МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«БРЕСТСКИЙГОСУДАРСТВЕННЫЙТЕХНИЧЕСКИЙУНИВЕРСИТЕТ» ФАКУЛЬТЕТ ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ

Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе №3

Специальность ИИ-23

Выполнил А.А. Шурина, студент группы ИИ-23

Проверил К.В. Андренко,

ст. преп. кафедры ИИТ,

« » 2025 г.

Брест 2025

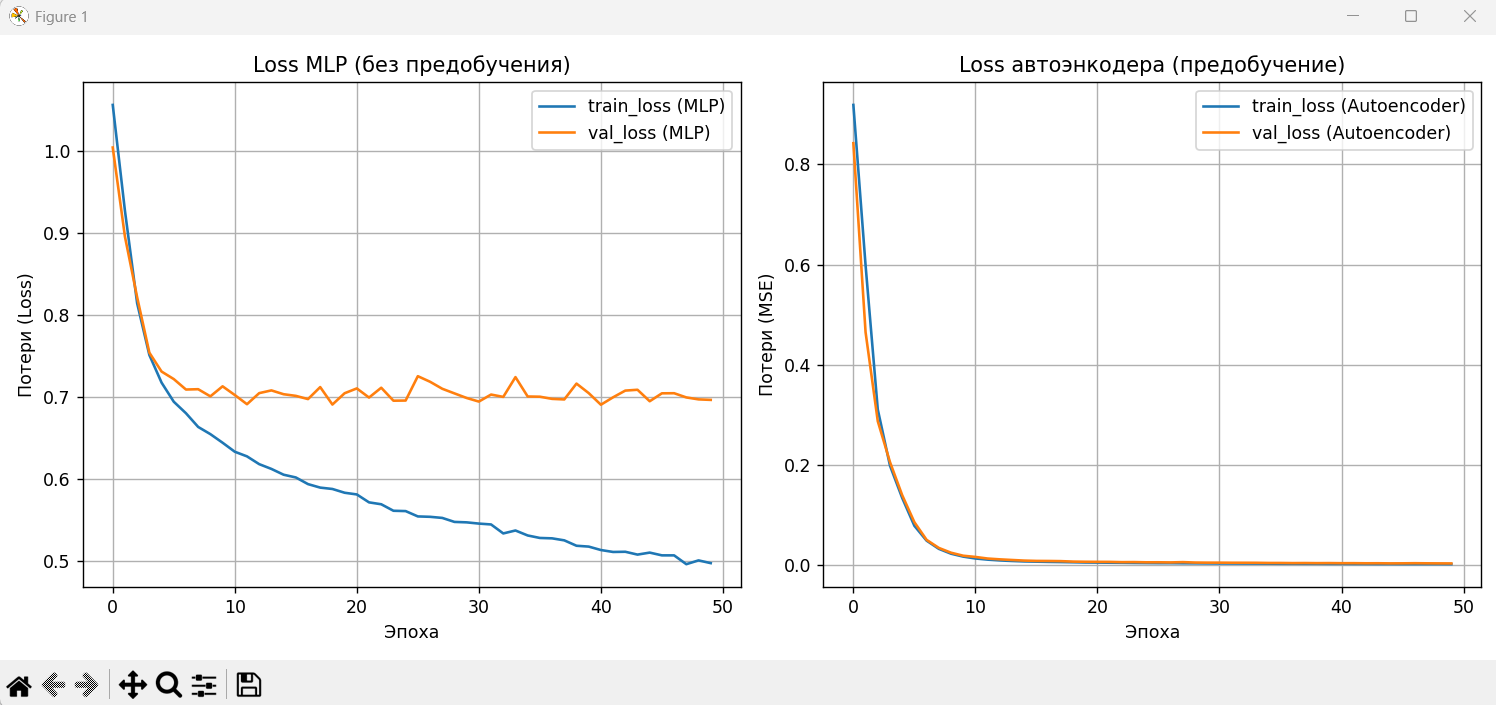
Цель работы: научиться осуществлять предобучение нейронных сетей с помощью автоэнкодерного подхода

