Министерство образования Республики Беларусь
Учреждение образования
«Брестский государственный технический университет»
ФАКУЛЬТЕТ ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ
Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе №4

Выполнил
А.В. Горобец,
студент группы АС66
Проверил
А. А. Крощенко,
ст. преп. кафедры ИИТ,
« __ » _____2025 г.

Брест 2025

Цель работы: Построить, обучить и оценить многослойный перцептрон (MLP) для решения задачи классификации

Вариант 3

Задание 1. классифицировать вино на "хорошее" (оценка >= 7) и "обычное" (бинарная классификация).

```
import pandas as pd
import torch
import torch.nn as nn
import torch.optim as optim
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, f1_score

# 1. Загрузка и подготовка данных
df = pd.read_csv("C:/Users/Anton/Downloads/winequality-white.csv", sep=";")
df["Target"] = (df["quality"] >= 7).astype(int)
X = df.drop(["quality", "Target"], axis=1)
y = df["Target"]

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3, ran-
dom_state=42)

X_train_tensor = torch.tensor(X_train, dtype=torch.float32)
X_test_tensor = torch.tensor(X_test, dtype=torch.float32)
y_train_tensor = torch.tensor(y_train.values, dtype=torch.float32).view(-1, 1)
y_test_tensor = torch.tensor(y_test.values, dtype=torch.float32).view(-1, 1)

# 2. Архитектура нейронной сети
class MLP(nn.Module):
    def __init__(self):
        super(MLP, self).__init__()
        self.model = nn.Sequential(
            nn.Linear(X_train.shape[1], 12),
            nn.ReLU(),
            nn.Linear(12, 12),
            nn.ReLU(),
            nn.Linear(12, 1)
        )

    def forward(self, x):
        return self.model(x)

# 3. Инициализация модели, функции потерь и оптимизатора
model = MLP()
```

```
criterion = nn.BCEWithLogitsLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)

# 4. Цикл обучения
epochs = 100  # увеличено вдвое для эксперимента

for epoch in range(epochs):
    model.train()
    y_pred = model(X_train_tensor)
    loss = criterion(y_pred, y_train_tensor)

    optimizer.zero_grad()
    loss.backward()
    optimizer.step()

    if (epoch + 1) % 10 == 0:
        print(f"Epoch {epoch+1}/{epochs}, Loss: {loss.item():.4f}")

# 5. Оценка модели
model.eval()
with torch.no_grad():
    y_logits = model(X_test_tensor)
    y_pred = torch.sigmoid(y_logits)
    y_pred_class = (y_pred > 0.5).int()

y_true = y_test_tensor.numpy()
y_pred_np = y_pred_class.numpy()

acc = accuracy_score(y_true, y_pred_np)
f1 = f1_score(y_true, y_pred_np)

print(f"Accuracy: {acc:.4f}")
print(f"F1-score: {f1:.4f}")
```

```
B:\PythonProject\OMO\.venv\Scripts\python.exe B:\PythonProject\OMO\.venv\Lib\site-packages\lab4.py
Epoch 10/100, Loss: 0.8153
Epoch 20/100, Loss: 0.7847
Epoch 30/100, Loss: 0.7562
Epoch 40/100, Loss: 0.7281
Epoch 50/100, Loss: 0.6997
Epoch 60/100, Loss: 0.6705
Epoch 70/100, Loss: 0.6407
Epoch 80/100, Loss: 0.6104
Epoch 90/100, Loss: 0.5808
Epoch 100/100, Loss: 0.5529
Accuracy: 0.7456
F1-score: 0.1762
```

Вывод: я построил, обучил и оценил многослойный перцептрон (MLP) для решения задачи классификациирезультат.