МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Вятский государственный университет»
(ФГБОУ ВО «ВятГУ»)
Институт математики и информационных систем
Факультет автоматики и вычислительной техники
Кафедра электронных вычислительных машин

«Теория принятия решений»
Отчёт по лабораторной работе №2 - python

Выполнил студент группы ИВТб-4301-04-00 _____/Самылов Д.Л.
Проверил преподаватель _____/Крутиков А.К.

Киров 2025

1 Цель работы

Освоение нейросетевой технологии для решения задач классификации и прогнозирования с помощью программ на Python.

2 Задание

Третья часть задания заключается в выполнении аналогичных действий (обучении сетей на аналогичной обучающей выборке) для двух простых моделей нейронных сетей: линейной нейронной сети и нейронной сети прямого распространения. Использовать специализированные библиотеки для языка Python.

3 Теория

Линейная нейронная сеть представляет собой простейшую структуру искусственной нейронной сети, состоящую из одного слоя нейронов, каждый из которых производит линейную комбинацию входных сигналов с последующим применением активационной функции (линейная функция активации). Эта архитектура используется преимущественно для решения простых задач классификации и регрессии, поскольку её возможности ограничены способностью моделировать лишь линейные зависимости между признаками и целевыми переменными. Линейные сети легко интерпретируются и быстро обучаются, однако неспособны эффективно обрабатывать сложные нелинейные взаимосвязи, характерные для большинства реальных задач машинного обучения.

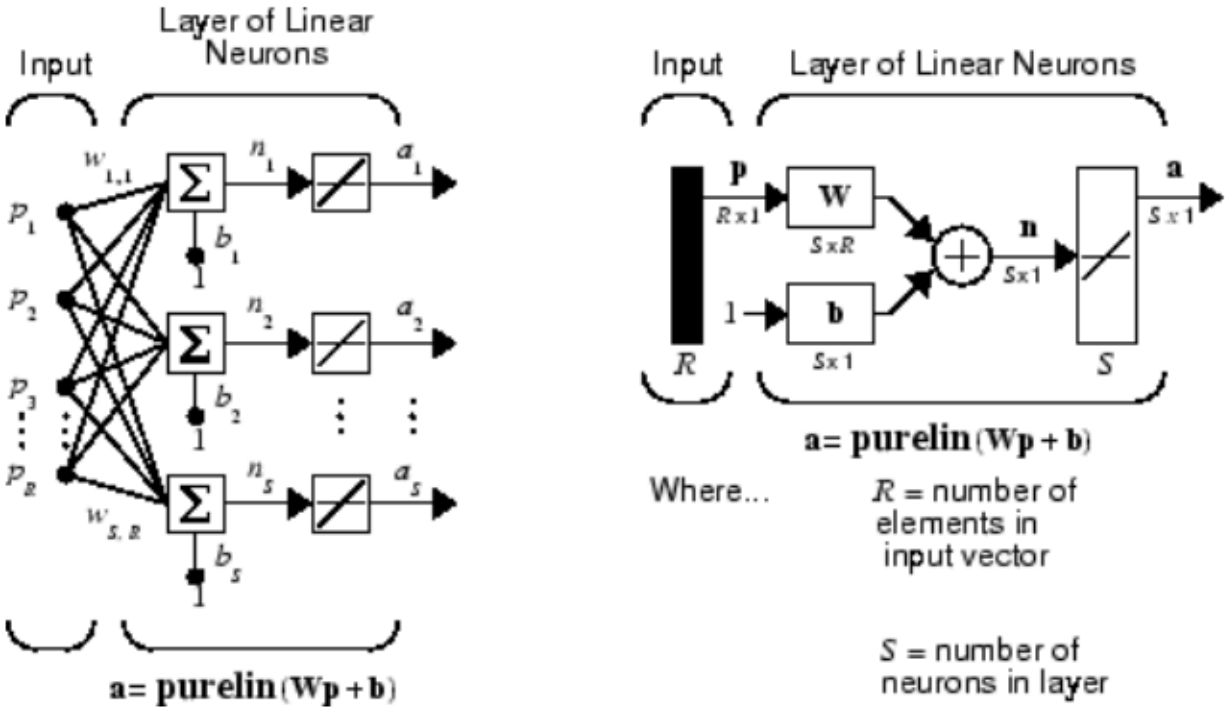

Рис. 1: Структура линейного нейрона

Линейные слои являются единственными слоями линейных нейронов. Они могут быть статическими, с входными задержками 0, или динамическими с входными задержками, больше, чем 0. Они могут быть обучены на простых линейных проблемах временных

рядов, но часто используются адаптивно, чтобы продолжить учиться, в то время как развернуто, таким образом, они могут настроить к изменениям в отношении между вводами и выводами, будучи используемым.

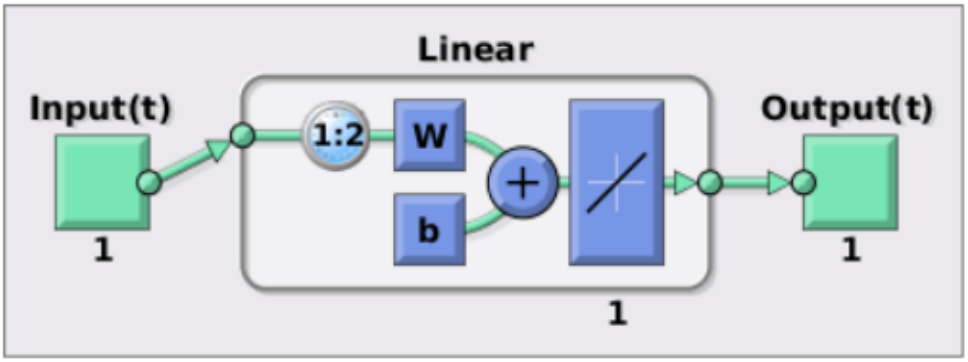

Рис. 2: Линейный слой

Нейронная сеть прямого распространения (Feedforward Neural Network) — это тип нейронной сети, в которой передача сигнала осуществляется исключительно в одном направлении, от входного слоя к выходному через промежуточные скрытые слои. Она не имеет обратных связей и способна эффективно решать задачи классификации, регрессии и распознавания образов. Обучение такой сети производится методом обратного распространения ошибки (Backpropagation), позволяющим автоматически настраивать веса соединений между нейронами для минимизации ошибки предсказания.

