Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Московский государственный технический университет имени
Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»
КАФЕДРА «Компьютерные системы и сети (ИУ-6)»

ОТЧЕТ О ВЫПОЛНЕНИИ ДОМАШНЕЙ РАБОТЫ

по дисциплине «Математические методы анализа данных и принятия решений»

Студент:	Козлов Владимир Михайлович
Группа:	ИУ6-13М
Тип задания:	домашняя работа
Тема:	Байесовский классификатор

Студент _____ Козлов В.М.
подпись, дата Фамилия, И.О.

Преподаватель _____ _____
подпись, дата Фамилия, И.О.

Москва, 2024

Содержание

1 Описание лабораторной работы 3
1.1 цель 3
1.2 Задание 3
2 Выполнение лабораторной работы 4
2.1 Код 4
2.2 Визуализация данных 9
2.3 Визуализация результатов 11
3 Контрольные вопросы 12
4 Вывод 13

1 Описание лабораторной работы

1.1 цель

Изучение процесса построения и обучения простой нейронной сети в PyTorch на примере задачи классификации ботов. Исследование влияния различных функций активации (Sigmoid, Tanh, ReLU) и оптимизаторов (SGD, SGD с моментом, RMSprop, Adam) на процесс обучения и качество модели. Определение оптимальной комбинации гиперпараметров для решения поставленной задачи классификации.

1.2 Задание

выполнить обучение и предсказание для задачи классификации/регрессии на примере:

1. загрузка исходных данных (загрузить датасет, выполнить визуализацию исходных данных);
2. подготовить данные для загрузки в модель (заполнить пропуски, выполнить замену текстовых
3. категориальных признаков, масштабировать значения при необходимости);
4. построить модель MLP (from sklearn.neural_network import MLPClassifier) и выполнить настройку параметров (количество слоев, число нейронов, функции активации);
    - * дополнительные баллы при реализации MLP вручную без импорта готовой модели из библиотеки;
5. обучить модель и выполнить оценку ее адекватности при помощи метрик;
6. ыполнить визуализацию результатов;
7. составить отчет о проделанной работе в соответствии с требованиями кафедры.

2 Выполнение лабораторной работы

2.1 Код

Для выполнения лабораторной был написан следующий код.

Листинг 1: Код лабораторной работы

```
#!/bin/python3
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import classification_report,
    confusion_matrix, accuracy_score

hidden_layer_sizes=(64, 32)
max_iter=300
learning_rate_init=0.001
random_state=42

dfs = []
folder_path = "datasets/"

for filename in os.listdir(folder_path):
    if filename.endswith('.csv'):
        file_path = os.path.join(folder_path, filename)
        dataset = pd.read_csv(file_path)
        dfs.append(dataset)
        print(f"Загружен файл: {filename} ({len(dataset)} строк)")

df = pd.concat(dfs, ignore_index=True)

# Гистограммы числовых признаков
df.hist(figsize=(15, 12), bins=30)
plt.suptitle("Распределение признаков датасета")
plt.savefig('lab1/plots/1.png')

# Матрица корреляций
plt.figure(figsize=(12, 10))
sns.heatmap(df.corr(), annot=False, cmap='coolwarm',
    linewidths=0.5)
plt.title("Матрица корреляций")
```

```
38 plt.savefig('lab1/plots/2.png')
39
40 # Подготовка данных
41 print("Пропуски в данных:")
42 print(df.isnull().sum())
43
44 if df.isnull().sum().any():
45     df.fillna(df.mean(), inplace=True)
46
47 # Разделение на признаки и целевую переменную
48 X = df.drop(columns=['label'])
49 y = df['label']
50
51 # Если метки строковые, кодируем их
52 if y.dtype == 'object':
53     le = LabelEncoder()
54     y = le.fit_transform(y)
55
56 X_train, X_test, y_train, y_test = train_test_split(X, y,
       test_size=0.2, random_state=42, stratify=y)
57
58 scaler = StandardScaler()
59 X_train_scaled = scaler.fit_transform(X_train)
60 X_test_scaled = scaler.transform(X_test)
61
62 # У MLP Выходной слой всегда использует softmax для многоклассовой классификации, а
       функция потерь всегда cross-entropy (log-loss)
63
64 mlp = MLPClassifier(
65     hidden_layer_sizes=hidden_layer_sizes,
66     activation='relu',
67     solver='adam',
68     alpha=0.0001,
69     batch_size='auto',
70     learning_rate='constant',
71     learning_rate_init=learning_rate_init,
72     max_iter=max_iter,
73     random_state=random_state,
74     verbose=True
75 ).fit(X_train_scaled, y_train)
76
77 y_pred = mlp.predict(X_test_scaled)
78
79 print("\Точностьn модели:", accuracy_score(y_test, y_pred))
80
81 # Матрица ошибок
```

```
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
    xticklabels=np.unique(y), yticklabels=np.unique(y))
plt.title('Матрица ошибок')
plt.xlabel('Предсказанный класс')
plt.ylabel('Истинный класс')
plt.savefig('lab1/plots/3.png')

# Визуализация процесса обучения
plt.figure(figsize=(8, 5))
plt.plot(mlp.loss_curve_, label='Ошибка на обучающей выборке')
plt.title('Кривая обучения')
plt.xlabel('Эпохи')
plt.ylabel('Функция потерь')
plt.legend()
plt.grid()
plt.savefig('lab1/plots/4.png')
```

