Министерство образования и науки Российской Федерации

Санкт-Петербургский политехнический университет Петра Великого

Высшая школа киберфизических систем и управления ИКНТ

Работа допущена к защите

Руководитель ОП

\_\_\_\_\_\_\_\_\_\_\_ А.В. Милицын

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_2023 г.

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

**РАБОТА БАКАЛАВРА**

**ВЕБ-ПРИЛОЖЕНИЕ ДЛЯ ИДЕНТИФИКАЦИИ ДИНАМИЧЕСКИХ СИСТЕМ**

направление 27.03.04 - Управление в технических системах

профиль 27.03.04\_05 - Интеллектуальные системы обработки информации и

управления

Выполнил

студент гр.3532704/90501 А.М. Мустафин

Руководитель

доцент В.М. Филиповский

Консультант

по нормоконтролю Л.А. Киселёва

Санкт-Петербург

2023

РЕФЕРАТ

На 49 с., 22 рисунка, 4 приложения.

КЛЮЧЕВЫЕ СЛОВА: ИДЕНТИФИКАЦИЯ СИСТЕМ, ДИНАМИЧЕСКАЯ СИСТЕМА, ВЕБ-ПРИЛОЖЕНИЕ,

Тема выпускной квалификационной работы: «Веб-приложение для идентификации динамических систем».

Данная работа посвящена проектированию веб-приложения для предоставления доступного инструмента, который позволяет решать задачи идентификации систем. Задачи, которые решались в ходе исследования:

1. Изучить работу автоматизированного теплового;
2. Изучить методы машинного обучения для прогнозирования временных рядов;
3. Проанализировать предоставленные данные для обучения, предобработать их;
4. Реализовать программные модули для прогнозирования одномерных временных рядов, используя изученные методы машинного обучения;
5. Сравнить качество получившихся моделей машинного обучения для прогнозирования одномерных временных рядов, оптимизировать модели, выбрать оптимальную;
6. Реализовать программный модуль прогнозирования технологических параметров.

Работа выполнена на базе автоматизированного теплового пункта. Были проведены исследования в области прогнозирования временных рядов и применены различные методы машинного обучения. Все модели методы реализованы на языке Python 3. Выбрана модель с наилучшим качеством предсказаний – LSTM. Эта модель встроена в модуль предиктивного управления.

ABSCTRACT

49 pages, 22 figures, 4 appendices.

KEYWORDS: PREDICTIVE CONTROL, TIME SERIES FORECASTING, LSTM, ARTIFICIAL NEURAL NETWORK, GRADIENT BOOSTING OVER DECISION TREES, RANDOM FOREST.

Topic of the final qualification work: "Forecasting the technological parameters of an automated heat supply unit using machine learning methods."

This work is devoted to research in the field of time series forecasting to create a predictive control module. Tasks that were solved during the study:

1. Explore the work of an automated heat supply unit;
2. Study machine learning methods for time series forecasting;
3. Analyze the provided data for training, pre-process it;
4. Implement software modules for predicting one-dimensional time series using the studied methods of machine learning;
5. Compare the quality of the resulting machine learning models for predicting one-dimensional time series, optimize the models, choose the optimal one;
6. Implement a software module for predicting technological parameters.

The work was carried out of an automated heat supply unit. Research has been done in the field of time series forecasting and various machine learning methods have been applied. All model methods are implemented in Python 3. The model with the best quality of predictions, LSTM, is selected. This model is built into the predictive control module.

**Содержание**

[**Введение** 4](#_Toc136291902)

[**1.** **Об идентификации динамических систем** 7](#_Toc136291903)

[**1.1** **Общие сведения** 7](#_Toc136291904)

[**1.2** **Основные типы моделей и их использование** 7](#_Toc136291905)

[**1.3** **Построение моделей** 8](#_Toc136291906)

[**1.4** **Основные компоненты процедуры идентификации систем** 9](#_Toc136291907)

[**2.** **Методы прогнозирования временных рядов** 10](#_Toc136291908)

[**2.1 Случайный лес** 10](#_Toc136291909)

[2.1.1 Решающие деревья 10](#_Toc136291910)

[2.1.2 Бэггинг над решающими деревьями 10](#_Toc136291911)

[**2.2 Градиентный бустинг над решающими деревьями** 10](#_Toc136291912)

[**1.3** **Искусственные глубокие нейронные сети** 10](#_Toc136291913)

[2.3.1 Общие сведения 10](#_Toc136291914)

[2.3.2 Обучение нейронных сетей 10](#_Toc136291915)

[**2.4 Искусственные рекуррентные нейронные сети LSTM** 10](#_Toc136291916)

[2.4.1 Общие положения о рекуррентных нейронных сетях 10](#_Toc136291917)

[**3.** **Сравнение методов прогнозирования** 11](#_Toc136291918)

[**1.1.** **Выбор метрики и оптимизация моделей** 11](#_Toc136291919)

[**1.2.** **Сравнение моделей** 11](#_Toc136291920)

[**Заключение** 11](#_Toc136291921)

