Министерство науки и высшего образования РФ

федеральное государственное бюджетное образовательное учреждение высшего образования «Югорский государственный университет»

УДК: 004.67

Рег. номер: АААА-А20-120020390011-7

УТВЕРЖДАЮ:

Проректор по научной работе

_____ Исламутдинов В.Ф.

«_____» _____ 2020.

ОТЧЁТ
О НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

«Разработка технологии прогнозирования динамики социально-экономического развития региона»

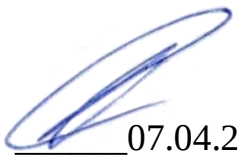
(заключительный)

Руководитель НИР:

в.н.с. ИЦЭ, к.э.н. _____ Татьянкин В.М.

г. Ханты-Мансийск 2020

СПИСОК ИСПОЛНИТЕЛЕЙ

Руководитель НИР,
ВНС института цифровой экономики
кандидат технических наук  _____07.04.2020 В.М. Татьянкин
подпись, дата (Глава 1-3)

Исполнители:

Преподаватель института
цифровой экономики

_____07.04.2020 А.В. Шицелов
подпись, дата (Приложение 1)

Нормоконтроль _____07.04.2020 Д.Р. Гареева
подпись, дата

РЕФЕРАТ

Отчёт 3 гл., 81 с., 24 рис., 1 схем., 1 прил.

Ключевые слова: нейронные сети, прогнозирование, математические модели, регрессионный анализ, социально-экономическая система.

Объект НИР: динамика параметров социальной и экономической системы региона.

Цель НИР: повышение качества прогнозирования на региональном и муниципальном уровне.

Методология НИР: используются современные способы и технологии математического моделирования, теории вероятности, системного анализа, аппарата искусственных нейронных сетей и разработки автоматизированных информационных систем.

Результаты НИР: в рамках научно-исследовательской работы разработана технология прогнозирования динамики социально-экономического развития региона. Технология обеспечивает анализ развития параметров, характеризующих регион РФ, с горизонтом прогнозирования до 5 лет и более. Технология состоит из информационной модели процесса прогнозирования социально-экономического состояния региона с точки зрения Пользователя, математических моделей, алгоритмов и программного продукта, реализованного в виде подключаемых библиотек.

Область применения:

1. Справочно-информационное (для заинтересованных лиц, в том числе, жителей и работодателей региона).
2. В подведомственных учреждениях региональных органов власти.
3. Управление в социально-экономических системах на региональном и муниципальном уровне.

Итоги внедрения результатов НИР: результаты внедрены на портале https://rms.admhmao.ru.

Значимость НИР: переход к цифровому и интеллектуальному управлению в области регионального и муниципального управления.

Прогнозные предположения о развитии объекта исследования: увеличение количества социально-экономических параметров для прогнозирования, совершенствование математических моделей и способов прогнозирования. Расширение географии предметной области.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 8

ГЛАВА 1. ИНФОРМАЦИОННАЯ МОДЕЛЬ «МОДУЛЯ ПРОГНОЗИРОВАНИЯ ДИНАМИКИ СОЦИАЛЬНО-ЭКОНОМИЧЕСКОГО РАЗВИТИЯ РЕГИОНА» 9

ГЛАВА 2. ТЕХНОЛОГИИ ПРОГНОЗИРОВАНИЯ И МОДЕЛИРОВАНИЯ ДИНАМИКИ СОЦИАЛЬНО-ЭКОНОМИЧЕСКОГО РАЗВИТИЯ РЕГИОНА 23

ГЛАВА 3. АПРОБАЦИЯ РАЗРАБОТАННОЙ ТЕХНОЛОГИИ ПРОГНОЗИРОВАНИЯ ДИНАМИКИ СОЦИАЛЬНО-ЭКОНОМИЧЕСКОГО РАЗВИТИЯ РЕГИОНА 33

ЗАКЛЮЧЕНИЕ 51

Приложение 1. Программная реализация разработанных модулей 53

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

В настоящем отчете о НИР применяются следующие термины с соответствующими определениями:

Персептрон – математическая или компьютерная модель восприятия информации мозгом.

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

В настоящем отчете о НИР применяют следующие сокращения и обозначения:

ХМАО-Югра – Ханты-Мансийский автономный округ-Югра.

ИНС – Искусственная нейронная сеть.

НСГД – Нейронная сеть глубокого доверия.

СНС – Сверточные нейронные сети.

МНС – Многослойная нейронная сеть.

ВВЕДЕНИЕ

На сегодняшний день отсутствует технология прогнозирования динамики социально-экономического развития региона, которая позволит специалисту в области государственного и муниципального управления моделировать долгосрочное развитие региона. Основная причина отсутствия данной технологии заключается в том, что для прогнозирования сложно формализованных процессов в социально-экономической системе используются искусственные нейронные сети, которые не имеют унифицированного подхода к обучению. Новизна работы заключается в разработке алгоритмов автоматизированного обучения нейронных сетей для моделирования динамики социально-экономических показателей региона.

Данная работа является продолжением диссертационного исследования «Способы и алгоритмы управления процессами кадрового обеспечения региона» по специальности 05.13.10 и научно-исследовательской работы № 0187200001717000446 от 14 июля 2017 г. по теме «Проведение исследований по оценке государственной миграционной политики в Ханты-Мансийском автономном округе – Югре с разработкой миграционной модели Ханты-Мансийского автономного округа – Югры».

Актуальность работы заключается в ежегодной потребности составления программы социального и экономического развития региона и муниципальных образований. При этом существующие подходы уже не удовлетворяют Заказчиков на предмет адекватности и достоверности полученных результатов моделирования и прогнозирования.

Обоснованием необходимости проведения НИР является договор № 05.5/19-ЮГУ-258 от 22.10.2019 «Разработка технологии прогнозирования динамики социально-экономического развития региона».

Данная технология составит основу системы прогнозирования, анализа, оповещения, процессного управления, сбора и обработки геокодированной информации из информационных систем ХМАО – Югры.

ГЛАВА 1. ИНФОРМАЦИОННАЯ МОДЕЛЬ «МОДУЛЯ ПРОГНОЗИРОВАНИЯ ДИНАМИКИ СОЦИАЛЬНО-ЭКОНОМИЧЕСКОГО РАЗВИТИЯ РЕГИОНА»

Модуль прогнозирования динамики социально-экономического развития региона обеспечит анализ динамики параметров, характеризующий автономный округ, с горизонтом прогнозирования до N[1] лет, а именно:

1. Тридцати семи параметров, которые поделены на три группы:

1.1. Существует строгая математическая модель для моделирования и прогнозирования параметра.

1.2. Без использования строгой модели, но при этом известны влияющие параметры на исследуемый параметр.

1.3. Параметры прогнозируются с использованием экспертного мнения или метода скользящего окна.

2. Произвольного параметра, который описывается однофакторной моделью на основе своей же существующей динамикой временного ряда

3. Произвольного параметра, который описывается многофакторной моделью. Среди факторов рассматриваются иные параметры, которые Пользователь посчитает нужным использовать для моделирования и прогнозирования[2].

Нейронные сети и регрессионный анализ будут инструментами прогнозирования и моделирования параметров, указанных в пункте 2-3.

Информационная модель процесса прогнозирования социально-экономического состояния региона с точки зрения Пользователя, представлена на схеме 1. Предложенная информационная модель включает одиннадцать этапов:

[1] Параметр N определяется пользователям и не ограничен сверху. Единственное условие — это существование начальных данных для прогнозирования/моделирования выбранного параметра.

[2] Далее под словом прогнозирование понимается моделирование и прогнозирование.

1. На первом этапе Пользователь выбирает показатель и временной интервал прогнозирования.