Вывод кода представлен ниже.

Листинг 2: Вывод кода лабораторной работы

```
Загружен
 файл: humans_total.csv (3631 строк)Загружен
 файл: koplayer_total.csv (56 строк)Загружен
 файл: ldplayer_total.csv (1404 строк)Загружен
 файл: memu_total.csv (3353 строк)Загружен
 файл: nox_total.csv (3169 строк)Пропуски
 в данных:
touches_size_exists     0
touches_size_min        0
touches_size_mean       0
touches_size_max        0
touches_size_std        0
gyroscope_exists        0
gyroscope_x_min         0
gyroscope_x_mean        0
gyroscope_x_max         0
gyroscope_x_std         0
gyroscope_y_min         0
gyroscope_y_mean        0
gyroscope_y_max         0
gyroscope_y_std         0
gyroscope_z_min         0
gyroscope_z_mean        0
gyroscope_z_max         0
```

```
gyroscope_z_std         0
accelerometer_exists    0
accelerometer_x_min     0
accelerometer_x_mean    0
accelerometer_x_max     0
accelerometer_x_std     0
accelerometer_y_min     0
accelerometer_y_mean    0
accelerometer_y_max     0
accelerometer_y_std     0
accelerometer_z_min     0
accelerometer_z_mean    0
accelerometer_z_max     0
accelerometer_z_std     0
light_exists            0
light_min               0
light_mean              0
light_max               0
light_std               0
label                   0
dtype: int64
Iteration 1, loss = 0.38702657
Iteration 2, loss = 0.11047386
Iteration 3, loss = 0.05734688
Iteration 4, loss = 0.04226725
Iteration 5, loss = 0.03312316
Iteration 6, loss = 0.02664729
Iteration 7, loss = 0.02181111
Iteration 8, loss = 0.01854171
Iteration 9, loss = 0.01546150
Iteration 10, loss = 0.01346133
Iteration 11, loss = 0.01160058
Iteration 12, loss = 0.01000531
Iteration 13, loss = 0.00906052
Iteration 14, loss = 0.00811599
Iteration 15, loss = 0.00727452
Iteration 16, loss = 0.00683171
Iteration 17, loss = 0.00623412
Iteration 18, loss = 0.00594642
Iteration 19, loss = 0.00559660
Iteration 20, loss = 0.00567037
Iteration 21, loss = 0.00505971
Iteration 22, loss = 0.00478833
Iteration 23, loss = 0.00559461
Iteration 24, loss = 0.00440574
Iteration 25, loss = 0.00443362
```

```
Iteration 26, loss = 0.00436302
Iteration 27, loss = 0.00416115
Iteration 28, loss = 0.00395033
Iteration 29, loss = 0.00389309
Iteration 30, loss = 0.00373203
Iteration 31, loss = 0.00373515
Iteration 32, loss = 0.00396885
Iteration 33, loss = 0.00370906
Iteration 34, loss = 0.00348140
Iteration 35, loss = 0.00337635
Iteration 36, loss = 0.00327947
Iteration 37, loss = 0.00328636
Iteration 38, loss = 0.00354753
Iteration 39, loss = 0.00323677
Iteration 40, loss = 0.00307666
Iteration 41, loss = 0.00338017
Iteration 42, loss = 0.00317773
Iteration 43, loss = 0.00285553
Iteration 44, loss = 0.00297616
Iteration 45, loss = 0.00278060
Iteration 46, loss = 0.00278897
Iteration 47, loss = 0.00269697
Iteration 48, loss = 0.00244217
Iteration 49, loss = 0.00335454
Iteration 50, loss = 0.00265874
Iteration 51, loss = 0.00248445
Iteration 52, loss = 0.00234776
Iteration 53, loss = 0.00229907
Iteration 54, loss = 0.00229728
Iteration 55, loss = 0.00221573
Iteration 56, loss = 0.00327662
Iteration 57, loss = 0.00213137
Iteration 58, loss = 0.00227376
Iteration 59, loss = 0.00217456
Training loss did not improve more than tol=0.000100 for 10 consecutive
    ↪ epochs. Stopping.Точность

 модели: 0.9969866551872578
```