[**Список литературы** 11](#_Toc136291922)

[**Приложение А.** 13](#_Toc136291923)

[**Модель случайного леса** 13](#_Toc136291924)

[**Приложение Б.** 16](#_Toc136291925)

[**Модель градиентного бустинга** 16](#_Toc136291926)

[**Приложение В.** 19](#_Toc136291927)

[**Модель искусственной нейронной сети** 19](#_Toc136291928)

[**Приложение Г.** 22](#_Toc136291929)

[**Модель LSTM** 22](#_Toc136291930)

# **Введение**

Веб-приложение для идентификации динамических систем обеспечит пользователя интерактивным, доступным и дружественным интерфейсом для выполнения различных задач в области идентификации. Приложение позволит оценивать параметры системы модели на основе входных и выходных данных, анализировать поведение системы.

*Цели данной работы:* облегчение задач идентификации систем, предоставление независимого и доступного программного решения в виде веб-приложения.

*Актуальность бакалаврской работы* заключается в том, что веб-приложение для идентификации систем поддерживает концепцию импортозамещения, предоставляя самодостаточное и надежное решение для задач идентификации систем. Импортозамещение направлено на снижение зависимости от иностранного программного обеспечения или инструментов путем содействия в разработке отечественных альтернатив. Такой подход способствует технологической независимости, обеспечивает доступность инструментов идентификации систем, адаптированных к конкретным потребностям местного сообщества.

В разделе 1 приведены общие сведения о науке идентификации систем. В этом разделе определяются основные термины, такие как: система, динамическая система, идентификация и прочие. Рассматриваются типы моделей, описывающие системы. Также уделяется внимание о способах построения модели.

Основные функции включают в себя: удобный интерфейс пользователя – веб-приложение обладает интуитивно понятным интерфейсом, который упрощает процесс идентификации системы; легкий импорт данных – пользователи могут легко внести данные входных и выходных сигналов, как правило, в виде временных рядов; оценка модели системы – приложение содержит алгоритмы и методы для оценки параметров модели, например, с помощью метода наименьших квадратов, метод максимального правдоподобия или с помощью метода идентификации подпространства; экспорт модели в виде коэффициентов полинома числителя и знаменателя передаточной функции, вывод передаточной функции пользователю; оценка качества идентификации – приложение позволяет оценивать достоверность моделей, отображая основной критерий качества: функционал. Вдобавок, возможно провести валидацию модели с помощью методов остаточного анализа и оценки соответствия модели; вывод графиков, а именно: кривые входных и выходных сигналов, кривая переходного процесса, графики частотных характеристик.

Стоит отметить достоинства веб-приложения. Доступность – к приложению доступ возможен с любого устройства, подключенного к глобальной сети Интернет, этот аспект устраняет необходимость в установке ПО, а также избавляет пользователя от особых требований к его оборудованию. Возможность создания профиля – это позволяет сохранять идентифицированные модели систем в профиле пользователя. Поддержка пользователей – приложение может предоставить сопроводительную документацию, которая поможет пользователям эффективно использовать доступные функции.

Данное решение разработано с открытым исходным кодом, это значит, что исходный код находится в свободном для проверки и модификаций доступе. Соблюдение принципов открытого исходного кода в разработке веб-приложения способствует развитию культуры обмена знаниями и даёт пользователям продукта возможность использовать опыт сообщества инженеров-программистов в области идентификации систем.

*Актуальность бакалаврской работы* заключается в том, что веб-приложение для идентификации динамических систем раскрывает возможности методов идентификации более широкой аудитории, предоставляя доступную и удобную в использовании платформу. Это упрощает процесс оценки динамических моделей, анализа поведения системы и его возможного синтеза.

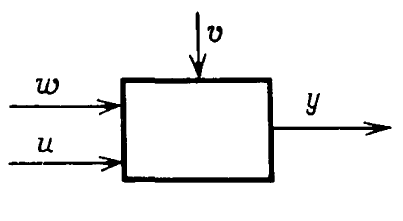
1. **Об идентификации динамических систем**
   1. **Общие сведения**

~~Данная~~ Наука идентификации исследует методы, которые позволяют формировать модели динамических систем на основе результатов наблюдений.

Система – это объект, в котором происходит взаимодействие между разнотипными переменными и формируются наблюдаемые сигналы (Льюинг)

Наблюдения представляют из себя входные и выходные данные. Входные данные – это любые управляющие воздействия, в том числе, шумы, представленные в формате временных рядов. Выходные данные – те данные, которые можно зафиксировать, например, приборами.

Динамические системы – такие системы, в которых текущее значение выходного сигнала зависит не только от текущих, но и более ранних значений внешних воздействий.