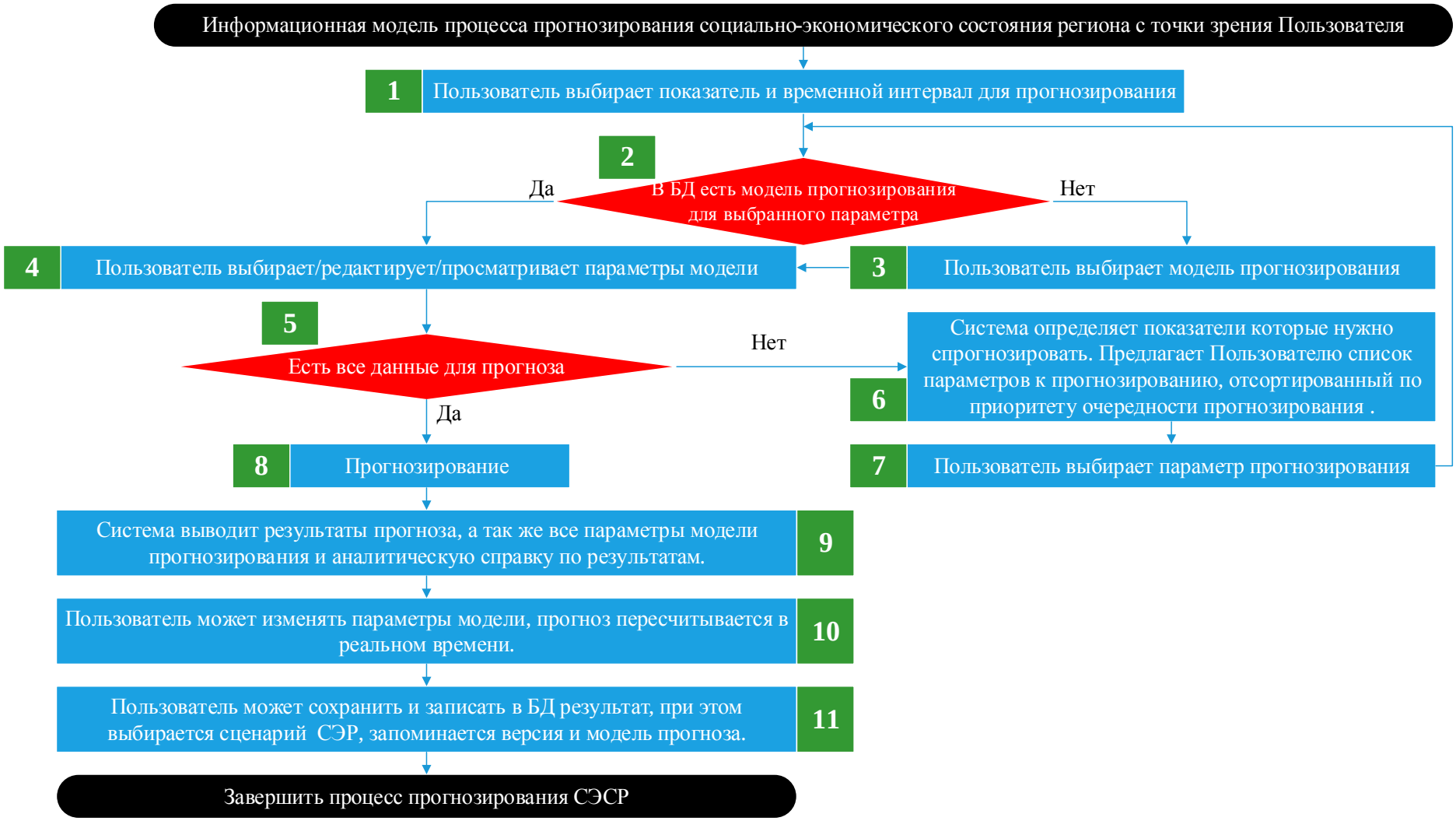


Схема 1. Информационная модель прогнозирования социально-экономического состояния региона с точки зрения Пользователя

2. На втором этапе модуль прогнозирования проверяет у выбранного показателя наличие математической модели, которая описывает его. Математическая модель может быть как строгая, так и произвольная, которая была уже выбрана предыдущим Пользователем. В случае успеха, переходим на 4 шаг, в противном случае переходим на 3 шаг.

3. На третьем этапе Пользователь выбирает модель прогнозирования, а именно:

- выбирает однофакторную модель или многофакторную
- выбирает факторы, влияющие на исследуемый параметр
- выбирает инструмент прогнозирования: нейронные сети и регрессионный анализ
- определяет настройки выбранного инструмента прогнозирования, самостоятельно или выбирает функцию автоматической настройки,

и переходит к 4 шагу.

4. На четвертом этапе Пользователь просматривает/редактирует параметры модели для прогнозирования

5. На пятом этапе происходит проверка на наличие требуемых всех данных для прогнозирования. В случае успеха переходим на 8 шаг, в противном случае переходим на 6 шаг.

6. На шестом этапе определяются входные данные выбранной математической модели, которые отсутствуют в базе данных на выбранный период прогнозирования. Далее они сортируются по приоритету первоочередного прогнозирования.

7. На седьмом этапе Пользователь выбирает из сформированного списка параметр для прогнозирования и переходит ко 2 шагу. Рекомендуется выбирать первый.

8. На восьмом этапе происходит процесс прогнозирования выбранного параметра.

9. На девятом этапе выводятся результаты прогноза, а также все параметры модели прогнозирования и аналитическая справка по результатам.

10. На десятом этапе Пользователю предлагается изменить параметры модели, прогноз пересчитывается в реальном времени.

11. На одиннадцатом этапе Пользователю предлагается сохранить и записать в базу данных результаты прогнозирования, при этом выбирается сценарий социально-экономического развития, запоминается версия и модель прогноза.

Строгие математические модели предложены для следующих параметров:

1. Численность населения. Математическая модель для прогнозирования численности населения представлена в выражении 1.

$$P_f_i = \sum_{j=1}^{150} P_f_a_y_m_{ij} + \sum_{j=1}^{150} P_f_a_y_w_{ij}, \quad (1)$$

где P_f_i – численность населения в году *i*, *i=1...N* – период прогнозирования, $P_f_a_y_m_{ij}$ – численность мужчин возрастом *j* в году *i*, $P_f_a_y_w_{ij}$ – численность женщин возрастом *j* в году *i*.

2. Численность населения по полу и возрасту. Математическая модель для прогнозирования численности населения по полу и возрасту представлена в выражениях 2-3.

$$P_f_g_a_y_{ij} = P_f_g_a_y_{i-1j-1} - D_f_g_a_y_{i-1j-1} + M_f_g_a_y_{ij}, \quad (2)$$

$$B_f_g_a_y_{i1} = \left|\sum_{j=2}^{150} P_f_a_y_w_{ij} * f_p_b_y_{ij}\right|, \quad (3)$$

где $P_f_g_a_y_{ij}$ – численность населения возрастом *j* в году *i*, $D_f_g_a_y_{ij}$ – численность умерших возрастом *j* в году *i*, $M_f_g_a_y_{ij}$ – численность миграции возрастом *j* в году *i*, $B_f_g_a_y_{i1}$ – численность рождённых в году *i*, $P_f_a_y_w_{ij}$ – численность женщин возрастом *j* в году *i*, $f_p_b_y_{ij}$ – прогноз вероятности родить женщины возрастом *j* в году *i*. С помощью

выражений (2-3) отдельно прогнозируется численность мужчин и отдельно численность женщин.

3. Число родившихся по полу. Математическая модель для прогнозирования родившихся по полу представлена в выражении 3.

4. Численность женщин по возрасту старше 18 лет. Математическая модель для прогнозирования численности женщин представлена в выражении 4.

$$W_o_18_i = \sum_{j=20}^{150} P_f_a_y_w_{ij}, \qquad (4)$$

где $W_o_18_i$ – численность женщин старше 18 лет в году *i*, $P_f_a_y_w_{ij}$ – численность женщин возрастом *j* в году *i*.

5. Число умерших по полу и возрасту. Математическая модель для прогнозирования численности умерших по полу и возрасту представлена в выражении 5.

$$D_f_g_a_y_{ij} = \left|\sum_{j=1}^{150} P_f_g_a_y_{ij} * f_p_d_y_{ij}\right|, \qquad (5)$$

где $D_f_g_a_y_{ij}$ – численность умерших возрастом *j* в году *i*, $P_f_g_a_y_{ij}$ – численность населения возрастом *j* в году *i*, $f_p_d_y_{ij}$ – прогноз вероятности смерти человека возрастом *j* в году *i*. С помощью выражения (5) отдельно прогнозируется смертность мужчин и отдельно смертность женщин.

6. Численность трудоспособного населения. Математическая модель для прогнозирования численности трудоспособного населения представлена в выражении 6.

$$W_a_p_i = \sum_{j=a_w}^{b_w} P_f_a_y_w_{ij} + \sum_{j=a_m}^{b_m} P_f_a_y_m_{ij}, \qquad (6)$$

где $W_a_p_i$ – численность трудоспособного населения в году *i*, $P_f_a_y_w_{ij}$ – численность женщин возрастом *j* в году *i*, $P_f_a_y_m_{ij}$ – численность мужчин возрастом *j* в году *i*, *a_w* – возраст начала трудовой деятельности у женщин, *b_w* – возраст окончания трудовой деятельности у женщин, *a_m* –

возраст начала трудовой деятельности у мужчин, *b_m* – возраст окончания трудовой деятельности у мужчин.

