УДК 004.032.26

А. Н. Мущук
Белорусский государственный технологический университет

ПРОГНОЗИРОВАНИЕ КОЛИЧЕСТВА ПРОИСШЕСТВИЙ С ИСПОЛЬЗОВАНИЕМ НЕЙРОННОЙ СЕТИ НА ОСНОВЕ ФРЕЙМВОРКА KERAS

В статье представлен процесс разработки архитектуры нейронной сети для прогнозирования количества происшествий на основе языка Python с использованием фреймворка Keras. Рассмотрен весь цикл проектирования подобных архитектур. Методика построения архитектуры включает несколько этапов: сбор, группировка и обработка данных с использованием соответствующих методов, формирование обучающей и тестовой выборки, подбор подходящей архитектуры сети с учетом специфических требований решаемой задачи прогнозирования. Далее осуществляется реализация, обучение и тестирование нейронной сети на независимом наборе данных для оценки производительности и точности. Проводится сравнительный анализ предложенной архитектуры с известными по критерию средней абсолютной ошибки.

Ключевые слова: нейронные сети, машинное обучение, Keras, долгая краткосрочная память.

Для цитирования: Мущук А. Н. Прогнозирование количества происшествий с использованием нейронной сети на основе фреймворка Keras // Труды БГТУ. Сер. 3, Физико-математические науки и информатика. 2024. № 1 (278). С. 58–63.
DOI: 10.52065/2520-6141-2024-278-9.

A. N. Mushchuk
Belarusian State Technological University

FORECASTING THE NUMBER OF INCIDENTS USING A NEURAL NETWORK BASED ON THE KERAS FRAMEWORK

The article presents the process of developing a neural network architecture for predicting the number of incidents based on socio-economic indicators in Python using the Keras framework. The entire design cycle of such architectures is considered. The methodology for constructing an architecture includes several stages: collecting, grouping and processing data using appropriate methods, forming a training and test set, selecting a suitable network architecture taking into account the specific requirements of the forecasting problem being solved. Next, the neural network is implemented, trained, and tested on an independent data set to evaluate performance and accuracy. A comparative analysis of the proposed architecture with the known ones is carried out according to the criterion of mean absolute error.

Keywords: neural network, machine learning, Keras, long short-term memory.

For citation: Mushchuk A. N. Forecasting the number of incidents using a neural network based on the Keras framework. *Proceedings of BSTU, issue 3, Physics and Mathematics Informatics*, 2024, no. 1 (278), pp. 58–63 (In Russian).
DOI: 10.52065/2520-6141-2024-278-9.

Введение. Машинное обучение в настоящий момент используются во многих сферах жизнедеятельности человека для решения большого количества разных задач [1]. Это обусловлено их способностью к обучению на больших объемах данных и выявлению сложных зависимостей. Вот некоторые задачи, успешно решаемые при помощи алгоритмов машинного обучения.

Классификация объектов по категориям на основе их признаков. Данного вида задачи подразделяется на бинарные и многоклассовые. Примерами могут служить распознавание объектов на изображениях, классификация почтовых писем, определение категории товаров и пр. [2].

Алгоритмы машинного обучения широко применяются для обнаружения и отслеживания объектов на изображениях и видео. Это может быть поиск лиц, транспортных средств, определение движущихся объектов и т. д. [3].

Все большую популярность приобретают алгоритмы, используемые для предоставления рекомендаций пользователям на основе их предыдущих действий и интересов, таких как рекомендации фильмов, музыки, товаров и др. [4].

Широкое применение машинное обучение получило для решения задач управления автономными системами и робототехникой, включая управление промышленными процессами, автопилотами, умными устройствами и пр. [5].

