Министерство науки и высшего образования Российской Федерации
Калужский филиал федерального государственного автономного образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИУК «Информатика и управление»

КАФЕДРА ИУК5 «Системы обработки информации»

ЛАБОРАТОРНАЯ РАБОТА №1.2

«РЕКУРРЕНТНЫЕ НЕЙРОННЫЕ СЕТИ.УЛУЧШЕННЫЕ МЕТОДЫ РЕКУРРЕНТНЫХ НЕЙРОННЫХ СЕТЕЙ»

по дисциплине: «Методы глубокого обучения»

Выполнил: студент группы ИУК5-21М _____ А. Э. Дармограй
(Подпись) (И.О. Фамилия)

Проверил: _____ Ю. С. Белов
(Подпись) (И.О. Фамилия)

Дата сдачи (защиты):

Результаты сдачи (защиты):

- Балльная оценка:
- Оценка:

Калуга, 2025

Цель: получение практических навыков построения глубоких рекуррентных нейронных сетей.

Задачи:
• разработка модели глубокой рекуррентной нейронной сети.

Задание:
1) Реализовать модель глубокой рекуррентной нейронной сети, используя набор данных, согласно варианту:

Вариант 2: Динамика продаж в супермаркете. Ссылка для скачивания: https://drive.google.com/file/d/1T3C0WsA9p6MPyuKG7naFrSPxtQroD6dj

/view?usp=drive_link

2) Вывести График точности на этапах обучения.

3) Вывести График изменения по набору данных.

4) Вывести График потерь на этапах обучения и проверки простой полносвязной сети в задаче.

Выполнение работы

Код доступен в репозитории GitHub:

https://github.com/Dariarty/Deep_Learning_Methods

Данную лабораторную работы выполнял на Python версии 3.12.7 и Keras версии 3.9.0

Код лабораторной работы №2:

https://github.com/Dariarty/Deep_Learning_Methods/blob/main/src/LAB2/bigmart_rnn.ipynb

Реализация модели RNN

В данной работе необходимо реализовать модель глубокой рекуррентной нейронной сети, используя набор данных продаж в супермаркете по наименованию.

Подключаем модули и задаем название файла с данными

```
#В данной работе использую Python 3.12.7 и Keras 3.9.0

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from keras.layers import SimpleRNN, Dense, Flatten
from keras.optimizers import RMSprop
from keras import Input
from sklearn.metrics import mean_absolute_error

fname = 'data/bigmart.csv'
```

Выводим верхние строчки таблицы и распределение данных

Данные здесь содержат записи о продаже в супермаркете по наименованию продукта.

1. Item_Identifier: ID продукта
2. Item_Weight: Вес продукта
3. Item_Fat_Content: Содержание жиров (Low Far - низкие жиры, Regular - обычный)

4. Item_Visibility: Процент от общей площади магазина, выделенный под данный продукт
5. Item_Type: Категория, к которой принадлежит продукт
6. Item_MRP: Макисмальная розничная цена
7. Outlet_Identifier: ID магазина
8. Outlet_Establishment_Year: Год открытия магазина
9. Outlet_Size: Размер магазина по его площади
10. Outlet_Location_Type: Тип города, в котором находится магазин
11. Outlet_Type: Тип магазина (Овощной, супермаркет)
12. Item_Outlet_Sales: Продажи данного продукта в данном магазине

