СЕМЕРИКОВ А. В., ГЛАЗЫРИН М. А. ПОСТРОЕНИЕ ЛОГИСТИЧЕСКОЙ РЕГРЕССИИ С ПОМОЩЬЮ БИБЛИОТЕКИ SCIKIT-LEARN

УДК 378.141.21:330.47, ВАК 1.2.2, ГРНТИ 28.29.51

Построение логистической регрессии с помощью библиотеки scikit-learn

А. В. Семериков[1], М. А. Глазырин[2]

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

В статье рассмотрено построение логистической регрессии для определения вероятности результата успешности завершения обучения потенциальным студентом университета. При построении логистической регрессии использовались библиотеки Python. В качестве исходных данных использовались данные в формате xlsx, состоящая из 10311 строк и 8 колонок. Целевая функция принимает бинарные значения. Качество построенной регрессии оценивалось с помощью статистических критериев. Представлен пример расчета вероятности итогового результата обучения абитуриента.

Ключевые слова: большие данные, нейронная сеть, целевой признак, логистическая регрессия, pandas, scikit-learn

Building a logistic regression using the scikit-learn library

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

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

The article considers the construction of a logistic regression to determine the probability of the result of the successful completion of training by a potential university student. When building a logistic regression, Python libraries were used. The data in the xlsx format, consisting of 10311 rows and 8 columns, was used as the initial data. The objective function accepts binary values. The quality of the constructed regression was assessed using statistical criteria. An example of calculating the probability of the final result of an applicant's education is presented.

Keywords: big data, neural network, target feature, logistic regression, pandas, scikit-learn

Введение

Логистическая регрессия – это алгоритм классификации в машинном обучении для прогнозирования вероятности категориально зависимой переменной. В логистической регрессии зависимые переменные – это двоичные (бинарные) переменные, содержащие 1 (успех) или 0 (неудача). Логистическая

регрессия прогнозирует значение (вероятность) зависимой переменной, как функцию от независимых переменных.

В этом случае задача регрессии может быть сформулирована иначе: вместо предсказания бинарной переменной, мы предсказываем непрерывную переменную со значениями на отрезке [0,1] при любых значениях независимых переменных. Это достигается применением уравнения (логит-преобразование)

$$P = \frac{1}{1+e^{-y}}, \quad (1)$$

где P – вероятность того, что произойдет интересующее событие;

e – основание натуральных логарифмов;

y – стандартное уравнение регрессии.

Логистическую регрессию можно представить в виде однослойной нейронной сети с сигмоидной функцией активации, веса которой есть коэффициенты логистической регрессии, а вес поляризации – константа регрессионного уравнения [1].

Однослойная нейронная сеть может успешно решить лишь задачу линейной сепарации. Поэтому возможности по моделированию нелинейных зависимостей у логистической регрессии отсутствуют.

Логистическая регрессия получила широкое применение для получения новых знаний об объектах, поэтому она присутствует в наборе методов Data Mining алгоритмов.

Экспериментальная часть

При поступлении в ВУЗ абитуриент предоставляет результаты ЕГЭ, которые представляют собой набор оценок: «Русский язык», «Математика», «Физика», «Информатика» и т.д.

После завершения обучения фиксируется результат обучения студента. Так, если студент по каким-либо причинам не закончил обучение с получением диплома в журнале с оценками, с которыми студент был принят в ВУЗ ставится отметка в виде цифры 0. В случае успешного окончания ВУЗа (получение диплома) в этом же журнале ставится цифра 1. Кроме того, в этом наборе данных отмечены пол студента и тип учебного заведения, которое он закончил перед поступлением в ВУЗ.

Для обучения и тестирования логистической регрессии 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 «Диплом» представляет собой целевой признак, который принимает значения 0 или 1. Значение этого бинарного признака, равного 0 или 1, зависит от результата обучения студента. При этом не имеет значения как зафиксирован факт получения или не получения диплома. Если принять 1 за факт получения диплома, то 0 будет означать, что студент закончил образование без получения диплома.

Таблица 1. Исходный набор данных по каждому абитуриенту

Студент	Пол	Образовательное учреждение	Математика	Русский язык	Физика	Диплом
15056	0	1	72	60	58	1
15060	1	3	49	43	42	0
15061	1	2	29	45	37	0
15062	0	3	41	70	30	1
15068	0	3	35	65	32	0
..	…	…	…	…	…	…

В настоящей статье рассматривается построение бинарной логистической регрессии с использованием интерактивной вычислительной среды Jupyter Notebook [2] и примера по нахождению регрессии [3,4].