Также алгоритмы машинного обучения применяются для задач регрессии. Под регрессией подразумевается прогнозирование числовых значений на основе входных данных. Это может быть предсказание цен на акции, потребления энергии,

температурных изменений и др. [6]. Помимо этого, регрессия используется для прогнозирования социальных факторов и явлений. Следует понимать, что предсказания в рамках данной тематики в достаточной мере стихийны и не могут быть сведены к каким-либо математическим формулам или моделям. Из чего следует, что получение однозначного прогноза является сложной задачей. Однако есть множество примеров, которые говорят о том, что использование алгоритмов машинного обучения все же позволяет добиться приемлемой точности. Примерами такого рода исследований служат реализации нейронных сетей для прогнозирования ожидаемой продолжительности жизни [7], академической успеваемости [8], миграционных потоков [9], уровня преступности [10] и др.

Стоит подчеркнуть, что прогнозирование социальных явлений – задача, для решения которой требуется не только разработать нейронную сеть, но и понимать причины, влияющие на них.

Для задач предсказания используется довольно большое количество алгоритмов. Самыми распространенными являются линейная регрессия, *k*-ближайших соседей, случайный лес, многослойный перцептрон, рекуррентная нейронная сеть и ее модификации [11].

Независимо от выбора архитектуры процесс разработки будет включать в себя сбор информации, ее подготовку с учетом используемых функций, разделение входных данных на обучающую и тестовую выборки, реализацию алгоритма, обучение на подготовленных данных и проверку точности полученных результатов на тестовом наборе. При необходимости, по завершению всех этапов, делается реконфигурация настроек, а затем повторяются процессы обучения и проверки точности [12].

Большинство существующих регрессионных алгоритмов неприменимы для решения проблемы прогнозирования количества происшествий по социально-экономическим показателям стран, так как обрабатывают данные в соответствии с заложенной в них математической функцией. В то же время нейронные сети позволяют найти закономерность, при этом не сводя преобразование данных к линейной функции. Однако существующие архитектуры также нельзя использовать для решаемой проблемы. Причиной этому служит то, что каждая отдельная реализации нейронной сети отличается от других количеством входных данных, необходимой точностью расчетов, числом слоев и другими параметрами архитектуры нейронной сети.

Ниже рассмотрим процесс проектирования архитектуры и реализации прогнозирующей происшествия по социально-экономическим показателям нейронной сети, разработанной на языке программирования *Python* с использованием фреймворка *Keras*.

Основная часть. Методика построения архитектуры нейронной сети включает в себя следующие этапы.

1. Сбор данных – это первейшая задача при разработке нейронных сетей. Для успешного прогноза необходимо собрать данные по значащим факторам. К таким относятся:

– исторические данные, представленные количеством, страной и годом произошедших инцидентов;

– социо-демографические показатели, представленные возрастной группой, численностью населения этой возрастной группы, полом, средним количеством лет на обучение и индексом развития человеческого потенциала;

– экономические показатели, представленные валовым внутренним продуктом;

– медицинские показатели, представленные статистикой психических расстройств в процентах от общего количества населения.

Сбор данных осуществляется из открытых источников (например *Kaggle*, сайт Всемирной организации здравоохранения и Всемирный банк данных).

2. Обработка данных. Для этого используются инструменты библиотек *Pandas* и *SkititLearn*. Первым этапом обработки является объединение всех данных в один набор. Для этого данные импортируются, после чего к ним применяется функция *merge* для объединения. Для слияния используется операция *Inner join*, поскольку потеря данных из-за отсутствия значений достаточно незначительна.

Следующим этапом является преобразование категориальных переменных в числовые значения. Категориальными переменными являются название страны, год, пол и возрастная группа. Для данного преобразования используется метод *get_dummies* [13]. Он имеет один достаточно существенный минус: каждая категориальная переменная преобразуется в отдельный столбец, из чего следует, что при большом разнообразии значений получится большое количество столбцов. Так, из исходных 10 столбцов был получен 131.