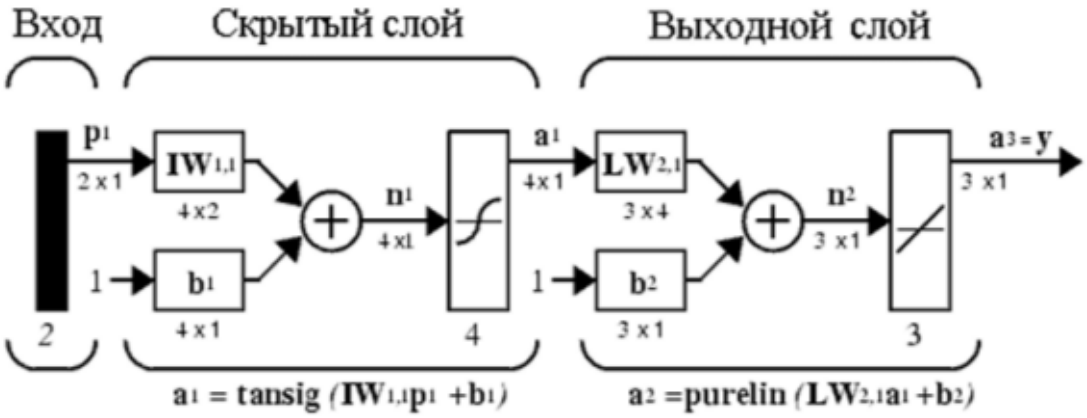

Рис. 3: Укрупненная структура FNN- сети

4 Выполнение лабораторной работы

Будем предсказывать рейтинг машины для покупки по 11 критериям. Рейтинг машин генерируется по линейной функции. Веса и критерии представлены ниже:

```
# Рейтинг автомобиля от 1 до 100
df['Рейтинг автомобиля (1-100)'] = (
    50  # базовый рейтинг
    + (df['Год выпуска'] - 2012) * 1.5  # чем новее, тем выше рейтинг
    - df['Пробег (тыс. км)'] * 0.1  # чем больше пробег, тем ниже рейтинг
    + (df['Объем двигателя (л)'] - 2.0) * 5  # оптимальный объем около 2.0 л
    + (df['Мощность (л.с.)'] - 200) * 0.05  # оптимальная мощность около 200 л.с.
    + df['Тип КПП (0-мех, 1-авто)'] * 3  # автомат лучше механики
    + np.where(df['Тип топлива (0-бенз,1-диз,2-гибр,3-элек)'] == 2, 5,
               np.where(df['Тип топлива (0-бенз,1-диз,2-гибр,3-элек)'] == 3, 8, 0))  # гибрид
    ↪  и электро лучше
    - (df['Количество владельцев'] - 1) * 2  # чем больше владельцев, тем ниже рейтинг
    + (df['Состояние кузова (баллы 1-10)'] - 5) * 2  # лучше состояние - выше рейтинг
    + (df['Состояние салона (баллы 1-10)'] - 5) * 2  # лучше состояние - выше рейтинг
    + df['Сервисная история (0-нет,1-есть)'] * 4  # наличие истории повышает рейтинг
    + np.random.normal(0, 5, n_rows)  # случайный шум
)
```

4.1 Исходный код линейной нейронной сети

```
X_tensor, y_tensor, dataset, dataloader, num_features, norm_params =
↪  load_dbf_data("car_rating_data.dbf")

# Определение структуры линейной нейронной сети с динамическим количеством признаков
class LinearModel(nn.Module):
    def __init__(self, input_features):
        super().__init__()
        self.linear_layer = nn.Linear(in_features=input_features, out_features=1)

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

# СОЗДАНИЕ МОДЕЛИ!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
model = LinearModel(num_features)
criterion = nn.MSELoss() # Функция потерь - среднеквадратичная ошибка
optimizer = torch.optim.SGD(model.parameters(), lr=0.001) # Уменьшили learning rate

# ОБУЧЕНИЕ!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
epochs = 100

for epoch in range(epochs):
    total_loss = 0
    for inputs, targets in dataloader:
        optimizer.zero_grad()
```

```
        outputs = model(inputs)
        loss = criterion(outputs, targets)
        loss.backward()
        optimizer.step()
        total_loss += loss.item()

    if epoch % 10 == 0:
        avg_loss = total_loss / len(dataloader)
        print(f'Epoch {epoch}, Loss: {avg_loss:.4f}')

# 3. Выводим результаты с ошибкой
print_predictions_results_with_error(test_predictions_denorm, test_features, actual_values,
↪  norm_params)
```

4.2 Исходный код FNN сети

```
X_tensor, y_tensor, dataset, dataloader, num_features, norm_params =
↪  load_dbf_data("car_rating_data.dbf")

# Определение структуры линейной нейронной сети с динамическим количеством признаков
class LinearModel(nn.Module):
    def __init__(self, input_features):
        super().__init__()
        self.linear_layer = nn.Linear(in_features=input_features, out_features=1)

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

# Создание модели с правильным количеством входных признаков
model = LinearModel(num_features)
criterion = nn.MSELoss() # Функция потерь - среднеквадратичная ошибка
optimizer = torch.optim.SGD(model.parameters(), lr=0.001) # Уменьшили learning rate

# Обучение модели
epochs = 100

for epoch in range(epochs):
    total_loss = 0
    for inputs, targets in dataloader:
        optimizer.zero_grad()
        outputs = model(inputs)
        loss = criterion(outputs, targets)
        loss.backward()
        optimizer.step()
        total_loss += loss.item()

    if epoch % 10 == 0:
        avg_loss = total_loss / len(dataloader)
        print(f'Epoch {epoch}, Loss: {avg_loss:.4f}')
```

```
33
34
35  # Тестирование модели - создаем тестовый ввод с правильным количеством признаков
36  # Берем средние значения всех признаков из обучающих данных (уже нормализованные)
37  mean_features = torch.mean(X_tensor, dim=0)
38  test_input = mean_features.unsqueeze(0)  # Добавляем dimension для batch
39
```