Теперь представим программный код, позволяющий создать логистическую функцию.

Вначале выполним загрузку библиотек pandas, numpy, sklearn, matplotlib.pyplot, seaborn, предназначение которых будет представлено по ходу их использования:

```
import pandas as pd
import numpy as np
from sklearn import preprocessing
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
import seaborn as sns
```

С помощью библиотеки pandas из таблицы формата Microsoft Excel загружаем данные в объект DataFrame:

```
df=pd.read_excel(io='f:\Исходные данные.xlsx', engine='openpyxl')
```

С помощью функции print определяем количество строк, столбцов и наименования последних:

```
print(df.shape)
print(df.columns)
```

В результате получаем количество строк 10311 и 8 столбцов с наименованиями: «Студент», «Пол», «Образование», «Математика», «Русский язык», «Физика», «Год поступления», «Диплом».

При построении логистической регрессии параметры «Студент», «Год поступления» не используются, поэтому их следует исключить из набора данных. Остальные параметры принимают участие в построении регрессии.

Вместе с тем параметры «Пол» и «Образование» являются категориальными, поэтому их следует представить в бинарном формате:

```
df = df.drop(['Студент'], axis=1)
df = df.drop(['Год поступления'], axis=1)
df['Пол']= np.where((df.Пол == 'M'), 0, 1)
df = df.join(pd.get_dummies(df['Образование'], prefix='Обр'))
```

После замены категориальной переменной «Образование» на четыре бинарных переменных удалям ее из набора данных:

```
df = df.drop(['Образование'], axis=1)
```

В результате получаем следующий набор данных (Рисунок 1).

	Пол	Математика	Русский язык	Физика	Диплом	Обр_высшее профессиональное	Обр_начальное профессиональное	Обр_общеобразовательное	Обр_среднее профессиональное
0	0	72	60	58	1	0	1	0	0
1	0	49	43	42	0	0	0	0	1
2	0	29	45	37	0	0	0	1	0
3	0	41	70	30	1	0	0	0	1
4	0	35	65	32	0	0	0	0	1
...	...	...	...	...	...	...	...	...	...
10306	1	45	80	40	0	0	0	0	1
10307	0	28	60	70	0	0	0	1	0
10308	0	28	40	45	0	1	0	0	0
10309	0	50	61	48	0	0	0	1	0
10310	0	28	50	40	0	0	0	1	0

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

Как видно из данных на Рисунке 1 значения параметров «Математика», «Русский язык», «Физика» на порядок превосходят остальные параметры. Тем самым построенная регрессионная функция будет выдавать результаты, опираясь только на них. Поэтому значения этих параметров необходимо нормализовать, поделив их на 100 (максимально возможный бал по дисциплине).

```
df['Математика'] = df['Математика'] / 100
df['Русский язык'] = df['Русский язык'] / 100
df['Физика'] = df['Физика'] / 100
```

Выведем общую статистику набора данных (Рисунок 2):

```
df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
Пол	10311.0	0.232082	0.422182	0.00	0.00	0.00	0.00	1.0
Математика	10311.0	0.505245	0.170039	0.17	0.38	0.50	0.63	1.0
Русский язык	10311.0	0.589320	0.146973	0.11	0.48	0.60	0.70	1.0
Физика	10311.0	0.481764	0.123615	0.15	0.40	0.45	0.54	1.0
Диплом	10311.0	0.517990	0.499700	0.00	0.00	1.00	1.00	1.0
Обр_высшее профессиональное	10311.0	0.048880	0.215627	0.00	0.00	0.00	0.00	1.0
Обр_начальное профессиональное	10311.0	0.084958	0.278832	0.00	0.00	0.00	0.00	1.0
Обр_общеобразовательное	10311.0	0.629134	0.483060	0.00	0.00	1.00	1.00	1.0
Обр_среднее профессиональное	10311.0	0.237028	0.425280	0.00	0.00	0.00	0.00	1.0

Рисунок 2. Общая статистика набора данных

Для построения регрессионной функции хорошего качества необходимо исключить высокую корреляцию независимых переменных. Поэтому вычислим взаимную корреляцию параметров, используя библиотеку seaborn (Рисунок 3):