2.2 Визуализация данных

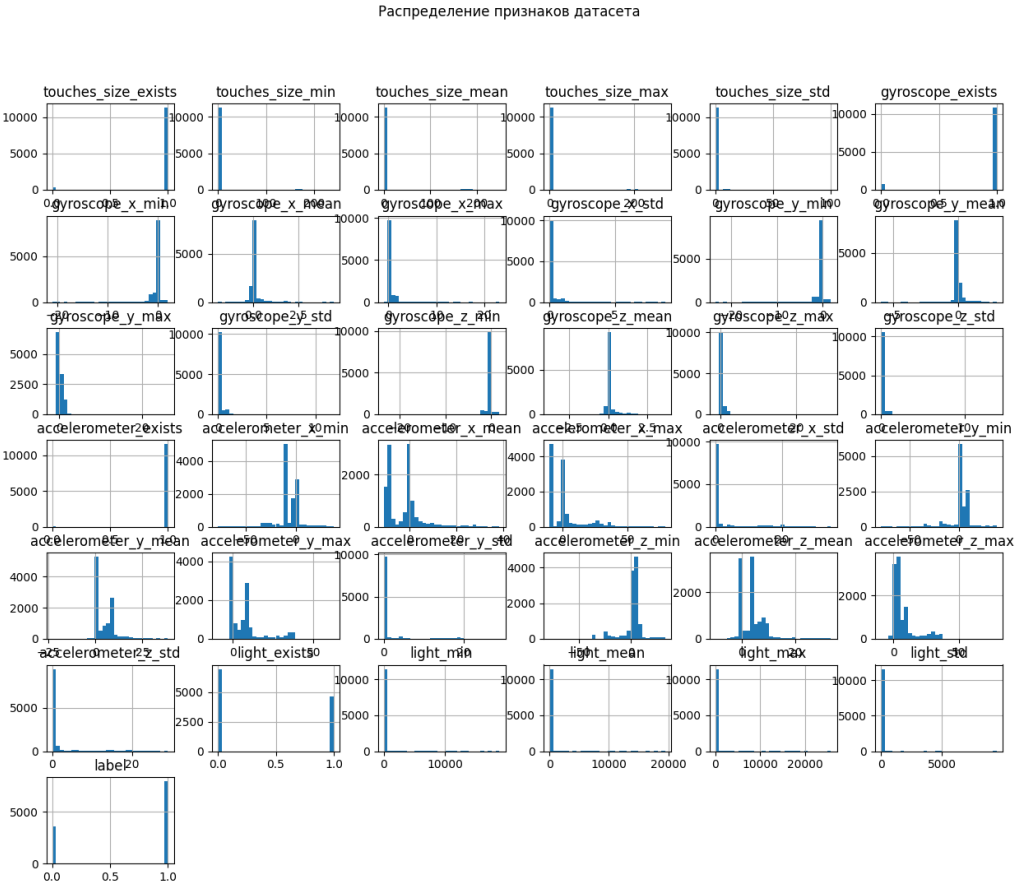

Рис. 1: Гистограммы признаков

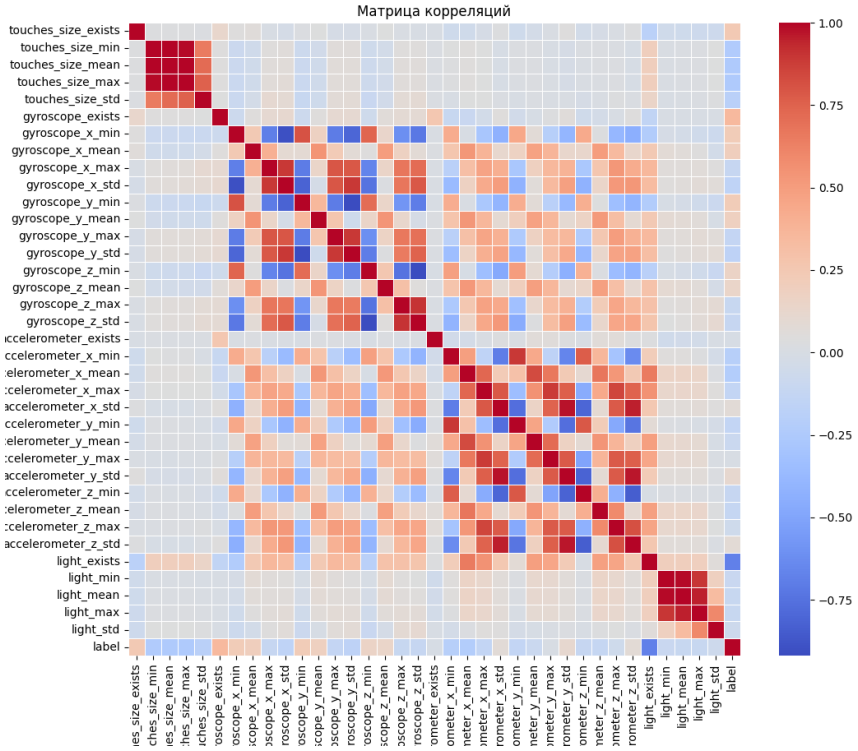

Рис. 2: Матрица корреляций

2.3 Визуализация результатов

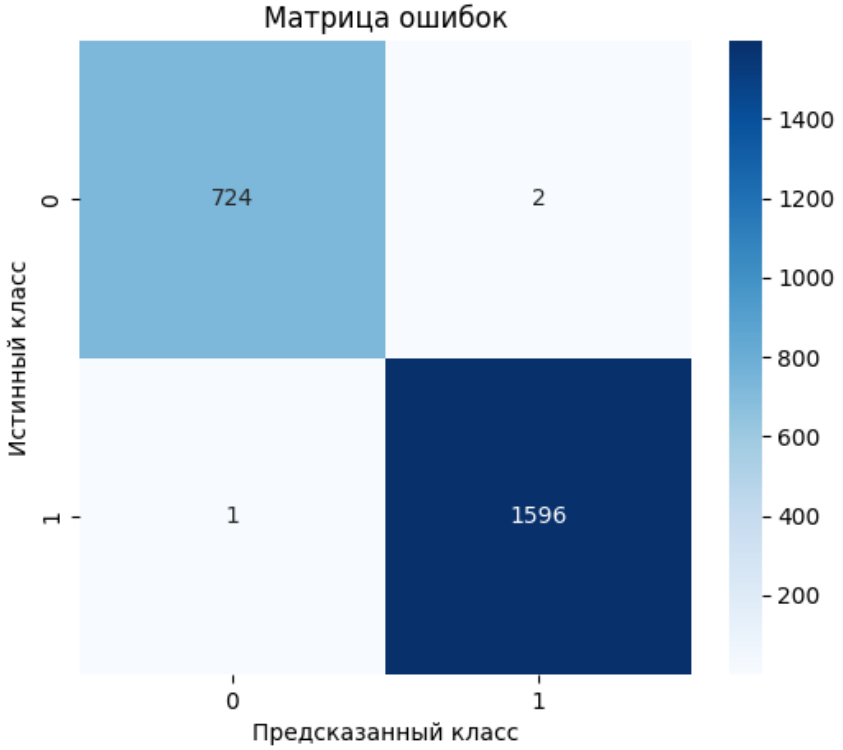

Рис. 3: Матрица ошибок

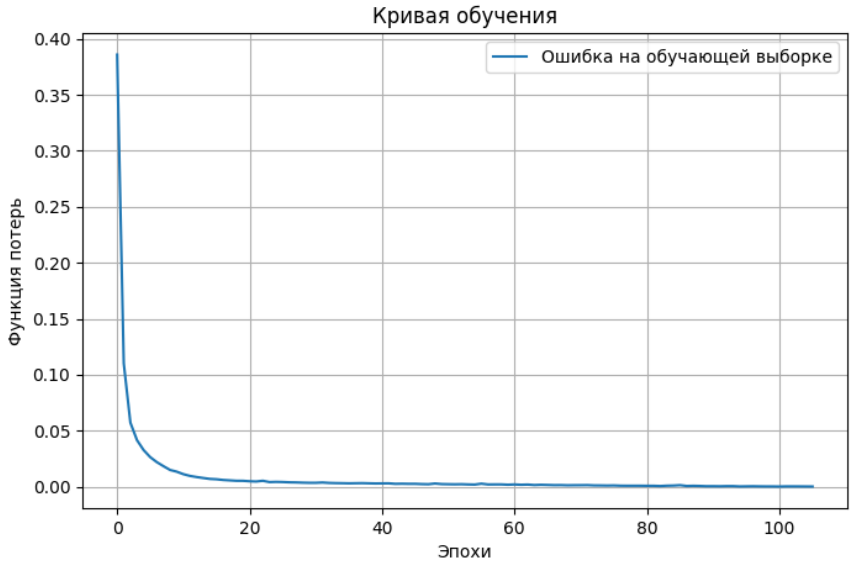

Рис. 4: Кривая обучения

3 Контрольные вопросы

1. Что такое искусственный нейрон и как он работает?

   Ответ: Искусственный нейрон — математическая модель: $y = f(\sum(w_i \cdot x_i) + b)$. Принимает входные сигналы x_i, умножает на веса w_i, суммирует, добавляет смещение b, применяет нелинейную функцию активации f.