4.3 Результаты тестирования линейной нейронной сети

На таблице ниже представлены результаты обучения линейной сети на различных по размеру обучающих выборках. Тестирование проводилось на одной и той же выборке размером 100 векторов сгенерированной отдельно. По результатом обучения можно сказать, что достигнув размера 100 векторов дальнейшее увеличение размера обучающей выборки не приносит никакого увеличения точности. В среднем сеть обученная на выборке в 100 векторов ошибается на 4 пункта, максимум на 10. При рейтинге от 0 до 100 ошибка в 10 пунктов - это 10%, что является довольно большим показателем. Но в целом, результат для такой простой сети довольно хороший.

Размер обучающей Выборки	Макс. Ошибка	Средняя Ошибка
20	28.2075	8.8434
50	25.0594	9.1477
100	10.1803	3.9251
500	11.8124	3.9734
1000	11.8940	3.9050

Рис. 4: Таблица для линейной нейронной сети

```
Epoch 0, Loss: 1.0511
Epoch 10, Loss: 0.0739
Epoch 20, Loss: 0.0696
Epoch 30, Loss: 0.0695
Epoch 40, Loss: 0.0695
Epoch 50, Loss: 0.0696
Epoch 60, Loss: 0.0695
Epoch 70, Loss: 0.0695
Epoch 80, Loss: 0.0695
Epoch 90, Loss: 0.0695
Загружено 10 тестовых векторов
Прогнозирование завершено успешно!

================================================================================
РЕЗУЛЬТАТЫ ПРОГНОЗИРОВАНИЯ С ОШИБКОЙ
================================================================================

Вектор #1:
  Реальный рейтинг: 18.00
  Прогнозируемый рейтинг: 16.36
  Ошибка: -1.64

Вектор #2:
  Реальный рейтинг: 41.00
  Прогнозируемый рейтинг: 39.81
  Ошибка: -1.19

Вектор #3:
  Реальный рейтинг: 61.00
  Прогнозируемый рейтинг: 69.96
  Ошибка: +8.96

Вектор #4:
  Реальный рейтинг: 56.00
  Прогнозируемый рейтинг: 56.44
  Ошибка: +0.44

Вектор #5:
  Реальный рейтинг: 59.00
  Прогнозируемый рейтинг: 57.80
  Ошибка: -1.20
```