```
plt.figure(figsize = (12, 8))
ax = sns.heatmap(df.corr(), annot = True, fmt = ".2f")
i, k = ax.get_ylim()
ax.set_ylim(i + 0.5, k - 0.5)
```

Согласно Рисунка 3 довольно большая корреляция наблюдается между двумя параметрами «Обр_общеобразовательное» и «Обр_среднее профессиональное». Таким образом одно из них может быть удалено.

На качество построения модели регрессии существенное влияние оказывает распределение плотности вероятности значений независимых переменных. Проведенные расчеты с помощью библиотеки seaborn показали, что распределение плотности у параметров «Физика», «Математика», «Русский язык» существенно отличаются от нормального распределения. Продемонстрируем это на примере распределения по параметру «Физика» (Рисунок 4):

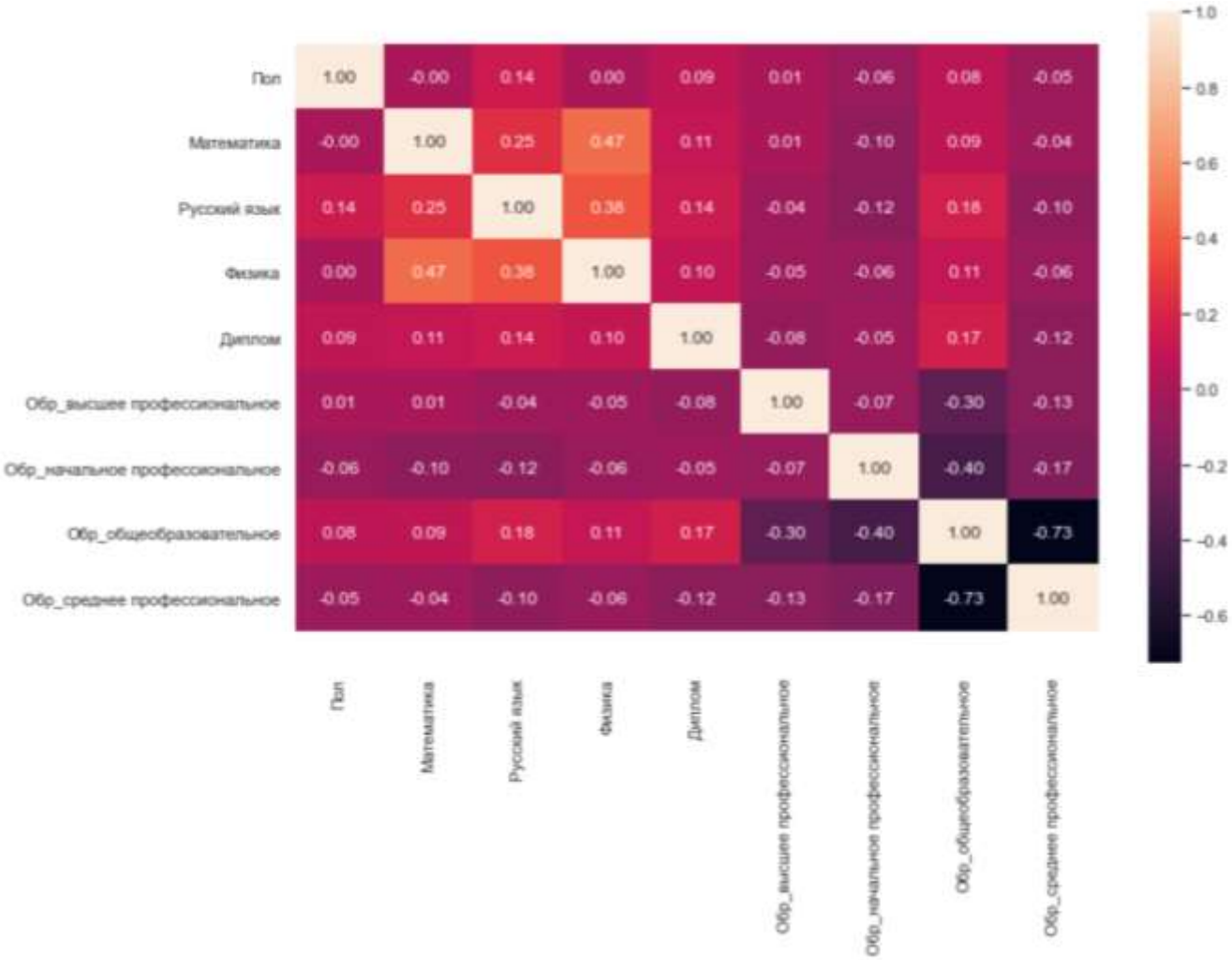

Рисунок 3. Корреляция параметров

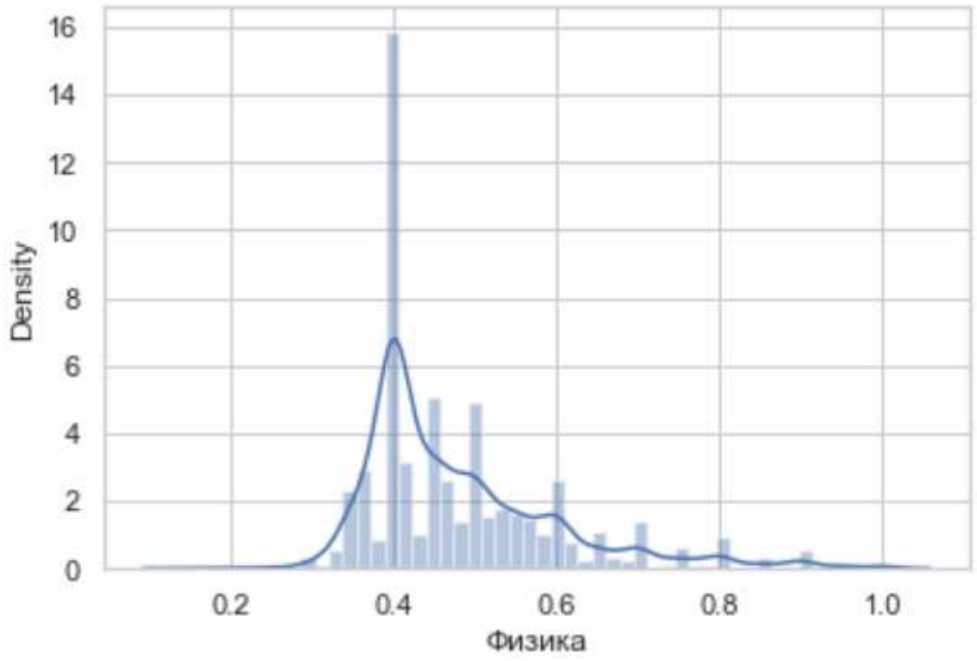

Рисунок 4. Распределение плотности вероятности параметра «Физика»

Этот факт указывает на возможную проблему получения регрессии высокого качества.

Теперь оценим качество исходных данных по соотношению количества студентов, получивших и не получивших диплом:

```
percent = df['Диплом'].value_counts(normalize=True, sort=False) * 100
print('Процент не получивших диплом', percent[0], '%')
print('Процент получивших диплом', percent[1], '%')
```

В результате имеем, что 48,2 % студентов не получили диплом, и соответственно 52,8 % получили диплом. Это указывает на сбалансированность целевого признака.

После анализа данных приступим к процессу построения модели.

Разделим набор данных на зависимые и независимые переменные. В нашем случае зависимая переменная будет «Диплом», а остальные переменные будут независимые:

```
dependent_col = ['Пол', 'Математика', 'Русский язык', 'Физика',
    'Обр_высшее профессиональное', 'Обр_начальное профессиональное',
    'Обр_общеобразовательное', 'Обр_среднее профессиональное']
X = df[dependent_col]
Y = df['Диплом']
```

Для построения логистической регрессии используем библиотеку Sklearn.

Для проведения машинного обучения поделим исходные данные для обучения и проведения тестовых испытаний:

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=300)
logreg = LogisticRegression()
logreg.fit(X_train, Y_train)
```

После обучения получаем предсказания зависимой переменной «Диплом»:

```
Y_predict = logreg.predict(X_test)
print('Предсказание ', Y_predict)
```

и общую точность модели логистической регрессии:

```
print('Точность предсказания: {:.2f}'.format(logreg.score(X_test, Y_test)))
```

Точность предсказания равна 0,61, что указывает на плохое качество модели.