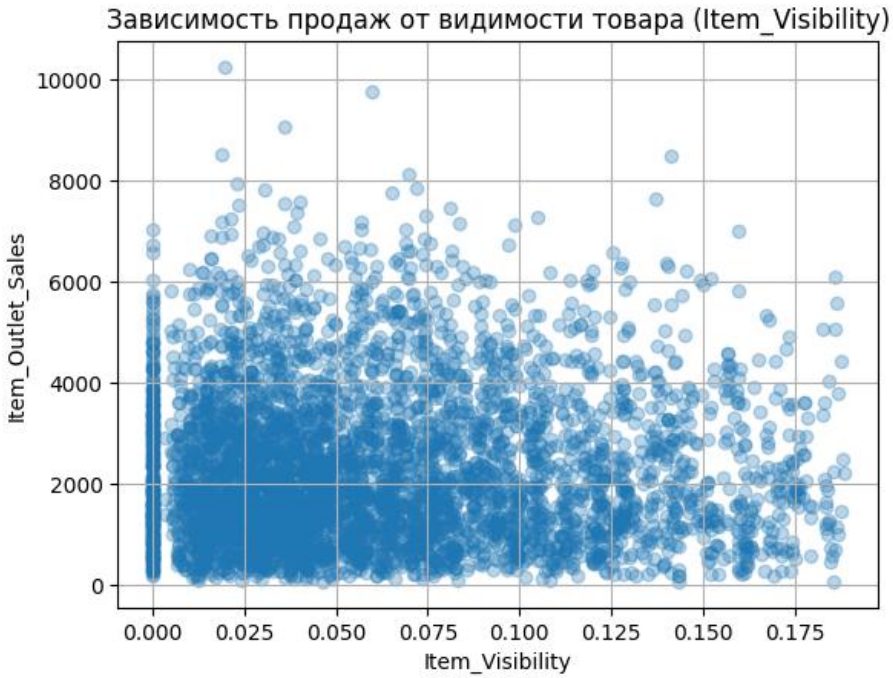
```
#Загрузка и очистка данных
df = pd.read_csv('data/bigmart.csv', sep=',', decimal='.', encoding='utf-8')
df = df.dropna()
df.head()
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year
0	FDA15	9.300	Low Fat	0.016047	Dairy	249.8092	OUT049	1999
1	DRC01	5.920	Regular	0.019278	Soft Drinks	48.2692	OUT018	2009
2	FDN15	17.500	Low Fat	0.016760	Meat	141.6180	OUT049	1999
4	NCD19	8.930	Low Fat	0.000000	Household	53.8614	OUT013	1987
5	FDP36	10.395	Regular	0.000000	Baking Goods	51.4008	OUT018	2009

Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales
Medium	Tier 1	Supermarket Type1	3735.1380
Medium	Tier 3	Supermarket Type2	443.4228
Medium	Tier 1	Supermarket Type1	2097.2700
High	Tier 3	Supermarket Type1	994.7052
Medium	Tier 3	Supermarket Type2	556.6088

Распределение данных

```
#График изменения продаж в зависимости от видимости товара
plt.figure()
plt.scatter(df['Item_Visibility'], df['Item_Outlet_Sales'], alpha=0.3)
plt.title('Зависимость продаж от видимости товара (Item_Visibility)')
plt.xlabel('Item_Visibility')
plt.ylabel('Item_Outlet_Sales')
plt.grid(True)
plt.show()
```



Создадим RNN-модель и выведем график средней абсолютной ошибки для обучения и проверки

```
#Кодирование категориальных признаков
categorical_cols = ['Item_Fat_Content', 'Item_Type', 'Outlet_Identifier',
                    'Outlet_Size', 'Outlet_Location_Type', 'Outlet_Type']

label_encoders = {}
for col in categorical_cols:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le

#Подготовка признаков
features = ['Item_Weight', 'Item_Visibility', 'Item_MRP',
'Outlet_Establishment_Year'] + categorical_cols
target = 'Item_Outlet_Sales'

x = df[features].values
```

```
y = df[target].values

#Первые 50 значений набора
plt.figure()
plt.plot(range(50), y[:50], label='Item_Outlet_Sales')
plt.title('График изменения продаж (первые 50 значений)')
plt.xlabel('Индекс образца')
plt.ylabel('Item_Outlet_Sales')
plt.grid(True)
plt.legend()
plt.show()