Рис. 5: Экранные формы линейной нейронной сети

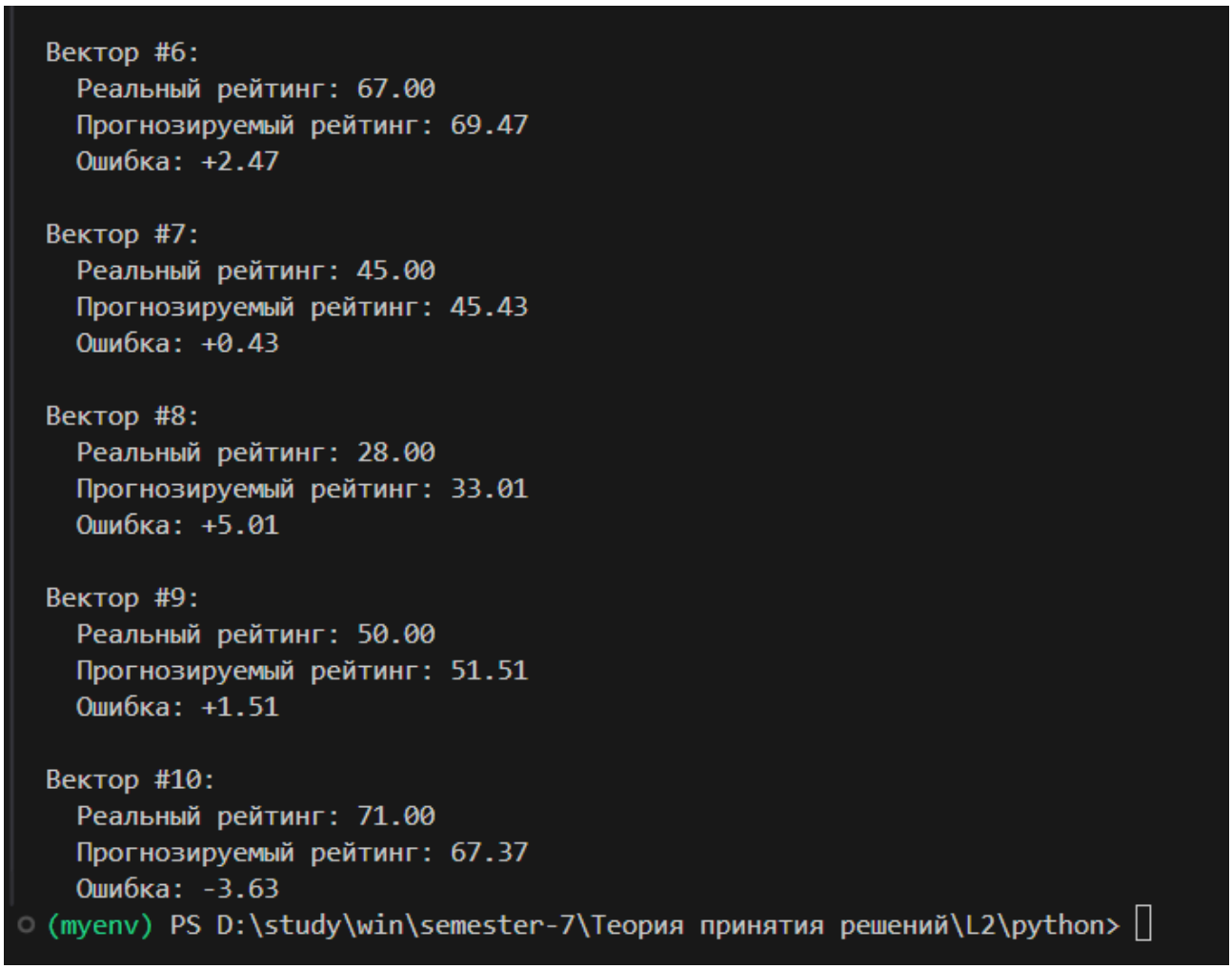

Рис. 6: Экранные формы линейной нейронной сети

4.4 Результаты тестирования нейронной сети прямого распространения

На таблице ниже представлены результаты обучения линейной сети на различных по размеру обучающих выборках, а так же для выборки размером 100 векторов с различным количеством нейронов в скрытом слое. Тестирование проводилось на одной и той же выборке размером 100 векторов сгенерированной отдельно. Так же как и для линейной сети лучшие показатели оказались у сети обученной на 100 векторах и 50 векторах с 64 нейронами в скрытом слое. Не смотря на более сложное строение для нашей задачи сети прямого распространения показала результаты хуже, чем линейная сеть.

Размер обучающей Выборки	Кол-во Нейронов в Скрытом слое	Макс. Ошибка	Средняя Ошибка
20	64	28.5422	7.7177
50	64	16.0203	5.2735
100	64	16.7191	5.1590
500	64	21.7464	5.7628
1000	64	16.2195	5.8166
100	32	21.4936	6.6759
100	128	16.9528	5.6201

Рис. 7: Таблица для FNN сети

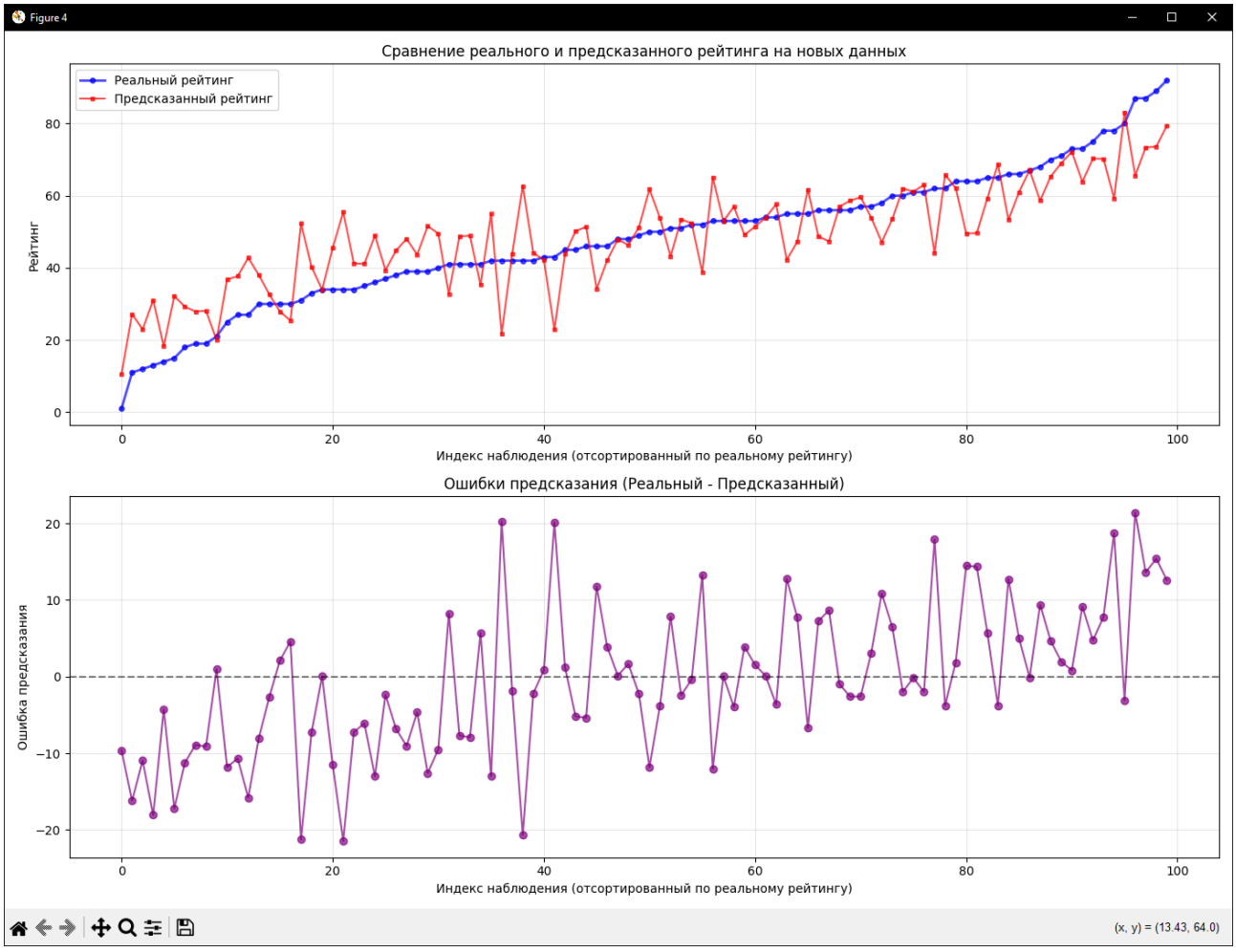

Рис. 8: Экранные формы нейронной сети прямого распространения - 20 векторов, 64 нейрона