Теперь после создания модели и проведения тестовых испытаний можно построить матрицу совпадений (Таблица 2):

```
from sklearn.metrics import confusion_matrix
confusion_matrix = confusion_matrix(y_test, y_pred)
print(confusion_matrix)
```

Таблица 2. Матрица совпадений

Модель	Факт Получение диплома	Факт Неполучение диплома
Получение диплома	TP=510	FP=461
Неполучение диплома	FN=350	TN=742
TP – количество примеров с верной классификацией получения диплома (так называемые истинно положительные случаи). TN – количество примеров с верной классификацией не получений диплома (истинно отрицательные случаи). FN – количество примеров с получением диплома классифицировано как не получившие диплом (ошибка I рода). FP – количество примеры с неполучением диплома классифицировано как получившие диплом (ошибка II рода).		

Согласно Таблице 2 доля (чувствительность) истинно получивших диплом равна:TP/(TP+FN)=510/(510+350)·100=59,3 %. Доля (специфичность) истинно не получивших диплом равна TN/(TN+FP)=742/(742+461)=61,7 %.

Таким образом, согласно расчетам, модель определяет факт получения диплома у 59,3 % фактически получивших диплом (обнаруживает положительные примеры получения диплома).

При этом точность модели составляет 510/(510+461)*100=53%.

Модель определяет факт не получения у 61,7 % фактически не получивших диплом (обнаруживает отрицательные примеры).

Невысокие значения чувствительности и специфичности указывают на низкое прогностическое качество модели, которое можно повысить, проведя дополнительное машинное обучение с использованием набор исходных данных увеличенного объема.

Для проведения прогноза с использованием функции логистической регрессии необходимо построить функциональную зависимость с независимыми переменными, которые должны соответствовать параметрам тренировочной выборки (Рисунок 5):

```
model_coeff = pd.DataFrame(logreg.coef_, columns=X_train.columns)
model_coeff['intercept'] = logreg.intercept_
model_coeff
```

Пол	0.324214
Математика	0.866445
Русский язык	1.108104
Физика	0.287704
Обр_высшее профессиональное	-0.467637
Обр_начальное профессиональное	0.085923
Обр_общеобразовательное	0.481581
Обр_среднее профессиональное	-0.100499
intercept	-1.507403

Рисунок 5. Параметры тренировочной выборки

Для визуального представления результатов бинарной классификации используем ROC-кривую (Receiver Operator Characteristic) – линию, которая показывает зависимость количества верно классифицированных положительных примеров от количества неверно классифицированных отрицательных примеров.

При построении графика предполагается, что у классификатора имеется некоторый параметр, варьируя который, можно получить то или иное разбиение на два класса. Этот параметр часто называют порогом, или точкой отсечения (cut-off value). В зависимости от него будут получаться различные величины ошибок 1 или 2 рода:

```
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
logit_roc_auc = roc_auc_score(Y_test, logreg.predict(X_test))
fpr, tpr, thresholds = roc_curve(Y_test, logreg.predict_proba(X_test)[:,1])
plt.figure()
plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('100-Специфичность')
plt.ylabel('Чувствительность')
# plt.title('График ROC')
plt.legend(loc="lower right")
plt.savefig('Log_ROC')
plt.show()
print(logit_roc_auc)
```

