НАУЧНАЯ СТАТЬЯ
УДК 004.9
DOI: https://doi.org/10.30932/1992-3252-2024-22-2-2

Мир транспорта. 2024. Т. 22. № 2(111). С. 12–24

Исследования интенсивности транспортного потока методом Deep learning

Игорь ПУГАЧЕВ

Николай ШЕШЕРА

Денис ГРИГОРОВ

Игорь Николаевич Пугачев[1], Николай Геннадьевич Шешера[2], Денис Евгеньевич Григоров[3]

[1]Хабаровский Федеральный исследовательский центр Дальневосточного отделения Российской академии наук (ХФИЦ ДВО РАН), Хабаровск, Россия.

[2, 3] Дальневосточный юридический институт МВД России, Хабаровск, Россия.

[1] ORCID 0000-0003-0345-4350; Web of Science Researcher ID: ABY-8399-2022; Scopus Author ID: 56386223400; РИНЦ SPIN-код: 1856-1556; РИНЦ Author ID: 416392.

[2] ORCID 0009-0006-3302-5572; Web of Science Researcher ID: LCI-5197-2024; Scopus Author ID: 57209470019; РИНЦ SPIN-код: 9869-8822; РИНЦ Author ID: 1033608.

[3] ORCID 0009-0005-4049-9488; Scopus Author ID: 57209470019; РИНЦ SPIN-код: 6146-4152; РИНЦ Author ID: 1084181.

✉ *[2] kolyaka239@mail.ru.*

АННОТАЦИЯ

В гармоничной транспортной системе автомобильные потоки рационально распределяются в зависимости от возможностей дорог и улиц по обеспечению пропускной способности с учетом систем регулирования. При этом не уделяется должного внимания изменениям погодных и природных условий, что в свою очередь значительно корректирует режимы движения, выводя их из стабильного, прогнозируемого состояния. Современные программно-аппаратные комплексы и информационные ресурсы больших городов имеют широкий диапазон регистрируемых показателей, оказывающих влияние на распределение транспортных потоков. Их автоматизированная обработка с использованием алгоритмического инструментария машинного обучения сформировала комплексное представление о закономерностях изменения показателя интенсивности, что является новым этапом повышения безопасности дорожного движения, стремящейся к нулевой смертности.

Научная новизна предлагаемого исследования заключается в приемах и подходах к исследованию погодно-климатических характеристик и факторов улично-дорожной сети, их предварительной обработке с использованием современных статистических и логических методов нормализации и исключения случайных выбросов.

Метод глубокого обучения открывает широкие возможности для анализа интенсивности транспортного потока. Путем обработки больших объемов данных такие алгоритмы способны выявлять сложные паттерны и взаимосвязи, что позволяет улучшить прогнозирование движения транспорта и оптимизировать управление его потоками. Для корректной работы нейронной сети по обучению модели и исследования интенсивности транспортного потока разработан комплекс программных инструментов по предварительной обработке данных, который включает поэтапный анализ структур массивов с последующей заменой значений или исключением ошибок.

После предварительной очистки данных в соответствии с синтаксисом программной логики и правилами статистического анализа был применен метод поиска и исключения аномалий – метод изолированного леса.

Данное направление вошло в состав большого исследования интенсивности транспортного потока, а продемонстрированные результаты являются совокупностью решений на основе системного взаимодействия разработанных авторами программ и методик статистических и аналитических преобразований.

Ключевые слова: транспортные системы, интенсивность транспортного потока, метод глубокого обучения, погодные и природные условия, нейронные сети, модель.

Для цитирования: Пугачев И. Н., Шешера Н. Г., Григоров Д. Е. Исследования интенсивности транспортного потока методом Deep learning // Мир транспорта. 2024. Т. 22. № 2 (111). С. 12–24. DOI: https://doi.org/10.30932/1992-3252-2024-22-2-2.

Полный текст статьи в переводе на английский язык публикуется во второй части данного выпуска.
English translation of the full text of the article is published in the second part of the issue.

ВВЕДЕНИЕ

Современные тенденции развития информационных технологий предлагают широкий перечень возможностей анализа данных. Большое количество признаков дает возможность сформировать шаблоны распределения транспортных сред и в каждой рассмотреть погодные условия[1] [1].

Искусственный интеллект (ИИ) является одной из перспективных областей в современной науке и технологиях. В последние десятилетия с использованием ИИ были созданы системы способные выполнять сложные задачи, требующие интеллектуальных навыков [2; 3].

Современный ИИ способен анализировать окружающую среду и взаимодействовать с ней, интерпретировать и перерабатывать данные, обучаться и адаптироваться по мере развития. Именно поэтому главной задачей работы является подготовка данных для объективной оценки исследуемого явления.

ИИ находит применение во многих областях, включая компьютерное зрение, обработку естественного языка, робототехнику, автономные транспортные средства, медицину, финансовую аналитику и т. д. В каждой из этих областей ИИ позволяет автоматизировать процессы, улучшать прогнозы и принимать решения на основе большого объема данных и анализа сложных моделей.

Принципы ИИ основываются на различных подходах и методах по созданию компьютерных систем, способных анализировать данные, обучаться и принимать решения на основе полученной информации. Методы машинного обучения, нейронные сети и глубокое обучение – основные инструменты в достижении этой цели.

В данной работе для прогнозирования интенсивности транспортного потока характеристики улично-дорожной сети (УДС) являются классификатором для категоризации транспортных сооружений, а погодные условия – независимыми переменными, по которым будет прогнозироваться изменения зависимой [4].

В работе присутствуют сложные элементы программирования. Первые операции будут поясняться, а для остальных, подобного рода, с целью снижения дублирующей информации, комментарии будут отсутствовать.

Рабочее окно среды для разработки PyCharm имеет пронумерованные строчки. Команды будут поясняться, ссылаясь на номер строчки приведенного скриншота рабочего поля.

РЕЗУЛЬТАТЫ

Архитектура нейронной сети обратного распространения

С целью подтверждения влияния погодных условий, геометрических элементов дорог и характеристик движения на интенсивность транспортного потока, а также проверки прогностических возможностей была использована нейронная сеть обратного распространения (BackPropagation)[2].

