СЕМЕРИКОВ А. В., ГЛАЗЫРИН М. А. ПРОГНОЗИРОВАНИЕ РЕЗУЛЬТАТА УСПЕШНОСТИ ЗАВЕРШЕНИЯ ОБУЧЕНИЯ ПОТЕНЦИАЛЬНЫМ СТУДЕНТОМ УНИВЕРСИТЕТА

УДК 378.141.21:330.47, ВАК 1.2.2/05.13.18, ГРНТИ 28.17.31

Прогнозирование результата успешности завершения обучения потенциальным студентом университета

А. В. Семериков[1], М. А. Глазырин[2]

[1]Ухтинский государственный технический университет, г. Ухта
[2]Вятский государственный университет, г. Киров

В статье представлена нейронная сеть для прогнозирования результата успешности завершения обучения в университете потенциальным студентом. При построении, обучении и анализе нейронной сети использовались библиотеки Python: NumPy, pandas, Keras. В качестве исходных данных использовалась структура pandas.DataFrame, состоящая из 36830 строк и 13 колонок. Целевая функция принимает бинарные значения. Качество построенной нейронной сети имеет значение 0.66, а процент ошибок 0.65. Представлен пример расчета вероятности итогового результата обучения абитуриента.

Ключевые слова: *большие данные, нейронная сеть, признаки, целевой признак, Python, pandas, Keras.*

Simulation of a process model of functioning of the enterprises for rendering of services

A.V. Semerikov[1], M.A. Glazyrin[2]

[1]Ukhta State Technical University, Ukhta
[2]Vyatka State University, Kirov

The article presents a neural network for predicting the outcome of successful completion of training by a potential uni-versity student. When building, training and analyzing the neural network, the Py-thon libraries were used: NumPy, pan-das, Keras. The pandas.DataFrame structure, consisting of 36830 rows and 13 columns, was used as the initial data. The objective function takes binary values. The quality of the constructed neural network is 0.66, and the error rate is 0.65. An example of calculating the probability of the final learning outcome of an applicant is presented.

Keywords: *big data, neural network, features, target feature, Python, pandas, Keras.*

Введение

При поступлении в ВУЗ абитуриент предоставляет персональные данные, которые состоят из набора следующих параметров: «Институт», «Специальность», «Форма обучения», «Категория», «Средний балл», «Работа», «Пол», «Общежитие», «Медаль», «Город окончания», «Тип школы», «Регион», «Город».

Кроме того, в этом наборе данных имеется два параметра: «Факт окончания» и «Процент окончания». После окончания обучения с помощью этих параметров осуществляется фиксирование результата обучения. В случае успешного обучения (получения диплома) параметру «Факт окончания» присваивается значение 1, параметру «Процент окончания» присваивается значение 100 (студент сдал все экзамены и зачеты). В противном случае (не получение диплома) параметру «Факт окончания» присваивается значение 0, а параметру «Процент окончания» присваивается выраженная в процентах доля количества успешно сданных экзаменов и зачетов от их общего количества.

«Факт окончания» и «Процент окончания» можно представить в виде целевой функции, которая принимает значения в зависимости от значения параметров, перечисленных выше.

Имея большое количество значений целевой функции, представляется возможным, используя методы машинного обучения, построить дерево решений, регрессионную функцию (обучение с учителем) и осуществить процесс кластеризации студентов (обучение без учителя). На основе дерева решений, регрессии и кластеризации можно предсказать будущее студента первокурсника, то есть определить наиболее вероятный исход [1].

Наряду с этим результат успешности окончания университета потенциальным студентом можно предсказать, используя нейронные сети (НС) [2]-[4].

Экспериментальная часть

В настоящей статье представлено описание построения нейронной сети для решения задачи предсказания успешности окончания университета потенциальным студентом.

Для обучения и тестирования нейронной сети использовались данные в виде таблицы Microsoft Excel в формате «xlsx», в которых строки отражают набор значений признаков для описания отдельного обезличенного студента, а столбцы соответствуют этим признакам (Таблица 1). Последний столбец «Факт окончания» представляет собой целевой признак.