*y* – выходной сигнал; – неизмеримая помеха; – измеримая помеха;

*u –* входной сигнал управления

Рисунок 1 – Система в общем виде

* 1. **Основные типы моделей и их использование**

В задачах исследования динамических систем необходимо соотнести между собой переменные, которые характеризуют систему. Назовём совокупность предполагаемых связей между наблюдаемыми сигналами **моделью**. Обзорно рассмотрим типы моделей:

1. Умозрительные (субъективные). Наиболее часто применяются в бытовой жизни, в этом типе моделей нет математики. Например, для увеличения скорости езды при управлении велосипедом достаточно знать, что при увеличении частоты и силы вращения педалей, увеличится скорость движения;
2. Графические модели. Такие модели описывают свойства некоторых систем в виде графиков, диаграмм и таблиц. Например, графики статических характеристик могут описать некоторые нелинейные звенья: реле, клапаны. Линейные системы могут быть представлены импульсными реакциями, реакциями на ступенчатое воздействие или частотными характеристиками;
3. Аналитические (или математические) модели. В этих моделях соотношения, описывающие связи между системными переменными, задаются в виде разностных и дифференциальных уравнений. В зависимости от типа используемых разностных или дифференциальных уравнений такие модели, в свою очередь, могут быть непрерывными или дискретными по времени, сосредоточенные или распределенные, детерминированные или стохастические, линейные или нелинейные и т. д.;
4. Программные модели. Программа для ЭВМ также может быть моделью, т. к. она описывает поведение сложных систем на языке программирования. Это описание может быть представлено в виде совокупности взаимодействующих между собой подпрограмм.
   1. **Построение моделей**

Построение моделей основывается на данных наблюдений. Так, например, субъективная модель динамики управления движением велосипеда основана на личном опыте велосипедиста. Графические модели наполняются результатами некоторых измерений.

Для построения математических моделей используют один из двух способов. Первый способ состоит в том, чтобы провести декомпозицию системы на такие подсистемы, свойства которых очевидны из ранее накопленного опыта, например, из известных законов физики или других надежных и доказанных соотношений. Далее необходимо формально на математическом языке объединить эти подсистемы. Полученное описание становится моделью всей системы.

Суть второго способа построения как математических, так и графических моделей заключается в использовании экспериментальных данных. В этом случае ведется фиксация входных и выходных сигналов системы, и модель формируется в результате обработки соответствующих данных. Этот способ называется – **идентификация**.

* 1. **Основные компоненты процедуры идентификации систем**

Введем основные компоненты, которые возникают при проектировании модели системы по данным наблюдения:

1. Данные наблюдений. Входные и выходные данные регистрируются в процессе проведения целенаправленных идентификационных экспериментов, когда пользователь может задать перечень и моменты изменения сигналов.

2. Множество моделей. Множество моделей-кандидатов формируется в результате применения методов моделирования. В этом множестве выполняется поиск наиболее подходящей модели. На этом этапе знание формальных свойств моделей необходимо соединить с априорным знанием, инженерным искусством и интуицией. Набор моделей, у которых параметры рассматриваются прежде всего, как варьируемые средства подстройки моделей к имеющимся данным и не отражают физики процесса, называется *черным ящиком*. Множества моделей с настраиваемыми параметрами, допускающими физическую интерпретацию, называют *серыми ящиками*.

3. Правило оценки степени соответствия испытываемой модели данным наблюдений. Оценка качества модели связана, как правило, с изучением поведения моделей в процессе их использования для воссоздания данных наблюдений. На этом этапе производится выбор из множества моделей-кандидатов наилучшей модели.

1. **Веб-приложение**

## 2.1 Общие сведения

## 2.2 Виды веб-приложений

## 3.3 Сведения об HTTP

3.4 Архитектура

## 3.4 Flask

### 2.1.1 Решающие деревья

## 2.1.2 Бэггинг над решающими деревьями

## 2.2 Градиентный бустинг над решающими деревьями

* 1. **Искусственные глубокие нейронные сети**

### 2.3.1 Общие сведения

### 2.3.2 Обучение нейронных сетей

**2.4 Искусственные рекуррентные нейронные сети LSTM**

### 2.4.1 Общие положения о рекуррентных нейронных сетях

1. **Сравнение методов прогнозирования**
   1. **Выбор метрики и оптимизация моделей**
   2. **Сравнение моделей**