Нейронная сеть обратного распространения – это тип нейронных сетей, который использует алгоритм обратного распространения ошибки для обучения. Основная идея заключается в том, что сеть проходит через два этапа: прямое распространение, где входные данные передаются от входного слоя через скрытые слои к выходному слою, и обратное распространение ошибки, где происходит корректировка весов сети [5; 6].

Процесс обратного распространения ошибки начинается с вычисления ошибки на выходном слое. Затем эта ошибка распространяется обратно через слои сети, пропорционально весам связей между нейронами, и рассчитывается ошибка на каждом слое [8; 9]. Далее, веса нейронов корректируются в направлении, противоположном градиенту ошибки, для уменьшения ошибки при последующих проходах по сети.

Для алгоритма обратного распространения ошибки необходимо определить ее функцию. Она измеряется разностью между ожидаемыми значениями и предсказанными. Среднеквадратичная ошибка (mean squared error) является наиболее распространенной для задач регрессии и перекрестной энтропии (cross entropy) для задач классификации [10].

Во время обучения сеть проходит через несколько эпох, где каждая эпоха представ-

[1] ОДМ 218.4.005-2010. Отраслевой дорожный методический документ. Рекомендации по обеспечению безопасности движения на автомобильных дорогах.

[2] Эксплуатация автомобильных дорог: Учеб. пособие / сост.: И. Н. Пугачев, А. В. Каменчуков, Н. С. Нестерова. – Хабаровск: Изд-во ДВГУПС, 2022. – 168 с.

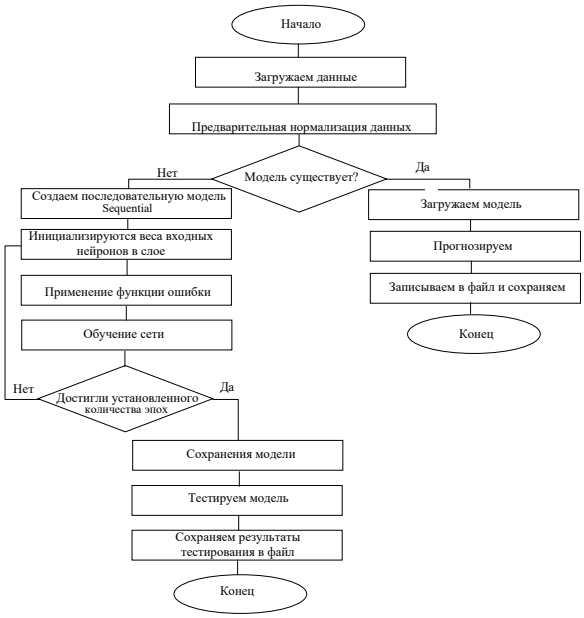

Рис. 1. Блок-схема нейронной сети обратного распространения [выполнено авторами].

ляет собой один проход по всем тренировочным примерам [11; 12]. Веса сети настраиваются таким образом, чтобы минимизировать ошибку на тренировочных данных.

Архитектуру нейронной сети было решено сформировать на языке программирования Python, используя надстройку Keras низкоуровневой библиотеки TensorFlow [13].

Keras был разработан с целью облегчить и ускорить процесс разработки моделей глубокого обучения. Он предоставляет высокоуровневые абстракции для построения нейронных сетей, скрывая от разработчика большую часть сложности и деталей низкоуровневых библиотек.

Архитектура нейронной сети обратного распространения состоит из нескольких основных компонентов:

1. Входной слой – принимает на вход данные, которые подаются на вход нейронной сети. В этом слое количество нейронов соответствует размерности поступающих данных.

2. Скрытые слои – выполняют промежуточные вычислительные операции. В зависимости от конкретной задачи или архитектуры нейронной сети изменяется количество скрытых слоев и нейронов в них. Каждый нейрон в скрытом слое берет на вход значения активации предыдущего слоя и вычисляет свое значение активации на основе весов, которые связывают его с предыдущим слоем.

3. Выходной слой – выполняет прогноз. Количество нейронов в выходном слое зависит от задачи классификации или регрессии. Например, если решается задача бинарной классификации, то в выходном слое будет один нейрон, отвечающий за вероятность принадлежности к одному классу.

4. Функция активации сети – используется для нелинейного преобразования входных данных. Наиболее распространенными функциями активации являются sigmoid, tanh и ReLU.

5. Обратное распространение ошибки – алгоритм для обновления весов в нейронной сети. Он вычисляет градиент функции ошибки по всем весам и использует его для обновления весов, чтобы минимизировать ошибку предсказания.

6. Оптимизатор – применяет оптимизационный алгоритм для обновления весов в нейронной сети на основе градиента ошибки. Некоторые популярные оптимизаторы включают в себя стохастический градиентный спуск (SGD), Adam и RMSprop.

7. Функция потерь – определяет, насколько хорошо нейронная сеть справляется с поставленной задачей. Она вычисляет разницу между предсказанными значениями и истинными значениями и используется в обратном распространении ошибки для вычисления градиента.

```
import os
import pandas as pd

df = pd.read_csv(os.getcwd() + '/dataset.csv', sep=";")
n = df.groupby('frontier') #Сортруем по рубежам
grp_idx = n.groups
for key in grp_idx.keys():
    r = n.get_group(key)
    #r.drop(columns=r.columns[0], axis=1, inplace=True)  # убираем лишний столбец
    #r.drop(columns=r.columns[16], axis=1, inplace=True)  # убираем лишний столбец
    r.to_csv(os.getcwd() + '/output/r/' + str(key) + '.csv', sep=";", encoding="utf-8-sig")
```

Рис. 2. Модуль – explode_start. Сортировка данных по рубежам и формирование файлов [выполнено авторами].

Основные компоненты архитектуры нейронной сети обратного распространения могут быть изменены и модифицированы в зависимости от конкретной задачи и требований [14; 15].

Блок-схема нейронной сети представлена на рис. 1.

Изначально структура нейронной сети строилась на анализе данных без предварительной обработки и валидации [16]. Тестирование модели после обучения показало неэффективность прогноза. Это связано с большим количеством входящих нейронов. Поэтому было принято решение сгруппировать количественные признаки. В результате пришлось изменить подходы к созданию нейронной сети [17].