Таблица 1. Исходный набор данных по каждому студенту

Институт	Специальность	Форма обучения	Категория	Средний балл	Работа	Пол	Общежитие	Медаль	Город окончания	Тип школы	Регион	Город	Факт окончания
СТИ	Электропривод и автоматика промышленных установок и технологических комплексов	очная	общий конкурс	63,3	нет	м	да	нет медали	Печора	школа	Республика Коми	Ухта	1
ИнЭУиИТ	Автоматизированные системы обработки информации и управления	заочная	общий конкурс	65	нет	м	да	серебряная	Инта	училище	Республика Коми	Ухта	1

Первичный анализ данных с использованием Python-библиотеки pandas показал, что в представленных данных имеются шумы (Таблица 2).

Таблица 2. Количество студентов в институтах и факта окончания

Факт окончания	Институт											
	1	2	3	4	5	6	7	8	9	10	11	Все
0 (не закончил)	6454	4533	4576	4477	152	127	1	12	1	106	37	20476
1 (закончил)	5704	2847	3965	4274	14	49	0	0	0	89	191	17133
Все	12158	7380	8541	8751	166	176	1	12	1	195	228	37609

Из таблицы видно, что в имеющихся в распоряжении данных 98 % студентов обучались в первых четырех институтах (с 1 по 4). Поэтому из первичного набора данных строки, содержащие данные об университетах с 5 по 11, были удалены. Количество строк при этом уменьшилось с 37609 до 36830.

Наряду с этим, значения параметров в имеющемся наборе данных необходимо представить в специальном формате: все категориальные признаки преобразуются в числовые и все числовые признаки нормализуются (значения параметров находятся в диапазоне от 0 до 1). Тем самым в проектируемой НС исключается чрезмерное влияние параметров с большими значениями, что в конечном счете позволит получить НС более высокого качества.

Таким образом создан набор числовых данных (Таблица 3), характеризующий каждого студента по признакам: «Институт», «Специальность», «Форма обучения», «Категория», «Средний балл», «Работа», «Пол», «Общежитие», «Медаль», «Город окончания», «Тип школы», «Регион», «Город», «Факт окончания».

Таблица 3. Набор числовых данных о студентах

Специальность	Форма обучения	Категория	Средний балл	Работа	Пол	Общежитие	Медаль	Город окончания	Тип школы	Регион	Город	Факт окончания
0,000	0,000	0,000	0,522	0,000	1,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
0,009	0,000	0,500	0,260	0,000	1,000	1,000	0,000	0,001	0,063	0,000	0,001	0,000
0,018	0,000	0,000	0,629	1,000	1,000	0,000	0,000	0,002	0,063	0,000	0,002	1,000
0,027	0,000	0,500	0,256	0,000	1,000	0,000	0,000	0,002	0,125	0,000	0,003	1,000
0,036	0,000	0,000	0,441	0,000	1,000	0,000	0,000	0,002	0,000	0,000	0,004	0,000

Для создания НС можно воспользоваться одним из имеющихся пакетов [4]. В настоящее время лидером по использованию для разработки нейросетей является платформа TensorFlow, разработанная компанией Google специально для создания нейросетей самой разной структуры. Кроме того, TensorFlow позволяет проводить обучение на графических процессорах (GPU) компьютеров, объединенных в сеть. TensorFlow – относительно низкоуровневый фреймворк. В то же время, проектировать НС лучше на языках высокого уровня. Это значительно

упрощает процесс проектирования НС. Одним из самых популярных пакетов является Keras, который представляет собой некую надстройку над TensorFlow.

Загрузка TensorFlow и Keras производится через систему управления пакетами pip:

```
pip install tensorflow
pip install keras
```

В настоящей статье рассматривается построение нейронной сети на основе пакета Keras с использованием интерактивной вычислительной среды Jupyter Notebook [5].

Опишем последовательность действий для подготовки к созданию НС.

