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

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

Киров 2025

1 Цель работы - часть 1 - NeuroPro

Освоение нейросетевой технологии для решения задач классификации и прогнозирования с помощью программы NeuroPro 0.25.

2 Выполнение лабораторной работы

2.1 Обучение нейронных сетей с разным количеством нейронов

В ходе выполнения лабораторной работы обучено 6 нейронных сетей, с разным количеством нейронов в слоях.

Название сети	Число слоев	Число нейронов	Число циклов обучения	Макс.Ошибка	Сред.Ошибка	Прогноз	Ошибка прогноза
n1	3	15-15-5	22	21.65	7,08	55,30	7,30
n2	3	15-18-11	14	17,44	7,24	64,23	16,23
n3	3	22-22-7	16	21,46	7,77	67,12	19,12
n4	3	21-30-13	11	12,55	5,98	69,53	21,53
n5	3	9-10-5	22	13,66	4,91	51,07	3,07
n6	3	10-10-10	18	23,77	7,03	61,48	13,48

Рис. 1: Нейронный сети

По результатам тестирования выбирается сеть n5, имеющая наименьшую ошибку прогноза, среднюю ошибку и вторую наименьшую максимальную ошибку.

Значимость входных сигналов для выбранной сети представлена на рисунке 2.

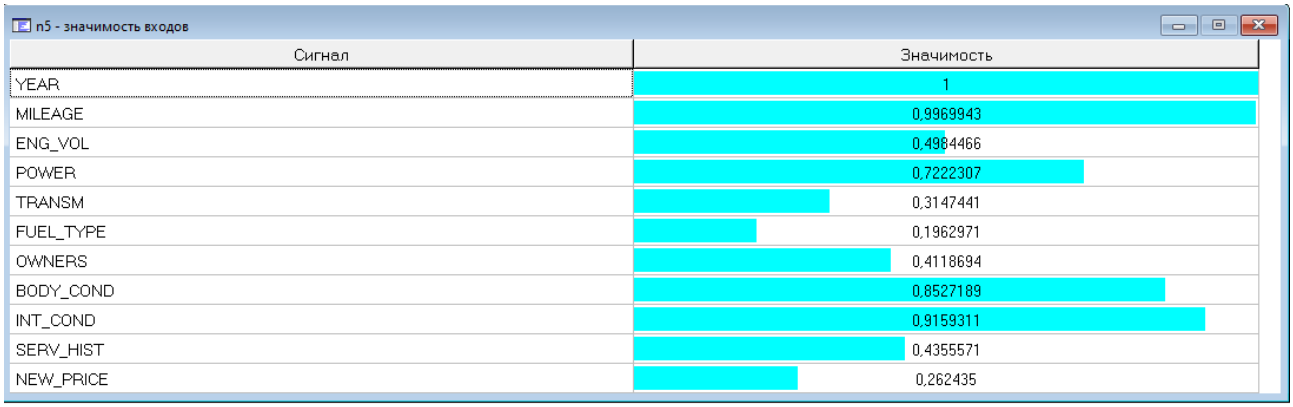


Рис. 2: Значимость входных сигналов

Тестирование проведенное для выбранной сети представлено на рисунках 3-4.

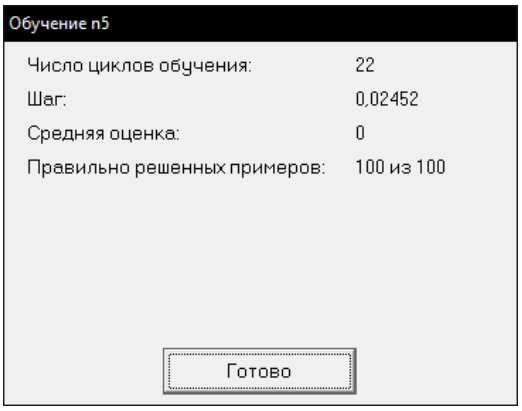
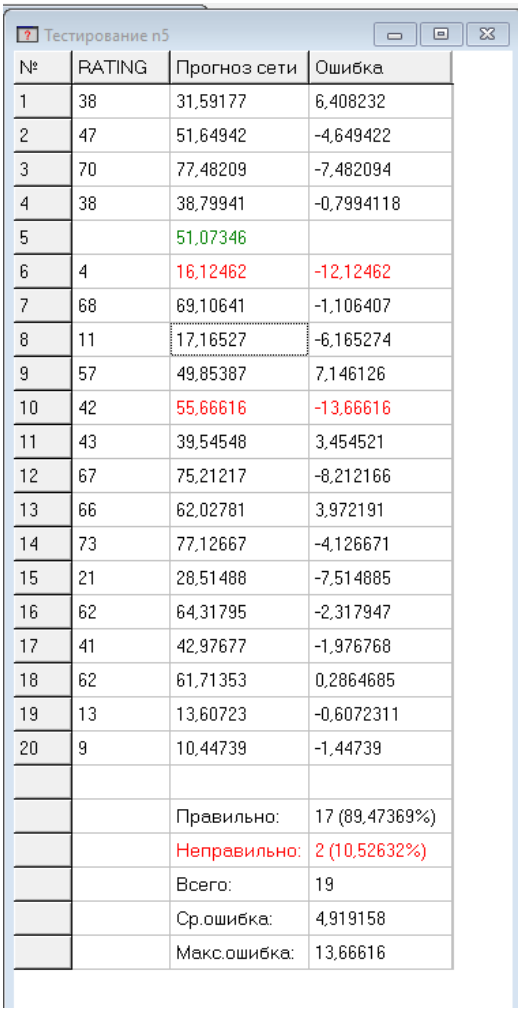


Рис. 3: Тестирование n5



№	RATING	Прогноз сети	Ошибка
1	38	31,59177	6,408232
2	47	51,64942	-4,649422
3	70	77,48209	-7,482094
4	38	38,79941	-0,7994118
5		51,07346	
6	4	16,12462	-12,12462
7	68	69,10641	-1,106407
8	11	17,16527	-6,165274
9	57	49,85387	7,146126
10	42	55,66616	-13,66616
11	43	39,54548	3,454521
12	67	75,21217	-8,212166
13	66	62,02781	3,972191
14	73	77,12667	-4,126671
15	21	28,51488	-7,514885
16	62	64,31795	-2,317947
17	41	42,97677	-1,976768
18	62	61,71353	0,2864685
19	13	13,60723	-0,6072311
20	9	10,44739	-1,44739
		Правильно:	17 (89,47369%)
		Неправильно:	2 (10,52632%)
		Всего:	19
		Ср.ошибка:	4,919158
		Макс.ошибка:	13,66616

Рис. 4: Тестирование n5

2.2 Упрощение выбранной нейронной сети n5

Результаты прогнозирования полученные при упрощениях сети представлены на рисунке 5.

Метод упрощения сети	Прогноз	Сред ошибка	Макс ошибка
Сокращение числа входных сигналов	53,82	6,11	9,74
Сокращение числа нейронов	29,27	5,09	9,75
Равномерное упрощение сети	13,63	5,39	9,06
Сокращение числа синапсов	71,76	6,01	9,79
Сокращение числа неоднородных входов	47,38	5,37	9,61
Бинаризация весов синапсов и неоднородных входов	50,51	5,31	9,45

Рис. 5: Методы упрощения сети

Наиболее приемлемым вариантом упрощения сети является вариант сокращение числа неоднородных входов, при котором сеть имеет минимальную среднюю ошибку, низкую минимальную ошибку прогнозирования и наиболее точный к исходному (48) прогноз - 47.38. Значимость входных сигналов для упрощенной сети представлена на рисунке 6.

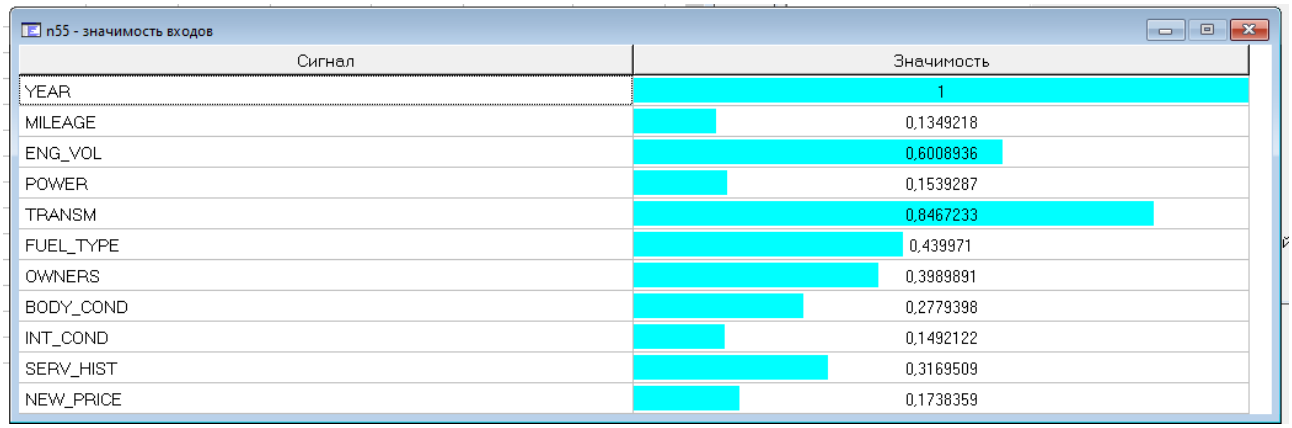
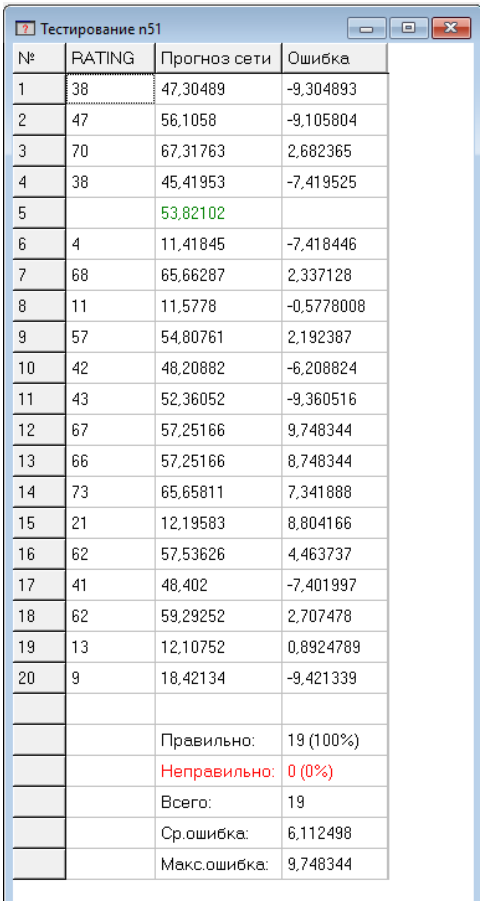