Последний этап – преобразование значимых данных. Это необходимо для их сведения к единому виду с целью облегчить обучение. Видов преобразования достаточно много, самый универсальный и подходящий – это нормализация. Чаще всего для нормализации используются функции *StandartScaler* и *MinMaxScaler* [14]. *StandartScaler* подходит для ситуаций, когда значения могут быть как положительными, так и отрицательными. В данном же случае все значения строго положительные, поэтому использована функция *MinMaxScaler*. Итоговое значение каждого

входного параметра рассчитывается по следующей формуле:

$$Xs = \frac{X - X.\min}{X.\max - X.\min}(\max - \min) + \min, \quad (1)$$

где X – входной параметр; $X.\min$ – минимальное значение параметра; $X.\max$ – максимальное значение параметра; max – предельный максимум, задаваемый при использовании функции; min – предельный минимум, задаваемый при использовании функции.

Другими словами, *MinMaxScaler* нормализует значения в диапазон от min до max. Для данной нейронной сети диапазон будет иметь значения от 0 до 1 с точностью того же порядка, что и максимальное число.

3. Разделение нормализованных данных на обучающую и тестовую части. Для этой цели используется метод *train_test_split*, в который в качестве параметров передаются входные и выходные параметры, размер тестовой выборки в процентах от общего числа и целочисленная переменная *random_state*, которая используется при рандомизации. Последняя необходима для однозначного воспроизведения случайных значений. Так, при многократном разделении данных, но неизменном *random_state* на выходе будут получаться одинаковые выборки, что, в свою очередь, очень полезно при сравнении эффективности различных архитектур.

4. Выбор подходящей архитектуры. В зависимости от специфики предсказываемого значения для задач предсказания используются рекуррентные нейронные сети (RNN – Recurrent Neural Network) и их подвиды, метод случайного леса или комбинации различных слоев. Подходящим выбором будет архитектура долгой краткосрочной памяти (LSTM – Long Short-Term Memory), которая является подвидом RNN [12]. LSTM имеет ряд преимуществ:

– быстрая скорость обучения;

– использование результатов каждой итерации для обучения последующих;

– отсутствие проблемы исчезающих градиентов.

Для реализации нейронной сети используется фреймворк *Keras*, который дает возможность настройки типа и количества слоев, а также количества нейронов на каждом из них.

Перед тем как настраивать архитектуру, необходимо привести данные к корректному виду. Большинство архитектур на вход могут принимать данные в виде двумерного массива. Однако LSTM принимает на вход двумерный массив, где в ячейках содержатся также двумерные массивы. Для данного преобразования используем функцию *reshape*, в которую в качестве параметров передаются три: исходный двумерный массив, единица, для указания того, что новый массив будет двумерным, и количество столбцов.

5. Реализация и настройка параметров нейронной сети. Экспериментальным путем было обнаружено, что оптимальной является сеть с четырьмя слоями, первые два из которых это, собственно, LSTM-слои, затем слой полносвязных нейронов, задача которых – скорректировать информацию, и последний – нейрон, объединяющий выходные данные в одно получаемое значение.

Для инициализации сети используется метод *Sequential*, после чего при помощи методов *add* в нее добавляются слои.

Первые два слоя добавляются с параметром LSTM, который, в свою очередь, является функцией со своими параметрами: количеством нейронов, формой ввода и функцией активации. Определить оптимальное количество нейронов можно лишь экспериментально, но в общем случае рекомендуется брать среднее значение количества входных и выходных параметров, при необходимости корректируя его. В данном случае экспериментально было установлено, что наиболее точный прогноз сеть выдает на значении 80 нейронов. Параметр формы ввода содержит в себе количество рассматриваемых в одну единицу времени срезов данных и число параметров в нем. Для данной сети единовременно рассматривается 1 срез со 131-м параметром.