В начале загружаются библиотеки для работы с данными:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
plt.style.use("ggplot")
%matplotlib inline
import os
```

Затем загружается собственно пакет Keras, позволяющий создать НС:

```
import tensorflow as tf
from tensorflow import keras
```

Далее загружаются функции Keras: sequential, Denes, np_utils, optimizer.

Sequential позволяет создать НС прямого распространения. В этом случае информация из нижележащих нейронов не передается в первоначальные нейроны:

```
from keras.models import sequential
```

Denes позволяет создавать внутренние слои сети (архитектуру НС):

```
from keras.layers import Dense
```

Np_utils используется для формирования данных в виде множества:

```
from keras.utils import np_utils
```

Optimizer используется для определения качества НС:

from keras import optimizers

Теперь представим программный код по определению итогового результата создания и обучения НС.

С помощью библиотеки pandas загружаем данные DataFrame которые находятся в таблице формата Microsoft Excel и просматриваем их (Рисунок 1):

```
df=pd.read_excel(io='d:/Big/ugtuFst.xlsx', engine='openpyxl')
df.head()
```

	Специальность	Форма обучения	Категория	Средний балл	Работа	Пол	Общежитие	Медаль	Город оконч	Тип школы	Регион	Город	Факт окончания
0	0.000000	0	0.0	0.522222	0	1	0	0.0	0.000000	0.0000	0.0	0.000000	0
1	0.009091	0	0.5	0.259596	0	1	1	0.0	0.001057	0.0625	0.0	0.001002	0
2	0.018182	0	0.0	0.629293	1	1	0	0.0	0.002114	0.0625	0.0	0.002004	1
3	0.027273	0	0.5	0.255556	0	1	0	0.0	0.002114	0.1250	0.0	0.003006	1
4	0.036364	0	0.0	0.441414	0	1	0	0.0	0.002114	0.0000	0.0	0.004008	0

Рисунок 1. Числовой набор данных по каждому студенту в представлении Pandas

Для определения итогового количества строк и столбцов в исходных данных используем свойство df.shape. Количество строк составляет 36830, количество столбцов – 13.

Одним из условий достижения высокого качества НС является сбалансированность значений целевого признака. При однобоких данных НС будет часто ошибаться. В рассматриваемом случае сбалансированность должна быть у параметра факт окончания:

```
df['Факт окончания'].value_counts(normalize=True)
```

Проверка показала, что доля отчисленных студентов и закончивших университет соответственно равны 0.54 и 0.46. Это значит, что целевой признак имеет сбалансированный вид, а НС должна иметь хорошее качество (небольшое количество ошибок).

Перед обучением НС необходимо разделить отклик и предикторы и проверить их количество:

```
y=df['Факт окончания']
x=df.drop(['Факт окончания' ], axis = 1)
x.shape
```

Количество откликов равно 36830, количество предикатов равно 12.

Теперь с помощью библиотеки sklearn разделим данные на две части (обучающую и тестовую). На тестовую выборку выделяем 33 % от всего набора данных. Разделение производим с помощью команды train_test_split:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,ran-
dom_state=12345,test_size=0.33)
```

При создании DataFrame использовалась библиотека pandas. Keras же работает с множествами array созданными NumPy. Поэтому выполним преобразование обучающей и тестовой выборок:

```
x_train=x_train.values
y_train=y_train.values
x_test=x_test.values
y_test=y_test.values
```

Для определения качества созданной НС необходимо выполнить разделения целевого признак на два массива (обучающий и тестовый):

```
y_train_bin=np_utils.to_categorical(y_train)
y_test_bin=np_utils.to_categorical(y_test)
```

Просмотрим содержание первых пяти элементов созданных массивов:

```
y_train_bin[0:5]
y_test_bin[0:5]
```

```
array([[0., 1.],
       [1., 0.],
       [1., 0.],
       [0., 1.],
       [0., 1.]], dtype=float32)
array([[0., 1.],
       [0., 1.],
       [0., 1.],
       [1., 0.],
       [1., 0.]], dtype=float32)
```

Теперь все готово для построения НС.

Определим модель НС, как последовательную, т.е. состоящую из слоев, идущих друг за другом:

```
model=keras.Sequential()
```

Добавим в НС первый слой, содержащий 12 входных предикатов и 12 нейронов. В качестве активационной функции используем функцию relu:

```
model.add(Dense(12,input_dim=12,activation='relu'))
```

Конструктор Dense формирует полносвязный слой, то есть, все входы будут связаны со всеми нейронами. Дополнительно, автоматически, для каждого нейрона добавляется смещение – bias.