Рис. 6: Значимость входных сигналов

Экранные формы при выборе упрощений представлены далее:

Рис. 7: Сокращение входных сигналов



№	RATING	Прогноз сети	Ошибка
1	38	47,30489	-9,304893
2	47	56,1058	-9,105804
3	70	67,31763	2,682365
4	38	45,41953	-7,419525
5		53,82102	
6	4	11,41845	-7,418446
7	68	65,66287	2,337128
8	11	11,5778	-0,5778008
9	57	54,80761	2,192387
10	42	48,20882	-6,208824
11	43	52,36052	-9,360516
12	67	57,25166	9,748344
13	66	57,25166	8,748344
14	73	65,65811	7,341888
15	21	12,19583	8,804166
16	62	57,53626	4,463737
17	41	48,402	-7,401997
18	62	59,29252	2,707478
19	13	12,10752	0,8924789
20	9	18,42134	-9,421339
		Правильно:	19 (100%)
		Неправильно:	0 (0%)
		Всего:	19
		Ср.ошибка:	6,112498
		Макс.ошибка:	9,748344

Рис. 8: Тестирование копии n5 - Сокращение входных сигналов

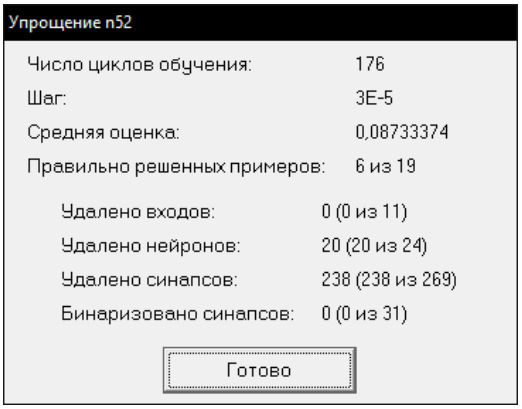
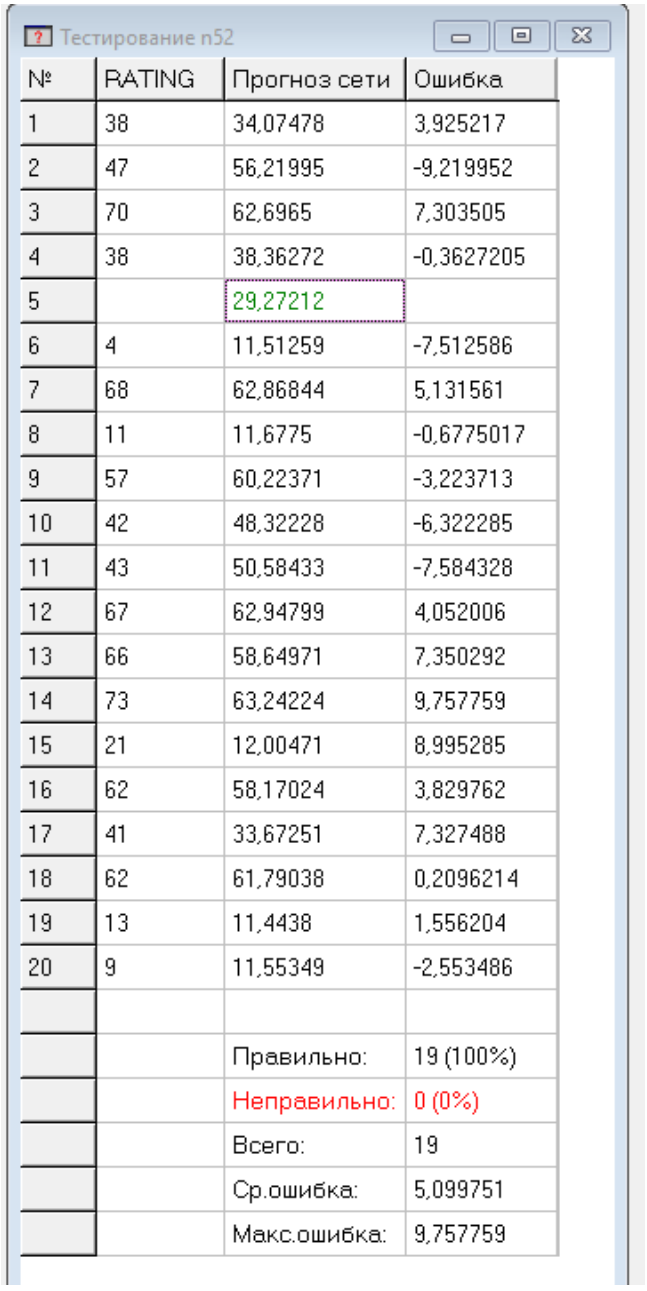


Рис. 9: Сокращение числа нейронов



№	RATING	Прогноз сети	Ошибка
1	38	34,07478	3,925217
2	47	56,21995	-9,219952
3	70	62,6965	7,303505
4	38	38,36272	-0,3627205
5		29,27212	
6	4	11,51259	-7,512586
7	68	62,86844	5,131561
8	11	11,6775	-0,6775017
9	57	60,22371	-3,223713
10	42	48,32228	-6,322285
11	43	50,58433	-7,584328
12	67	62,94799	4,052006
13	66	58,64971	7,350292
14	73	63,24224	9,757759
15	21	12,00471	8,995285
16	62	58,17024	3,829762
17	41	33,67251	7,327488
18	62	61,79038	0,2096214
19	13	11,4438	1,556204
20	9	11,55349	-2,553486
		Правильно:	19 (100%)
		Неправильно:	0 (0%)
		Всего:	19
		Ср.ошибка:	5,099751
		Макс.ошибка:	9,757759

Рис. 10: Тестирование копии n5 - Сокращение числа нейронов

Рис. 11: Равномерное упрощение сети

Тестирование n53

№	RATING	Прогноз сети	Ошибка
1	38	36,62318	1,376816
2	47	49,61562	-2,615616
3	70	62,23395	7,766048
4	38	30,2829	7,717096
5		13,63856	
6	4	12,75411	-8,754107
7	68	66,97507	1,024925
8	11	18,45276	-7,452759
9	57	64,7673	-7,767296
10	42	51,07	-9,069996
11	43	51,10724	-8,107243
12	67	65,88068	1,119316
13	66	66,92264	-0,9226379
14	73	66,19131	6,808685
15	21	12,34182	8,658178
16	62	55,85601	6,143993
17	41	40,99397	0,006031036
18	62	53,04452	8,955482
19	13	12,5781	0,421896
20	9	16,85609	-7,856091
		Правильно:	19 (100%)
		Неправильно:	0 (0%)
		Всего:	19
		Ср.ошибка:	5,397064
		Макс.ошибка:	9,069996

Рис. 12: Тестирование копии n5 - Равномерное упрощение сети

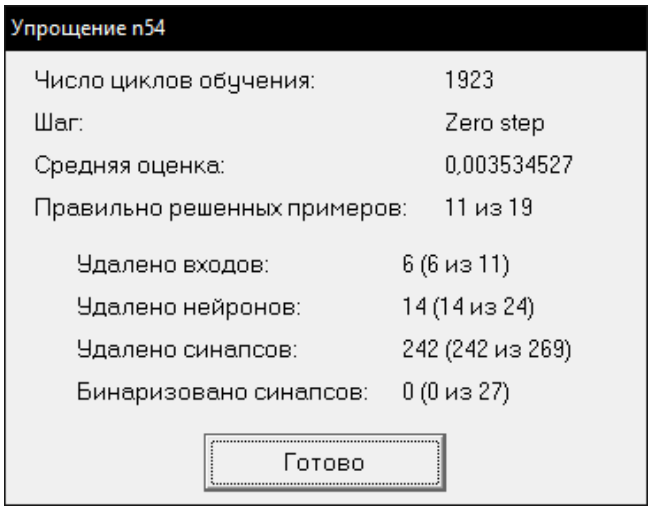
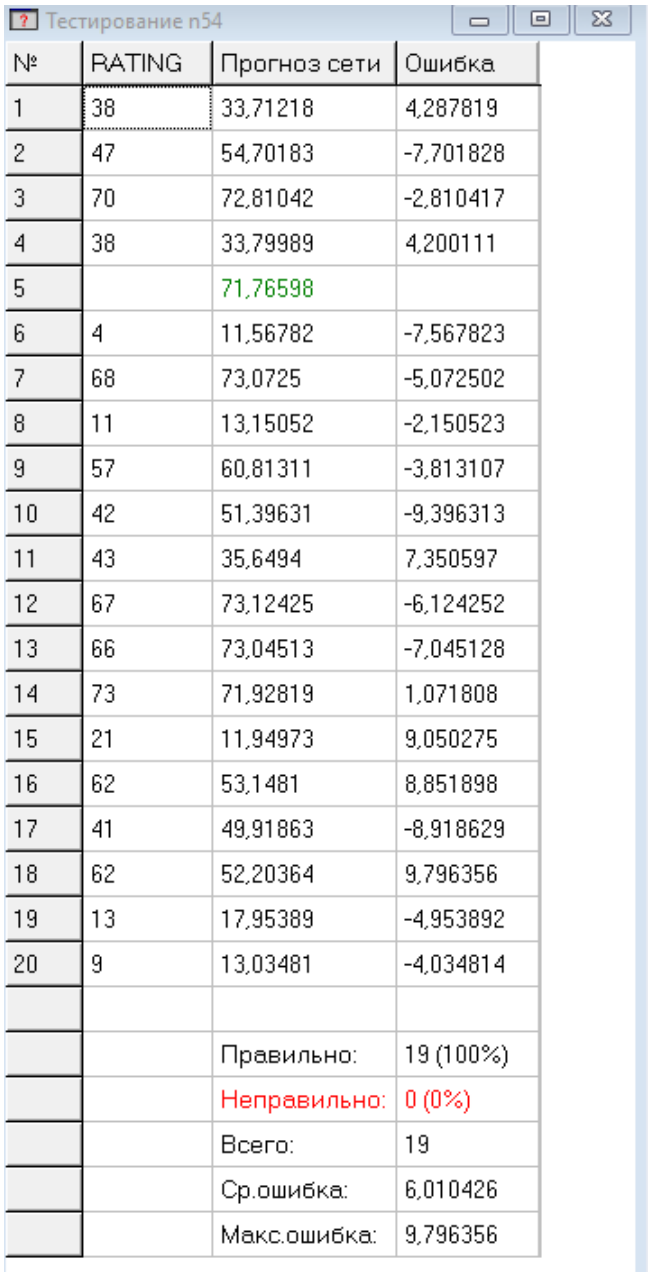