# **Заключение**

# **Список литературы**

1. Будыльский, Д. В. GRU и LSTM: современные рекуррентные нейронные сети / Д. В. Будыльский, - Текст : непосредственный // Молодой ученый. – 2015. – № 15(95). – С. 51-54.
2. Boas, Matheus. Optimal Decision Trees for the Algorithm Selection Problem: Integer Programming Based Approaches / Matheus Boas, G. Vilas [et al]. - Int. Trans. Oper. Res  28 (2021): 2759-2781. – Текст непосредственный.
3. McQuistan Adam. Using Machine Learning to Predict the Weather: Part 1. / Adam McQuistan. – URL : <https://stackabuse.com/using-machine-learning-to-predict-the-weather-part-1> (дата обращения : 4.05.2022). – Текст : электронный.
4. Pialunga Piero. Weather forecasting with Machine Learning, using Python / Piero Pialung/ - URL: <https://towardsdatascience.com/weather-forecasting-with-machine-learning-using-python-55e90c346647> (Дата обращения 30/05/2022). - Текст : электронный.
5. Воронцов К. В. Глубокие нейронные сети / Воронцов К. В./ – URL: <http://www.machinelearning.ru/wiki/images/7/71/Voron-ML-DeepLearning-slides.pdf> (Дата обращения 3/05/2022). - Текст : электронный.
6. Joseph Rocca. Ensemble methods: bagging, boosting and stacking / Joseph Rocca / - URL: <https://towardsdatascience.com/ensemble-methods-bagging-boosting-and-stacking-c9214a10a205> (Дата обращения 10/05/2022). - Текст : электронный.
7. Steven Yu. Stacking and Blending — An Intuitive Explanation / Steven Yu / - URL: <https://medium.com/@stevenyu530_73989/stacking-and-blending-intuitive-explanation-of-advanced-ensemble-methods-46b295da413c> (Дата обращения 25/05/2022). - Текст : электронный.
8. Документация к библиотеке машинного обучения XGBoost. / - URL: <https://xgboost.readthedocs.io/en/stable/> (Дата обращения 23/05/2022). - Текст : электронный.
9. Документация к библиотеке машинного обучения CatBoost. / - URL: <https://catboost.ai/en/docs/references/training-parameters/> (Дата обращения 14/05/2022). - Текст : электронный.
10. Документация к библиотеке машинного обучения Keras. / - URL: <https://www.tensorflow.org/api_docs/python/tf/keras/layers/LSTM> (Дата обращения 15/05/2022). - Текст : электронный.
11. Документация к библиотеке машинного обучения PyTorch / - URL: <https://pytorch.org/docs/stable/generated/torch.nn.LSTM.html> (Дата обращения 13/05/2022). - Текст : электронный.
12. Документация к библиотеке машинного обучения scikit learn / - URL: [https://scikit-learn.org/stable/modules/ensemble.html#](https://scikit-learn.org/stable/modules/ensemble.html) (Дата обращения 25/05/2022). - Текст : электронный.
13. Учебник по машинному обучению от ШАД / - URL: <https://ml-handbook.ru/> (Дата обращения 25/05/2022). - Текст : электронный.

**Приложение А.**

**Модель случайного леса**

from datetime import timedelta

import numpy as np

import tensorflow as tf

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import mean\_absolute\_error

from matplotlib import pyplot as plt

import plotly.express as px

from typing import List

from statsmodels.graphics.tsaplots import plot\_acf

from statsmodels.graphics.tsaplots import plot\_pacf

import pandas as pd

data = pd.read\_csv('dataTnv.csv',index\_col=False)

dataTemp = pd.read\_csv('dataTnv.csv',index\_col=False, usecols=["avg\_value"])

datainter=dataTemp.fillna(data.interpolate())

dataParse\_time=pd.to\_datetime(data.btime, format="%Y-%m-%d %X+03")

df=pd.concat([dataParse\_time,datainter],axis=1)

train\_size = int(df.shape[0] \* 0.7)

val\_size = int(df.shape[0] \* 0.2)

test\_size = int(df.shape[0] \* 0.1)

train\_df = df.iloc[:train\_size]

val\_df = df.iloc[train\_size:-test\_size]

test\_df = df.iloc[train\_size+val\_size:]

train\_df.shape, val\_df.shape, test\_df.shape

df.loc[:train\_size, 'partition'] = 'train'

df.loc[train\_size:train\_size+val\_size, 'partition'] = 'validation'

df.loc[train\_size+val\_size:, 'partition'] = 'test'

fig = px.line(

    data\_frame = df.reset\_index(),

    x      = 'btime',

    y      = 'avg\_value',

    color  = 'partition',

    width  = 900,

    height = 500)

fig.update\_xaxes(

    rangeslider\_visible=True,

    rangeselector=dict(

        buttons=list([

            dict(count=1, label="h", step="hour", stepmode="backward"),

   Продолжение приложения А

         dict(count=24, label="24h", step="hour", stepmode="backward"),

            dict(count=24\*7, label="w", step="hour", stepmode="backward"),

            dict(count=1, label="m", step="month", stepmode="backward"),

            dict(step="all")

        ])

    )

)

fig.show()

df=df.drop(columns='partition')

def make\_dataset\_for\_rf(df, window\_size,front\_number, use\_scaler=False):

    features\_arr = df[["avg\_value"]].iloc[:-(front\_number)]

    features\_arr = np.array(features\_arr, dtype=np.float32)

    if use\_scaler:

        features\_arr = scaler.transform(features\_arr)

    target = df[["avg\_value"]].iloc[(window\_size+front\_number):]

    target = np.array(target, dtype=np.float32)

    features = []

    for i in range(len(target)):

        features.append(features\_arr[i:window\_size+i])

    features = np.asarray(features)

    return features,target

window\_size = 15

front\_number = 7

x\_train,y\_train = make\_dataset\_for\_rf(df=train\_df, window\_size=window\_size, front\_number=front\_number, use\_scaler=False)