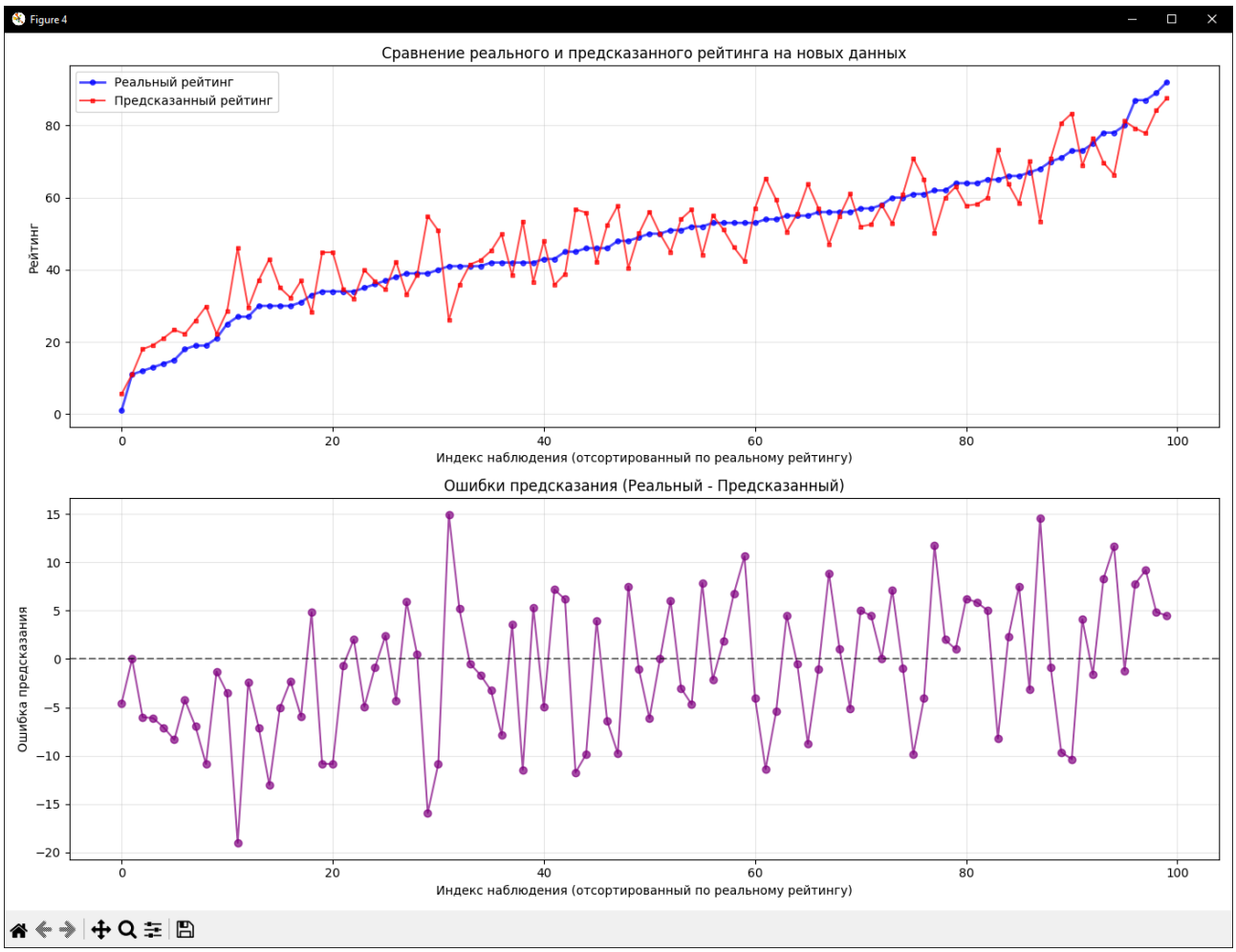

Рис. 9: Экранные формы нейронной сети прямого распространения - 50 векторов, 64 нейрона