Рис. 13: Сокращение числа синапсов



№	RATING	Прогноз сети	Ошибка
1	38	33,71218	4,287819
2	47	54,70183	-7,701828
3	70	72,81042	-2,810417
4	38	33,79989	4,200111
5		71,76598	
6	4	11,56782	-7,567823
7	68	73,0725	-5,072502
8	11	13,15052	-2,150523
9	57	60,81311	-3,813107
10	42	51,39631	-9,396313
11	43	35,6494	7,350597
12	67	73,12425	-6,124252
13	66	73,04513	-7,045128
14	73	71,92819	1,071808
15	21	11,94973	9,050275
16	62	53,1481	8,851898
17	41	49,91863	-8,918629
18	62	52,20364	9,796356
19	13	17,95389	-4,953892
20	9	13,03481	-4,034814
		Правильно:	19 (100%)
		Неправильно:	0 (0%)
		Всего:	19
		Ср.ошибка:	6,010426
		Макс.ошибка:	9,796356

Рис. 14: Тестирование копии n5 - Сокращение числа синапсов

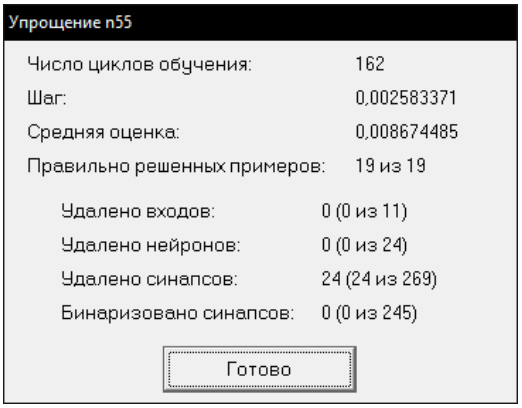


Рис. 15: Сокращение числа неоднородных входов

Тестирование n55

№	RATING	Прогноз сети	Ошибка
1	38	28,89656	9,103441
2	47	49,94372	-2,943722
3	70	62,23209	7,76791
4	38	34,95099	3,049011
5		47,38995	
6	4	13,56015	-9,560147
7	68	63,16115	4,838852
8	11	17,32219	-6,322189
9	57	60,07425	-3,074249
10	42	40,50418	1,495819
11	43	51,29609	-8,296085
12	67	61,08087	5,919128
13	66	57,66396	8,336044
14	73	65,1315	7,8685
15	21	15,12181	5,878193
16	62	63,50065	-1,500648
17	41	40,17327	0,8267326
18	62	52,38215	9,617851
19	13	13,70615	-0,7061491
20	9	14,01224	-5,012243
		Правильно:	19 (100%)
		Неправильно:	0 (0%)
		Всего:	19
		Ср.ошибка:	5,374574
		Макс.ошибка:	9,617851

Рис. 16: Тестирование копии n5 - Сокращение числа неоднородных входов

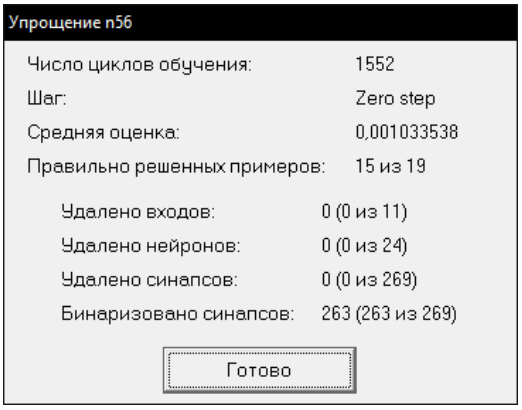
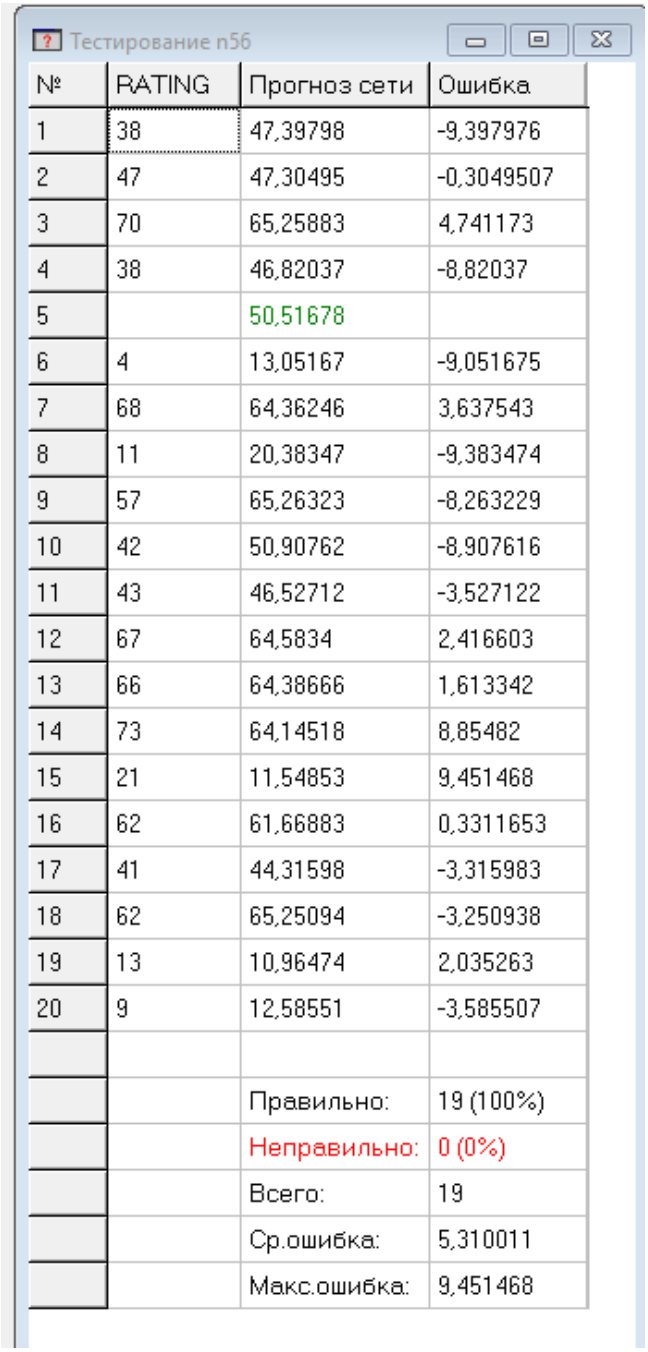


Рис. 17: Бинаризация весов синапсов и неоднородных входов



№	RATING	Прогноз сети	Ошибка
1	38	47,39798	-9,397976
2	47	47,30495	-0,3049507
3	70	65,25883	4,741173
4	38	46,82037	-8,82037
5		50,51678	
6	4	13,05167	-9,051675
7	68	64,36246	3,637543
8	11	20,38347	-9,383474
9	57	65,26323	-8,263229
10	42	50,90762	-8,907616
11	43	46,52712	-3,527122
12	67	64,5834	2,416603
13	66	64,38666	1,613342
14	73	64,14518	8,85482
15	21	11,54853	9,451468
16	62	61,66883	0,3311653
17	41	44,31598	-3,315983
18	62	65,25094	-3,250938
19	13	10,96474	2,035263
20	9	12,58551	-3,585507
		Правильно:	19 (100%)
		Неправильно:	0 (0%)
		Всего:	19
		Ср.ошибка:	5,310011
		Макс.ошибка:	9,451468

Рис. 18: Тестирование копии n5 - Бинаризация весов синапсов и неоднородных входов

2.3 Методы оптимизации

Метод	Число циклов обучения	Макс.Ошибка	Сред.Ошибка	Прогноз	Ошибка прогноза
Градиентный спуск	575	27,56	10,47	69,56	21,56
Модифицированный Par Tan	120	20,97	6,68	66,86	18,86
Сопряженные градиенты	29	14,76	14,76	63,18	15,18
BFGS	31	11,88	6,43	67,27	19,27

Рис. 19: Методы оптимизации

Наилучшей оптимизацией оказался метод BFGS, имеющий более точный прогноз и наименьшие ошибки.

