги по загрузке модели для дальнейшего использования

Сохраненная модель загружается из файла temperature\_predictor.pkl.

model = joblib.load('temperature\_predictor.pkl')

weather\_data = joblib.load('weather\_data.pkl')

cities\_data = joblib.load('cities\_data.pkl')

aggregated\_predictions = joblib.load('aggregated\_predictions.pkl')

# 6. Приложение прогноза погоды

## 6.1 Обзор функциональности скрипта app.py

Скрипт app.py отвечает за создание веб-интерфейса для взаимодействия с пользователем. Интерфейс позволяет пользователю вводить данные (год, месяц, день и город) и получать предсказанную температуру.

## 6.2 Объяснение пользовательского интерфейса на основе Streamlit

• **Настройка конфигурации страницы**: Устанавливаются параметры страницы приложения с помощью st.set\_page\_config.

st.set\_page\_config(

page\_title="Weather Forecast",

page\_icon="🌤",

layout="centered",

initial\_sidebar\_state="expanded",

)

• **Загрузка модели и данных**: Загружаются обученная модель и данные из файлов temperature\_predictor.pkl, weather\_data.pkl, cities\_data.pkl и aggregated\_predictions.pkl.

model = joblib.load('temperature\_predictor.pkl')

weather\_data = joblib.load('weather\_data.pkl')

cities\_data = joblib.load('cities\_data.pkl')

aggregated\_predictions = joblib.load('aggregated\_predictions.pkl')

* **Функция для получения флага страны**: Функция get\_flag принимает код страны и возвращает соответствующий флаг.

def get\_flag(country\_code):

offset = ord('🇦') - ord('A')

return chr(ord(country\_code[0]) + offset) + chr(ord(country\_code[1]) + offset)

• **Функция для преобразования температуры из Цельсия в Фаренгейты**: Функция celsius\_to\_fahrenheit преобразует температуру из градусов Цельсия в градусы Фаренгейта.

def celsius\_to\_fahrenheit(celsius):

return celsius \* 9/5 + 32

• **Заголовок приложения**: Устанавливается заголовок приложения с помощью st.title.

st.title("🌤 Weather Forecast 🌤")

• **Создание колонок для расположения элементов интерфейса**: Используются st.columns для создания двух колонок.

col1, col2 = st.columns([5, 1])

• **Ввод параметров пользователем**:

• В первой колонке пользователь выбирает страну и город, а также дату.

• Во второй колонке пользователь выбирает единицы измерения температуры (Цельсий или Фаренгейт).

with col1:

country = st.selectbox("🌍 Select Country", cities\_data['country'].unique(), key="country")

cities\_in\_country = cities\_data[cities\_data['country'] == country]['city\_name']

city = st.selectbox("🏙 Select City", cities\_in\_country, key="city")

selected\_date = st.date\_input("📅 Select Date", key="date")

with col1:

temp\_unit = st.selectbox("Show weather in", ["Celsius", "Fahrenheit"], key="temp\_unit")

• **Получение кода страны и флага**: Код страны используется для получения флага с помощью функции get\_flag.

country\_code = cities\_data[cities\_data['country'] == country]['iso2'].values[0]

flag = get\_flag(country\_code)

• **Отображение заголовка с выбранными параметрами**: Заголовок отображает выбранную страну, город и дату.

st.subheader(f"{flag} Temperature for {country} - {city} on {selected\_date.strftime('%d.%m.%Y')}:")

## 6.3 Генерация прогноза на следующие 7 дней

• **Заголовок для прогноза на 7 дней**: Отображается заголовок для прогноза на следующую неделю.

st.subheader("Weather Forecast for the Next 7 Days:")

• **Генерация прогноза на следующие 7 дней**: Прогнозируются температуры на следующие 7 дней с использованием aggregated\_predictions.

next\_7\_days = pd.date\_range(selected\_date, periods=7)

forecast = []

for date in next\_7\_days:

month = date.month

day = date.day

prediction = aggregated\_predictions[(aggregated\_predictions['city\_name'] == city) &