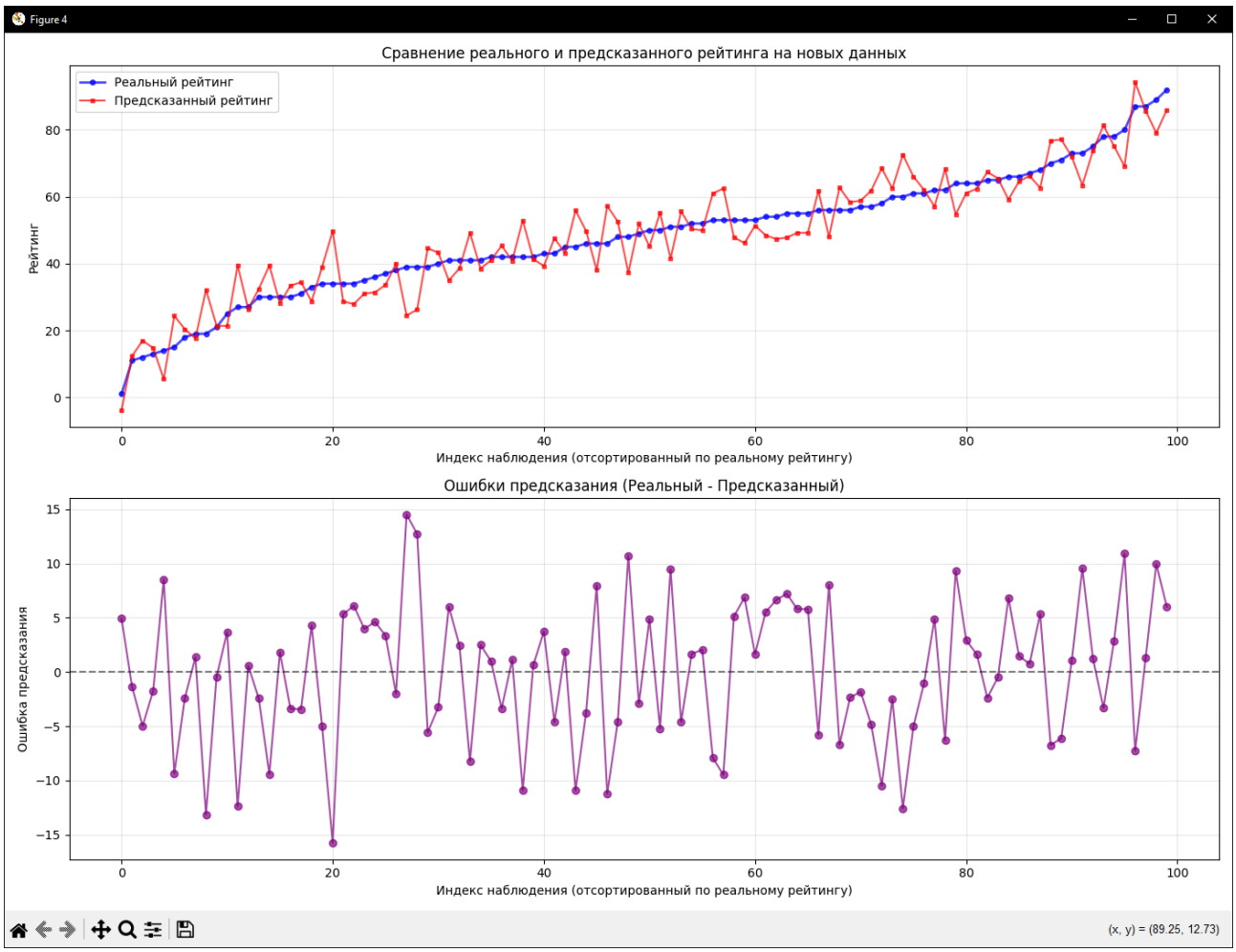

Рис. 10: Экранные формы нейронной сети прямого распространения - 100 векторов, 64 нейрона

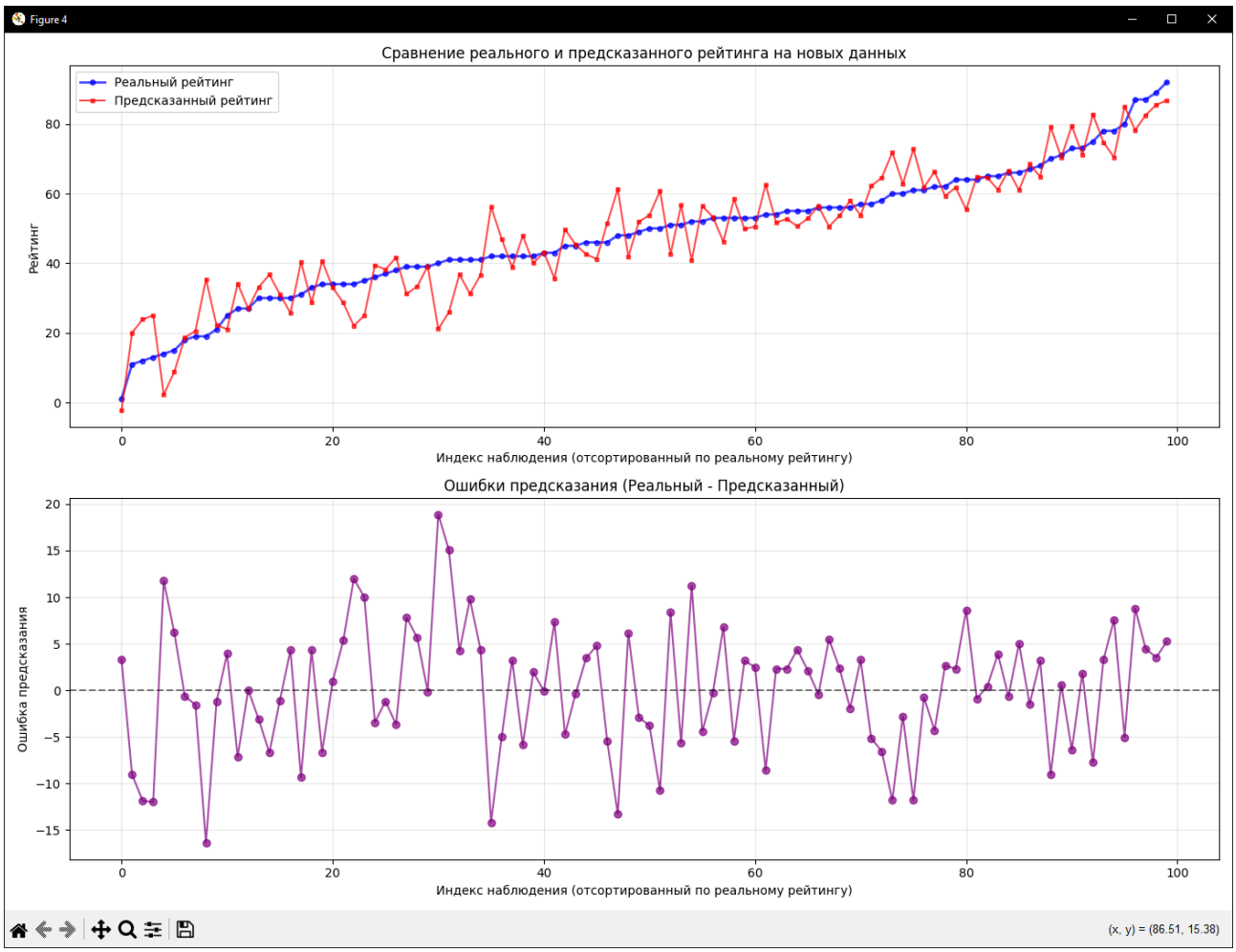

Рис. 11: Экранные формы нейронной сети прямого распространения - 500 векторов, 64 нейрона

