|  |  |
| --- | --- |
|  | **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

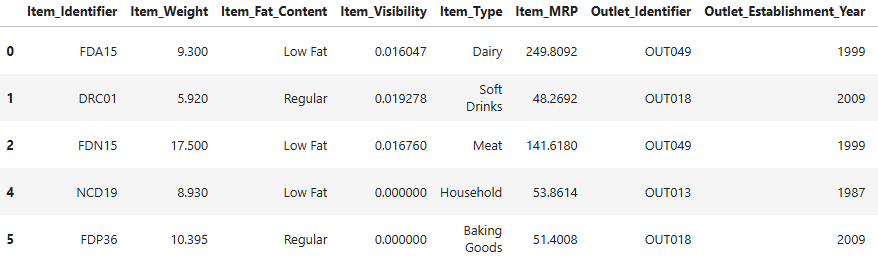
|  |
| --- |
| *#В данной работе использую Python 3.12.7 и Keras 3.9.0*  **import** pandas **as** pd  **import** numpy **as** np  **import** matplotlib.pyplot **as** plt  **from** sklearn.preprocessing **import** LabelEncoder, StandardScaler  **from** sklearn.model\_selection **import** train\_test\_split  **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' |

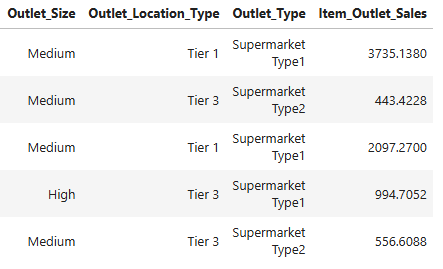
Выводим верхние строчки таблицы и распределение данных

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

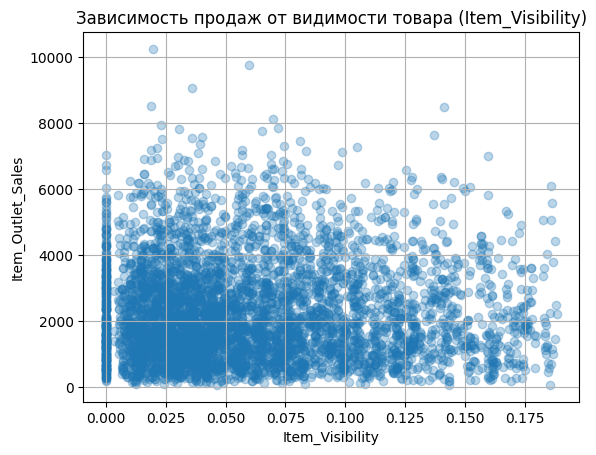
|  |
| --- |
| *#Загрузка и очистка данных*  df **=** pd**.**read\_csv('data/bigmart.csv', sep**=**',', decimal**=**'.', encoding**=**'utf-8')  df **=** df**.**dropna()  df**.**head() |





Распределение данных

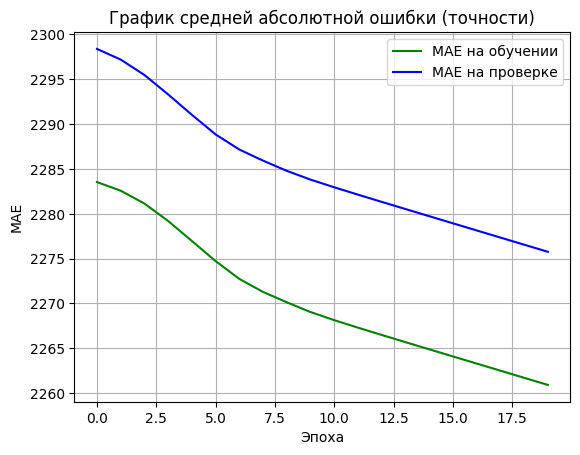
|  |
| --- |
| *#График изменения продаж в зависимости от видимости товара*  plt**.**figure()  plt**.**scatter(df['Item\_Visibility'], df['Item\_Outlet\_Sales'], alpha**=**0.3)  plt**.**title('Зависимость продаж от видимости товара (Item\_Visibility)')  plt**.**xlabel('Item\_Visibility')  plt**.**ylabel('Item\_Outlet\_Sales')  plt**.**grid(**True**)  plt**.**show() |



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

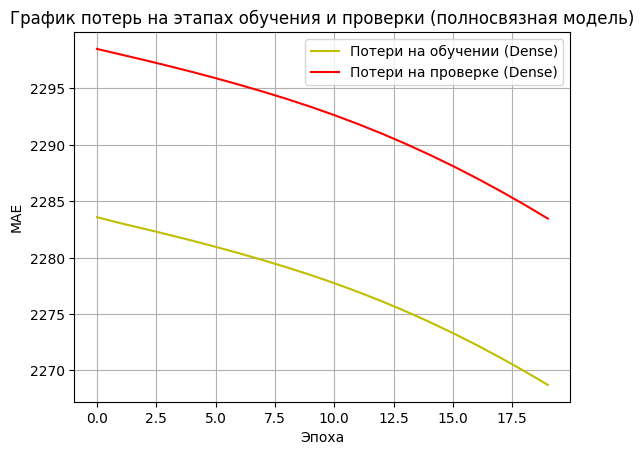
|  |
| --- |
| *#Кодирование категориальных признаков*  categorical\_cols **=** ['Item\_Fat\_Content', 'Item\_Type', 'Outlet\_Identifier',  'Outlet\_Size', 'Outlet\_Location\_Type', 'Outlet\_Type']  label\_encoders **=** {}  **for** col **in** categorical\_cols:  le **=** LabelEncoder()  df[col] **=** 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() |





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