Функции активации и повторной активации являются теми элементами нейрона, которые определяют, пойдут данные дальше по сети или нет [15]. Именно с ними сеть при обучении получала минимальное значение ошибки. Функция повторной активации осуществляет внутренние рекуррентные вычисления. Для обоих слоев функцией повторной активации является сигмоида, рассчитываемая по следующей формуле:

$$f(x) = \frac{1}{1 + e^{-x}}, \quad (2)$$

где e – экспонента (число Эйлера), а x – входное значение. Функция активации вычисляет конечный результат слоя. Функцией активации для LSTM-слоев является тангенсоида, рассчитываемая по следующей формуле:

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}, \quad (3)$$

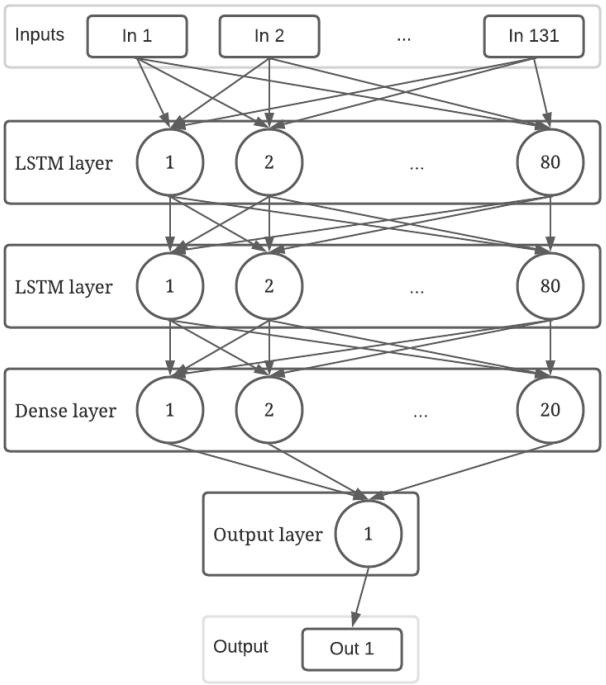
где e – экспонента (число Эйлера), а x – входное значение.

Отличие тангенсоиды от синусоиды в том, что синусоида более сглаженная. Из чего следует, что она дает больше возможностей для предварительной подгонки весов. В свою очередь тангенсоида необходима для распознавания сложных зависимостей в данных.

Следующий слой является полносвязным, определяемым при помощи функции *Dense*. Она содержит двадцать нейронов, а функцией активации является выпрямленная единица (ReLu), которая подает на выход либо 0, либо преобразованное внутри нейрона значение. Это имеет свои недостатки. В процессе обучения многие нейроны могут получить отрицательное значение, отключиться и в дальнейшем не использоваться на протяжении обучения. Поэтому данная функция активации используется на промежуточном слое, по сути, для отсечения явно некорректных данных. Все функции активации были выбраны в результате экспериментов.

Последний слой является копией предыдущего, за исключением того, что в нем только один нейрон. Это необходимо для того, чтобы сеть давала на выход только одно значение.

Полученная архитектура сети представлена на рисунке.



Архитектура сети

Следующий шаг – это сборка сети. Она производится при помощи метода *compile*, в который передаются параметры о функции, подсчитывающей потери, оценщике и методе оптимизации. В качестве метода оптимизации был выбран алгоритм Адам со значением обновления весов 0.001 [15]. Данный метод взят в основном из-за того, что он дает наиболее существенный прирост производительности при незначительной потере точности. Функция потери используется при корректировке весов модели на каждой итерации обучения. В данной сети используется среднеквадратическая ошибка (MSE – Mean Squared Error), рассчитываемая по следующей формуле:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y})^2, \quad (4)$$

где n – общее количество предсказанных значений; y_i – i-е возвращаемое значение; $\hat{y}$ – среднее возвращаемых значений. В свою очередь, оценщик не влияет на обучение, а служит исключительно показателем того, насколько точные значения сеть предсказывает. В качестве оценщика взята функция средней абсолютной ошибки (MAE – Mean Absolute Error), рассчитываемая по следующей формуле:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - x_i|, \quad (5)$$

где n – общее количество предсказанных значений; y_i – возвращаемое значение; x_i – тестовое значение.