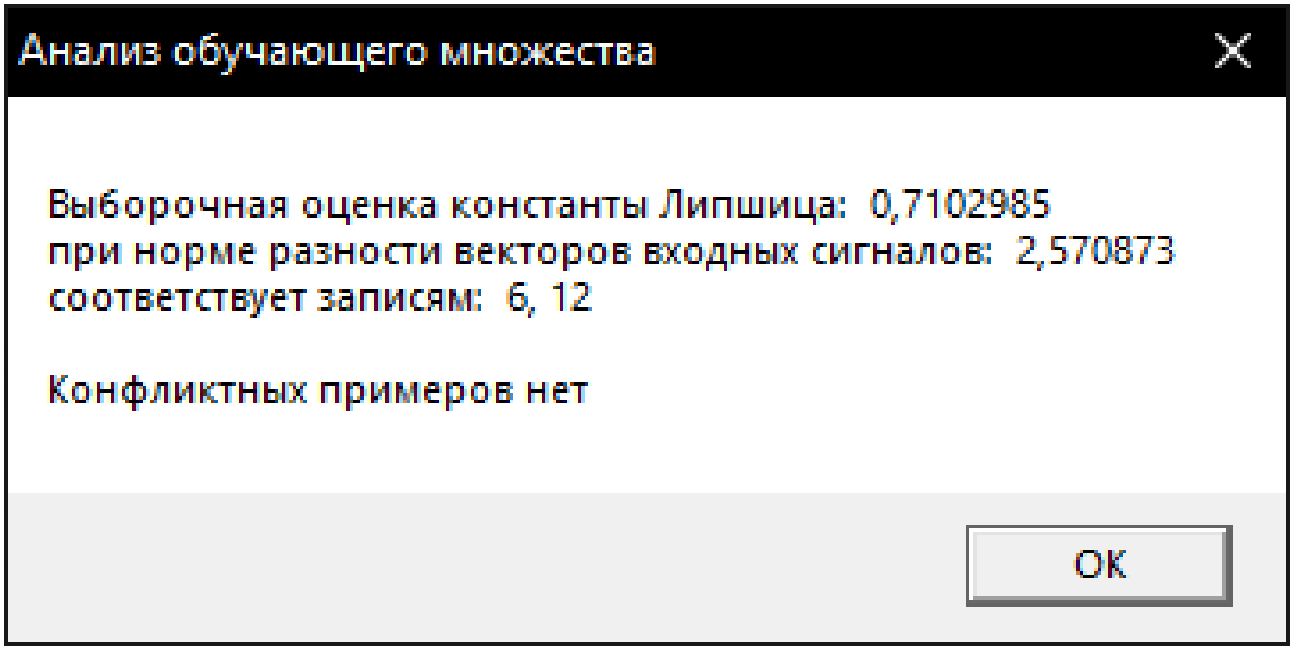
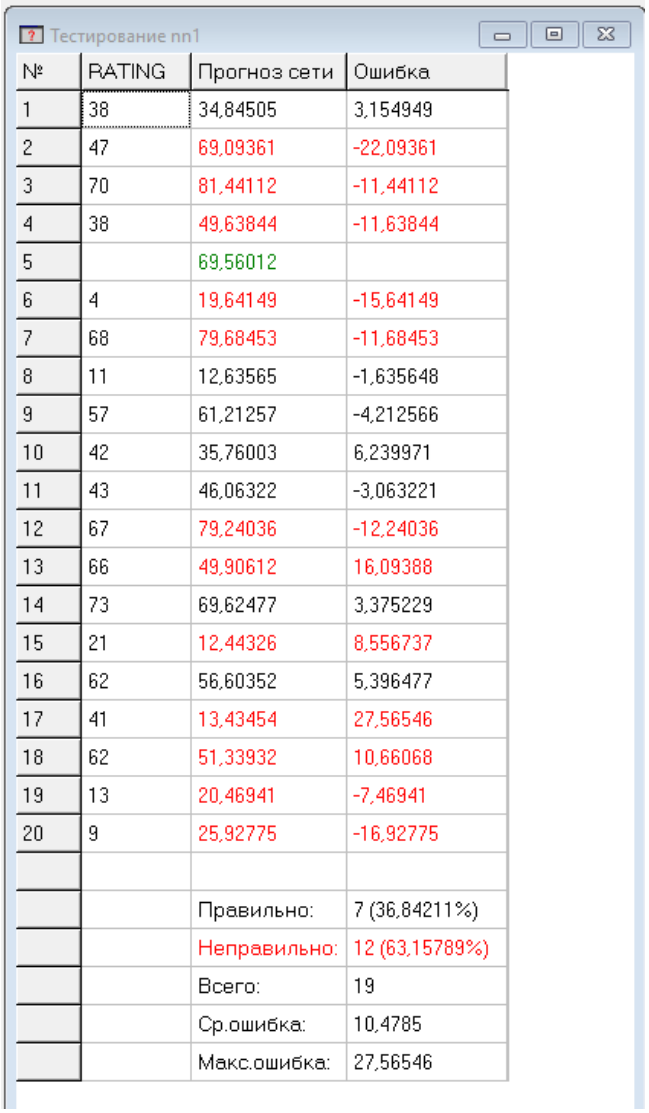


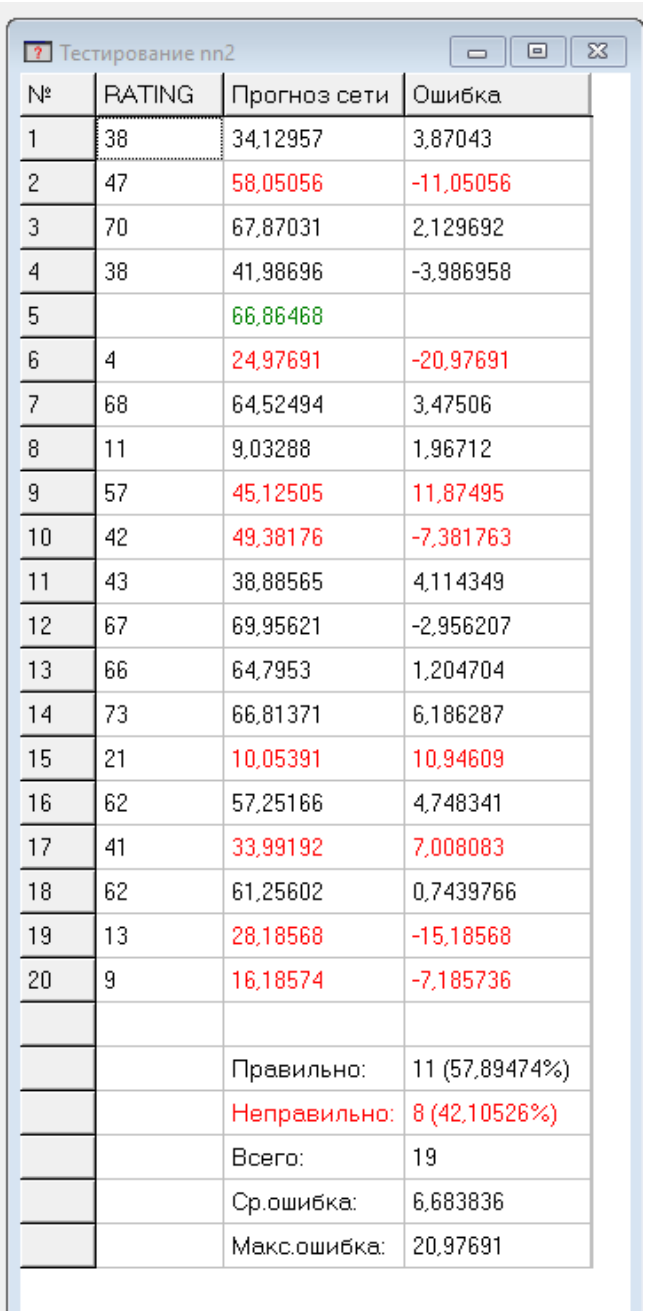
Рис. 20: Анализ обучающего множества

Результаты тестирования при различных метода оптимизации представлены далее.



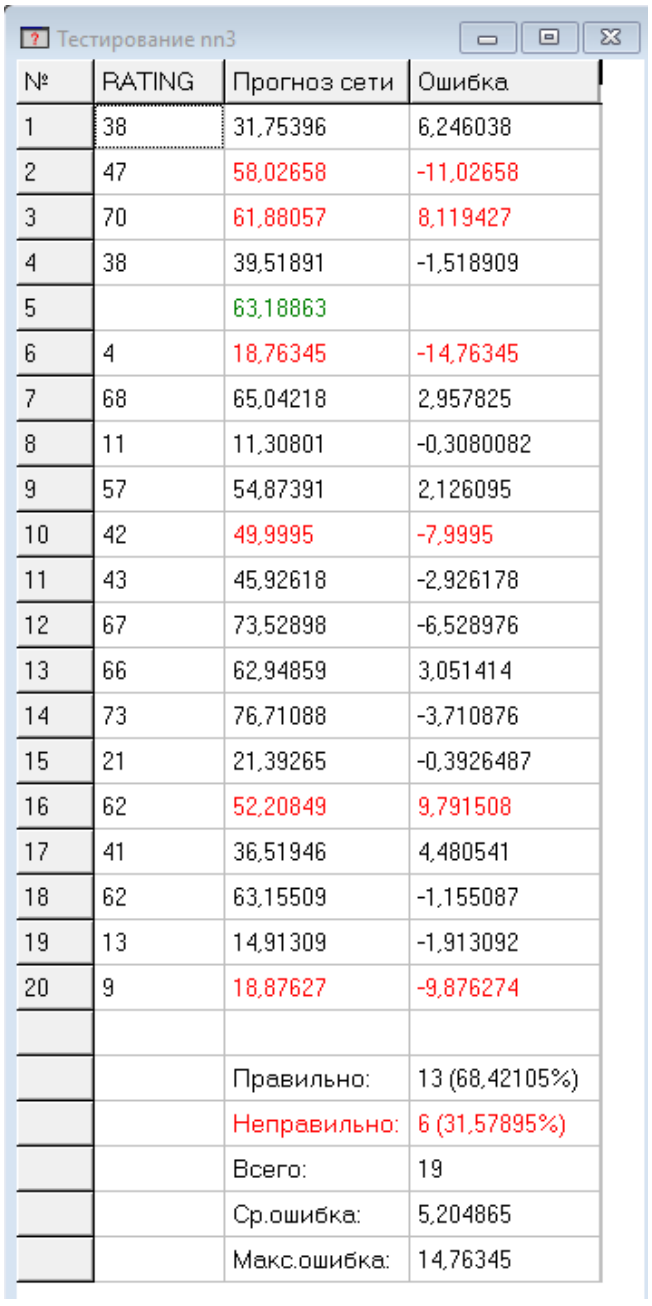
№	RATING	Прогноз сети	Ошибка
1	38	34,84505	3,154949
2	47	69,09361	-22,09361
3	70	81,44112	-11,44112
4	38	49,63844	-11,63844
5		69,56012	
6	4	19,64149	-15,64149
7	68	79,68453	-11,68453
8	11	12,63565	-1,635648
9	57	61,21257	-4,212566
10	42	35,76003	6,239971
11	43	46,06322	-3,063221
12	67	79,24036	-12,24036
13	66	49,90612	16,09388
14	73	69,62477	3,375229
15	21	12,44326	8,556737
16	62	56,60352	5,396477
17	41	13,43454	27,56546
18	62	51,33932	10,66068
19	13	20,46941	-7,46941
20	9	25,92775	-16,92775
		Правильно:	7 (36,84211%)
		Неправильно:	12 (63,15789%)
		Всего:	19
		Ср.ошибка:	10,4785
		Макс.ошибка:	27,56546