Для реализации программы по приведенному выше алгоритму в интегрированной среде разработки PyCharm было создано виртуальное окружение. Это, в первую очередь, необходимо было сделать для более удобного управления зависимостями проекта, то есть сделать процесс разработки более удобным и сконцентрированным.

В созданном виртуальном окружении размещены модули (файлы с расширением «ру»). Они используются для управления зависимостями проекта, создания изолированной среды разработки и работы с различными версиями пакетов и библиотек.

Проект состоит их трех этапов:

1) формирование данных;

2) валидация данных;

3) обучение модели.

Формирование данных

Статистические данные импортированы в директорию виртуального окружения проекта в формате файла *csv*. Такое расширение применяется для удобства хранения данных (в виде текста), чтобы было удобно их обрабатывать и передавать [18].

Для исследования влияния погодных, природных условий и геометрических элементов дорог на интенсивность транспортного потока было собрано и сгруппировано 562612 часов с различными характеристиками со 107 рубежей. Предварительная обработка данных всего массива показала плохие результаты влияния независимых признаков на зависимый. Объясняя это недостаточным перечнем анализируемых факторов, было решено каждый рубеж анализировать отдельно, с перспективой создать не одну, а 107 моделей для прогнозирования. Таким образом, будет исключено влияние неучтенных статических независимых переменных [19; 20].

Для достижения поставленной выше цели с учетом правил обработки нейронной сетью из одного файла со всеми данными требовалось сформировать 107 файлов с расширением *csv*. В каждом из них – данные конкретного рубежа.

Для сортировки генеральной совокупности по рубежам с последующим формированием файлов был создан и запущен алгоритм *explode_start.py* (рис. 2), который находится в корне приложения.

В модуле *explode_start* были импортированы функции операционной системы – OS (*import os*) для обеспечения чтения файлов и библиотека *pandas* (*import pandas as pd*) для работы с данными (рис. 2). Дополнительно, к библиотеке применяется *alias* (переименование для простого обращения к библиотеке) с упрощением его названия до *pd.*

На четвертой строчке программа «читает» файл, дополнительно был указан вид сепаратора – точка с запятой.

На пятой строчке делается группировка по рубежам.

На шестой строчке программа получает список рубежей.

С седьмой строки в цикле поочередно программа перебирает рубежи и сохраняет их в отдельные файлы, записывая в наименование их номер.

После выполнения скрипта *explode_start* в директории *r*, следуя динамическому маршруту '*/output/r/*' (рис. 2), появляются файлы с данными о рубежах (рис. 3).

Подготовка данных

После формирования файлов с рубежами для исключения ошибок при обучении необходима подготовка данных, которая будет включать контроль отсутствия значений в позиции (предупреждать оператора, заменять на 0, удалять всю строку и т. п.) и диапазон допустимых величин, валидацию и т. д. [21].

Для подготовки данных был создан и запущен алгоритм *validate_start* (рис. 4).

Командой «*from app.validator import start as validate*» была импортирована функция валидации «*start*» из модуля «*validator*» директории «*app*» (рис. 5).

В цикле алгоритма *validate_start* (рис. 4) программа получает список файлов из директории */output/r,* перебирая их по одному. На шестой строчке разъединяется название файла по разделителю – точка. В результате чего получается массив из двух частей. Первая часть (нулевая) – это имя файла, вторая часть (единица) – это расширение. В данной функции учитывается только имя файла.

На восьмой строчке, используя функцию *start* (переименованной в *predict*), из модуля *validator* запускается валидация (рис. 5).

При выполнении алгоритма программы модуля *validator* импортируются вспомогательные функции и переменные из директории columns (*app/domain/columns*) и модуль *logger* из директории *helpers* (*app/domain/ helpers*).

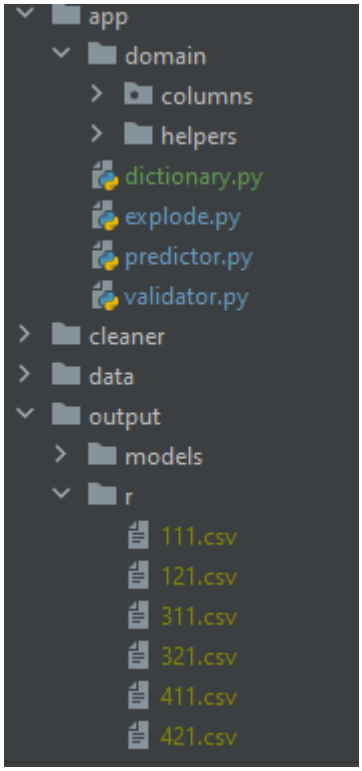

Рис. 3. Сформированные файлы с рубежами в директории виртуального окружения проекта [выполнено авторами].

Задача модуля *logger* менять цвет шрифта сообщения, которое выводится в консоль (оформление не обязательно в работе, поэтому детализация алгоритма данного модуля опускается).

Учитывая, что *columns* – это директория, а не исполняемый файл, то по умолчанию в нем инициализируется модуль *_init_*. Соответственно, строчкой 2 (рис. 5) импортируются в модуль *validator* из модуля *_init_* методы *names_input* и *names_output* – это инкапсуляция, один из принципов объектно-ориентированного программирования, сокрытия реализации для обеспечения возможности внедрения дополнительной логики в формирование *names_input* и *names_output* (список названий колонок входных параметров, которые используются в шапке данных) (рис. 6, 7).

```
1  import os
2
3  from app.validator import start as validate
4
5  for file in os.listdir(os.getcwd() + '/output/r'):
6      split = os.path.splitext(file)
7      filename = split[0]
8      validate(True, filename)
```

Рис. 4. Скрипт validate_start [выполнено авторами].

```python
import os
from app.domain.columns import names_input
from app.domain.columns import names_output
from app.domain.columns import validators as get_validators
from app.domain.helpers import logger
import pandas as pd
# NSheshera *
def start(need_train, filename):
    # 1-й шаг - готовим данные
    data = pd.read_csv(os.getcwd() + '/output/r/'+filename+'.csv', sep=';')
    validators = get_validators()
    new_array = dict()
    for data_name, data_values in data.items():
        if data_name in validators:
            l = validators[data_name]
            s = data_values
            encoded = list(map(l, s))
            new_array[data_name] = encoded
            logger.success(''' + str(data_name) + ''' + ' validated success!')
    df = pd.DataFrame(new_array)
    if need_train:
        d = names_input() + names_output()
    else:
        d = names_input()
    df.to_csv(os.getcwd() + '/output/validated/'+filename+'.csv', sep=";",
              columns=d, encoding='utf-8-sig')
```

Рис. 5. Функция валидации start из модуля validator директории app [выполнено авторами].