Затем следует обучить модель, для чего вызывается метод *fit*, в который передаются в качестве параметров данные для обучения и сравнения и количество эпох обучения. Каждая эпоха представляет собой полный цикл обучения на всех тестовых данных. Главное, что следует помнить при установке данного параметра, это то, что в какой-то момент сеть достигает оптимальной точности, а в дальнейшем при прохождении каждой эпохи точность начинает изменяться как вниз, так и вверх. Соответственно, необходимо обнаружить, при каком значении это происходит, и указать его. Можно указать и большее значение, однако дополнительные циклы обучения уже не дадут существенного результата. В данной сети повышение точности прогноза перестает происходить приблизительно на 180-й эпохе, однако для надежности параметр установлен на значении 200.

После окончания обучения происходит тестирование разработанной сети. Для этого применяется ранее выделенная тестовая выборка. Используя метод *predict* с параметром в виде тестовых входных данных, получаем на выходе предсказанные данные, которые, в свою очередь, сравниваются с тестовым результатом.

Для того чтобы оценить эффективность полученной сети, было проведено ее сравнение с более классической реализацией, состоящей из одного LSTM-слоя с 60 нейронами и слоя из одного нейрона для выходящего значения, а также реализацией алгоритма случайного леса из библиотеки *SkititLearn*. Оценка проводилась по значению средней абсолютной ошибки десяти циклов обучения на случайно перемешанных данных. Значения средней абсолютной ошибки для каждой реализации представлены в таблице.

Сравнение значений средней абсолютной ошибки

№	Реализованная сеть	LSTM с одним слоем	Алгоритм случайного леса
1	2,5813	3,9163	2,922
2	2,6718	3,5941	2,6938
3	2,669	3,8631	2,7279
4	2,6964	3,781	2,7665
5	2,5597	4,0208	2,4035
6	2,6508	3,8405	3,2644
7	2,6978	3,8022	2,7965
8	2,7274	3,9179	3,2066
9	2,695	3,7753	2,7722
10	2,6793	4,0544	2,9457

Среднее значение MAE для тестовых данных реализованной сети составило 2,6629. В свою очередь для однослойной LSTM и алгоритма случайного леса этот показатель составил 3,8566 и 2,8499 соответственно. Следовательно, реализованная сеть значительно лучше предсказывает результат, чем более классическая, и имеет небольшое преимущество в точности по сравнению с методом случайного леса. Однако стоит отметить, что реализованная сеть обучается дольше, чем аналоги.

После оценки модели могут быть предприняты дальнейшие шаги для улучшения ее эффективности. Это включает в себя расширение набора данных, изменение архитектуры сети, подбор оптимальных параметров или использование различных методик оптимизации.

Заключение. В данной статье описан весь процесс создания и реализации архитектуры прогнозирующей нейронной сети. Рассмотрены методы, используемые для нормализации входных данных, функции активации и формулы подсчета потерь. Представлена архитектура нейронной сети для решения задачи прогнозирования количества происшествий по социально-экономическим показателям стран.

Предложенная архитектура способна предсказывать значения с достаточно высокой точностью. В сравнении с известными архитектурами реализованная имеет меньшее значение средней абсолютной ошибки, но при этом тратит вдвое больше времени на обучение.

Разработанная модель может использоваться как по прямому назначению, так и как вспомогательный инструмент при развитии программ и мер по предотвращению происшествий.

Список литературы

1. Tufail Sh., Riggs H., Tariq M., Sarwat A. I. Advancements and Challenges in Machine Learning: A Comprehensive Review of Models, Libraries, Applications, and Algorithms // Electronics. 2023. Vol. 12, no. 8. P. 1789. DOI: 10.3390/electronics12081789.

