УДК 004.032

РАЗРАБОТКА И ОБУЧЕНИЕ МОДЕЛИ МАШИННОГО ОБУЧЕНИЯ ДЛЯ TELEGRAM-БОТА

М. А. Фесенко, *студент группы АСб-20Э1;*
Д. К. Давыденко, *студент группы АСб-20Э1*
Федеральное государственное бюджетное образовательное учреждение высшего образования «Сибирский государственный автомобильно-дорожный университет (СибАДИ)», Омск, Россия

Аннотация. *В статье описывается создание нейронной сети с использованием многослойного персептрона для предсказания вероятности выживания пассажиров на борту парохода «Титаник». В качестве массива данных система получает определенный набор характеристик пассажиров и делает на основе их оценку вероятности. Также был реализован Telegram-бот с использованием данной обученной нейронной сети.*

Ключевые слова: *Нейронные сети, машинное обучение, Telegram-бот, Титаник, keras, python-telegram-bot.*

DEVELOPING AND TRAINING A MACHINE LEARNING MODEL FOR A TELEGRAM BOT

M. A. Fesenko, *student of group ASb-20I1;*
D. K. Davydenko, *student of group ASb-20I1*
Federal State Budget Educational Institution of Higher Education «The Siberian State Automobile and Highway University», Omsk, Russia

Abstract. *The article describes the creation of a neural network using a multilayer perceptron to predict the probability of survival of passengers on board the Titanic.As a data array, the system receives a certain set of passenger characteristics and makes a probability estimate based on them. A Telegram bot was also implemented using this trained neural network.*

Keywords: *Neural networks, machine learning, Telegram-bot, Titanic, keras, python-telegram-bot.*

Введение

В наше время технологии нейронных сетей настолько продвинулись, что позволяют создавать высокоточные модели для различных задач, включая прогнозирование. В данной статье мы рассмотрим, как создать Telegram-бота, который будет использовать нейросеть для определения вероятности выживания на Титанике. Титаник был знаменитым лайнером, который затонул в 1912 году. Трагическое событие потрясло весь мир, и с тех пор многие исследователи пытались понять, какие факторы могли повлиять на вероятность выживания на борту.

Основная часть

Основной задачей в начале разработки Telegram-бота было создание модели машинного обучения, которая будет предсказывать вероятность выживания пассажира на основе его характеристик. Здесь стоит отметить, что это является задачей классификации, так как модель должна определить, к какому из двух классов (выжил или не выжил) относится каждый пассажир. В задаче будут использоваться данные о пассажирах для обучения модели и проверки ее точности. Для начала были подготовлены данные для обучения. Набор данных о пассажирах Титаника содержит информацию о 891 пассажире, которые находились на борту во время крушения. Каждый пассажир описывается определенными характеристиками, которые приведены в таблице 1. Наша модель будет использовать только числовые данные, поэтому нам нужно преобразовать категориальные данные, такие как пол и порт посадки, в числовые. Нейронная сеть не может обработать данные, которые содержат отсутствующие значения (NaN), поэтому мы удаляем строки. Для преобразования категориальных данных в числовые мы будем использовать класс LabelEncoder из библиотеки sklearn [1]. Для масштабирования данных, чтобы значения каждого признака были примерно в одном диапазоне, используется класс StandardScaler из библиотеки sklearn. Теперь наши данные готовы для обучения модели. Фрагмент кода представлен на рисунке 1.

Таблица 1 – Характеристики пассажира

Pclass	Класс билета (1 - первый класс, 2 - второй класс, 3 - третий класс)
Sex	Полпассажира (мужской или женский)
Age	Возраст пассажира
SibSp	Количество братьев и сестер или супругов на борту
Parch	Количество родителей или детей на борту
Fare	Стоимость билета
Embarked	Портпосадки (C - Cherbourg, Q - Queenstown, S - Southampton)

```python
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Загрузка датасета
data = pd.read_csv('train.csv')

# Удаление строк с отсутствующими значениями
data = data.dropna()

# Преобразование категориальных признаков в числовые
le = LabelEncoder()
data['Sex'] = le.fit_transform(data['Sex'])
data['Embarked'] = le.fit_transform(data['Embarked'])

# Масштабирование признаков
scaler = StandardScaler()
X = scaler.fit_transform(data.drop(['Survived', 'PassengerId', 'Name', 'Ticket', 'Cabin'], axis=1))
y = data['Survived']

# Разделение данных на обучающую и тестовую выборки
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5)
```

Рисунок 1 – Подготовка данных для обучения

Для обучения модели мы будем применять нейронную сеть с несколькими слоями. Для создания модели используется библиотека keras языка программирования Python. Сначала мы разделим данные на обучающую и тестовую выборки. Создадим модель и добавим несколько слоев: первый слой имеет 64 нейрона и функцию активации relu; второй слой имеет 32 нейрона и также использует функцию активации relu [2]. Выходной слой имеет один нейрон и функцию активации sigmoid, которая позволяет модели выдавать вероятности [3]. Далее мы скомпилируем модель, используя бинарную кросс-энтропию как функцию потерь и оптимизатор Adam. Модель обучается в течение 100 эпох, используя пакеты размером 32 объекта, и использует тестовую выборку для проверки качества модели на каждой эпохе [4].Фрагмент кода приведен на рисунке 2.

```python
# Разделение данных на обучающую и тестовую выборки
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5)
from keras.models import Sequential
from keras.layers import Dense

# Создание модели
model = Sequential()
model.add(Dense(64, input_dim=7, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

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

# Обучение модели
model.fit(X_train, y_train, epochs=100, batch_size=32, validation_data=(X_test, y_test))
```