7. Естественный прирост населения. Математическая модель для прогнозирования естественного прироста населения представлена в выражении 7.

$$N_g_p_i = \frac{(B_f_a_y_w_i + B_f_a_y_m_i - \sum_{j=1}^{150} D_f_a_y_w_{ij} - \sum_{j=1}^{150} D_f_a_y_m_{ij}) * 1000}{\sum_{j=1}^{150} P_f_a_y_w_{ij} + \sum_{j=1}^{150} P_f_a_y_m_{ij}}, \quad (7)$$

где $N_g_p_i$ – естественный прирост населения в году *i*, $B_f_a_y_w_i$ – численность женщин рожденных в году *i*, $B_f_a_y_m_i$ – численность мужчин рожденных в году *i*, $D_f_a_y_w_{ij}$ – численность женщин в возрасте *j* умерших в году *i*, $D_f_a_y_m_{ij}$ – численность мужчин в возрасте *j* умерших в году *i*, $P_f_a_y_m_{ij}$ – численность мужчин возрастом *j* в году *i*, $P_f_a_y_w_{ij}$ – численность женщин возрастом *j* в году *i*.

8. Число родившихся. Математическая модель для прогнозирования численности родившихся представлена в выражении 8.

$$B_f_g_a_y_{i1} = B_f_a_y_w_{i1} + B_f_a_y_m_{i1}, \quad (8)$$

где $B_f_g_a_y_{i1}$ – численность родившихся, $B_f_a_y_w_i$ – численность женщин рожденных в году *i*, $B_f_a_y_m_i$ – численность мужчин рожденных в году *i*.

9. Число умерших. Математическая модель для прогнозирования численности умерших представлена в выражении 9.

$$D_f_g_a_y_{ij} = \sum_{j=1}^{150} D_f_a_y_m_{ij} + \sum_{j=1}^{150} D_f_a_y_w_{ij}, \quad (9)$$

где $D_f_y_i$ – численность умерших, $D_f_a_y_w_{ij}$ – численность женщин в возрасте *j* умерших в году *i*, $D_f_a_y_m_{ij}$ – численность мужчин в возрасте *j* умерших в году *i*.

10. Уровень образованности. Математическая модель для прогнозирования уровня образованности представлена в выражении 10.

$$L_education_i = \frac{(s_e_s_i + s_p_e_15_72_i + s_e_15_19_i) * 100}{3 * f_t_l_i}, \quad (10)$$

где $L_education_i$ – уровень образованности населения в году *i*, $s_e_s_i$ – доля расходов консолидированного бюджета субъекта Российской Федерации в году *i*, на дошкольное, общее, среднее профессиональное образование и дополнительное образование детей в общей сумме расходов консолидированного бюджета субъекта Российской Федерации $s_p_e_15_72_i$ – коэффициент профессиональной образованности экономически активного населения (занятого и безработного) в возрасте от 15 до 72 лет, имеющего среднее профессиональное и высшее образование в году *i*, $s_e_15_19_i$ – доля населения в возрасте 15-19 лет, охваченного образованием в году *i*, $f_t_l_i$ – коэффициент нагрузки учителей в году *i*.

11. Доля населения в возрасте 15-19 лет, охваченного образованием. Математическая модель для прогнозирования доли населения в возрасте 15-19 лет, охваченного образованием представлена в выражении 11.

$$s_e_15_19_i = e_15_19_i / All_e_15_19_i \quad (11)$$

где $s_e_15_19_i$ – доля населения в возрасте 15-19 лет, охваченного образованием, $e_15_19_i$ – суммарное количество человек, в возрасте 15-19 лет охваченное образованием, $All_e_15_19_i$ – суммарное количество человек, в возрасте 15-19 лет.

12. Коэффициент нагрузки учителей. Математическая модель для прогнозирования коэффициента нагрузки учителей представлена в выражении 12.

$$f_t_l_i = f_t_l_{iy} * t_{iy} * N_s_i / t_i \quad (12)$$

где $f_t_l_i$ – коэффициент нагрузки учителей в году *i*, $f_t_l_{ly}$ – коэффициент нагрузки учителей на последний известный год, t_{ly} – количество учителей на последний известный год, N_s_i – количество школьников в году *i*, t_i – количество учителей в году *i*.

13. Коэффициент профессиональной образованности экономически активного населения в возрасте от 15 до 72 лет, имеющего среднее профессиональное и высшее образование. Математическая модель для прогнозирования коэффициента профессиональной образованности экономически активного населения в возрасте от 15 до 72 лет, имеющего среднее профессиональное и высшее образование представлена в выражении 13.

$$s_p_e_15_72_i = \frac{\sum_{j=16}^{78} P_f_a_y_w_h_{ij} + \sum_{j=16}^{78} P_f_a_y_m_h_{ij} + \sum_{j=16}^{78} P_f_a_y_w_m_{ij} + \sum_{j=16}^{78} P_f_a_y_m_m_{ij}}{\sum_{j=16}^{78} P_f_a_y_w_{ij} + \sum_{j=16}^{78} P_f_a_y_m_{ij}}, \quad (13)$$

где $s_p_e_15_72_i$ – коэффициент профессиональной образованности экономически активного населения в возрасте от 15 до 72 лет, $P_f_a_y_w_{ij}$ – численность женщин возрастом *j* в году *i*, $P_f_a_y_m_{ij}$ – численность мужчин возрастом *j* в году *i*, $P_f_a_y_w_h_{ij}$ – численность женщин с высшим образованием возрастом *j* в году *i*, $P_f_a_y_m_h_{ij}$ – численность мужчин с высшим образованием возрастом *j* в году *i*, $P_f_a_y_w_m_{ij}$ – численность женщин со средним образованием возрастом *j* в году *i*, $P_f_a_y_m_m_{ij}$ – численность мужчин со средним образованием возрастом *j* в году *i*.

14. Численность населения по полу, возрасту, уровню образования. Математическая модель для прогнозирования численности населения по полу, возрасту, уровню образования представлена в выражении 14.

$$P_f_g_a_y_e_{ij} = P_f_g_a_y_e_{i-1j-1} - D_f_g_a_y_e_{i-1j-1} + M_f_g_a_y_e_{ij} + P_edu_{ij}, \quad (14)$$

где $P_f_g_a_y_e_{ij}$ – численность населения с профессиональным образованием, возрастом *j* в году *i*, данный показатель прогнозируется отдельно по каждому уровню профессионального образования, $D_f_g_a_y_e_{ij}$ – численность умерших с рассматриваемым уровнем профессионального образования, возрастом *j* в году *i*, $M_f_g_a_y_e_{ij}$ – численность миграции с рассматриваемым уровнем профессионального образования, возрастом *j* в году *i*, P_edu_{ij} – число жителей региона получивших рассматриваемое профессиональное образование, возрастом *j* в году *i*.

15. Число жителей региона получивших образование по возрасту, полу и уровню профессионального образования. Математическая модель для прогнозирования численности жителей региона получивших образование по возрасту, полу и уровню профессионального образования представлена в выражении 15.

$$P_edu_{ij} = P_edu_region_{ij} + P_edu_noregion_{ij}, \qquad (15)$$

где P_edu_{ij} – число жителей региона получивших рассматриваемое профессиональное образование, возрастом *j* в году *i*, $P_edu_region_{ij}$ – число жителей региона, получивших рассматриваемое профессиональное образование в регионе проживания, возрастом *j* в году *i*, $P_edu_noregion_{ij}$ – число жителей региона, получивших рассматриваемое профессиональное образование не в регионе проживания, возрастом *j* в году *i*.

16. Число жителей региона, получивших образование в регионе проживания по возрасту, полу, уровню профессионального образования и годам на период прогнозирования. Математическая модель для прогнозирования численности жителей региона получивших образование в

регионе проживания по возрасту, полу, уровню профессионального образования и годам представлена в выражение 16.

$$P_edu_region_{ij} = s_m_c_fc_{i-nj-n} * p_of_SE_{i-nj-n}, \quad (16)$$

где $P_edu_region_{ij}$ – число жителей региона, получивших рассматриваемое профессиональное образование в регионе проживания, возрастом *j* в году *i*, $s_m_c_fc_{i-nj-n}$ – количество студентов первого курса возрастом *j-n* в году *i-n*, $p_of_SE_{ij}$ – вероятность получить профессиональное образование студентом возрастом *j* в году *i*, *n* – период обучения.

17. Численность занятых в сегменте МСП. Математическая модель для прогнозирования численности занятых в сегменте МСП представлена в выражении 17.

$$E_MSP_i = Ind_p_i + Mic_b_i + Sm_b_i + Med_b_i, \quad (17)$$

где E_MSP_i – число занятых в сегменте МСП в году *i*, Ind_p_i – число индивидуальных предпринимателей в году *i*, Mic_b_i – численность работников занятых на микропредприятии в году *i*, Sm_b_i – численность работников занятых на малых предприятиях в году *i*, Med_b_i – численность работников занятых на средних предприятиях в году *i*.