2. Распознавание подстилающей поверхности Земли с помощью сверточной нейронной сети на одноплатном микрокомпьютере /Д. А. Павленко [и др.] // Информатика. 2020. Т. 17, № 3. С. 36–43. DOI: 10.37661/1816-0301-2020-17-3-36-43.

3. Сорокина В. В., Абламейко С. В. Выделение отдельных участков тела человека на изображении с использованием нейронных сетей и модели внимания // Журнал Белорусского государственного университета. Математика. Информатика. 2022. Т. 2. С. 94–106. DOI: 10.33581/2520-6508-2022-2-94-106.

4. Portugal I., Alencar P., Cowan D. The use of machine learning algorithms in recommender systems: A systematic review // Expert Systems with Applications. 2018. Vol. 97. P. 205–227. DOI: 10.1016/j.eswa.2017.12.020.

5. О возможностях применения машинного обучения в системах управления освещением / С. В. Рослякова [и др.] // Научный результат. Сер.: Информационные технологии. 2021. Т. 6, № 4. С. 48–63. DOI: 10.18413/2518-1092-2021-6-4-0-7.

6. Regression Analysis for COVID-19 Infections and Deaths Based on Food Access and Health Issues / A. Almalki [et al.] // Healthcare. 2022. Vol. 10, no. 2. P. 324. DOI: 10.3390/healthcare10020324.

7. Life Expectancy: Prediction & Analysis using ML / V. Bali [et al.] // ICRITO: 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions). 2021. P. 1–8. DOI: 10.1109/ICRITO51393.2021.9596123.

8. Rajendran S., Chamundeswari S., Sinha A. Predicting the academic performance of middle-and high-school students using machine learning algorithms // Social Sciences & Humanities Open. 2022. Vol. 6, no. 1. P. 100357. DOI: 10.1016/j.ssaho.2022.100357.

9. Predicting the Income Groups and Number of Immigrants by Using Machine Learning (ML) / B. Aydemir [et al.] // International Journal of Multidisciplinary Studies and Innovative Technologies. 2022. Vol. 6, no. 2. P. 162–168. DOI: 10.36287/ijmsit.6.2.162.

10. Shah N., Bhagat N., Shah M. Crime forecasting: a machine learning and computer vision approach to crime prediction and prevention // Visual Computing for Industry, Biomedicine, and Art. 2021. Vol. 4. P. 1–14. DOI: 10.1186/s42492-021-00075-z.

11. Sarker I. Machine learning: Algorithms, real-world applications and research directions // SN computer science. 2021. Vol. 2, no. 3. P. 160. DOI: 10.1007/s42979-021-00592-x.

12. Шолле Ф. Глубокое обучение на Python. СПб.: Питер, 2018. 400 с.
13. McKinney W. In Python for data analysis // O'Reilly Media. 2013. Issue 2. P. 207–209.
14. Орельен Ж. Прикладное машинное обучение с помощью Scikit-Learn и TensorFlow. СПб.: Альфа-книга, 2018. 684 с.
15. Goodfellow I., Bengio Y., Courville A. Deep Learning. Massachusetts: Massachusetts Institute of Technology, 2017. 804 с.