Метод *validators* в модуле *_init_* возвращает список всех валидаторов, которые находятся в двух константах, представляющих собой массивы данных (рис. 8):

1) *_columns_inputs+_columns_outputs*;

2) *_validators_input+_validators_output*.

Эти массивы после обработки данных конкатенируются, и устанавливается соответствие между названием колонки и ее валидатором. Это значит, что очередность валидатора в константе – *validators_input* должна соответствовать очередности ее колонки.

Каждая колонка из списков колонок (*_columns_inputs, _columns_outputs, _validators_input, _validators_output*) описывается в директории *columns*, а обрабатывается в модуле *_init_* с использованием приведенных на рис. 4 методов. Для обеспечения этого в модуль *_init_* командой *from app.domain.columns* были импортированы модули и методы проверки данных:

1. День недели:

– week_day import validate as week_validator;

– week_day import COLUMN_NAME as WEEK_COLUMN_NAME.

2. Время суток:

– time import COLUMN_NAME as TIME_COLUMN_NAME;

– time import validate as time_validator.

3. Интенсивность:

– intensivity import COLUMN_NAME as INTENSIVE_COLUMN_NAME;

– intensivity import validate as intensive_validator.

4. Температура воздуха:

– t_air import COLUMN_NAME as T_AIR_COLUMN_NAME;

– t_air import validate as t_air_validator.

5. Температура почвы:

– t_soil import COLUMN_NAME as T_SOIL_COLUMN_NAME;

– t_soil import validate as t_soil_validator.

6. Температура точки росы:

– t_dew import COLUMN_NAME as T_DEW_COLUMN_NAME;

– t_dew import validate as t_dew_validator.

7. Парциальное давление водяного пара (ps), Па:

```
# Колонки names_input которые использовать в анализе данных
5 usages  NSheshera
def names_input():
    return _columns_inputs
2 usages  NSheshera
def names_output():
    return _columns_outputs
# Получить список колонка-валидатор
NSheshera *
def validators():
    return dict(zip(_columns_inputs+_columns_outputs,
                    _validators_input+_validators_output))
```

Рис. 6. Метод names_input модуля _init_ [выполнено авторами].

```
# Колонки по порядку
_columns_inputs = [
    WEEK_COLUMN_NAME,# День недели
    TIME_COLUMN_NAME,#   Время суток
    T_AIR_COLUMN_NAME,#   Температура воздуха
    T_SOIL_COLUMN_NAME,#   Температура почвы
    T_DEW_COLUMN_NAME,#   Температура точки росы
    PRESSURE_COLUMN_NAME,# Парциальное давление водяного пара (ps), Па
    F_PERCENT_COLUMN_NAME,# Относительная влажность воздуха (ϕ), %;
    VISIBILITY_CIPHER_COLUMN_NAME,# Видимость, шифр (VV);
    SATURATION_DEFICIT_COLUMN_NAME,# Дефицит насыщения (d), г/м³
    P_LEVEL_STARION_COLUMN_NAME,# Атмосферное давление на уровне станции (P станции), г.Па
    P_SEA_LEVEL_COLUMN_NAME,# Атмосферное давление на уровне моря (P моря), г.Па
    WEATHER_CODE_WW_COLUMN_NAME,# Погода, шифр (ww)
    DIRECTION_WIND_COLUMN_NAME,#Направление ветра, градусы
    WIND_SPEED_COLUMN_NAME,# Скорость ветра, м/с
    PRECIPITATION_COLUMN_NAME,# Осадки, мм
    DAYLIGHT_COLUMN_NAME,# Естественное освещение...
]
_columns_outputs = [INTENSIVE_COLUMN_NAME]
```

Рис. 7. Список названий колонок входных и выходных параметров в модуле _init_ [выполнено авторами].

```
# Валидаторы по порядку
_validators_input = [
    week_validator,# День недели
    time_validator,#   Время суток
    t_air_validator,#   Температура воздуха
    t_soil_validator,#   Температура почвы
    t_dew_validator,#   Температура точки росы
    pressure_validator,# Парциальное давление водяного пара (ps), Па
    f_percent_validator,# Относительная влажность воздуха (ϕ), %;
    visibility_cipher_validator,# Видимость, шифр (VV);
    saturation_deficit_validator,# Дефицит насыщения (d), г/м³
    p_level_station_validator,# Атмосферное давление на уровне станции (P станции), г.Па
    p_sea_level_validator,# Атмосферное давление на уровне моря (P моря), г.Па
    weather_code_ww_validator,# Погода, шифр (ww)
    direction_wind_validator,#Направление ветра, градусы
    wind_speed_validator,# Скорость ветра, м/с
    precipitation_validator,# Осадки, мм
    daylight_validator,# Естественное освещение...]
_validators_output = [intensive_validator]
```

Рис. 8. Список названий колонок входных и выходных параметров в модуле _init_ [выполнено авторами].

– pressure import COLUMN_NAME as PRESSURE_COLUMN_NAME;

– columns.pressure import validate as pressure_validator.

8. Относительная влажность воздуха (ϕ), %:

– f_percent import COLUMN_NAME as F_PERCENT_COLUMN_NAME;

– f_percent import validate as f_percent_validator.

9. Видимость, шифр (VV):

– from app.domain.columns.visibility_cipher import COLUMN_NAME as VISIBILITY_CIPHER_COLUMN_NAME;

– visibility_cipher import validate as visibility_cipher_validator.