x\_val,y\_val = make\_dataset\_for\_rf(df=val\_df, window\_size=window\_size, front\_number=front\_number, use\_scaler=False)

x\_test,y\_test = make\_dataset\_for\_rf(df=test\_df, window\_size=window\_size, front\_number=front\_number, use\_scaler=False)

x\_train = x\_train.reshape(len(y\_train),window\_size)

x\_val = x\_val.reshape(len(y\_val),window\_size)

x\_test = x\_test.reshape(len(y\_test),window\_size)

y\_train = y\_train.reshape(len(y\_train))

y\_test = y\_test.reshape(len(y\_test))

from sklearn.ensemble import RandomForestRegressor

from sklearn.datasets import make\_regression

Окончание приложения А

regr = RandomForestRegressor(n\_estimators=1000, max\_depth=10, min\_samples\_split=2, min\_samples\_leaf=1,verbose=1)

regr.fit(x\_train, y\_train)

prediction = regr.predict(x\_test)

val\_time = val\_df[:len(-y\_test)].drop(columns=["avg\_value"])

data\_plot = pd.DataFrame({

                'test': np.reshape(y\_test,len(y\_test)),

                'prediction': prediction,

                 })

data\_plot.index.name = 'date\_time'

fig = px.line(

    data\_frame = data\_plot.reset\_index(),

    x      = val\_time['btime'],

    y      = data\_plot.columns,

    width  = 900,

    height = 500

    )

fig.update\_xaxes(

    rangeslider\_visible=True,

    rangeselector=dict(

        buttons=list([

            dict(count=1, label="h", step="hour", stepmode="backward"),

            dict(count=4, label="4h", step="hour", stepmode="backward"),

            dict(count=24, label="24h", step="hour", stepmode="backward")

            dict(step="all")

        ])

    )

)

fig.show()

import sys

np.set\_printoptions(threshold=sys.maxsize)

dif\_xgb = np.array(prediction-y\_test.reshape(len(y\_test)))

plt.figure(figsize=(10, 7))

plt.plot(val\_time,dif\_xgb)

plt.title('mistake plot')

plt.ylabel('dif\_temperature')

plt.xlabel('date,time',labelpad=10)

plt.xticks( rotation='vertical')

dif\_xgb = np.abs(dif\_xgb)

print(dif\_xgb.mean())

Рисунок А.1 – Листинг программы ПЛК метода Случайный лес

**Приложение Б.**

**Модель градиентного бустинга**

from datetime import timedelta

import numpy as np

import tensorflow as tf

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import mean\_absolute\_error

from matplotlib import pyplot as plt

import plotly.express as px

from typing import List

from statsmodels.graphics.tsaplots import plot\_acf

from statsmodels.graphics.tsaplots import plot\_pacf

import pandas as pd

data = pd.read\_csv('dataTnv.csv',index\_col=False)

dataTemp = pd.read\_csv('dataTnv.csv',index\_col=False, usecols=["avg\_value"])

datainter=dataTemp.fillna(data.interpolate())

dataParse\_time=pd.to\_datetime(data.btime, format="%Y-%m-%d %X+03")

df=pd.concat([dataParse\_time,datainter],axis=1)

train\_size = int(df.shape[0] \* 0.7)

val\_size = int(df.shape[0] \* 0.2)

test\_size = int(df.shape[0] \* 0.1)

train\_df = df.iloc[:train\_size]

val\_df = df.iloc[train\_size:-test\_size]

test\_df = df.iloc[train\_size+val\_size:]

train\_df.shape, val\_df.shape, test\_df.shape

df.loc[:train\_size, 'partition'] = 'train'

df.loc[train\_size:train\_size+val\_size, 'partition'] = 'validation'

df.loc[train\_size+val\_size:, 'partition'] = 'test'

fig = px.line(

    data\_frame = df.reset\_index(),

    x      = 'btime',

    y      = 'avg\_value',

    color  = 'partition',

    width  = 900,

    height = 500)

fig.update\_xaxes(

    rangeslider\_visible=True,

    rangeselector=dict(

        buttons=list([

            dict(count=1, label="h", step="hour", stepmode="backward"),

            dict(count=24, label="24h", step="hour", stepmode="backward"),

            dict(count=24\*7, label="w", step="hour", stepmode="backward"),

            dict(count=1, label="m", step="month", stepmode="backward"),

Продолжение приложения Б

            dict(step="all") ])))

fig.show()

df=df.drop(columns='partition')

def make\_dataset\_for\_gbdt(df, window\_size,front\_number, use\_scaler=False):

    features\_arr = df[["avg\_value"]].iloc[:-(front\_number)]

    features\_arr = np.array(features\_arr, dtype=np.float32)

    if use\_scaler:

        features\_arr = scaler.transform(features\_arr)

    target = df[["avg\_value"]].iloc[(window\_size+front\_number):]

    target = np.array(target, dtype=np.float32)

    features = []

    for i in range(len(target)):

        features.append(features\_arr[i:window\_size+i])

    features = np.asarray(features)