18. Кадровая потребность в разрезе уровней образования. Математическая модель для прогнозирования кадровой потребность в разрезе уровней образования представлена в выражении 18.

$$S_need_edu_i = labor_demand_i - labor_supply_i, \quad (18)$$

где $S_need_edu_i$ – кадровая потребность для каждого уровня образования в году *i*, $labor_demand_i$ – спрос на трудовые ресурсы образования в году *i*, $labor_supply_i$ – спрос на трудовые ресурсы образования в году *i*.

19. Финансирование дошкольного и общего образования. Математическая модель для прогнозирования финансирования дошкольного и общего образования представлена в выражении 19.

$$Financing_ES_i = Financing_pres_i + Financing_school_i, \quad (19)$$

где $Financing_ES_i$ – финансирование дошкольного и общего образования в году *i*, $Financing_pres_i$ – финансирование дошкольного образования в году *i*, $Financing_school_i$ – финансирование общего образования в году *i*.

20. Нагрузка на систему дошкольного образования. Математическая модель для прогнозирования нагрузки на систему дошкольного образования представлена в выражении 20.

$$Load_ps_ES_i = pre_schoolboy_i * 100/e_s_p_pre_s_i, \quad (20)$$

где $Load_ps_ES_i$ – нагрузка на систему дошкольного образования в году *i*, $pre_schoolboy_i$ – количество дошколят в году *i*, $e_s_p_pre_s_i$ – мощность системы дошкольного образования в году *i*.

21. Нагрузка на систему общего образования. Математическая модель для прогнозирования нагрузки на систему общего образования представлена в выражении 21.

$$Load_s_ES_i = schoolboy_i * 100/e_s_p_s_i, \quad (21)$$

где $Load_s_ES_i$ – нагрузка на систему школьного образования в году *i*, $schoolboy_i$ – количество школьников в году *i*, $e_s_p_s_i$ – мощность системы общего образования в году *i*.

Нестрогие математические модели (когда известны только влияющие параметры на исследуемый параметр) предложены для следующих параметров:

1. Численность индивидуальных предпринимателей. Нестрогая математическая модель для прогнозирования численности индивидуальных предпринимателей представлена в выражении 22.

$$Ind_p_i = F_1(Ind_p_{i-1}, x_2(i), x_3(i), x_4(i)), \quad (22)$$

где Ind_p_i – число индивидуальных предпринимателей в году *i*, $x_2(i)$ – инвестиции в экономику региона в году *i*, $x_3(i)$ – поддержка малого бизнеса в году *i*, $x_4(i)$ – численность трудоспособного населения в году *i*.

2. Численность работников, занятых на средних предприятиях. Не строгая математическая модель для прогнозирования численности индивидуальных предпринимателей представлена в выражении 23.

$$Med_b_i = F_1(Med_b_{i-1}, x_2(i), x_3(i), x_4(i)), \quad (23)$$

где Mic_b_i – численность занятых на микропредприятии в году *i*, $x_2(i)$ – инвестиции в экономику региона в году *i*, $x_3(i)$ – поддержка малого бизнеса в году *i*, $x_4(i)$ – численность трудоспособного населения в году *i*.

3. Численность работников, занятых на малых предприятиях. Не строгая математическая модель для прогнозирования численности индивидуальных предпринимателей представлена в выражении 24.

$$Sm_b_i = F_1(Sm_b_{i-1}, x_2(i), x_3(i), x_4(i)), \quad (24)$$

где Sm_b_i – численность занятых на малых предприятиях в году *i*, $x_2(i)$ – инвестиции в экономику региона в году *i*, $x_3(i)$ – поддержка малого бизнеса в году *i*, $x_4(i)$ – численность трудоспособного населения в году *i*.

4. Численность работников, занятых на микропредприятиях. Не строгая математическая модель для прогнозирования численности индивидуальных предпринимателей представлена в выражении 5.

$$Mic_b_i = F_1(Mic_b_{i-1}, x_2(i), x_3(i), x_4(i)), \quad (25)$$

где Med_b_i – численность занятых на средних предприятиях в году *i*, $x_2(i)$ – инвестиции в экономику региона в году *i*, $x_3(i)$ – поддержка малого бизнеса в году *i*, $x_4(i)$ – численность трудоспособного населения в году *i*.

Для определения модели прогнозирования параметров в пунктах 22-25, будут доступны два инструмента: нейронные сети и регрессионный анализ.

ГЛАВА 2. ТЕХНОЛОГИИ ПРОГНОЗИРОВАНИЯ И МОДЕЛИРОВАНИЯ ДИНАМИКИ СОЦИАЛЬНО-ЭКОНОМИЧЕСКОГО РАЗВИТИЯ РЕГИОНА

Регрессионный анализ. Для влияющих параметров подбираются коэффициенты согласно следующим выражениям:

$$Y(t) = \sum_{i=1}^{n} b_i * X_i(t)$$

$$\sum_{t=1}^{m}(Y(t) - Yk(t))^2 \rightarrow min,$$

где $Y(t)$ – значения исследуемого параметр полученные в результате вычислений, $t=1...m$ – период времени, b_i – коэффициенты влияющих параметров, $i=1...n$ – количество влияющих параметров, X_i – значения влияющих параметров, $Yk(t)$ – значения исследуемого параметра.

Автоматическое прогнозирование данных на указанный период с использованием регрессионного анализа. Задача автоматического прогнозирования направленна на минимизацию последствий субъективного мнения Пользователя и повышения точности прогнозирования параметров. Автоматическое прогнозирование будет осуществляться с использованием полного перебора количества влияющих параметров от 3 до 0.3*n+1, где n – количество элементов временного ряда.

В качестве критерия выбора, будет использоваться наименьшая ошибка по модулю, полученная в процессе кросс валидации, как при прогнозировании абсолютных величин, так и при классификации:

$$\boldsymbol{E = \sum_{l=1}^{n} |Y(l) - I(l)|},$$

$$\boldsymbol{E = Mismatch},$$

где E – ошибка по модулю, Y – значения получение при прогнозирование, I – эталонные значения, l=1…n, размер выборки для кросс валидации, $Mismatch$ – суммарное количество не совпадений.

Нейронные сети. При выборе этого инструмента будет доступен функционал, представленный в таблице 1.

Таблица 1. Архитектуры нейронных сетей.

№	Топология	Алгоритмы обучения
1	Однослойный персептрон.	1. Правило обучения Видроу-Хоффа. 2. Алгоритм обратного распространения ошибки
2	Многослойный персептрон.	1. Алгоритм обратного распространения ошибки. 2. Алгоритм глубокого обучения. Для предобучения использовать автоэнкодер.
3	Сверточная нейронная сеть.	Алгоритм глубокого обучения. Для предобучения использовать автоэнкодер.

Для реализации правила обучения Видроу-Хоффа, будет использоваться следующий алгоритм[3]:

1. Задается скорость обучения α и минимальная среднеквадратичная ошибка сети *Em*, которой необходимо достичь в процессе обучения.

2. Случайным образом инициализируются весовые *w* коэффициенты и порог нейронной сети *T*.

3. Подаются входные образы на нейронную сеть и вычисляются векторы выходной активности сети.

4. Производится изменение весовых коэффициентов и порога нейронной сети согласно выражениям:

[3] Нейронные сети: обучение, организация и применение. Кн. 10: Учеб. пособие для вузов / Общая ред. А. И. Галушкина. - М.: ИПРЖР, 2000.: ил. (Нейрокомпьютеры и их применение).

$$\omega_{j1}(k+1) = \omega_{j1}(k) - \alpha(y_1{}^k - t^k)x_i{}^k,$$

$$T(k+1) = T(k) + \alpha(y_1{}^k - t^k),$$

где x – нейроны входного слоя, k – номер итерации, y – паттерн выходной активности сети, t^k – эталонное значение, $j = \overline{1,n}$ – количество нейронов.

5. Алгоритм продолжается до тех пор, пока суммарная среднеквадратичная ошибка сети не станет меньше заданной, т. е. $E \le E_m$:

$$E(k) = \frac{1}{2}(y_1{}^k - t^k)^2$$

где *E(k)* – суммарная среднеквадратичная ошибка сети.

Алгоритм обратного распространения ошибки будет реализован следующим образом[4]:

1. Задается шаг обучения α $(0 < \alpha < 1)$ и желаемая среднеквадратичная ошибка нейронной сети E_m.