Создадим внутренний слой из 10 нейронов. В качестве активационной функции используем relu:

model.add(Dense(10,activation='relu'))

Создадим выходной слой из 2 нейронов и активационной функции softmax, которая позволяет определить частоту появления бинарного нейрона:

model.add(Dense(2,activation='softmax'))

Теперь, когда структура НС определена, ее нужно скомпилировать:

model.compile(loss='binary_crossentropy', optimizer='adam',metrics=['accuracy'])

В функции compile присутствуют параметры, предназначенные для назначения критерия качества НС loss, алгоритма поиска весовых коэффициентов optimizer и способа оценки значений metrics на выходе НС.

Критерии качества loss выбирается в зависимости от вида НС. В рамках данной задачи определение качества осуществляется с помощью функции правдоподобия Бернулли binary_crossentropy, которая используется, когда на выходе сети присутствуют два нейрона. В этом случае определяется частота появления бинарного события 1 и 0. Для случая с количеством выходных нейронов больше 2 используется функция categorical_crossentropy.

Определение весовых коэффициентов нейронов осуществляется на основе алгоритма градиентного спуска, представленного в методе adam.

Определение процента правильно указанных значений целевой функции на тестовой выборке выполняется методом accuracy.

Таким образом, на этапе компиляции определены: функция качества, метод градиентного спуска и метод нахождения ошибки НС. Сеть автоматически инициализируется начальными значениями весов связей. Так, что теперь она полностью готова к этапу обучения.

Для запуска процесса обучения используется метод fit:

log=model.fit(x_train,y_train_bin,epochs=150,batch_size=100,verbose=False)

На вход метода передается обучающая выборка x_train, затем y_train_bin, число обучений epochs (количество итераций обучающей выборки) и параметр batch_size. Таким образом обучающая выборка x_train будет пропущена через сеть 150 раз и через каждый набор в размере 100 будут корректироваться весовые коэффициенты нейронов и вычисляться значение критерия качества.

Последний параметр verbose=False указывает что не нужно отображать текущую информацию при обучении сети.

Наконец НС сформирована и можно посмотреть, как обучается НС и как изменяется качество НС в зависимости от номера эпохи (Рисунок 2):

```
plt.plot(log.history['loss'])
plt.grid(True)
plt.show()
```

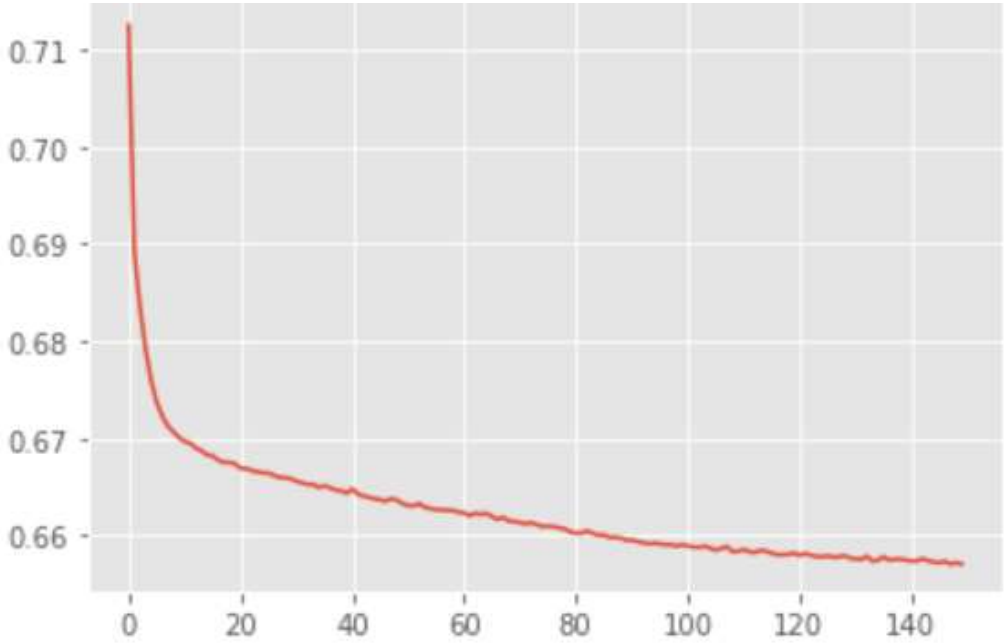

Рисунок 2. Зависимость качества сети от номера эпохи

Согласно рисунку 2 можно сделать заключение, что на эпохе 150 качество модели не уменьшается и оценивается в 0.65, то есть НС определяет правильно результат обучения студента с вероятностью 0.65.