Рис. 21: Тестирование при оптимизации методом градиентного спуска



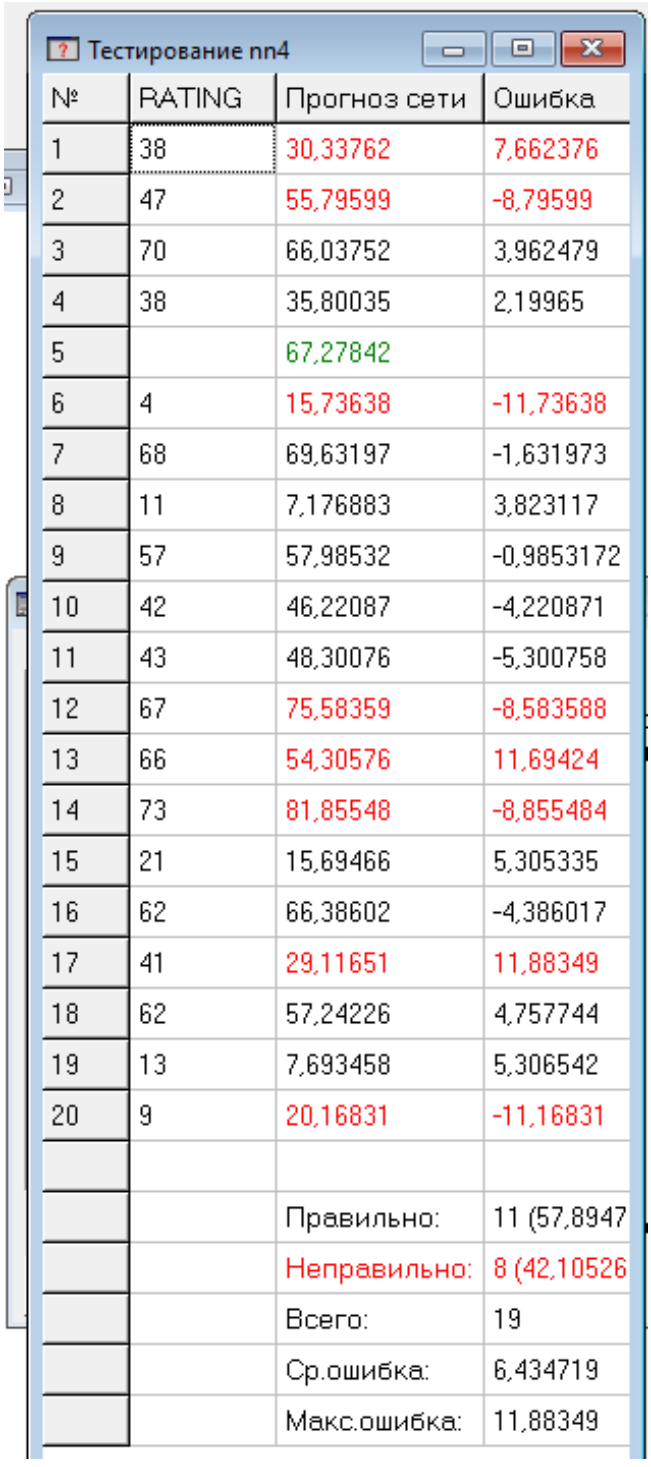
№	RATING	Прогноз сети	Ошибка
1	38	34,12957	3,87043
2	47	58,05056	-11,05056
3	70	67,87031	2,129692
4	38	41,98696	-3,986958
5		66,86468	
6	4	24,97691	-20,97691
7	68	64,52494	3,47506
8	11	9,03288	1,96712
9	57	45,12505	11,87495
10	42	49,38176	-7,381763
11	43	38,88565	4,114349
12	67	69,95621	-2,956207
13	66	64,7953	1,204704
14	73	66,81371	6,186287
15	21	10,05391	10,94609
16	62	57,25166	4,748341
17	41	33,99192	7,008083
18	62	61,25602	0,7439766
19	13	28,18568	-15,18568
20	9	16,18574	-7,185736
		Правильно:	11 (57,89474%)
		Неправильно:	8 (42,10526%)
		Всего:	19
		Ср.ошибка:	6,683836
		Макс.ошибка:	20,97691

Рис. 22: Тестирование при оптимизации методом модифициров анный Par Tan



№	RATING	Прогноз сети	Ошибка
1	38	31,75396	6,246038
2	47	58,02658	-11,02658
3	70	61,88057	8,119427
4	38	39,51891	-1,518909
5		63,18863	
6	4	18,76345	-14,76345
7	68	65,04218	2,957825
8	11	11,30801	-0,3080082
9	57	54,87391	2,126095
10	42	49,9995	-7,9995
11	43	45,92618	-2,926178
12	67	73,52898	-6,528976
13	66	62,94859	3,051414
14	73	76,71088	-3,710876
15	21	21,39265	-0,3926487
16	62	52,20849	9,791508
17	41	36,51946	4,480541
18	62	63,15509	-1,155087
19	13	14,91309	-1,913092
20	9	18,87627	-9,876274
		Правильно:	13 (68,42105%)
		Неправильно:	6 (31,57895%)
		Всего:	19
		Ср.ошибка:	5,204865
		Макс.ошибка:	14,76345

Рис. 23: Тестирование при оптимизации методом сопряженные градиенты



№	RATING	Прогноз сети	Ошибка
1	38	30,33762	7,662376
2	47	55,79599	-8,79599
3	70	66,03752	3,962479
4	38	35,80035	2,19965
5		67,27842	
6	4	15,73638	-11,73638
7	68	69,63197	-1,631973
8	11	7,176883	3,823117
9	57	57,98532	-0,9853172
10	42	46,22087	-4,220871
11	43	48,30076	-5,300758
12	67	75,58359	-8,583588
13	66	54,30576	11,69424
14	73	81,85548	-8,855484
15	21	15,69466	5,305335
16	62	66,38602	-4,386017
17	41	29,11651	11,88349
18	62	57,24226	4,757744
19	13	7,693458	5,306542
20	9	20,16831	-11,16831
		Правильно:	11 (57,8947
		Неправильно:	8 (42,10526
		Всего:	19
		Ср.ошибка:	6,434719
		Макс.ошибка:	11,88349

Рис. 24: Тестирование при оптимизации методом BFGS

3 Выводы по заданию NeuroPro

В рамках данной лабораторной работы были освоены нейросетевые технологии для решения задач классификации и прогнозирования с помощью программы NeuroPro 0.25.

На подготовительном этапе был изучен интерфейс программного обеспечения и приобретены навыки загрузки и обработки данных. Ключевым результатом стала самостоятельная разработка и формирование обучающей выборки в формате DBF.

Практическая значимость работы заключается в успешном прохождении полного цикла создания прогнозной модели — от сбора и подготовки данных до конфигурирования архитектуры сети, её обучения, оптимизации и анализа полученных результатов. В итоге были проведены сравнения полученных нейронных сетей и выбраны сети, которые наиболее точно предсказывают результат.

В результате лабораторной работы были получено базовое представление о простых нейронных сетях и этапах их обучения. Полученные знания пригодятся при дальнейшем изучении перемета теория принятия решений.

4 Цель работы - часть 2 - Python

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

5 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.

6 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.

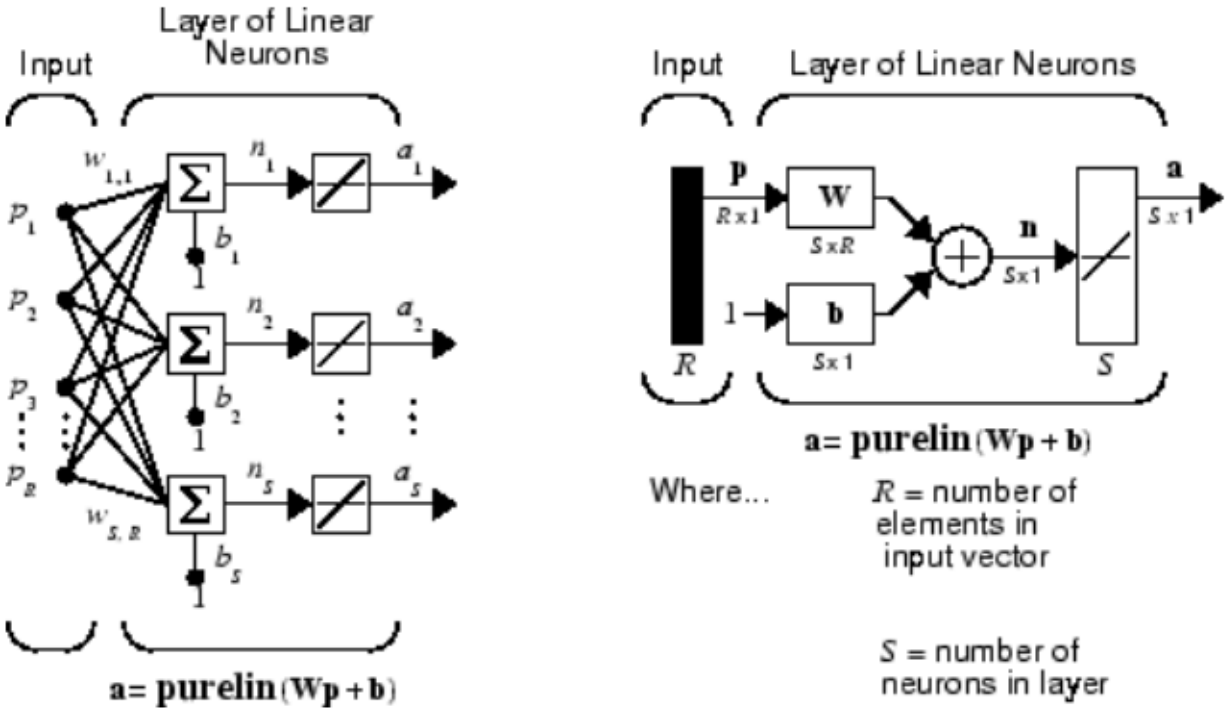


Рис. 25: Структура линейного нейрона

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.

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.