2. Случайным образом инициализируются весовые коэффициенты и пороговые значения нейронной сети.

3. Последовательно подаются образы из обучающей выборки на вход нейронной сети. При этом для каждого входного образа выполняются следующие действия:

a) производится фаза прямого распространения входного образа по нейронной сети. При этом вычисляется выходная активность всех нейронных элементов сети:

$$y_j = F(\sum_i \omega_{ij} y_i - T_j),$$

где индекс *j* характеризует нейроны следующего слоя по отношению к слою *i*.

b) производится фаза обратного распространения сигнала, в результате которой определяется ошибка $\gamma_j, j = 1,2,\cdots$ нейронных элементов для всех слоев сети. При этом соответственно для выходного и скрытого слоев:

[4] Нейронные сети: обучение, организация и применение. Кн. 10: Учеб. пособие для вузов / Общая ред. А. И. Галушкина. - М.: ИПРЖР, 2000.: ил. (Нейрокомпьютеры и их применение).

$$\gamma_j = y_j - t_j,$$

$$\gamma_j = \sum_i \gamma_i F'(S_i)\omega_{ji}.$$

В последнем выражении индекс i характеризует нейронные элементы следующего слоя по отношению к слою j.

с) для каждого слоя нейронной сети происходит изменение весовых коэффициентов и порогов нейронных элементов:

$$\omega_{ij}(t+1) = \omega_{ij}(t) - \alpha\gamma_j F'(S_j) y_i,$$

$$T_j(t+1) = T_j(t) + \alpha\gamma_j F'(S_j).$$

4. Вычисляется суммарная среднеквадратичная ошибка нейронной сети:

$$E = \frac{1}{2}\sum_{k=1}^{L}\sum_j (y_j{}^k - t_j{}^k)^2,$$

где L — размерность обучающей выборки.

5. Если $E > E_m$ то происходит переход к шагу 3 алгоритма. В противном случае алгоритм обратного распространения ошибки заканчивается.

При выборе алгоритма обратного распространения ошибки, будут доступны следующие функции активации:

1. Сигмоидная функция. Обобщенное дельта правило для сигмоидной функции активации можно представить в следующем виде:

$$\omega_{ij}(t+1) = \omega_{ij}(t) - \alpha\gamma_j y_j(1-y_j) y_i,$$

$$T_j(t+1) = T_j + \alpha\gamma_j y_j(1-y_j).$$

Ошибка для *j*-го нейрона выходного слоя определяется, как

$$\gamma_j = y_j - t_j$$

Для *j*-го нейронного элемента скрытого слоя:

$$\gamma_j = \sum_{i=1}^{m} \gamma_i y_i (1-y_i)\omega_{ji},$$

где *m*— количество нейронных элементов следующего слоя по отношению к слою *j*.

2. Биполярная сигмоидная функция. Выражения для обучения нейронной сети с биполярной сигмоидной функцией активации:

$$\omega_{ij}(t+1) = \omega_{ij}(t) - \alpha\gamma_j 2(1-y_j)y_j y_i,$$

$$T_j(t+1) = T_j(t) + \alpha\gamma_j 2(1-y_j)y_j.$$

Ошибка для *j*-го нейрона выходного и скрытого слоев определяется соответственно, как:

$$\gamma_j = y_j - t_j,$$

$$\gamma_j = \sum_i \gamma_i 2(1-y_i)y_i\omega_{ji}$$

3. Гиперболический тангенс. Правило обучения представляется в виде следующего выражения:

$$\omega_{ij}(t+1) = \omega_{ij}(t) - \alpha\gamma_j(1-y_j^2)y_i,$$

$$T_j(t+1) = T_j(t) + \alpha\gamma_j(1-y_j^2).$$

Ошибка для *j*-го нейрона выходного и скрытого слоев соответственно равняется:

$$\gamma_j = y_j - t_j$$

$$\gamma_j = \sum_i \gamma_i(1-y_i^2)\omega_{ji}$$

Алгоритм глубокого обучения с предобучением автоэнкодера[5], будет реализован следующим образом:

1) предобучение нейронной сети методом послойного обучения, начиная с первого слоя (pre-training); данный этап обучения осуществляется без учителя:

1.1. Конструируется автоассоциативная сеть с входным слоем, скрытым слоем и выходным слоем .

1.2. Обучается автоассоциативная сеть, например, при помощи алгоритма обратного распространения ошибки (как правило не более 100 эпох)и фиксируются синаптические связи первого слоя .

[5] Головко В.А. От многослойных персептронов к нейронным сетям глубокого доверия: парадигмы обучения и применение. Лекции по нейроинформатике, Publisher: Москва: НИЯУ "МИФИ", pp.47-84.

1.3. Берется следующий слой и формируется автоассоциативная сеть аналогичным образом.

1.4. Используя настроенные синаптические связи предыдущего слоя, подаем входные данные на вторую автоассоциативную сеть и обучаем ее аналогичным образом. В результате получаются весовые коэффициенты второго слоя.

1.5. Процесс продолжается до последнего слоя нейронной сети.

1.6. Берется последний слой нейронной сети и обучается с учителем.

1.7. Обучается вся сеть для точной настройки параметров при помощиалгоритма обратного распространения ошибки.

2) настройка синаптических связей всей сети при помощи алгоритма обратного распространения ошибки.

Автоматическое прогнозирование данных на указанный период с использованием нейронных сетей и подбором параметров. Задача автоматического прогнозирования направленна на минимизацию последствий субъективного мнения Пользователя и повышения точности прогнозирования параметров. Автоматическое прогнозирование будет осуществляться с использованием полного перебора параметров настройки нейронной сети, представленного в таблице 2.

В качестве критерия выбора окончательных параметров нейронной сети, будет использоваться наименьшая ошибка по модулю, полученная в процессе кросс валидации, как при прогнозировании абсолютных величин, так и при классификации:

$$E = \sum_{l=1}^{n} |Y(l) - I(l)|,$$

$$E = Mismatch,$$

где E – ошибка по модулю, Y – значения полученные при прогнозировании, I – эталонные значения, l=1…n, размер выборки для кросс валидации, $Mismatch$ – суммарное количество не совпадений.

При выборе автоматического прогнозирования, Пользователю будет доступна возможность оценить необходимое время для данной операции и выбрать критерий остановки обучения нейронной сети: время или количество итераций.

Таблица 2. Параметры настройки нейронной сети

Количество обрабатывающих слоёв в искусственной нейронной сети	От одного слоя до 7 слоёв.
Количество нейронов в обрабатывающих слоях	От трёх нейронов до 2*n+1, где n – количество нейронов входного слоя.
Количество нейронов выходного слоя	Один.
Количество нейронов входного слоя	Формируется согласно размеру обучающей выборки
Функции активации нейронных элементов.	Сигмоидная Биполярная сигмоидная Гиперболический тангенс. Линейная.
Шаг обучения	Адаптивный шаг обучения. Постоянный. Значения от 0.01 до 0.1. С шагом 0.01.

Адаптивный шаг обучения будет реализован следующим образом[6]:

Для сигмоидной функции:

$$\alpha(t) = \frac{4\sum_j \gamma_j{}^2 y_j(1-y_j),}{(1+\sum_i y_i{}^2)\cdot\sum_j \gamma_j{}^2 y_j{}^2(1-y_j)^2},$$

где ошибка *j*-ого нейронного элемента скрытого слоя равняется:

$$\gamma_j = \sum_i \gamma_i y_i(1-y_i)\omega_{ji}$$

[6] Нейронные сети: обучение, организация и применение. Кн. 10: Учеб. пособие для вузов / Общая ред. А. И. Галушкина. - М.: ИПРЖР, 2000.: ил. (Нейрокомпьютеры и их применение).

Здесь *i* характеризует нейроны следующего слоя по отношению к слою *j*.

Для гиперболического тангенса:

$$\alpha(t) = \frac{\sum_j \gamma_j{}^2 (1 - y_j{}^2),}{(1 + \sum_i y_i{}^2) \cdot \sum_j \gamma_j{}^2 (1 - y_j{}^2)},$$

ошибка *j*-ого нейрона скрытого слоя определяется как:

$$\gamma_j = \sum_i \gamma_i (1 - y_i{}^2) \omega_{ji}$$

Для биполярной сигмоидной функции:

$$\alpha(t) = \frac{\sum_j \gamma_j{}^2 y_j (1 - y_j),}{(1 + \sum_i y_i{}^2) \cdot \sum_j \gamma_j{}^2 y_j{}^2 (1 - y_j)^2},$$

где ошибка *j*-ого нейрона скрытого слоя:

$$\gamma_j = \sum_i \gamma_i 2 \cdot (1 - y_i) y_i \omega_{ji}$$

Для прогнозирования параметров:

- Миграция населения по полу, возрасту и уровню образования
- Распределение рожениц по возрасту
- Доля родившихся мужчин
- Вероятность смерти по возрасту, полу и уровню образования.
- Доля расходов консолидиранного бюджета на дошкольное, общее, среднее профессиональной образование и дополнительное образование детей в общей сумме расходов консолидированного бюджета субъекта РФ
- Число жителей региона получивших образование в не региона проживания по возрасту, полу и уровню профессионального образования.
- Коэффициенты занятости населения по полу, возрасту и уровню образования
- Спрос на трудовые ресурсы по уровню образования
- Затраты на одного человека в системе дошкольного образования.