    return features,target

window\_size = 15

front\_number = 7

x\_train,y\_train = make\_dataset\_for\_gbdt(df=train\_df, window\_size=window\_size, front\_number=front\_number, use\_scaler=False)

x\_val,y\_val = make\_dataset\_for\_gbdt(df=val\_df, window\_size=window\_size, front\_number=front\_number, use\_scaler=False)

x\_test,y\_test = make\_dataset\_for\_gbdt(df=test\_df, window\_size=window\_size, front\_number=front\_number, use\_scaler=False)

x\_train = x\_train.reshape(len(y\_train),window\_size)

x\_val = x\_val.reshape(len(y\_val),window\_size)

x\_test = x\_test.reshape(len(y\_test),window\_size)

import xgboost

from xgboost import XGBRegressor

model = XGBRegressor(n\_estimators=1000, max\_depth=7, learning\_rate=0.01)

model.fit(x\_train,y\_train,

         eval\_set=[(x\_train,y\_train),(x\_val,y\_val)],

         eval\_metric='mae',

         verbose=True, callbacks=[xgboost.callback.EarlyStopping(rounds=15,save\_best=True)])

model.best\_ntree\_limit

results = model.evals\_result()

plt.figure(figsize=(10,7))

plt.plot(results["validation\_0"]["mae"], label="Training loss")

Окончание приложения Б

plt.plot(results["validation\_1"]["mae"], label="Validation loss")

plt.axvline(model.best\_ntree\_limit, color="gray", label="Optimal tree number")

plt.xlabel("Number of trees")

plt.ylabel("Loss")

plt.legend()

prediction = model.predict(x\_test)

val\_time = val\_df[:len(-y\_test)].drop(columns=["avg\_value"])

data\_plot = pd.DataFrame({

                'test': np.reshape(y\_test,len(y\_test)),

                'prediction': prediction, })

data\_plot.index.name = 'date\_time'

fig = px.line(

    data\_frame = data\_plot.reset\_index(),

    x      = val\_time['btime'],

    y      = data\_plot.columns,

    width  = 900,

    height = 500

    )

fig.update\_xaxes(

    rangeslider\_visible=True,

    rangeselector=dict(

        buttons=list([

            dict(count=1, label="h", step="hour", stepmode="backward"),

            dict(count=4, label="4h", step="hour", stepmode="backward"),

            dict(count=24, label="24h", step="hour", stepmode="backward"),

            dict(step="all")])))

fig.show()

import sys

np.set\_printoptions(threshold=sys.maxsize)

dif\_xgb = np.array(prediction-y\_test.reshape(len(y\_test)))

plt.figure(figsize=(10, 7))

plt.plot(val\_time,dif\_xgb)

plt.title('mistake plot')

plt.ylabel('dif\_temperature')

plt.xlabel('date,time',labelpad=10)

plt.xticks( rotation='vertical')

dif\_xgb = np.abs(dif\_xgb)

print(dif\_xgb.mean())

Рисунок Б.1 – Листинг программы для модели градиентного бустинга

**Приложение В.**

**Модель искусственной нейронной сети**

from datetime import timedelta

import numpy as np

import tensorflow as tf

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import mean\_absolute\_error

from matplotlib import pyplot as plt

import plotly.express as px

from typing import List

from statsmodels.graphics.tsaplots import plot\_acf

from statsmodels.graphics.tsaplots import plot\_pacf

import pandas as pd

data = pd.read\_csv('dataTnv.csv',index\_col=False)

dataTemp = pd.read\_csv('dataTnv.csv',index\_col=False, usecols=["avg\_value"])

datainter=dataTemp.fillna(data.interpolate())

dataParse\_time=pd.to\_datetime(data.btime, format="%Y-%m-%d %X+03")

df=pd.concat([dataParse\_time,datainter],axis=1)

df = df.iloc[::7, :]

df['dif\_avg\_value'] = np.append(0,(np.array(df['avg\_value'][1:]) - np.array(df['avg\_value'][:-1])))

df.loc[df.dif\_avg\_value > 0.3,'dif\_avg\_value'] = 0

df.loc[df.dif\_avg\_value < -0.3,'dif\_avg\_value'] = 0

train\_size = int(df.shape[0] \* 0.7)

val\_size = int(df.shape[0] \* 0.2)

test\_size = int(df.shape[0] \* 0.1)

train\_df = df.iloc[:train\_size]

val\_df = df.iloc[train\_size:-test\_size]

test\_df = df.iloc[train\_size+val\_size:]

df.loc[:train\_size, 'partition'] = 'train'

df.loc[train\_size:train\_size+val\_size, 'partition'] = 'validation'

df.loc[train\_size+val\_size:, 'partition'] = 'test'

fig, ax = plt.subplots(figsize=(11, 4))

train\_df['dif\_avg\_value'].plot(ax=ax, label='train')

val\_df['dif\_avg\_value'].plot(ax=ax, label='validation')

test\_df['dif\_avg\_value'].plot(ax=ax, label='test')

ax.legend()

def make\_dataset\_for\_NN(df, window\_size,ahead\_number, use\_scaler=False):

    features\_arr = df[["dif\_avg\_value"]].iloc[:-(ahead\_number)]

    features\_arr = np.array(features\_arr, dtype=np.float32)

    if use\_scaler:

      Продолжение приложения В

  features\_arr = scaler.transform(features\_arr)

    target = df[["dif\_avg\_value"]].iloc[(window\_size+ahead\_number):]

    target = np.array(target, dtype=np.float32)

    features = []

    for i in range(len(target)):

        features.append(features\_arr[i:window\_size+i])

    features = np.asarray(features)

    return features,target

window\_size = 15

ahead\_number = 1

x\_train,y\_train = make\_dataset\_for\_NN(df=train\_df, window\_size=window\_size, ahead\_number=ahead\_number, use\_scaler=False)

x\_val,y\_val = make\_dataset\_for\_NN(df=val\_df, window\_size=window\_size, ahead\_number=ahead\_number, use\_scaler=False)

x\_test,y\_test = make\_dataset\_for\_NN(df=test\_df, window\_size=window\_size, ahead\_number=ahead\_number, use\_scaler=False)

x\_train = x\_train.reshape(len(y\_train),window\_size)

x\_val = x\_val.reshape(len(y\_val),window\_size)

x\_test = x\_test.reshape(len(y\_test),window\_size)

from keras.models import Sequential

from keras.layers import Activation, Dense,Dropout

model = Sequential([

    Dense(128, activation='selu', input\_shape=(window\_size,)),

    Dropout(0.5),

    Dense(256, activation='selu'),

    Dense(256, activation='selu'),

    Dense(256, activation='selu'),

    Dropout(0.5),

    Dense(256, activation='selu'),

    Dense(256, activation='selu'),

    Dropout(0.5),

     Dense(256, activation='selu'),

   Dropout(0.5),

    Dense(1)

])

def compile\_and\_fit(model,x\_train,y\_train, x\_val,y\_val, batch\_size, num\_epochs):

    model.compile(optimizer='adam', loss='mae', metrics=['mse'])

    return model.fit(x\_train, y\_train, validation\_data=(x\_val, y\_val),

                    batch\_size=batch\_size, epochs=num\_epochs, verbose=1,

                    callbacks=

Окончание приложения B

         [tf.keras.callbacks.ModelCheckpoint('best\_models\_NN/', monitor='val\_loss', verbose=1,save\_best\_only=True, save\_weights\_only=False,

                            mode='min', save\_freq='epoch')])

batch\_size = 8

num\_epochs = 25

history =  compile\_and\_fit(model, x\_train, y\_train, x\_val, y\_val, batch\_size,num\_epochs)

np.round(model.evaluate(x\_val,y\_val),4)

np.round(model.evaluate(x\_test,y\_test),4)

from keras.models import load\_model

model = load\_model('best\_models\_NN/')

prediction = model.predict(x\_test)

val\_time = val\_df[:len(-y\_test)].drop(columns=["dif\_avg\_value","avg\_value"])

plt.figure(figsize=(10, 7))

plt.plot(val\_time,y\_test)

plt.plot(val\_time,prediction)

plt.title('prediction')

plt.ylabel('temperature')

plt.xlabel('date,time')

plt.legend(['real', 'prediction'], loc='lower left')

plt.xticks( rotation='vertical')

dif = np.array(prediction-y\_test)

plt.figure(figsize=(10, 7))

plt.plot(val\_time,dif)

plt.title('mistake plot')

plt.ylabel('diftemperature')

plt.xlabel('date,time',labelpad=10)

plt.xticks( rotation='vertical')

dif = np.abs(dif)

print(dif.mean())

Рисунок В.1 – Листинг программы для модели искусственной нейронной сети

**Приложение Г.**

**Модель LSTM**

from datetime import timedelta

import numpy as np

import tensorflow as tf

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import mean\_absolute\_error

from matplotlib import pyplot as plt

import plotly.express as px

from typing import List

from statsmodels.graphics.tsaplots import plot\_acf

from statsmodels.graphics.tsaplots import plot\_pacf

import pandas as pd

data = pd.read\_csv('dataTnv.csv',index\_col=False)

dataTemp = pd.read\_csv('dataTnv.csv',index\_col=False, usecols=["avg\_value"])

datainter=dataTemp.fillna(data.interpolate())

dataParse\_time=pd.to\_datetime(data.btime, format="%Y-%m-%d %X+03")

df=pd.concat([dataParse\_time,datainter],axis=1)

df = df.iloc[::7, :]

df['dif\_avg\_value'] = np.append(0,(np.array(df['avg\_value'][1:]) - np.array(df['avg\_value'][:-1])))

df.loc[df.dif\_avg\_value > 0.3,'dif\_avg\_value'] = 0

df.loc[df.dif\_avg\_value < -0.3,'dif\_avg\_value'] = 0

train\_size = int(df.shape[0] \* 0.7)

val\_size = int(df.shape[0] \* 0.2)