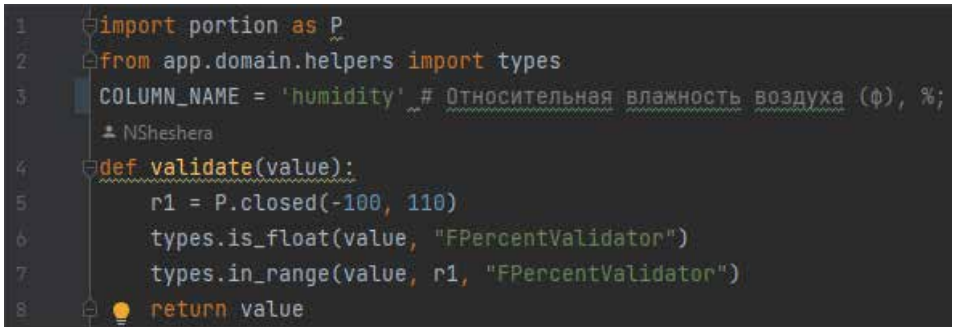

Рис. 9. Структура модуля контроля ошибок модуля f_percent [выполнено авторами].

10. Дефицит насыщения (d), г/м³:

– columns.saturation_deficit import COLUMN_NAME as SATURATION_DEFICIT_COLUMN_NAME;

– saturation_deficit import validate as saturation_deficit_validator.

11. Атмосферное давление на уровне станции (P станции), г. Па:

– p_level_station import COLUMN_NAME as P_LEVEL_STARION_COLUMN_NAME;

– p_level_station import validate as p_level_station_validator.

12. Атмосферное давление на уровне моря (P моря), г. Па:

– p_sea_level import COLUMN_NAME as P_SEA_LEVEL_COLUMN_NAME;

– p_sea_level import validate as p_sea_level_validator.

13. Погода, шифр (ww):

– weather_code_ww import COLUMN_NAME as WEATHER_CODE_WW_COLUMN_NAME;

– weather_code_ww import validate as weather_code_ww_validator.

14. Направление ветра, градусы:

– direction_wind import COLUMN_NAME as DIRECTION_WIND_COLUMN_NAME;

– direction_wind import validate as direction_wind_validator.

15. Скорость ветра, м/с:

– wind_speed import COLUMN_NAME as WIND_SPEED_COLUMN_NAME;

– wind_speed import validate as wind_speed_validator.

16. Осадки, мм:

– precipitation import COLUMN_NAME as PRECIPITATION_COLUMN_NAME;

– precipitation import validate as precipitation_validator.

17. Естественное освещение:

– daylight import COLUMN_NAME as DAYLIGHT_COLUMN_NAME;

– daylight import validate as daylight_validator.

На девятой строчке функция *start* модуля *validator* директории – *app/ domain/columns* читает файл *csv* (рис. 5). По строчке шаблона формируется путь к файлу (*os.getcwd*() + ‘/*output/r/*’).

Переменная *filename* – это название файла, оно передается аргументом в данную функцию, а *need_train* – это показатель того, что валидация идет на уровне тренировки модели или же на уровне прогнозирования (рис. 5). Дело в том, что валидация используется как на этапе тренировки модели, так и на этапе прогнозирования.

При помощи валидаторов в функции *start* модуля *validator* (рис. 5) формируются новые проверенные данные, которые записываются в директорию – /*utput/validated.*

Структура самого модуля, отражающего информацию о колонке, на примере переменной «Относительная влажность воздуха» приведена на рис. 9.

На строчке 2 импортируется *helpers*, которая обозначает кастомные функции проекта для проверки типов значения колонки таблицы *csv.*

К константе *COLUMN_NAME* на четвертой строчке привязывается наименование заголовка колонки. В данном случае это – «*humidity*».

С пятой по восьмую строчки функция *validate* проверяет соответствие значений установленным требованиям, в противном случае программа информирует об ошибке. В конкретном примере (рис. 9) значения в колонке *humidity* (относительная влажность воздуха) не должны быть ниже 100 и выше

110. Другие значения будут считаться ошибкой.

Правила контроля данных на языке программирования Python колонок файла *csv* на этапе подготовки приведены ниже.

Естественное освещение, модуль daylight

```
import math
import app.domain.helpers.utils as f
COLUMN_NAME = ‘daylight’ #Естественное
освещение
def validate(value):
if 0.5 < value < 1:
value = float(f.toFixed(value, 2))
types.in_range(value, [0, 0.5, 0.85, 1], «Daylight_
Validator»)
return value
```

Направление ветра, градусы, модуль direction_wind

```
import portion as P
from app.domain.helpers import types
COLUMN_NAME = ‘wind_direction’ #Направление
ветра, градусы
def validate(value):
r1 = P.closed(0, 360)
types.is_float(value, «direction_windValidator»)
types.in_range(value, r1, «direction_windValidator»)
return value
```

Относительная влажность воздуха, модуль f_percent

```
import portion as P
from app.domain.helpers import types
COLUMN_NAME = ‘humidity’ # Относительная
влажность воздуха (ϕ), %;
def validate(value):
r1 = P.closed(–100, 110)
types.is_float(value, «FPercentValidator»)
types.in_range(value, r1, «FPercentValidator»)
return value
```

Интенсивность транспортного потока, модуль intensivity

```
import portion as P
from app.domain.helpers import types
COLUMN_NAME = ‘intensity’ # Интенсивность
def validate(value):
r1 = P.closed(0, float(‘inf’))
types.is_int(value, «IntensivityValidator»)
types.in_range(value, r1, «IntensivityValidator»)
return value
```

Атмосферное давление на уровне станции, модуль p_level_station

```
import portion as P
from app.domain.helpers import types
COLUMN_NAME = ‘pressure_station’
# Атмосферное давление на уровне станции
(P станции), г. Па
def validate(value):
r1 = P.closed(900, 1100)
types.is_float(value, «p_level_stationValidator»)
types.in_range(value, r1, «p_level_stationValidator»)
return value
```

Атмосферное давление на уровне моря, модуль p_sea_level

```
import portion as P
from app.domain.helpers import types
COLUMN_NAME = ‘pressure_sea’ # Атмосферное
давление на уровне моря (P моря), г. Па
def validate(value):
r1 = P.closed(900, 1100)
types.is_float(value, «p_sea_levelValidator»)
types.in_range(value, r1, «p_sea_leveValidator»)
return value
```

Осадки, модуль precipitation

```
import portion as P
from app.domain.helpers import types
import math
COLUMN_NAME = ‘precipitation’ # Осадки, мм
def validate(value):
if math.isnan(value):
value = float(0)
r1 = P.closed(0, 100)
types.is_float(value, «Precipitation_Validator»)
types.in_range(value, r1, «Precipitation_Validator»)
return value
```