-
- Затраты на одного человека в системе общего образования.
- Затраты на строительство одного места в системе дошкольного образования.
- Затраты на строительство одного места в системе общего образования,

будет возможность выбрать инструмент: скользящее окно или экспертное мнение.

Скользящее окно. Способ прогнозирования с использованием скользящего окна будет реализован следующим образом[7].

Пусть дан временной ряд $x(t)$ на промежутке $t = \overline{1,m}$. Тогда задача прогнозирования состоит в том, чтобы найти продолжение временного ряда на неизвестном промежутке, т. е. необходимо определить $x(m+1)$, $x(m+2)$ и так далее. Совокупность известных значений временного ряда образуют обучающую выборку, размерность которой характеризуется значением *m*. Для прогнозирования временных рядов используется метод "скользящего окна". Он характеризуется длиной окна p, которая равняется количеству элементов ряда, одновременно подаваемых на нейронную сеть. Это определяет структуру нейронной сети, которая состоит из p распределительных нейронов и одного выходного нейрона.

Такая модель соответствует линейной авторегрессии и описывается следующим выражением:

$$\overline{x(n)} = \sum_{k=1}^{p} \omega_k x(n - p + k - 1),$$

где $\omega_k, k = \overline{1,p}$ — весовые коэффициенты нейронной сети; $\overline{x(n)}$ — оценка значения ряда $x(n)$ в момент времени n.

Ошибка прогнозирования определяется, как

$$e(n) = x(n) - \overline{x(n)}.$$

[7] Нейронные сети: обучение, организация и применение. Кн. 10: Учеб. пособие для вузов / Общая ред. А. И. Галушкина. - М.: ИПРЖР, 2000.: ил. (Нейрокомпьютеры и их применение).

Модель линейной авторегрессии формирует значение ряда $x(n)$, как взвешенную сумму предыдущих значений ряда. Обучающую выборку нейронной сети можно представить в виде матрицы, строки которой характеризуют векторы, подаваемые на вход сети:

$$X = \begin{bmatrix} x(1) & x(2) & \dots & x(p) \\ x(2) & x(3) & \dots & x(p+1) \\ \dots & \dots & \dots & \dots \\ x(m-p) & x(m-p+1) & \dots & x(m-1) \end{bmatrix}.$$

Это эквивалентно перемещению окна по ряду $x(t)$ с единичным шагом. Эталонные значение соответственно:

$$I = \begin{bmatrix} x(p+1) \\ x(p+2) \\ \dots \\ x(m) \end{bmatrix}.$$

Экспертное мнение. При выборе способа прогнозирования «экспертное мнение», Пользователю будет доступна возможность сформировать динамику временного ряда, исследуемого параметра вручную.

ГЛАВА 3. АПРОБАЦИЯ РАЗРАБОТАННОЙ ТЕХНОЛОГИИ ПРОГНОЗИРОВАНИЯ ДИНАМИКИ СОЦИАЛЬНО-ЭКОНОМИЧЕСКОГО РАЗВИТИЯ РЕГИОНА

Для того чтобы воспользоваться сервисом обеспечивающей поиск связанности данных, необходимо перейти по ссылке https://rms.admhmao.ru/viewer?dashboardGuid=7d1da124b8b84772a1ad20e75651b51d&showNav=true&fit=true и авторизоваться (рисунок 1):

Рисунок 1 – Страница авторизации

После чего мы попадаем на страницу «Настройка параметров» (рисунок 2).

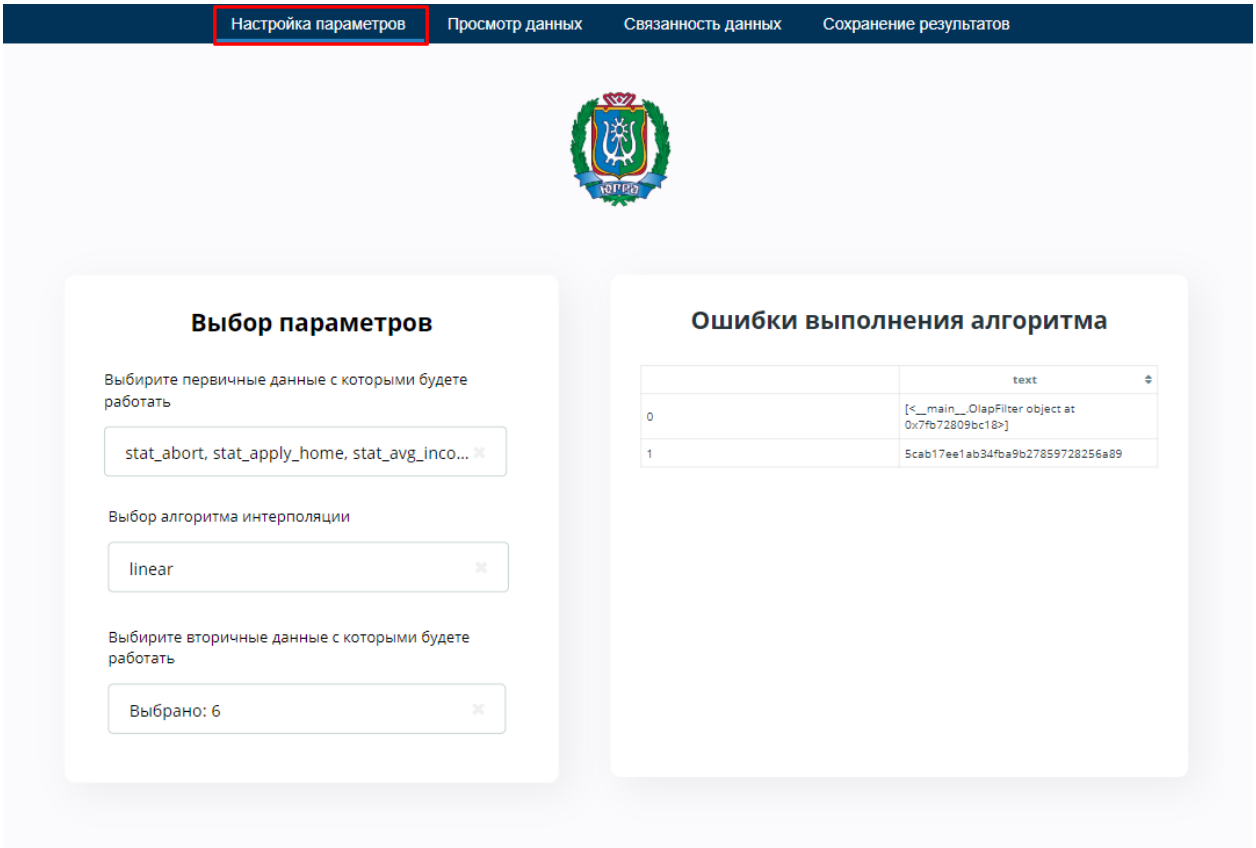


Рисунок 2 – Страница «Настройка параметров»

На странице «Настройка параметров» расположено два блока «Выбор параметров» и «Ошибки выполнения алгоритма».

Для того, чтобы графически увидеть связанность данных, в блоке «Выбор параметров» (рисунок 3) нужно выбрать первичные и вторичные данные, а так же алгоритм интерполяции и нажать применить.