test\_size = int(df.shape[0] \* 0.1)

train\_df = df.iloc[:train\_size]

val\_df = df.iloc[train\_size:-test\_size]

test\_df = df.iloc[train\_size+val\_size:]

df.loc[:train\_size, 'partition'] = 'train'

df.loc[train\_size:train\_size+val\_size, 'partition'] = 'validation'

df.loc[train\_size+val\_size:, 'partition'] = 'test'

fig, ax = plt.subplots(figsize=(11, 4))

train\_df['dif\_avg\_value'].plot(ax=ax, label='train')

val\_df['dif\_avg\_value'].plot(ax=ax, label='validation')

test\_df['dif\_avg\_value'].plot(ax=ax, label='test')

ax.legend()

def make\_dataset\_for\_LSTM(df, window\_size,ahead\_number, use\_scaler=False):

    features\_arr = df[["dif\_avg\_value"]].iloc[:-(ahead\_number)]

    features\_arr = np.array(features\_arr, dtype=np.float32)

    if use\_scaler:

      Продолжение приложения Г

  features\_arr = scaler.transform(features\_arr)

    target = df[["dif\_avg\_value"]].iloc[(window\_size+ahead\_number):]

    target = np.array(target, dtype=np.float32)

    features = []

    for i in range(len(target)):

        features.append(features\_arr[i:window\_size+i])

    features = np.asarray(features)

    return features,target

window\_size = 15

ahead\_number = 1

x\_train,y\_train = make\_dataset\_for\_LSTM(df=train\_df, window\_size=window\_size, ahead\_number=ahead\_number, use\_scaler=False)

x\_val,y\_val = make\_dataset\_for\_LSTM(df=val\_df, window\_size=window\_size, ahead\_number=ahead\_number, use\_scaler=False)

x\_test,y\_test = make\_dataset\_for\_LSTM(df=test\_df, window\_size=window\_size, ahead\_number=ahead\_number, use\_scaler=False)

x\_train = x\_train.reshape(len(y\_train),window\_size)

x\_val = x\_val.reshape(len(y\_val),window\_size)

x\_test = x\_test.reshape(len(y\_test),window\_size)

lstm\_model = tf.keras.models.Sequential([

    tf.keras.Input((window\_size,1)),

    #tf.keras.layers.LSTM(32, return\_sequences=True),

    tf.keras.layers.LSTM(32, return\_sequences=False),

    tf.keras.layers.Dense(1)

])

def compile\_and\_fit(model,x\_train,y\_train, x\_val,y\_val, batch\_size, num\_epochs):

    model.compile(loss=tf.losses.MeanSquaredError(), optimizer=tf.optimizers.Adam(), metrics=[tf.metrics.MeanAbsoluteError()])

    return model.fit(x=x\_train, y=y\_train, batch\_size=batch\_size, epochs=num\_epochs, verbose=1,

                        callbacks =

                        tf.keras.callbacks.ModelCheckpoint(

                        'best\_models\_lstm/', monitor='val\_loss', verbose=1,

                        save\_best\_only=True, save\_weights\_only=False,

                        mode='min', save\_freq='epoch'), validation\_data=(x\_val, y\_val), shuffle=True,

                       )

batch\_size = 8

num\_epochs = 25

history =  compile\_and\_fit(lstm\_model, x\_train, y\_train, x\_val, y\_val, batch\_size, num\_epochs)

Окончание приложения Г

plt.figure(figsize=(10, 7))

plt.plot(history.history['val\_loss'])

plt.plot(history.history['loss'])

plt.ylabel('loss')

plt.xlabel('epoch')

plt.legend(['val', 'train'], loc='upper left')

plt.figure(figsize=(10, 7))

plt.plot(history.history['val\_mean\_absolute\_error'])

plt.plot(history.history['mean\_absolute\_error'])

plt.ylabel('mae')

plt.xlabel('epoch')

plt.legend(['val', 'train'], loc='upper left')

np.round(lstm\_model.evaluate(x\_val,y\_val),4)

np.round(lstm\_model.evaluate(x\_test,y\_test),4)

from keras.models import load\_model

lstm\_model = load\_model('best\_models\_lstm/')

prediction = lstm\_model.predict(x\_test)

val\_time = val\_df[:len(-y\_test)].drop(columns=["dif\_avg\_value",'avg\_value'])

plt.figure(figsize=(10, 7))

plt.plot(val\_time,y\_test)

plt.plot(val\_time,prediction)

plt.title('prediction')

plt.ylabel('temperature')

plt.xlabel('date,time')

plt.legend(['real', 'prediction'], loc='lower left')

plt.xticks( rotation='vertical')

dif = np.array(prediction-y\_test)

plt.figure(figsize=(10, 7))

plt.plot(dif)

plt.ylim(0, 2)

plt.title('mistake plot')

plt.ylabel('MAE on sample')

plt.xlabel('sample number',labelpad=10)

dif = np.abs(dif)

print(dif.mean(

Рисунок Г.1. – Листинг программы для модели LSTM