Парциальное давление водяного пара, модуль pressure

```
import portion as P
from app.domain.helpers import types
COLUMN_NAME = ‘partial_pressure’ # Парциальное
давление водяного пара (ps), Па
def validate(value):
r1 = P.closed(0, float(‘inf’))
types.is_float(value, «PressureValidator»)
types.in_range(value, r1, «PressureValidator»)
return value
```

Дефицит насыщения, модуль saturation_deficit

```
import portion as P
from app.domain.helpers import types
COLUMN_NAME = ‘saturation_deficit’ # Дефицит
насыщения (d), г/м³
def validate(value):
r1 = P.closed(0, 40)
types.is_float(value, «SaturationValidator»)
types.in_range(value, r1, «SaturationValidator»)
return value
```

Температура воздуха (t воздуха), модуль t_air

```
import portion as P
from app.domain.helpers import types
COLUMN_NAME = ‘t_air’ #Температура воздуха
(t воздуха), C°
def validate(value):
r1 = P.closed(–100, 100)
types.is_float(value, «TAirValidator»)
types.in_range(value, r1, «TAirValidator»)
return value
```

Температура точки росы (DP), модуль t_dew

```
import portion as P
from app.domain.helpers import types
COLUMN_NAME = ‘t_dew_point’ # Температура точки
```

росы (DP), C°
def validate(value):
r1 = P.closed(–100, 100)
types.is_float(value, «TDewValidator»)
types.in_range(value, r1, «TDewValidator»)
return value

Температура почвы (t почвы), модуль t_soil

import portion as P
from app.domain.helpers import types
COLUMN_NAME = 't_soil' # Температура почвы (t почвы), C°
def validate(value):
r1 = P.closed(–100, 100)
types.is_float(value, «TSoilValidator»)
types.in_range(value, r1, «TSoilValidator»)
return value

Время суток, модуль time

import portion as P
from app.domain.helpers import types
COLUMN_NAME = 'day_time' # Время суток
def validate(value):
r1 = P.closed(0, 23)
types.is_int(value, «TimeValidator»)
types.in_range(value, r1, «TimeValidator»)
return value

Видимость, шифр (VV), модуль visibility_cipher

from app.domain.helpers import types
import math
COLUMN_NAME = 'visibility_VV' # Видимость, шифр (VV);
def validate(value):
if math.isnan(value):
value = 0
types.is_int(int(value), «VisibilityValidator»)
types.in_range(value, [90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 0], «VisibilityValidator»)
return value

Погода, шифр (ww), модуль weather_code_ww

from app.domain.helpers import types
import math
COLUMN_NAME = 'weather_WW' # Погода, шифр (ww)
def validate(value):
if math.isnan(value):
value = 0
types.is_int(int(value), «Weather_code_ww_Validator»)
types.in_range(value, [0, 1, 2, 3, … 99], «Weather_code_ww_Validator»)
return value

День недели, модуль week_day

import portion as P
from app.domain.helpers import types
COLUMN_NAME = 'week_day' # День недели
def validate(value):
r1 = P.closed(1, 7)
types.is_int(value, «WeekValidator»)
types.in_range(value, r1, «WeekValidator»)
return value

Скорость ветра, модуль wind_speed

import portion as P
from app.domain.helpers import types
import math
COLUMN_NAME = 'wind_speed' # Скорость ветра, м/с
def validate(value):
if math.isnan(value):
value = float(0)
r1 = P.closed(0, 40)
types.is_float(value, «Wind_speedValidator»)
types.in_range(value, r1, «Wind_speedValidator»)
return value

Очистка от выбросов методом изолированного леса

Забегая вперед, нужно обозначить, что решение об очистке данных от выбросов было следствием малой точности прогноза тестовых данных (табл. 1).

Test loss и test acc являются метриками, используемыми для оценки производительности модели машинного обучения на тестовых данных.

Test loss (тестовая ошибка) – это мера ошибки или разницы между предсказанными значениями и фактическими значениями на тестовых данных. Test loss обычно вычисляется с использованием некоторой функции потерь (например, Mean Squared Error или Cross-Entropy Loss) и показывает, насколько хорошо модель справляется с предсказанием правильных значений на тестовых данных. Чем меньше значение test loss, тем лучше производительность модели.

Test accuracy (точность на тестовых данных) – это процент правильных предсказаний модели на тестовых данных. Он показывает, насколько хорошо модель классифицирует правильные классы на тестовых данных. Test accuracy вычисляется путем сравнения предсказанных значений с фактическими метками классов и вычисления доли правильных ответов. Чем выше значение test accuracy, тем лучше производительность модели.

Обе метрики, test loss и test accuracy, используются для оценки производительности модели на данных, которые не использовались при обучении. Они позволяют проверить, насколько хорошо модель обобщает знания и способна предсказывать правильные значения на новых данных.

Проверке на аномалии подвергались все факторы, но только интенсивность транспортного потока показала значительное загрязнение данных случайными выбросами. Это

Таблица 1

Фрагмент данных оценки моделей методами test loss и test acc до и после исключения аномалий из показателя интенсивности транспортного потока [составлено авторами]

Рубеж	С аномалиями		Без аномалий	
	test loss	test acc	test loss	test acc
1011	0.21	0.44	0.36	0.69
1021	0.26	0.38	0.30	0.67
1031	0.25	0.31	0.62	0.52
111	0.52	0.26	0.89	0.43
1111	0.42	0.24	0.74	0.20
1121	0.25	0.35	0.33	0.65
1131	0.23	0.40	0.32	0.62
1141	0.25	0.47	0.40	0.60

следствие ДТП, обслуживания дорог или прилегающих сооружений с временным ограничением движения, поломка фиксирующего устройства и т. п.

Для поиска аномалий лучше всего подошел метод «изолированного леса», так как показатель интенсивности транспортного потока не имеет ярко выраженного нормального распределения.

«Изолированный лес» (Isolation Forest) – это алгоритм машинного обучения для обнаружения аномалий в данных. В отличие от других алгоритмов, таких как метод кластеризации или плотность, изолированный лес использует деревья решений для поиска аномалий.