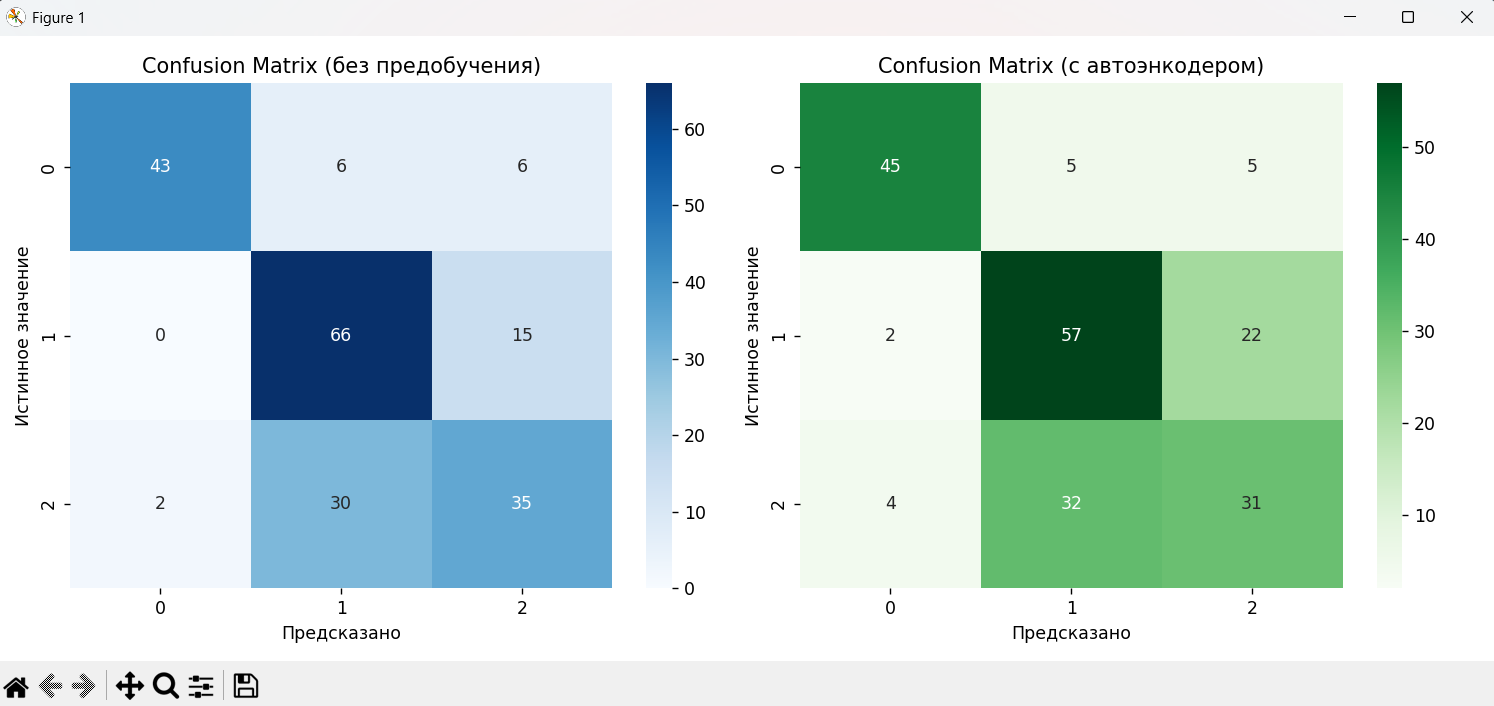
Вариант13



Код программы :

import os  
os.environ['TF\_CPP\_MIN\_LOG\_LEVEL'] = '3'  
  
import pandas as pd  
from sklearn.model\_selection import train\_test\_split  
from sklearn.preprocessing import StandardScaler, LabelEncoder  
from sklearn.metrics import f1\_score, confusion\_matrix  
import matplotlib.pyplot as plt  
import seaborn as sns  
import tensorflow as tf  
from tensorflow.keras import layers, models, Input  
  
# -------------------------  
# 1. Загрузка и подготовка данных  
# -------------------------  
df = pd.read\_csv("maternal.csv")  
  
le = LabelEncoder()  
df['RiskLevel'] = le.fit\_transform(df['RiskLevel'])  
  
X = df.drop('RiskLevel', axis=1).values  
y = df['RiskLevel'].values  
  
scaler = StandardScaler()  
X = scaler.fit\_transform(X)  
  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(  
 X, y, test\_size=0.2, random\_state=42, stratify=y  
)  
  
# -------------------------  
# 2. Базовая MLP модель (без предобучения)  
# -------------------------  
model = models.Sequential([  
 Input(shape=(X.shape[1],)),  
 layers.Dense(64, activation='relu'),  
 layers.Dense(32, activation='relu'),  
 layers.Dense(16, activation='relu'),  
 layers.Dense(3, activation='softmax')  
])  
  
model.compile(optimizer='adam', loss='sparse\_categorical\_crossentropy', metrics=['accuracy'])  
print("Обучаем базовую MLP модель...")  
history\_mlp = model.fit(X\_train, y\_train, epochs=50, batch\_size=16, validation\_split=0.2, verbose=0)  
  