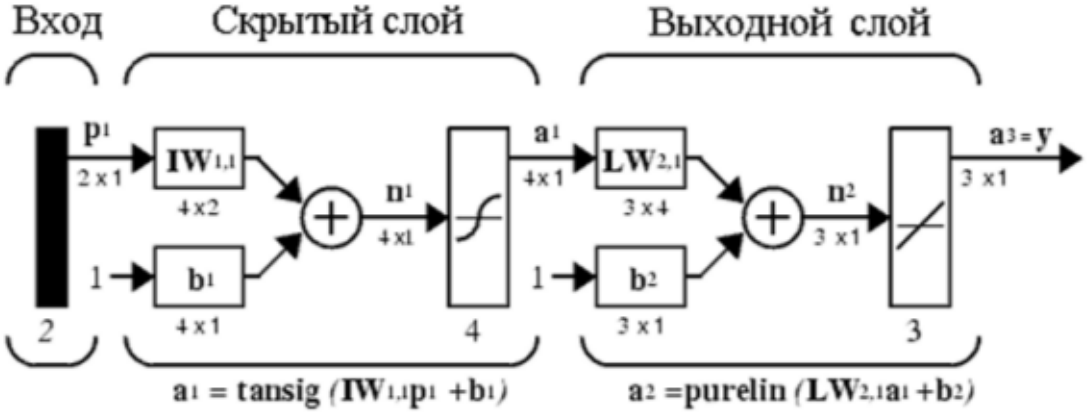


Рис. 26: Укрупненная структура FNN- сети

7 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
)
```

7.1 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)
```

7.2 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}')