2. Чем отличается линейная модель от многослойного персептрона?

   Ответ: Линейная модель — один слой, только линейные зависимости: $y = Wx+b$. MLP — несколько слоёв с нелинейными активациями: $y = f_n(\ldots f_2(f_1(W_1x+b_1)+b_2)\ldots)$, что позволяет моделировать сложные нелинейные зависимости.

3. Для чего нужны функции активации?

   Ответ:

   - Вводят нелинейность (без них MLP эквивалентен линейной модели);
   - Ограничивают выход;
   - Управляют потоком градиентов;
   - Создают разреженные представления (например, ReLU).

4. Как выбрать количество эпох и скрытых слоёв?

   Ответ: Эпохи: по кривой обучения или ранней остановке. Слои: начинать с 1–2, увеличивать при недообучении. Для простых задач — 0–1 слой, для сложных — 3+.

5. Какие метрики применяются для оценки качества классификации?

   Ответ: Accuracy, Precision, Recall, F1-score, ROC-AUC. Для многоклассовой — macro/micro/weighted average. Выбор зависит от задачи (например, для медицины важен Recall).

4 Вывод

Построен и обучен MLP для классификации. Данные успешно подготовлены. Модель с 2 скрытыми слоями (64→32 нейрона, ReLU) обучена Adam за 300 эпох с L2-регуляризацией.

Результаты: достигнута точность 0.99%

Выводы: MLP эффективен для нелинейных задач. Ключевые факторы успеха — предобработка данных и правильный выбор гиперпараметров.