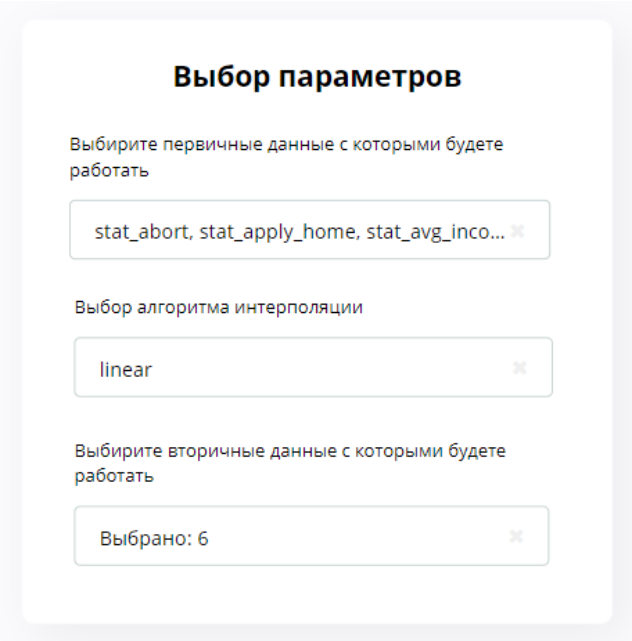


Рисунок 3 – Блок выбора параметров

После чего произойдет применение данных, сопровождающееся прелоадером.

На странице «Просмотр данных» (рисунок 4) отображается график связанных данных, выбранных на предыдущем шаге.

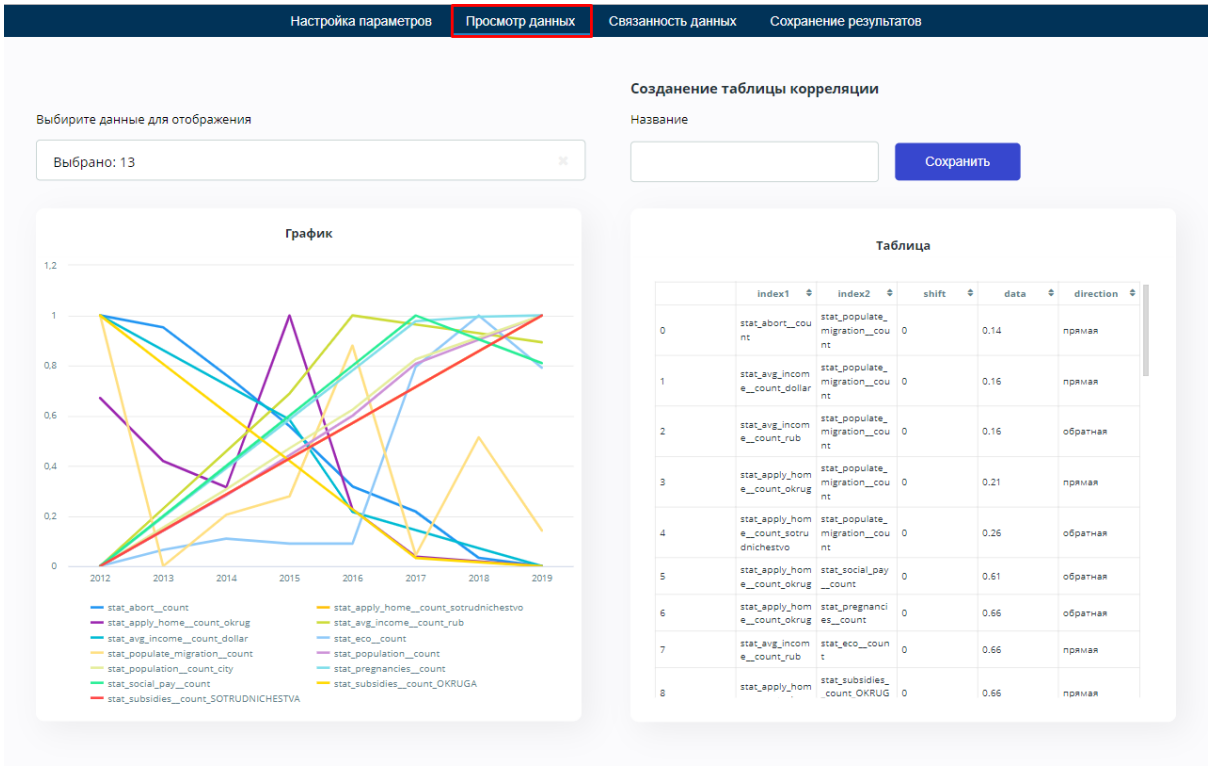


Рисунок 4 – Блок выбора параметров

Выбранные данные можно сохранить в виде таблицы корреляции (рисунок 5), и экспортировать в формате .xls.

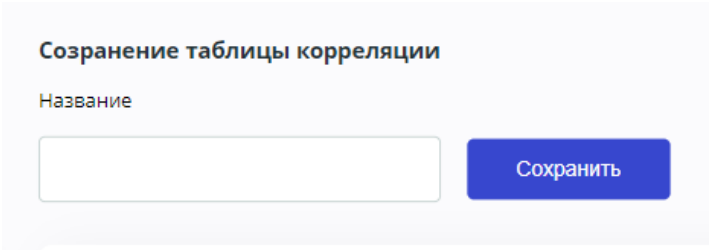


Рисунок 5 – Блок выбора параметров

На странице «Связанность данных» (рисунок 6) отображаются:

- Матрица корреляций по строкам и столбцам в виде диаграммы
- Матрица корреляций по строкам и столбцам в виде таблицы градаций

- Корреляция по шагу в виде круговой диаграммы

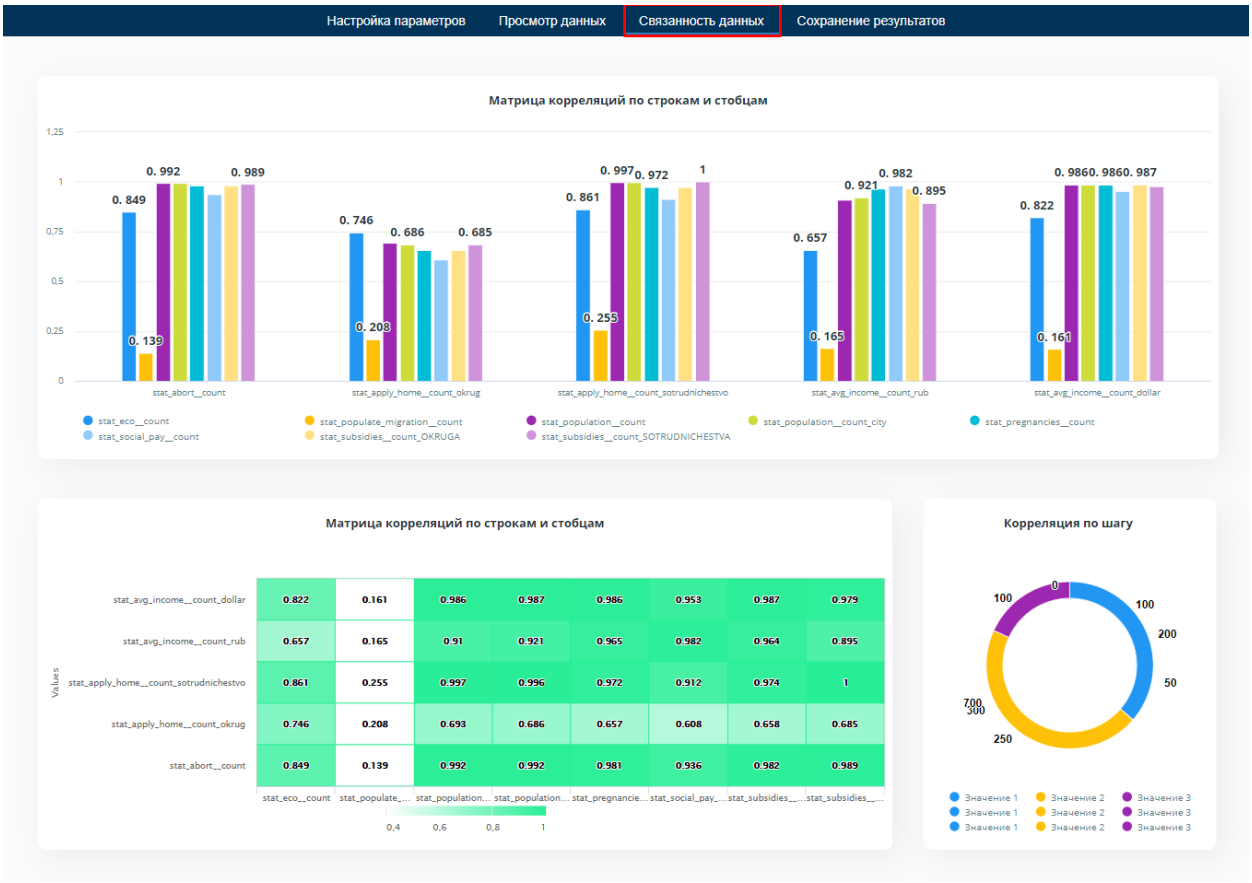


Рисунок 6 – Страница «Связанность данных»

На странице «Сохранение результатов» (рисунок 7) отображается таблица сохраненных анализируемых данных.