#Масштабирование
scaler = StandardScaler()
X_scaled = scaler.fit_transform(x)
X_rnn = X_scaled.reshape((X_scaled.shape[0], 1, X_scaled.shape[1]))

#Разделение выборки
X_train, X_test, y_train, y_test = train_test_split(X_rnn, y, test_size=0.2,
random_state=42)

#Модель глубокой RNN
model = Sequential()
model.add(Input(shape=(1, X_rnn.shape[2])))
model.add(SimpleRNN(64, return_sequences=True))
model.add(SimpleRNN(32))
model.add(Dense(1))
model.compile(optimizer='rmsprop', loss='mae', metrics=['mae'])

#Обучение
history = model.fit(X_train, y_train,
                    epochs=20,
                    batch_size=128,
                    validation_split=0.2)

#График MAE
plt.figure()
plt.plot(history.history['mae'], 'g', label='MAE на обучении')
plt.plot(history.history['val_mae'], 'b', label='MAE на проверке')
plt.title('График средней абсолютной ошибки (точности)')
plt.xlabel('Эпоха')
plt.ylabel('MAE')
plt.legend()
plt.grid(True)
plt.show()
```


График изменения продаж (первые 50 значений)
Item_Outlet_Sales
Item_Outlet_Sales
Индекс образца
0
1000
2000
3000
4000
5000
6000
7000
0
10
20
30
40
50

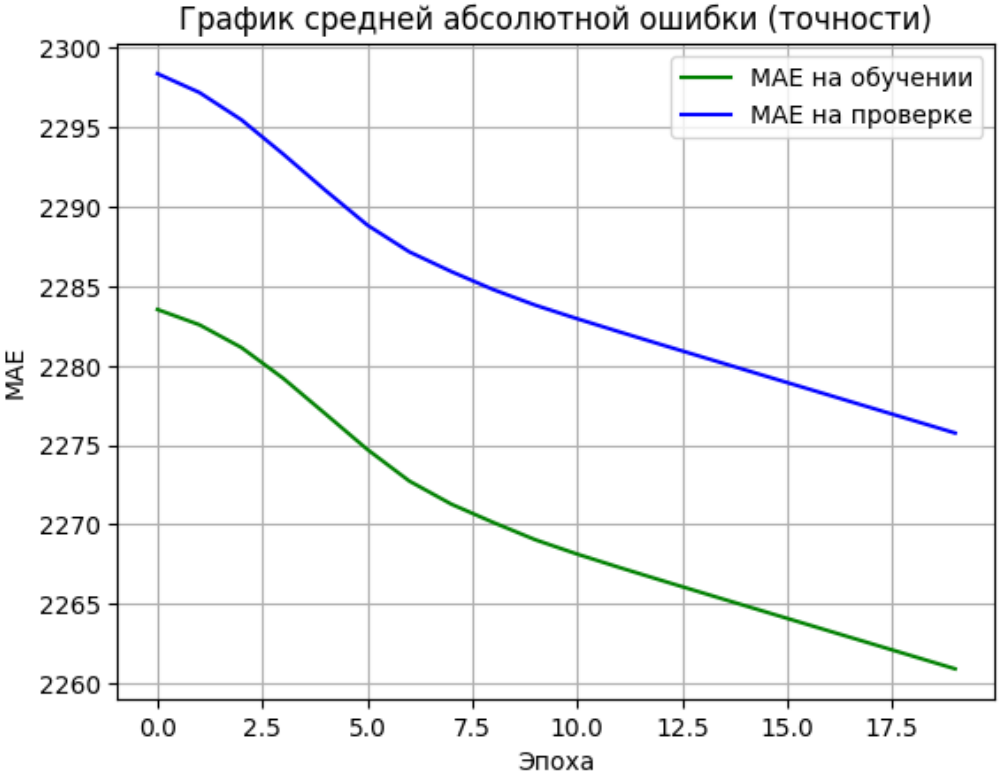


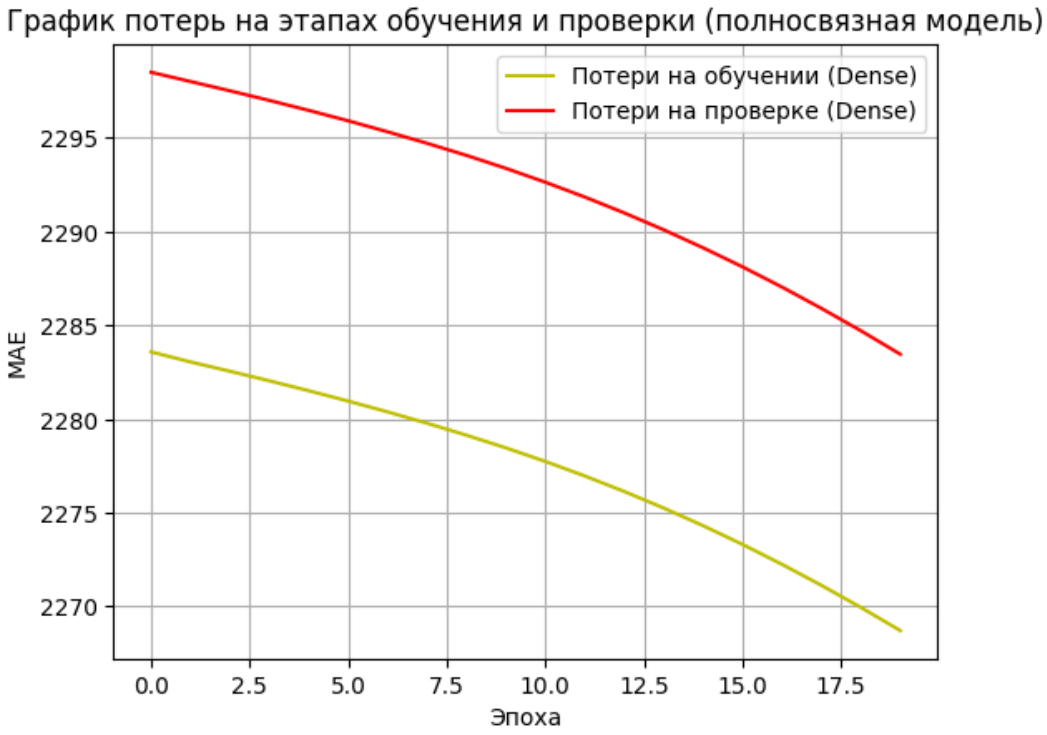