При завершении обучения НС можно узнать значения весовых коэффициентов:

```
print(model.get_weights())
```

[0.15583228, 0.3463995, -0.5327399, 0.1638723, -0.06374401, 0.43925568, 0.29024795, -0.5132758, -0.43350118, -0.16405626, -0.6230817, -0.01503731]

Для отображения значений выходных нейронов на тестовой выборке используем функцию predict:

```
classes=model.predict(x_test,batch_size=128)
print(classes)
```

[0.765464 0.23453605]

[0.64950967 0.35049036]
[0.61011064 0.3898894]
 ...

Согласно полученным данным первый в этом списке студент с вероятностью 0.77 не получит диплом об образовании и с вероятностью 0.23 получит диплом после курса обучения.

Для определения ошибки НС на тестовой выборке используем функцию evaluate:

```
scores=model.evaluate(x_test,y_test_bin)
print(scores)
```

[0.6645331978797913, 0.6020240187644958]

Построенная и обученная НС с критерием качества 0.66 может быть использована для прогноза возможного итогового результата обучения вновь поступающего абитуриента.

Для получения прогноза возьмем 12 входных параметров согласно Рисунку 1. Затем проведем нормализацию этих данных и в результате получим следующий массив, который подается на вход функции predict:

```
c=[[0, 0,0.0,0.0,0.52,0,1,0.0,0.0,0,0,0]]
predictions=model.predict(c)
print(predictions)
```

В результате получаем следующее значение нейрона:

[0.75093013   0.2490699]

Согласно полученным данным вновь поступающий с вероятностью 0.75 закончит обучение без получения диплома.

Результаты

1. Построена нейронная сеть для прогнозирования успешностии окончания университета потенциальны студентом.
2. Выполнено обучение нейронной сети. В качестве учителя использованы данные о 36830 студентах.
3. Определено качество нейронной сети и процент ошибочных предсказаний, которые соответственно равны 0.65 и 0.66.

Список использованных источников и литературы:

1. Открытый курс по машинному обучению [Электронный ресурс]. – Режим доступа: https://www.youtube.com/watch?v=p9Hny3Cs6rk. (дата обращения: 24.06.2021).

2. Нейронные сети. Теория и первый пример (Анализ данных на Python в примерах и задачах. Ч2) [Электронный ресурс]. – Режим доступа: https://www.youtube.com/watch?v=RKKhzFBmEBg. (дата обращения: 20.06.2021).

3. Обучение нейронных сетей в Keras (Анализ данных на Python в примерах и задачах. Ч2) [Электронный ресурс]. – Режим доступа: https://www.youtube.com/watch?v=fyktZrnqOKs (дата обращения: 21.06.2021).

4. Ускорение обучения, начальные веса, стандартизация, подготовка выборки [Электронный ресурс]. – Режим доступа: https://proproprogs.ru/neural_network/uskorenie-obucheniya-nachalnye-vesa-standartizaciya-podgotovka-vyborki (дата обращения: 21.06.2021).

List of references:

1. Open Course in Machine Learning, https://www.youtube.com/watch?v=p9Hny3Cs6rk, accessed June 24, 2021.

2. Neural networks. Theory and the first example (Data analysis in Python in examples and problems. P2), https://www.youtube.com/watch?v=RKKhzFBmEBg, accessed June 20, 2021.

3. Training neural networks in Keras (Data analysis in Python in examples and tasks. P2), https://www.youtube.com/watch?v=fyktZrnqOKs, accessed June 21, 2021.

4. Acceleration of learning, starting weights, standardization, sample preparation https://proproprogs.ru/neural_network/uskorenie-obucheniya-nachalnye-vesa-standartizaciya-podgotovka-vyborki, accessed June 21, 2021.