References

1. Tufail Sh., Riggs H., Tariq M., Sarwat A. I. Advancements and Challenges in Machine Learning: A Comprehensive Review of Models, Libraries, Applications, and Algorithms *Electronics*, 2023, vol. 12, no. 8, p. 1789. DOI: 10.3390/electronics12081789.
2. Paulenka D. A., Kovalev V. A., Snezhko E. V., Liauchuk V. A., Pechkovsky E. I. Recognition of underlying surface using a convolutional neural network on a single-board computer. *Informatika* [Informatics], 2020, vol. 17, no. 3, pp. 36–43. DOI: 10.37661/1816-0301-2020-17-3-36-43 (In Russian).
3. Sorokina V. V, Ablameyko S. V. Detection of human body parts on the image using the neural networks and the attention model. *Zhurnal Belorusskogo gosudarstvennogo universiteta. Matematika. Informatika* [Journal of the Belarusian State University. Mathematics and Informatics], 2022, vol. 2, pp. 94–106. DOI: 10.33581/2520-6508-2022-2-94-106. (In Russian).
4. Portugal I., Alencar P., Cowan D. The use of machine learning algorithms in recommender systems: A systematic review. *Expert Systems with Application*, 2018, vol. 97, pp. 205–227. DOI: 10.1016/j.eswa. 2017.12.020.
5. Roslyakova S., Bragina T., Zemlyanova E., Korotkova D., Merkulova P., Laushkina A., Filippov I. On the possibilities of applying machine learning in lighting control systems. *Nauchnyy rezul'tat. Ser.: Informatsionnyye tekhnologii* [Research result. Information technologies], 2021, vol. 6, no. 4, pp. 48–63 (In Russian). DOI: 10.18413/2518-1092-2021-6-4-0-7.
6. Almalki A., Gokaraju B., Acquaah Y., Turlapaty A. Regression Analysis for COVID-19 Infections and Deaths Based on Food Access and Health Issues. *Healthcare*, 2022, vol. 10, no. 2, p. 324. DOI: 10.3390/healthcare10020324
7. Bali V. [et al.]. Life Expectancy: Prediction & Analysis using ML. *ICRITO: 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)*, 2021. P. 1–8. DOI: 10.1109/ICRITO51393.2021.9596123.
8. Rajendran S., Chamundeswari S., Sinha A. Predicting the academic performance of middle-and high-school students using machine learning algorithms. *Social Sciences & Humanities Open*, 2022, vol. 6, no. 1, p. 100357. DOI: 10.1016/j.ssaho.2022.100357.
9. Aydemir B. [et al.]. Predicting the Income Groups and Number of Immigrants by Using Machine Learning (ML). *International Journal of Multidisciplinary Studies and Innovative Technologies*, 2022, vol. 6, no. 2, pp. 162–168. DOI: 10.36287/ijmsit.6.2.162.
10. Shah N., Bhagat N., Shah M. Crime forecasting: a machine learning and computer vision approach to crime prediction and prevention. *Visual Computing for Industry, Biomedicine, and Art*, 2021, vol. 4, pp. 1–14. DOI: 10.1186/s42492-021-00075-z.
11. Sarker I. Machine learning: Algorithms, real-world applications and research directions. *SN computer science*, 2021, vol. 2, no. 3, p. 160. DOI: 10.1007/s42979-021-00592-x.
12. Chollet F. *Glubokoye obucheniye na Python* [Deep Learning with Python]. St. Petersburg, Piter Publ., 2018. 400 p. (In Russian).
13. McKinney W. In Python for data analysis. *O'Reilly Media*, 2013, issue 2, pp. 207–209.
14. Aurelien J. *Prikladnoye mashinnoye obucheniye s pomoshch'yu Scikit-Learn i TensorFlow* [Applied machine learning with Scikit-Learn and TensorFlow]. St. Petersburg, Al'fa-kniga Publ., 2018. 684 p. (In Russian).
15. Goodfellow I., Bengio Y., Courville A. Deep Learning. Massachusetts, Massachusetts Institute of Technology Publ., 2017. 804 p.

Информация об авторе

Мушук Артур Николаевич – магистрант кафедры программной инженерии. Белорусский государственный технологический университет (220006, г. Минск, ул. Свердлова, 13а, Республика Беларусь). E-mail: mushuk-artur@mail.ru

Information about the author

Mushchuk Artur Nikolaevich – Master's degree student, the Department of Software Engineering. Belarusian State Technological University (13a, Sverdlova str., 220006, Minsk, Republic of Belarus). E-mail: mushuk-artur@mail.ru

Поступила после доработки 09.02.2024