Рисунок 2 – Обучение модели

После того, как модель прошла обучение, необходимо оценить её качество на тестовой выборке. Для этого используется метод evaluate, который позволяет вычислить потери и метрики модели

на тестовых данных. Фрагмент программного кода приведен на рисунке 3. После оценки качества обученной модели была написана функция для теста данной модели. Пример работы программного кода показан на рисунке 4.

```
#оценка качества модели
loss, accuracy = model.evaluate(X_test, y_test)
print(f'Test loss: {loss:.4f}')
print(f'Test accuracy: {accuracy:.4f}')
```

```
3/3 [==============================] - 0s 7ms/step - loss: 0.5363 - accuracy: 0.7283
Test loss: 0.5363
Test accuracy: 0.7283
```

Рисунок 3 – Оценка качества обученной модели

```
def predict_survival(model, pclass, sex, age, sibsp, parch, fare, embarked):
    data = [[pclass, sex, age, sibsp, parch, fare, embarked]]
    data = scaler.transform(data)
    prediction = model.predict(data)
    return prediction[0][0]

# Пример использования
result = predict_survival(model, 1, 0, 21, 0, 0, 30, 3)
print('Вероятность выживания: %.2f%%' % (result * 100))
result = predict_survival(model, 0, 1, 30, 2, 0, 30, 1)
print('Вероятность выживания: %.2f%%' % (result * 100))
```

```
1/1 [==============================] - 0s 29ms/step
Вероятность выживания: 99.56%
1/1 [==============================] - 0s 29ms/step
Вероятность выживания: 70.49%
```

Рисунок 4 – Тест работы обученной модели

После того, как модель была обучена и протестирована, был реализован Telegram-бот, который будет принимать данные о пассажирах Титаника и, с использованием уже обученной модели, определять вероятность их выживания. Для создания Telegram бота была использована библиотека python-telegram-bot. Пройдя тесты, бот был откорректирован [5]. Пример работы Telegram-бота показан на рисунке 5.

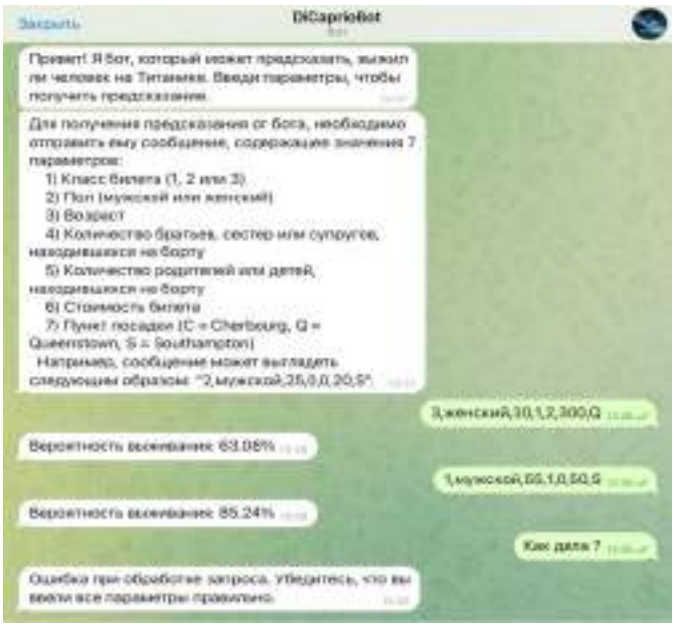

Рисунок 5 – Пример работы бота

Данная реализация модели оказалась очень полезной для изучения классификации, кластеризации, и самих нейронных сетей. Модель не нуждается в доработке, так как тренировочных данных достаточно для 89% точности результата. В будущем на основе бота будет реализовано мобильное приложение с использованием визуальных эффектов.

Библиографический список

1. Барский А. Б. Нейронные сети: распознавание, управление, принятие решений / А. Б. Баркий. – М.: Финансы и статистика, 2004. – 176 с. (Прикладные информационные технологии).
2. Матвеев, М. Г. Модели и методы искусственного интеллекта. Применение в экономике: учебное пособие / М. Г. Матвеев, А. С. Свиридов, Н. А. Алейников.– М.: Финансы и статистика; ИНФРА-М, 2008.– 488 с.
3. Сверточная нейронная сеть на Python и Keras. – URL: https://линуксблог.рф/svertochnaya-nejronnaya-set-na-python-i-keres/ (дата обращения: 16.03.2023).
4. Нейронные сети, генетические алгоритмы и нечеткие системы / пер. с польск. И. Д. Рудинского; Д. Рутковская, М. Пилиньский, Л. Рутковский. – М.: Горячая линия – Телеком, 2006. – 452 с.
5. Ясницкий, Л. Н. Введение в искусственный интеллект: учебное пособие для студ. высш. учеб. заведений / Л. Н. Ясницкий. – М.: Издательский центр «Академия», 2005.–176 с.

Научный руководитель – Филимонова О. А., старший преподаватель кафедры «Цифровые технологии» СибАДИ.