Основная идея изолированного леса заключается в том, что аномалии в данных обычно имеют меньшее число связей и более короткие расстояния до остальных объектов. Алгоритм строит несколько деревьев решений, случайным образом выбирая признаки и разделяя данные на каждом шагу. Затем он оценивает, как быстро аномалия может быть изолирована в этих деревьях.

При обучении изолированного леса:

1. Задается максимальная глубина дерева и максимальное количество разбиений.

2. Данные разбиваются на случайные подвыборки, и для каждой подвыборки строится дерево решений.

3. Каждое дерево решений разделяет данные на две части, выбирая случайный признак и случайное разделение.

4. Повторяются шаги 2–3 до достижения максимальной глубины или количества разбиений.

5. По окончании обучения и для нового объекта используется валидационная оценка для определения аномальности.

При использовании изолированного леса для обнаружения аномалий:

1. Вычисляется «путь» в дереве для каждого объекта, представляющий количество разбиений, необходимых для изоляции объекта.

2. Вычисляется средняя длина пути для всех объектов, которая используется для определения аномалий. Объекты с более короткими путями обычно считаются более аномальными.

Преимущества использования изолированного леса:

1. Хорошо масштабируется для обработки больших объемов данных.

2. Мощный и эффективный в обнаружении аномалий.

3. Не требует предварительной подготовки данных, так как нечувствителен к выбросам и мультиколлинеарности.

Существующий способ реализации метода случайного леса на языке программирования Python требует определенные версии специальных библиотек самой программы, которые не совместимы с данным проектом, поэтому в среде разработки PyCharm было создано дополнительное виртуальное окружение с ключевым модулем *clean_start* (рис. 10).

При запуске модуля *clean_start* на третьей строчке импортируется метод изолированного леса библиотеки *sklearn* надстройки *ensemble*. Запускается циклическая функция для каждого рубежа из директории */output/validated* и отрабатывается функция *cleaner* (рис. 10), где:

– на шестой строчке загружаются данные;

– на седьмой строчке удаляется первая индексная колонка;

– на десятой строчке формируется модель и задаются параметры;

Рис. 10. Очистка данных интенсивности транспортного потока методом изолированного леса [выполнено авторами].

– на одиннадцатой строчке в модель передаются данные об интенсивности транспортного потока;

– на двенадцатой-тринадцатой строчках определяются и помечаются аномалии. Создаются дополнительные две колонки. Одна отражает баллы значений (*scores_intensity*), вторая (*anomaly_intensity*) помечает в соответствии с баллами все аномальные значения цифрой «-1», а нормальные «1».

В конце каждого цикла преобразованные данные снова записываются в директорию / *output/validated*.

Для исключения аномалий перед обучением была произведена сортировка данных по значению «1» столбца *anomaly_intensity*, командой *data_frame = data_frame.loc[data_frame[‘anomaly_intensity’] == 1*].

ЗАКЛЮЧЕНИЕ

Предварительное преобразование данных с целью их подготовки к обработке методами машинного обучения является важным компонентом достижения точности полученных результатов, объективного сравнения признаков и определения зависимостей.

Использование информационно-аналитических ресурсов больших городов становится перспективным для анализа интенсивности транспортного потока. С этой целью использовались алгоритмические инструментарии, основанные на полученных теоретических результатах и программных прототипах их основных компонентов, методов и алгоритмов предиктивной аналитики языка программирования Python, среды разработки PyCharm.

Дальнейшая реализация результатов исследования направлена на построение нейронной сети глубокого обучения, обучения модели и прогнозирование, поэтому, предварительная обработка является важным компонентом логики программы, позволяет улучшить качество данных, упростить модель и сделать обучение более эффективным.

Из-за высокой динамичности интенсивности транспортного потока и большого количества нестабильных параметров, оказывающих на него влияние, наблюдаются непрогнозируемые (аномальные) значения, в число которых входят ДТП, строительные работы, сбои фиксирующей аппаратуры и т. п. Для их исключения, после предварительной обработки данных применен метод изолированного леса (Isolation Forest).

Изолированный лес использовался для поиска аномалий путем построения ансамбля изолированных деревьев, в каждом из которых случайным образом выбираются разбиения по случайным признакам и точки данных для построения дерева. Затем аномалии исключаются путем уменьшения высоты аномальных точек в дереве.

Данная работа стала платформой для обучения моделей зависимости интенсивности транспортного потока от погодно-климатических характеристик и факторов УДС методом глубокого обучения. В табл. 1 продемонстрировано качество модели нейронной сети до и после обработки данных, в том числе методом изолированного леса. Результаты исследований апробированы и с научной точки зрения взвешены, свидетельствуют о повышении качества прогностической модели на тестовых данных.

Работа актуальна с теоретической и практической точек зрения и уже имеет примеры применения, которые будут опубликованы в следующих трудах.

СПИСОК ИСТОЧНИКОВ

1. Бабков В. Ф. Дорожные условия и безопасность движения. – М.: Транспорт, 1993. – 271 с. ISBN 5-277-01402-0.

2. Pavlyuk, D. Robust and Responsive Learning of Spatiotemporal Urban Traffic Flow Relationships. IEEE Transactions on Intelligent Transportation Systems, 2022, Vol. 23, Iss. 9, pp. 14524–14541. DOI: 10.1109/TITS.2021.3130146.

3. Oktarina, Yu., Sastiani, D. Z., Dewi, T. Simulation Design of Artificial Intelligence Controlled Goods Transport Robot. Computer Engineering and Applications Journal, 2022, Vol. 11, Iss. 2, pp. 155–165. DOI: 10.18495/COMENGAPP.V11I2.411.

4. Пугачев И. Н., Щеглов В. И. Реализация программ комплексного развития транспортных инфраструктур агломераций и соседствующих субъектов Российской Федерации на основе создания информационной системы // Транспорт и сервис. – 2021. – № 9. – С. 7–16. EDN: AMDQXT.

5. Lugbade, S., Ojo, S., Imoize, A. L., Isabona, J., Alaba, M. O. A Review of Artificial Intelligence and Machine Learning for Incident Detectors in Road Transport Systems. Mathematical and Computational Applications, 2022, Vol. 27, Iss 5, 77. DOI: 10.3390/mca27050077.