(aggregated\_predictions['month'] == month) &

(aggregated\_predictions['day'] == day)]

if not prediction.empty:

avg\_temp = prediction['avg\_predicted\_temp'].values[0]

if temp\_unit == "Fahrenheit":

avg\_temp = celsius\_to\_fahrenheit(avg\_temp)

forecast.append((date.strftime("%d.%m.%Y"), date.strftime("%a"), f"{avg\_temp:.2f} °{temp\_unit[0]}"))

else:

forecast.append((date.strftime("%d.%m.%Y"), date.strftime("%a"), "No data"))

**Построение графика изменения температуры**: Если данных достаточно, строится график изменения температуры на неделю с помощью Plotly.

if len(forecast\_df.columns) > 1:

dates = forecast\_df.columns[1:]

temps = forecast\_df.iloc[1, 1:].str.replace(f' °{temp\_unit[0]}', '').astype(float)

fig = go.Figure()

fig.add\_trace(go.Scatter(

x=dates,

y=temps,

mode='lines+markers+text',

text=[f'{temp:.2f}°{temp\_unit[0]}' for temp in temps],

textposition='top center',

line=dict(color='royalblue', width=2),

marker=dict(color='red', size=10)

))

fig.update\_layout(

title={

'text': 'Average Temperature Changes During the Week',

'x': 0.5,

'xanchor': 'center'

},

xaxis\_title='Date',

yaxis\_title=f'Avg Temperature (°{temp\_unit[0]})',

template='plotly\_dark',

plot\_bgcolor='rgba(0,0,0,0)',

paper\_bgcolor='rgba(0,0,0,0)',

)

st.plotly\_chart(fig)

else:

st.write("❌ Insufficient data to plot the graph.")

**Подробное описание кода:**

1. **Проверка наличия данных для построения графика**: Код проверяет, достаточно ли данных для построения графика изменения температуры.

2. **Извлечение дат и температур**: Из таблицы прогноза извлекаются даты и температуры.

3. **Создание графика с использованием Plotly**: Создается график с линиями и маркерами для отображения изменения температуры на протяжении недели.

4. **Настройка графика**: Настраиваются заголовки осей, шаблон и цвета графика.

5. **Отображение графика**: График отображается в приложении с помощью st.plotly\_chart.

# 7. Запуск приложения

## 7.1 Инструкции по запуску приложения Streamlit

Для запуска приложения выполните следующую команду в терминале:

streamlit run app.py

Эта команда запустит локальный сервер, на котором будет доступен веб-интерфейс приложения.

## 7.2 Примеры использования и ожидаемые результаты

Пример использования:

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

Автоматически созданное описание

# 8. Устранение неполадок и часто задаваемые вопросы

## 8.1Общие проблемы и их решения

• **Ошибка при загрузке данных**: Убедитесь, что файл weather\_data.pkl находится в правильном месте и доступен для чтения.

• **Ошибка при установке зависимостей**: Проверьте, что у вас установлена правильная версия Python и что все библиотеки указаны в requirements.txt.

## 8.2 Часто задаваемые вопросы

• **Вопрос**: Как изменить набор данных?

**Ответ**: Замените файл weather\_data.pkl новым набором данных и обновите код для предобработки, если это необходимо.

• **Вопрос**: Как изменить модель предсказания?

**Ответ**: Измените код в model.py, заменив LinearRegression другой моделью из библиотеки scikit-learn или любой другой библиотеки машинного обучения.

# 9. Будущие улучшения

## 9.1 Потенциальные улучшения и функции для будущих версий

• Добавить поддержку большего количества городов.

• Улучшить пользовательский интерфейс с использованием более сложных визуализаций.

• Добавить возможность предсказания других метеорологических параметров, таких как влажность или осадки.

**10.** Дополнительные ресурсы и ссылки

• [Документация Streamlit](https://docs.streamlit.io/)

• [Документация scikit-learn](https://scikit-learn.org/stable/index.html)

• [Документация pandas](https://pandas.pydata.org/docs/)