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

```

7.3 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. 27: 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. 28: Экранные формы линейной нейронной сети

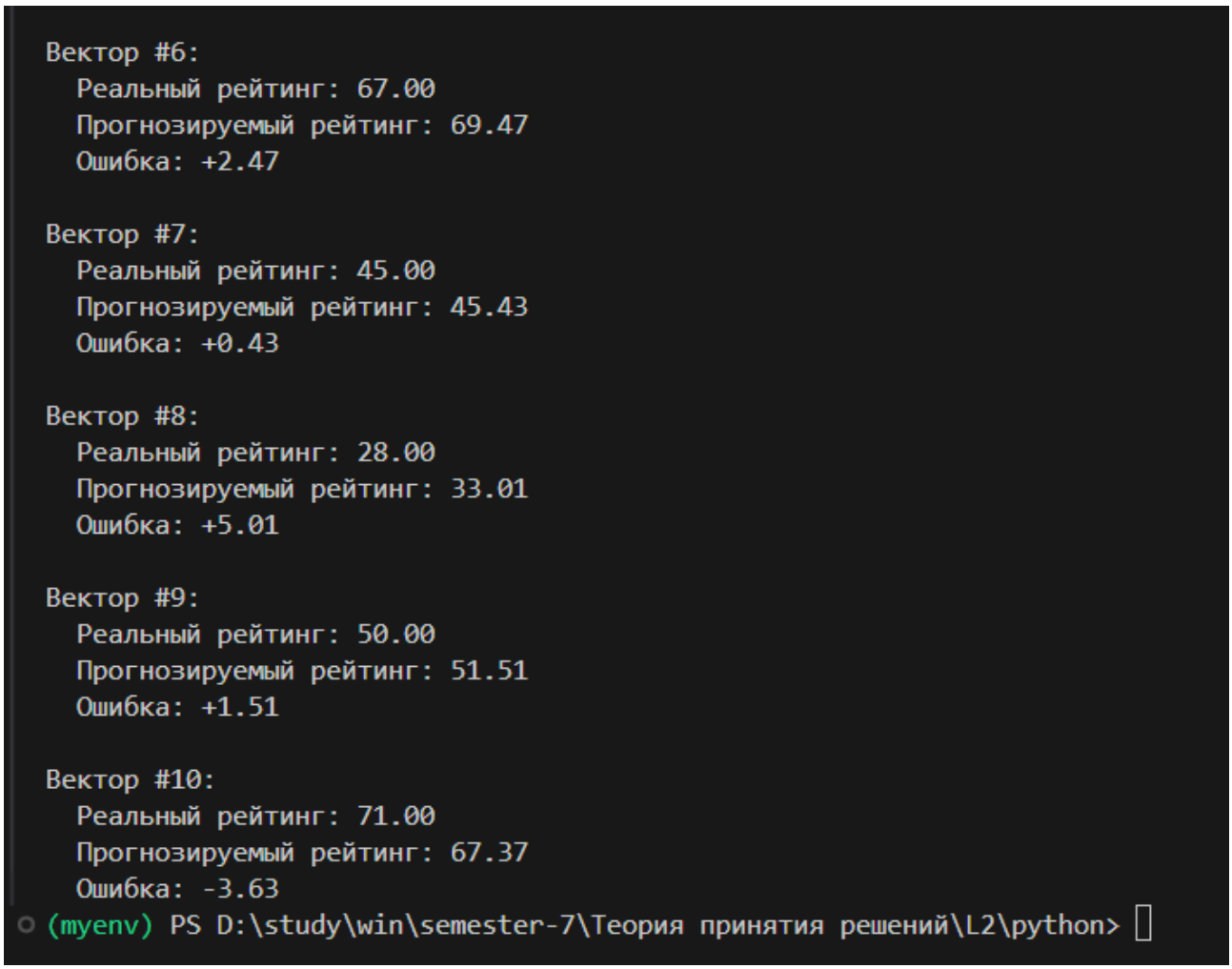


Рис. 29: Экранные формы линейной нейронной сети

7.4 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. 30: Таблица для FNN сети

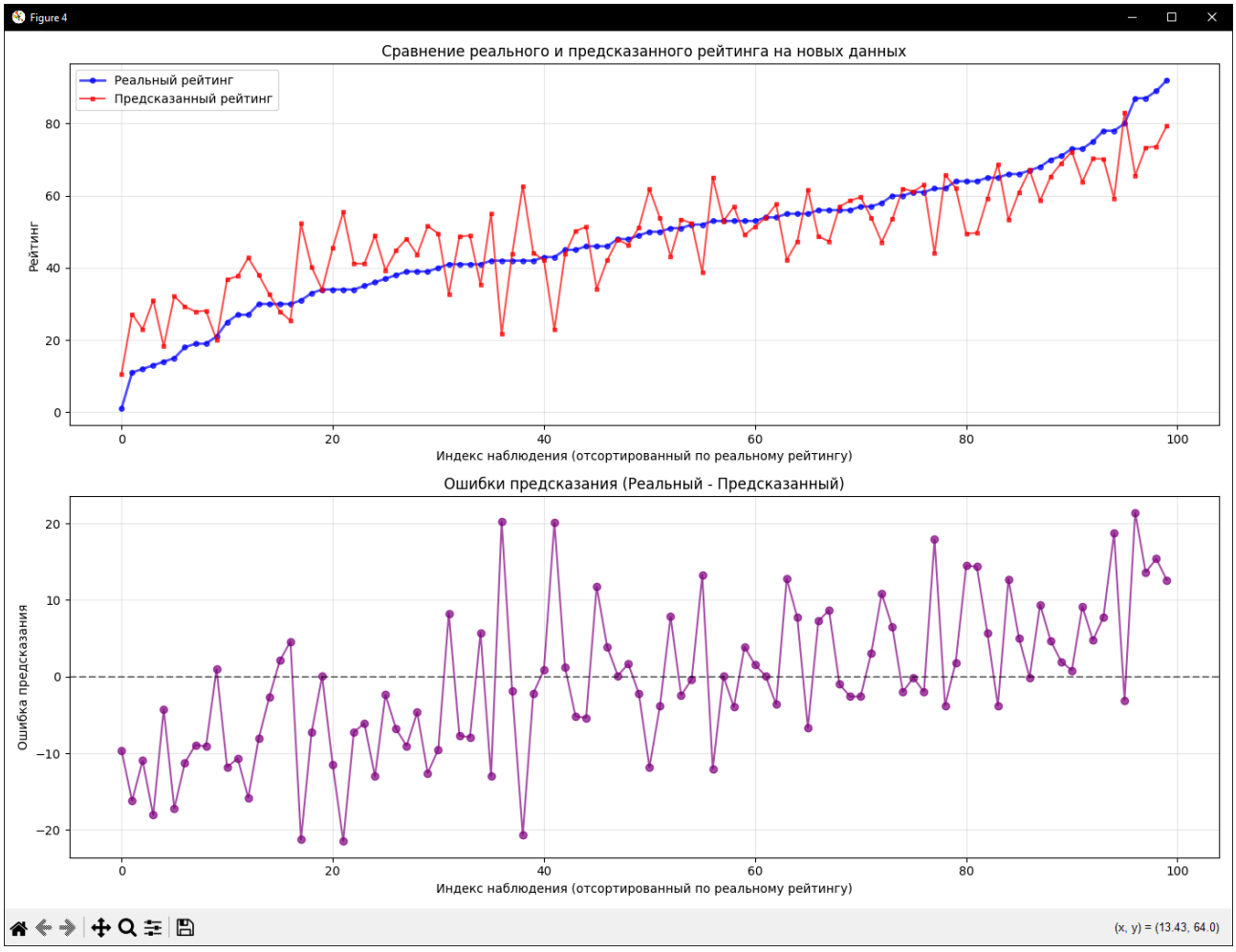


Рис. 31: Экранные формы нейронной сети прямого распространения - 20 векторов, 64 нейрона

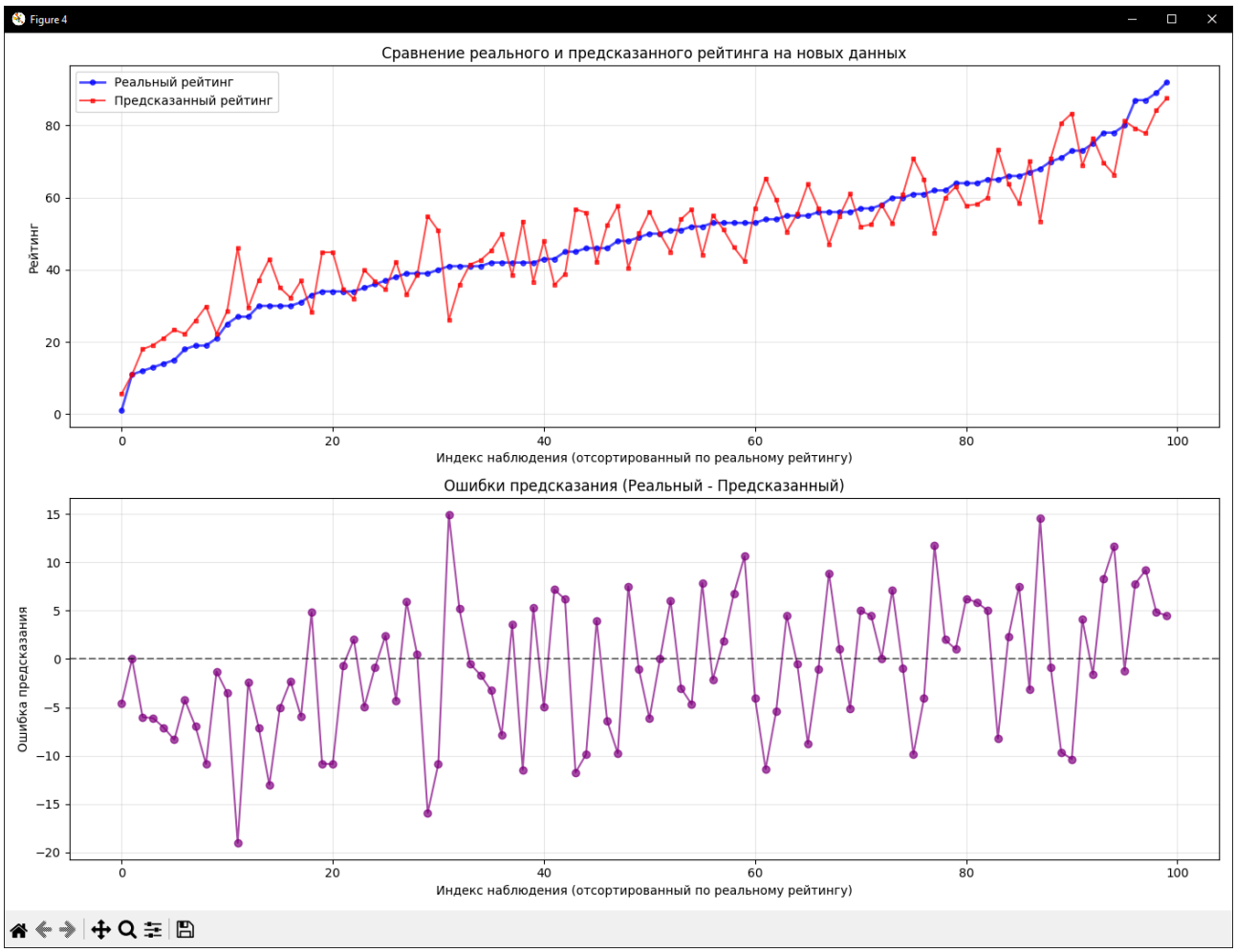


Рис. 32: Экранные формы нейронной сети прямого распространения - 50 векторов, 64 нейрона

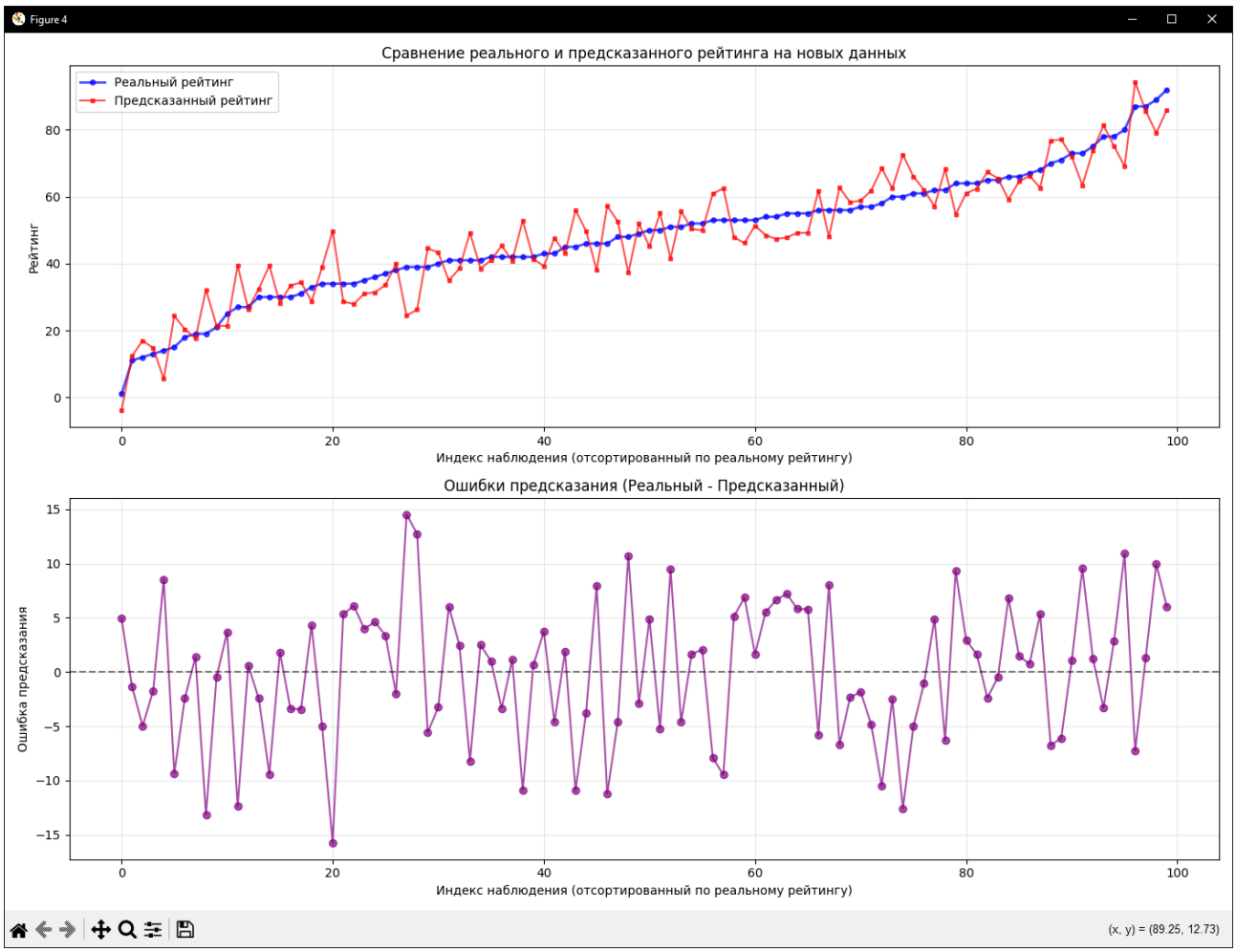