# Оценка  
y\_pred = model.predict(X\_test, verbose=0).argmax(axis=1)  
print("F1-score (macro):", f1\_score(y\_test, y\_pred, average='macro'))  
print("Confusion Matrix:\n", confusion\_matrix(y\_test, y\_pred))  
  
# -------------------------  
# 3. Автоэнкодер  
# -------------------------  
input\_dim = X.shape[1]  
autoencoder\_input = Input(shape=(input\_dim,))  
encoded = layers.Dense(32, activation='relu')(autoencoder\_input)  
encoded = layers.Dense(16, activation='relu')(encoded)  
decoded = layers.Dense(32, activation='relu')(encoded)  
decoded = layers.Dense(input\_dim, activation='linear')(decoded)  
  
autoencoder = models.Model(autoencoder\_input, decoded)  
autoencoder.compile(optimizer='adam', loss='mse')  
  
print("\nОбучаем автоэнкодер...")  
history\_auto = autoencoder.fit(X\_train, X\_train, epochs=50, batch\_size=16, validation\_split=0.2, verbose=0)  
  
# -------------------------  
# 4. Модель с предобучением (encoder часть автоэнкодера)  
# -------------------------  
encoder = models.Model(autoencoder\_input, encoded)  
X\_encoded\_train = encoder.predict(X\_train, verbose=0)  
X\_encoded\_test = encoder.predict(X\_test, verbose=0)  
  
classifier = models.Sequential([  
 Input(shape=(16,)),  
 layers.Dense(16, activation='relu'),  
 layers.Dense(3, activation='softmax')  
])  
  
classifier.compile(optimizer='adam', loss='sparse\_categorical\_crossentropy', metrics=['accuracy'])  
print("\nОбучаем классификатор на закодированных данных...")  
history\_pre = classifier.fit(X\_encoded\_train, y\_train, epochs=50, batch\_size=16, validation\_split=0.2, verbose=0)  
  
y\_pred\_enc = classifier.predict(X\_encoded\_test, verbose=0).argmax(axis=1)  
print("=== Результаты с предобучением автоэнкодером ===")  
print("F1-score (macro):", f1\_score(y\_test, y\_pred\_enc, average='macro'))  
print("Confusion Matrix:\n", confusion\_matrix(y\_test, y\_pred\_enc))  
  
# -------------------------  
# 5. Визуализация  
# -------------------------  
plt.figure(figsize=(12, 5))  
  
# --- Потери MLP ---  
plt.subplot(1, 2, 1)  
plt.plot(history\_mlp.history['loss'], label='train\_loss (MLP)')  
plt.plot(history\_mlp.history['val\_loss'], label='val\_loss (MLP)')  
plt.title('Loss MLP (без предобучения)')  
plt.xlabel('Эпоха')  
plt.ylabel('Потери (Loss)')  
plt.legend()  
plt.grid(True)  
  
# --- Потери автоэнкодера ---  
plt.subplot(1, 2, 2)  
plt.plot(history\_auto.history['loss'], label='train\_loss (Autoencoder)')  
plt.plot(history\_auto.history['val\_loss'], label='val\_loss (Autoencoder)')  
plt.title('Loss автоэнкодера (предобучение)')  
plt.xlabel('Эпоха')  
plt.ylabel('Потери (MSE)')  
plt.legend()  
plt.grid(True)  
  
plt.tight\_layout()  
plt.show()  
  
# -------------------------  
# 6. Confusion matrices  
# -------------------------  
fig, axes = plt.subplots(1, 2, figsize=(12, 5))  
  
sns.heatmap(confusion\_matrix(y\_test, y\_pred), annot=True, fmt="d", cmap="Blues", ax=axes[0])  
axes[0].set\_title("Confusion Matrix (без предобучения)")  
axes[0].set\_xlabel("Предсказано")  
axes[0].set\_ylabel("Истинное значение")  
  
sns.heatmap(confusion\_matrix(y\_test, y\_pred\_enc), annot=True, fmt="d", cmap="Greens", ax=axes[1])  
axes[1].set\_title("Confusion Matrix (c автоэнкодером)")  
axes[1].set\_xlabel("Предсказано")  
axes[1].set\_ylabel("Истинное значение")  
  
plt.tight\_layout()  
plt.show()

Результаты:  
 



Вывод: научилась осуществлять предобучение нейронных сетей с помощью автоэнкодерного подхода.