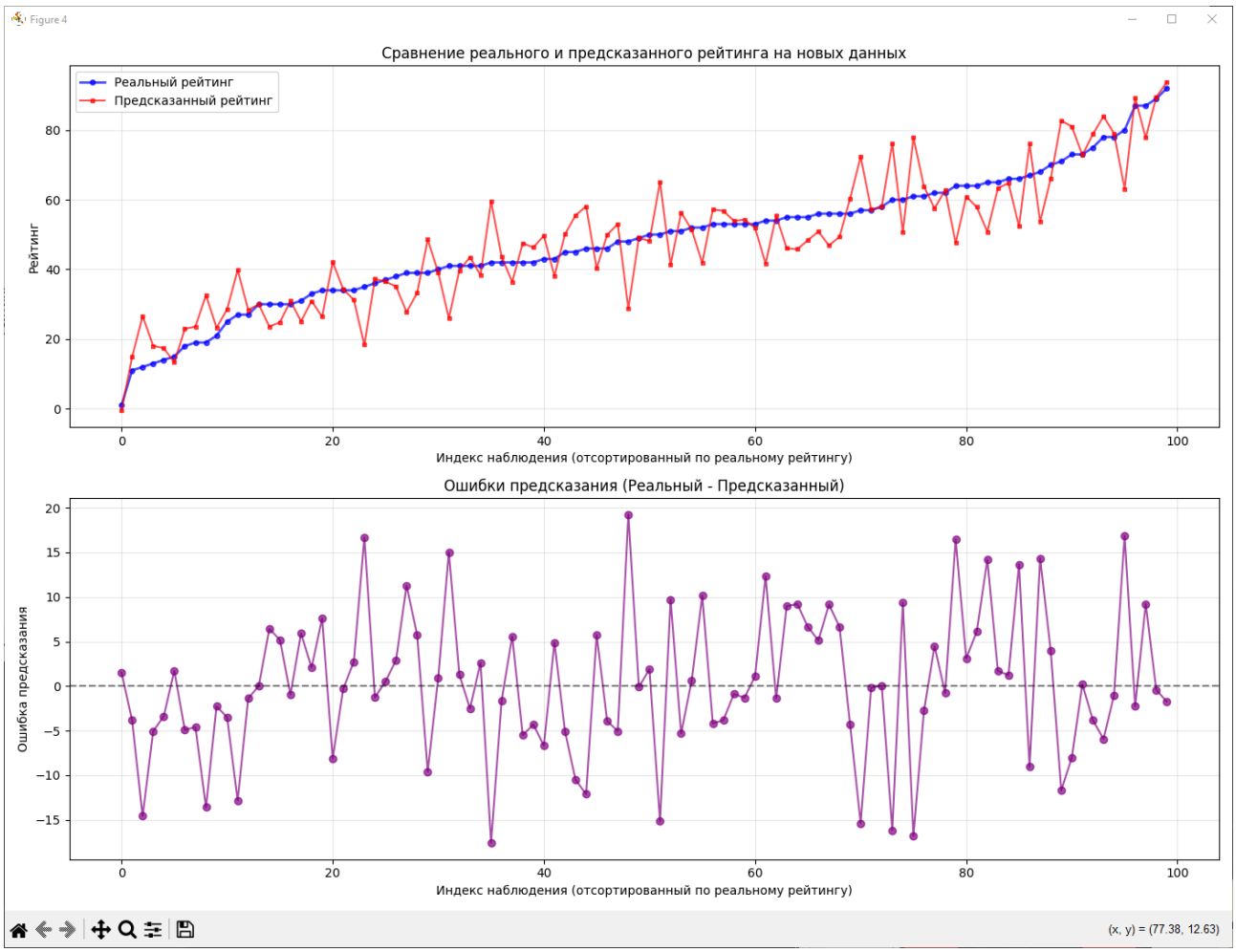

Рис. 12: Экранные формы нейронной сети прямого распространения - 1000 векторов, 64 нейрона

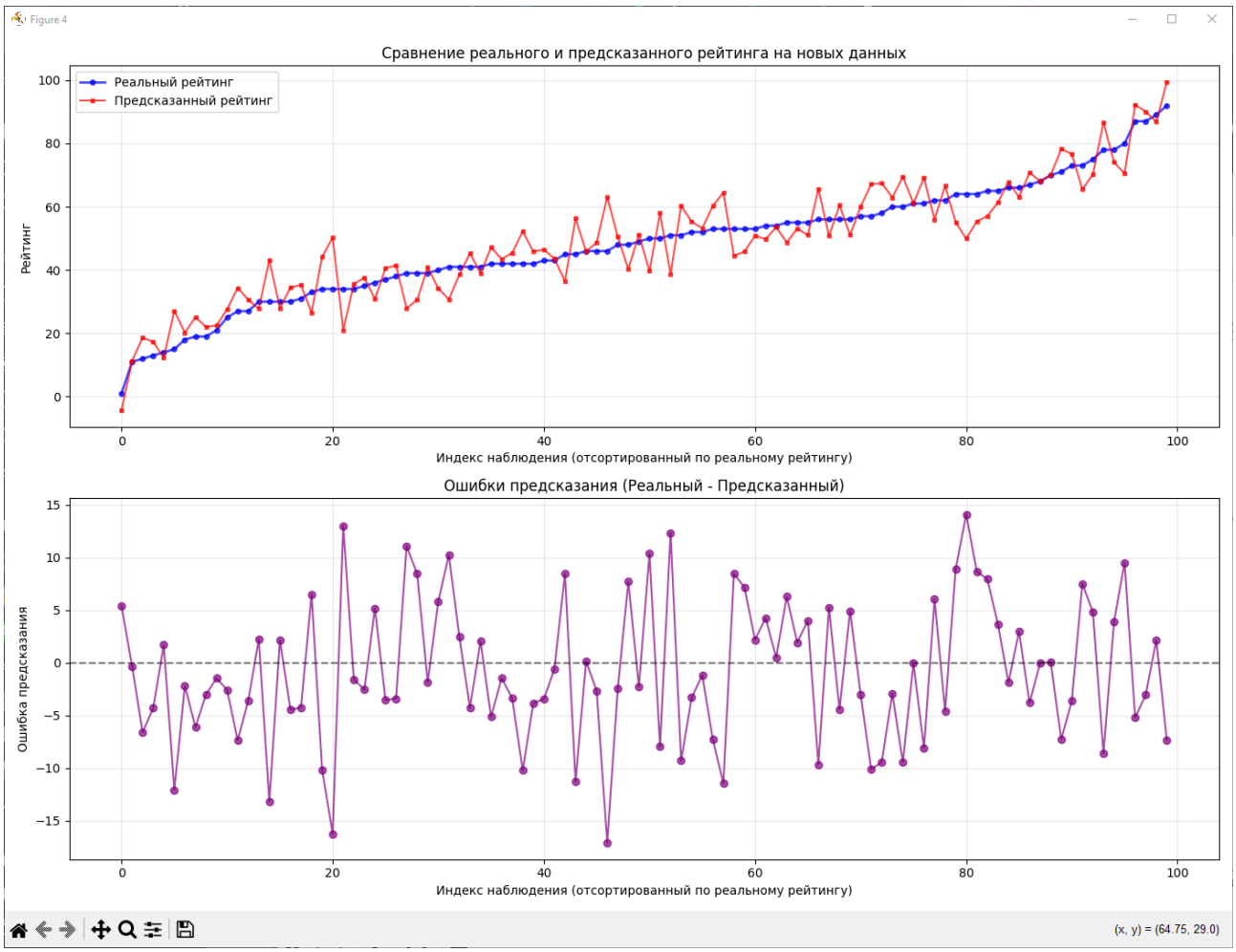

Рис. 13: Экранные формы нейронной сети прямого распространения - 100 векторов, 32 нейрона