Рис. 33: Экранные формы нейронной сети прямого распространения - 100 векторов, 64 нейрона

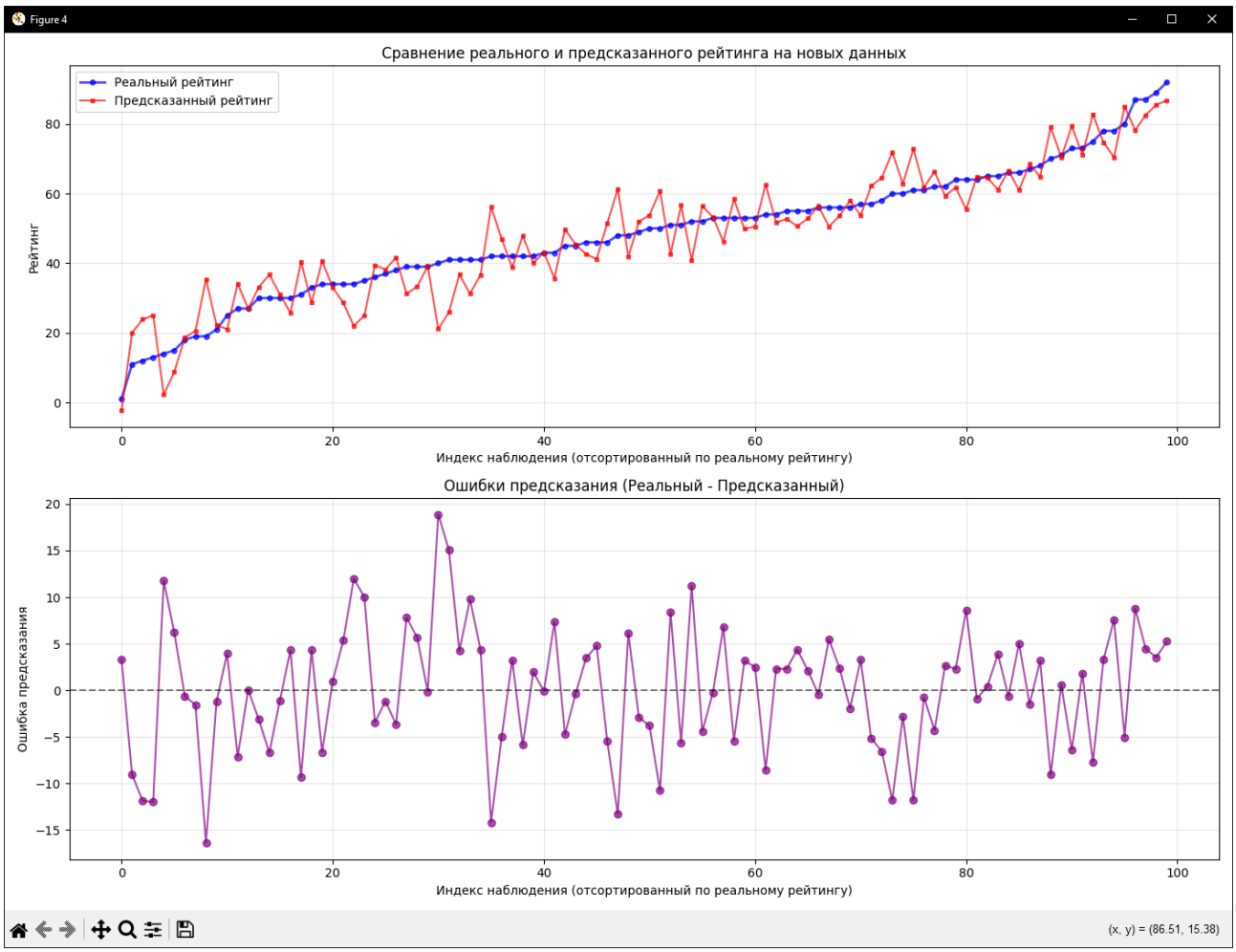


Рис. 34: Экранные формы нейронной сети прямого распространения - 500 векторов, 64 нейрона

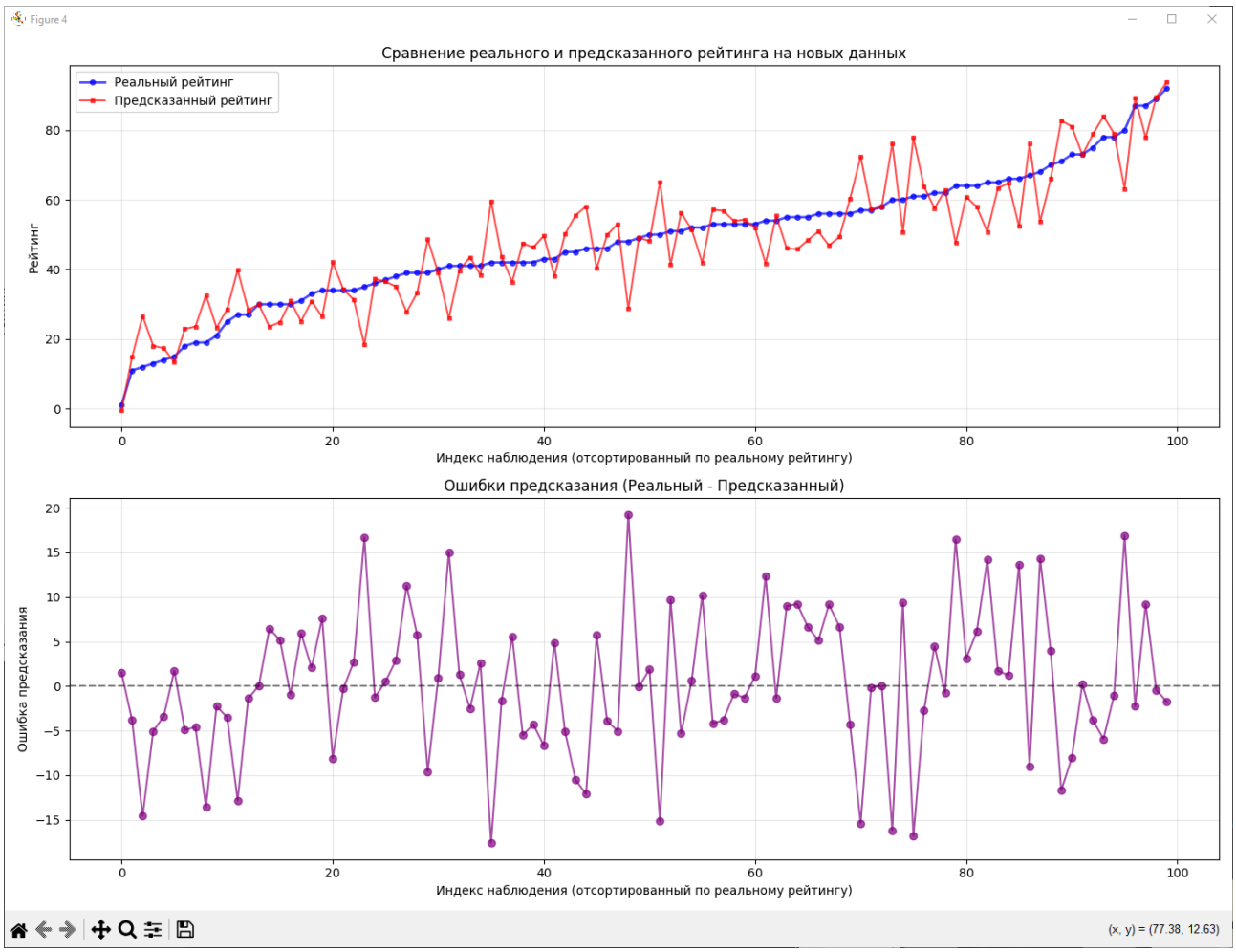


Рис. 35: Экранные формы нейронной сети прямого распространения - 1000 векторов, 64 нейрона

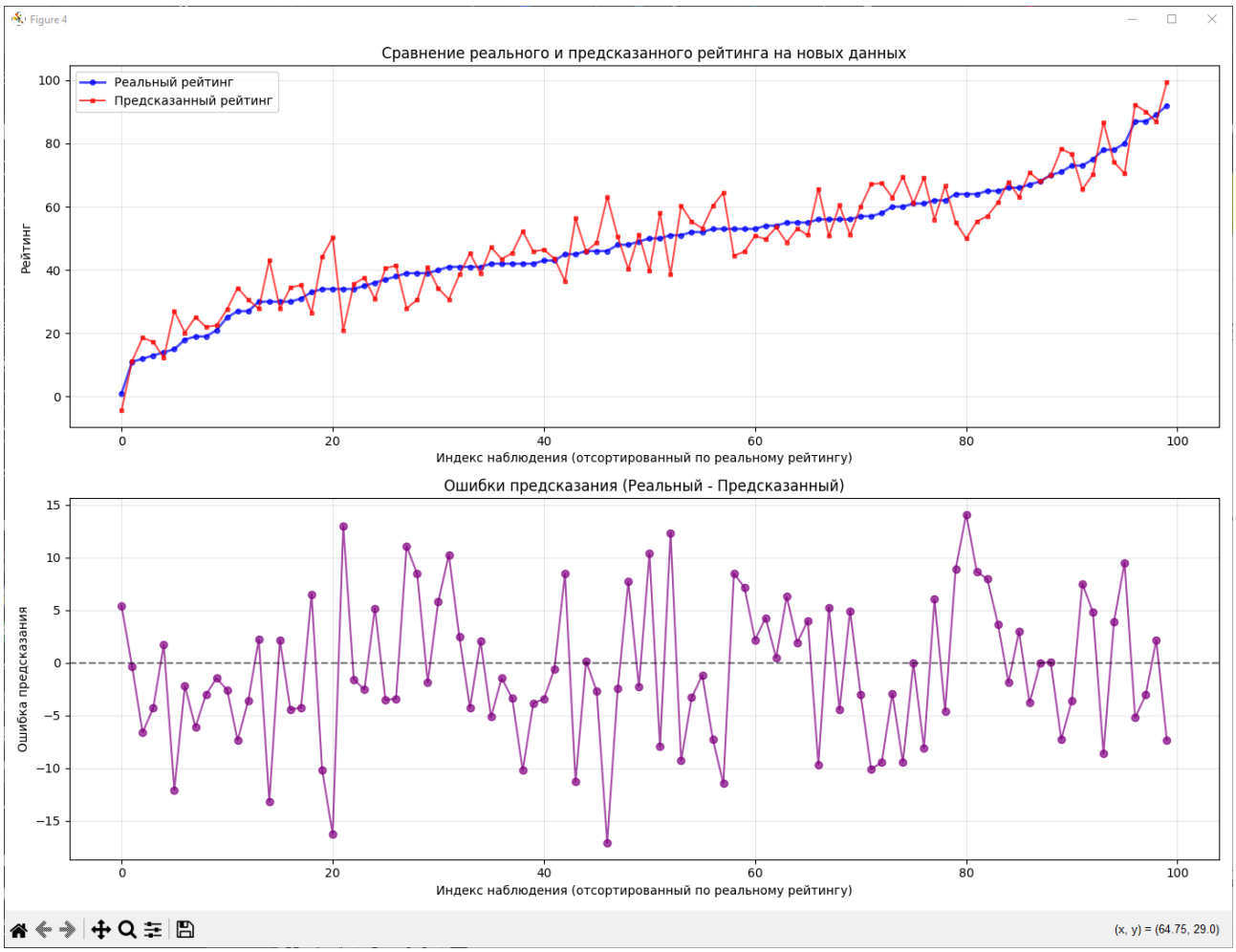


Рис. 36: Экранные формы нейронной сети прямого распространения - 100 векторов, 32 нейрона

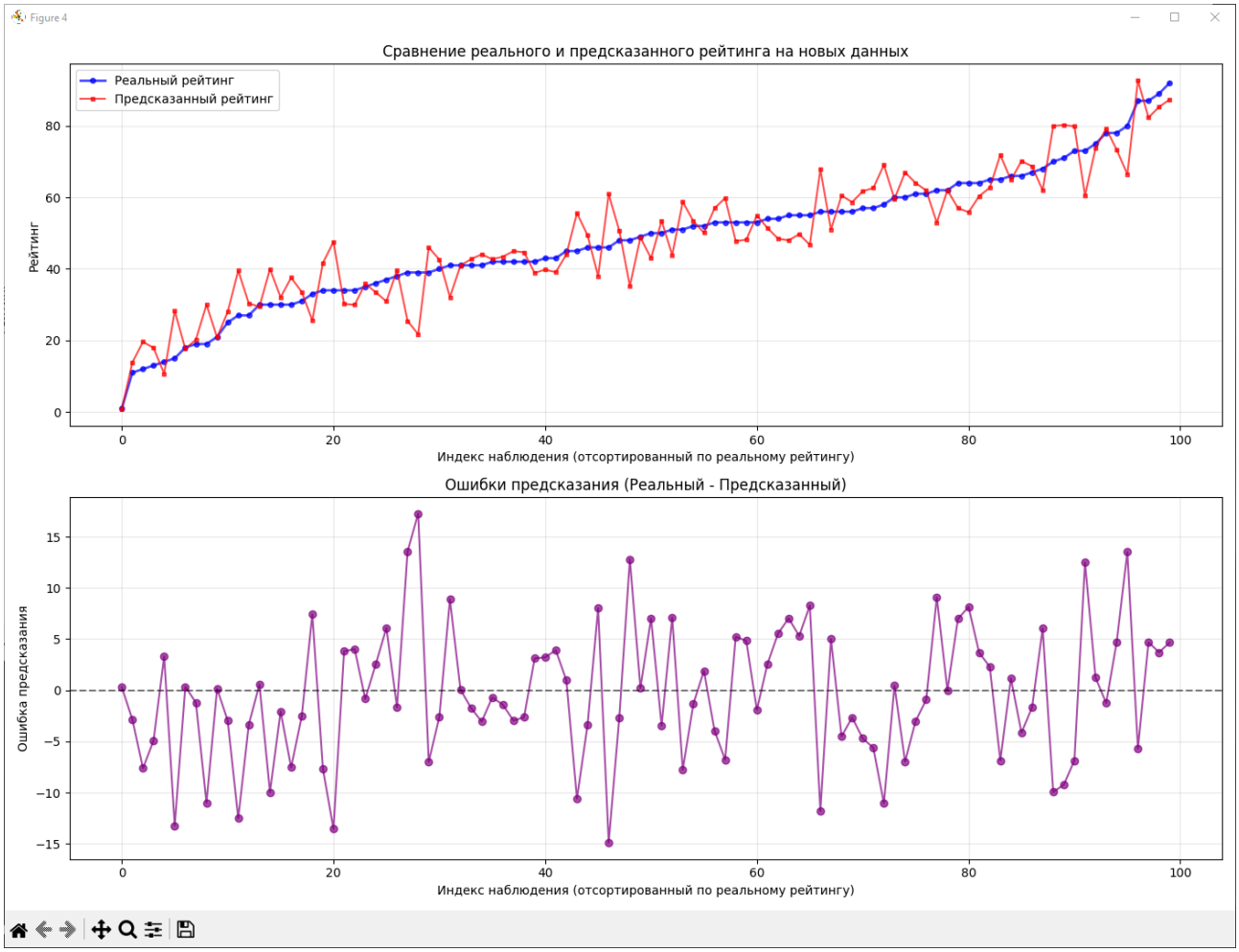


Рис. 37: Экранные формы нейронной сети прямого распространения - 100 векторов, 128 нейрона

8 Выводы по заданию Python

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.