График средней абсолютной ошибки (точности)
MAE на обучении
MAE на проверке
MAE
Эпоха
2260
2265
2270
2275
2280
2285
2290
2295
2300
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

В сравнении с полносвязной моделью в данной задаче

```
#Для сравнения обучим простую модель (без RNN)
model_dense = Sequential()
model_dense.add(Input(shape=(1, X_rnn.shape[2])))
model_dense.add(Flatten())
model_dense.add(Dense(32, activation='relu'))
model_dense.add(Dense(1))
model_dense.compile(optimizer=RMSprop(), loss='mae')

history_dense = model_dense.fit(X_train, y_train,
                                 epochs=20,
                                 batch_size=128,
                                 validation_split=0.2)

plt.figure()
plt.plot(history_dense.history['loss'], 'y', label='Потери на обучении (Dense)')
plt.plot(history_dense.history['val_loss'], 'r', label='Потери на проверке
(Dense)')
plt.title('График потерь на этапах обучения и проверки (полносвязная модель)')
plt.xlabel('Эпоха')
plt.ylabel('MAE')
plt.legend()
plt.grid(True)
plt.show()
```



Вывод: в ходе выполнения лабораторной работы была реализована рекуррентная нейронная сеть, а также были построены графики потерь на обучающем и тестовом наборах данных по эпохам