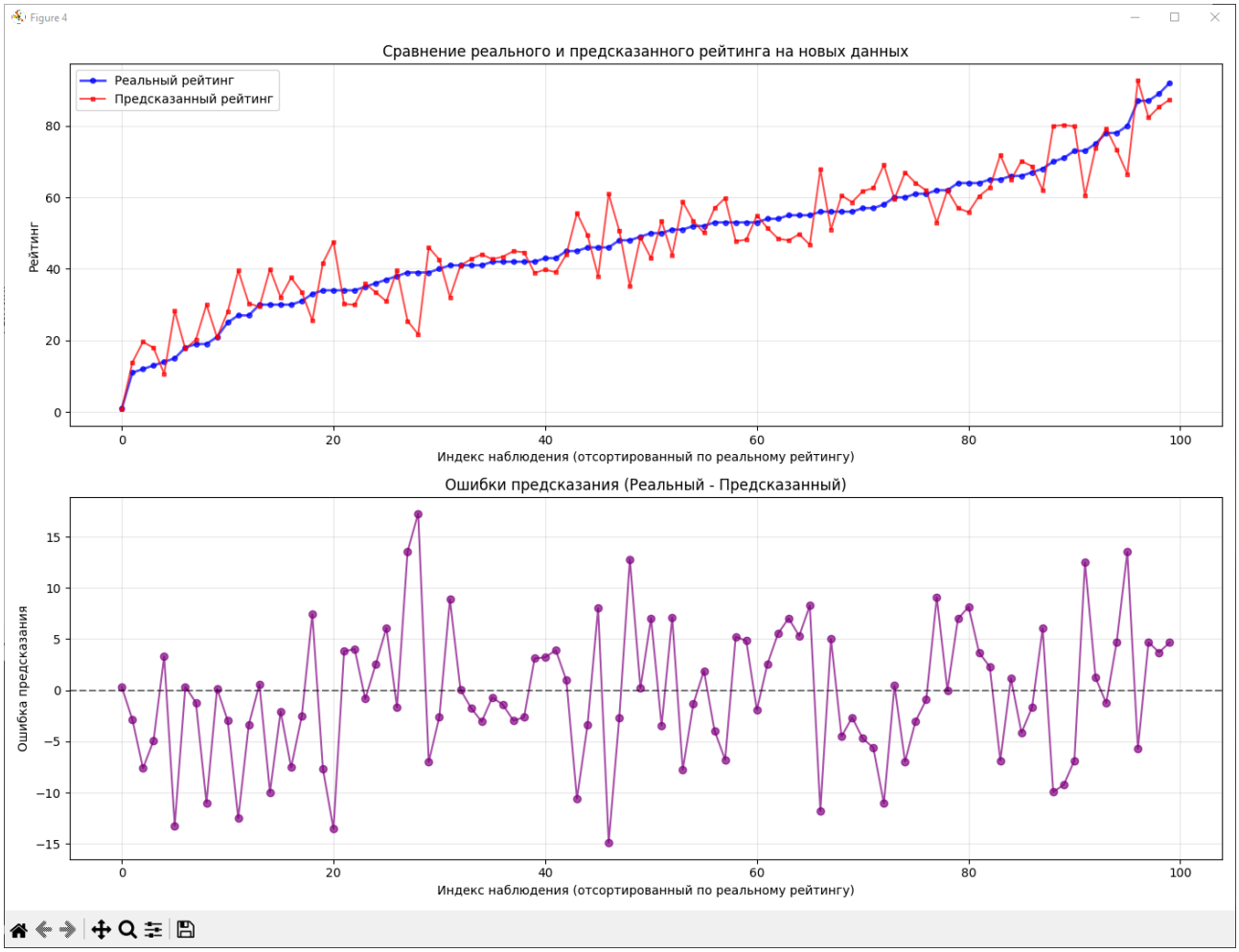

Рис. 14: Экранные формы нейронной сети прямого распространения - 100 векторов, 128 нейрона

5 Выводы по работе

В ходе лабораторной работы были рассмотрены и реализованы на языке программирования Python линейные нейронные сети и нейронные сети прямого распространения.

На основе 5 различных по размеру выборок были обучены 5 линейных нейронных сетей, наиболее точной из которой оказалась третья нейронная сеть обученная на 100 векторах. Так же было обучено 7 нейронных сетей прямого распространения - на 5 различных размерностях векторов при 64 нейронов в скрытом слое, и 2 сети обученные на 100 входных векторах и 32 и 128 нейронах в скрытом слое. Среди сетей прямого распространения наиболее точными оказались сети обученные на 50 и 100 векторах и 64 нейронах в скрытом слое.

При поставленной задаче - спрогнозировать рейтинг машины в зависимости от входных параметров наиболее точной оказалась линейная сеть обученная на 100 входных векторах. Сеть прямого распространения оказалась менее точной для данной задачи, но теоретически такие сети могут решать куда более обширный спектр задач, чем линейные сети. Возможно, что сети прямого распространения могут показать более точные при других параметрах.

Полученные в ходе выполнения лабораторной работы знания станут основой для более глубокого изучения нейронных сетей и пригодятся при дальнейшем изучении перемета теория принятия решений.