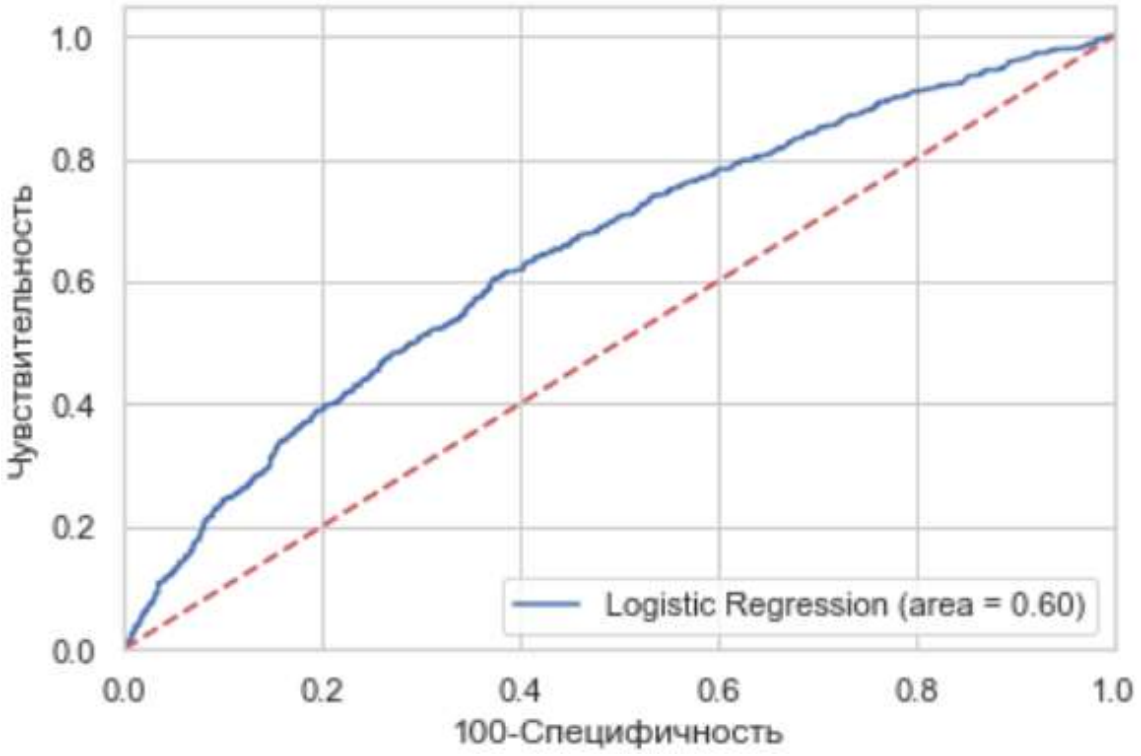

Рисунок 6. График ROC

Идеальная модель обладает стопроцентной чувствительностью и специфичностью. В рассматриваем примере график расположен чуть выше диагональной линии, что говорит о плохой предсказательной способности модели. Для ее повышения необходимо подобрать порог отсечения, значения которого влияют на соотношение чувствительности и специфичности. То есть возникает задача нахождения оптимального значения порога, при котором эти два показателя будут равны.

Порог отсечения нужен для того, чтобы применять модель на практике – относить новые примеры к одному из двух классов. Для определения оптимального порога нужно задать критерий его определения.

Визуальное сравнение кривых ROC не всегда позволяет выявить наиболее эффективную модель. Своеобразным методом сравнения ROC-кривых является оценка площади под кривыми. Теоретически она изменяется от 0 до 1, но, поскольку модель всегда характеризуются кривой, расположенной выше положительной диагонали, то обычно говорят об изменениях от 0,5 («бесполезный» классификатор) до 1,0 («идеальная» модель).

Эта оценка может быть получена непосредственно вычислением площади под многогранником. Численное значение можно вычислить с помощью численного метода трапеций или функции roc_auc_score, который равняется 0,60.

Результаты

В результате исследований выполнено обучение нейронной сети. В качестве учителя использованы данные о 10311 студентах.

Построена логистическая регрессия для прогнозирования успешности окончания университета потенциальными студентами. Определено качество модели, которое равно 0,60.

Построенная модель может быть использована в качестве основы для определения логистической регрессии с другими исходными данными.

Список использованных источников и литературы

1. Нейронная сеть (Neural network) [Электронный ресурс]. – Режим доступа: https://wiki.loginom.ru/articles/neural-network.html (дата обращения: 21.02.2023).

2. Пошаговое построение логистической регрессии в Python [Электронный ресурс]. – Режим доступа: https://medium.com/nuances-of-programming (дата обращения: 01.02.2023).

3. Логистическая регрессия на Python / Хабр (habr.com) [Электронный ресурс]. – Режим доступа: https://habr.com/ru/company/skillfactory/blog/701530/ (дата обращения: 18.02.2023).

List of references

1. Neural network, https://wiki.loginom.ru/articles/neural-network.html, (date of access: 02/21/2023).

2. Building Logistic Regression Step by Step in Python, https://medium.com/nuances-of-programming, (date of access: 02/01/2023).

3. Logistic Regression in Python https://habr.com/ru/company/skillfactory/blog/701530/, (date of access: 02/18/2023).