6. Москвитин В. М., Семенова Н. И. Влияние шума на рекуррентные нейронные сети с нелинейными нейронами // Известия высших учебных заведений. Прикладная нелинейная динамика. – 2023. – Т. 31. – № 4. – С. 484–500. DOI: https://doi.org/10.18500/0869-6632-003052.

7. Пугачев И. Н., Тормозов В. С. Разработка нового метода детектирования и классификация транспортных средств по спутниковым изображениям // Дороги и мосты. – 2023. – Вып. 49–1. – С. 199–221. [Электронный ресурс]: https://rosdornii.ru/upload/iblock/de5/4m83hirrtm29nzdyiknu9vg4rse3q7me/11.-Pugachev-Razrabotka-novogo-metoda.pdf. Доступ 29.12.2023.

8. Пугачев И. Н., Скрипко П. Б., Шешера Н. Г. Программный подход к комплексному сбору и подготовки данных об интенсивности движения транспортных средств, погодных условий и естественной освещенности в часовых интервалах // T-Comm: Телекоммуникации и транспорт. – 2023. – Т. 17. – № 10. – С. 43–51. DOI: 10.36724/2072-8735-2023-17-10-43-51.

9. Kopp, T., Weitemeyer, R., Beyer, J., Ziegler, D., Hess, R. Entscheidungsunterstützung in Leitstellen des Personennahverkehrs – Technische und sozio-technische Herausforderungen [Artificial Intelligence for Decision Support in Local Public Transport Control Centers–Technical and Socio-technical Challenges]. HMD Praxis der Wirtschaftsinformatik, 2023, Vol. 60, Iss. 6, pp. 1156–1173. DOI: https://doi.org/10.1365/s40702-023-00996-8.

10. Пугачев И. Н., Шешера Н. Г. Применение методов статистического анализа для оценки параметров транспортных потоков и характеристик улично-дорожной сети // Дальневосточный юрид. ин-т МВД России. – Хабаровск: РИО ДВЮИ МВД России, 2020. – 108 с. ISBN 978-5-9753-0313-4.

11. Черных В. С., Жихарев А. Г., Федосеев А. Д., Мартон Н. А. Сравнение эффективности различных методов обучения нейронных сетей // Научный результат Информационные технологии. – 2023. – Т. 8. – № 1. – С. 83–93. DOI: 10.18413/2518-1092-2022-8-1-0-8.

12. Михалев О. Н., Янюшкин А. С. Автоматизация технологических процессов на основе нейронной сети. Автоматизация. Современные технологии. – 2022. – Т. 76. – № 4. – С. 147–152. EDN: DIEMHO.

13. Nguyen, H. T., Nguyen, L. T., Afanasiev, A. D., Pham, L. T. Classification of Road Pavement Defects Based on Convolution Neural Network in Keras. Automatic Control and Computer Sciences, 2022, Vol. 56, Iss. 1, pp. 17–25. DOI: https://doi.org/10.3103/S0146411622010084.

14. Сокерин Д. Д. Введение в искусственные нейронные сети // Информация и образование: границы коммуникаций. – 2023. – № 15 (23). – С. 284–286. EDN: WKHNMZ.

15. Пугачев И. Н., Шешера Н. Г., Щеглов В. И. Анализ геометрических элементов дорог с помощью современных геоинформационных систем при оценке их аварийности // Вестник гражданских инженеров. – 2021. – № 3 (86). – С. 127–133. DOI: 10.23968/1999-5571-2021-18-3-127-133.

16. Аррыкова Г. К., Эзизова С. Э., Гараев Г. Б., Ходжакаева Д. М. Искусственный интеллект и нейронные сети: современные технологии в решении ключевых проблем. Ceteris Paribus, 2023, Iss. 12, pp. 16–18. [Электронный ресурс]: https://sciartel.ru/arhiv-journal/CP-2023–12.pdf?ysclid=m06mw02bpt288436451 [полный текст выпуска]. Доступ 29.12.2023.

17. Шашев Д. В., Шатравин В. В. Реализация сигмоидной функции активации с помощью концепции перестраиваемых вычислительных сред // Вестник Томского государственного университета. Управление, вычислительная техника и информатика. – 2022. – № 61. – С. 117–127. EDN: PDIJZM.

18. Дулесов А. С., Байшев А. В., Карандеев Д. Ю., Дулесова Н. В., Карандеева И. Ю. Предварительная обработка статистических данных о состоянии однородных технических объектов // Научно-технический вестник Поволжья. – 2023. – № 4. – С. 80–83. EDN: EFIDGU.

19. Резова Н. Л., Казаковцев Л. А., Шкаберина Г. Ш., Цепкова М. И. Предварительная обработка данных для анализа поведения сложных систем // Системы управления и информационные технологии. – 2022. – № 2 (88). – С. 40–45. EDN: BYGESB.

20. Акимов А. А., Валитов Д. Р., Кубряк А. И. Предварительная обработка данных для машинного обучения // Научное обозрение. Технические науки. – 2022. – № 2. – С. 26–31. EDN: GWGJSK.

21. Безменов И. В. Метод очистки измерительных данных от выбросов: поиск оптимального решения с минимальным количеством отбракованных результатов измерений // Измерительная техника. – 2023. – № 1. – С. 16–23. EDN: KISWIN.

Информация об авторах:

***Пугачев Игорь Николаевич** – доктор технических наук, доцент, заместитель директора по научной работе Хабаровского Федерального исследовательского центра Дальневосточного отделения Российской академии наук (ХФИЦ ДВО РАН), Хабаровск, Россия, ipugachev64@mail.ru.*

***Шешера Николай Геннадьевич** – кандидат технических наук, доцент кафедры информационного и технического обеспечения ОВД, Дальневосточный юридический институт МВД России, Хабаровск, Россия, kolyaka239@mail.ru.*

***Григоров Денис Евгеньевич** – начальник кабинета специальных дисциплин кафедры информационного и технического обеспечения ОВД, Дальневосточный юридический институт МВД России, Хабаровск, Россия, glowfisch8lan@gmail.com.*

Статья поступила в редакцию 29.12.2023, одобрена после рецензирования 24.04.2024, принята к публикации 15.05.2024.