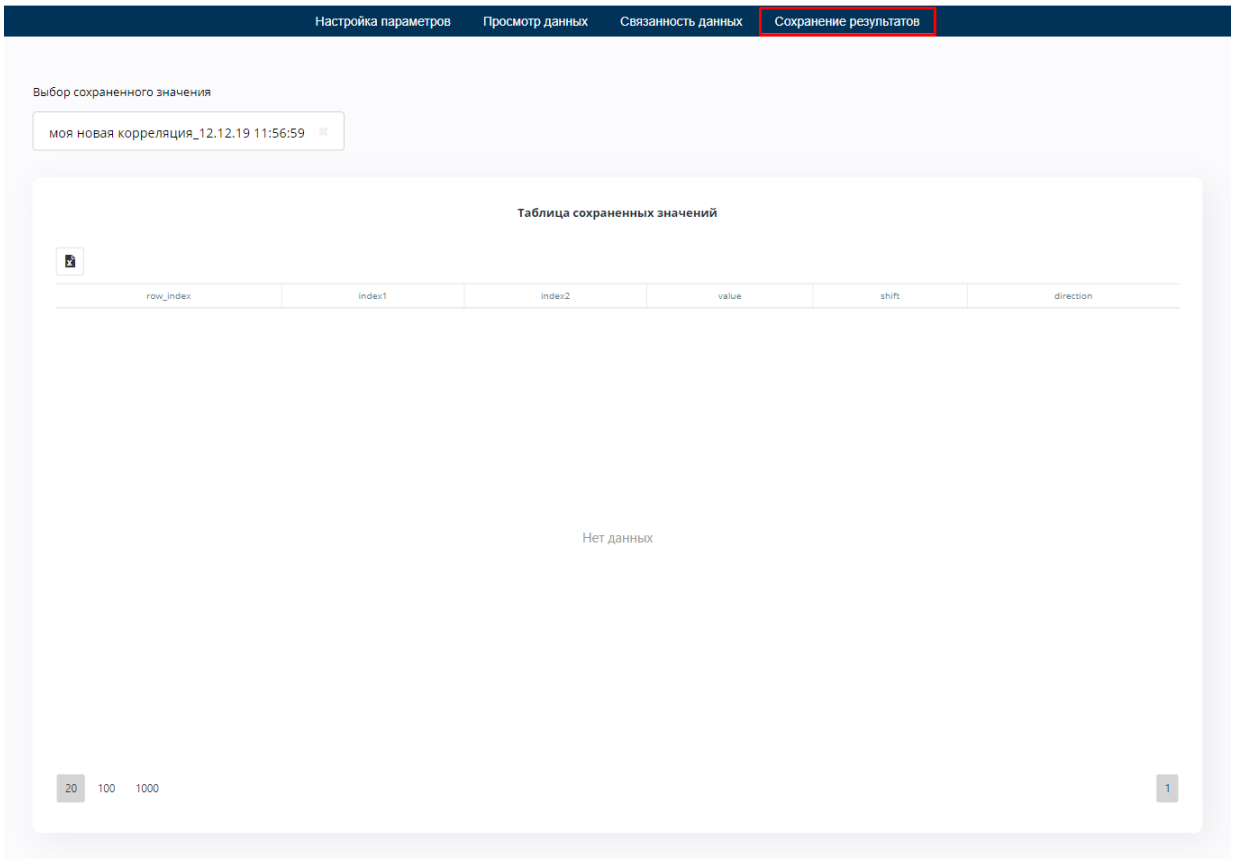


Рисунок 7 – Страница «Сохранение результатов»

В блоке «Выбор сохраненного значения» (рисунок 8) можно из списка выбрать ранее сохраненные данные, и посмотреть значения корреляции в табличном виде, а также экспортировать данные текущей таблицы в формате .xlsx, нажав на соответствующий значок (рисунок 9).

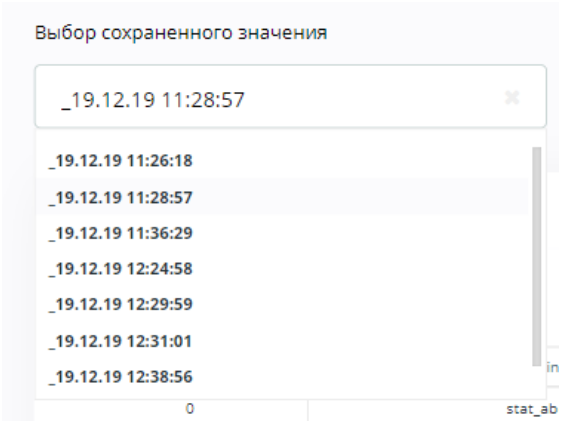
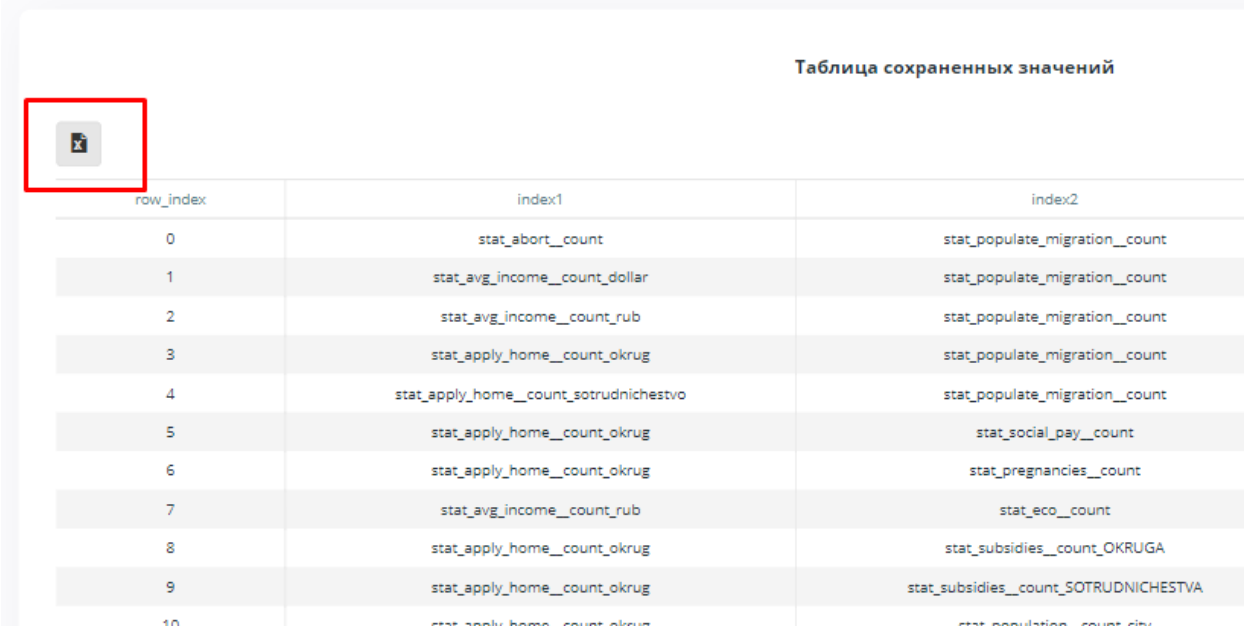


Рисунок 8 – Блок «Выбор сохраненного значения»



row_index	index1	index2
0	stat_abort__count	stat_populate_migration__count
1	stat_avg_income__count_dollar	stat_populate_migration__count
2	stat_avg_income__count_rub	stat_populate_migration__count
3	stat_apply_home__count_okrug	stat_populate_migration__count
4	stat_apply_home__count_sotrudnichestvo	stat_populate_migration__count
5	stat_apply_home__count_okrug	stat_social_pay__count
6	stat_apply_home__count_okrug	stat_pregnancies__count
7	stat_avg_income__count_rub	stat_eco__count
8	stat_apply_home__count_okrug	stat_subsidies__count_OKRUGA
9	stat_apply_home__count_okrug	stat_subsidies__count_SOTRUDNICHESTVA

Рисунок 9 – Экспортирование таблицы

Для того чтобы воспользоваться сервисом регрессионного анализа, необходимо перейти по ссылке https://rms.admhmao.ru/viewer?dashboardGuid=6d9391309c7d472a9260485e5a91a36f&showNav=true&fit=true и авторизоваться (рисунок 10):

Рисунок 10 – Страница авторизации

После авторизации сразу попадаем на страницу «Регрессионный анализ ОП» (рисунок 11). На данной странице отображаются изменяемые поля для отображения данных, график, таблица изменений, позволяющая симулировать прогноз при определенных условиях, а так же системный журнал.

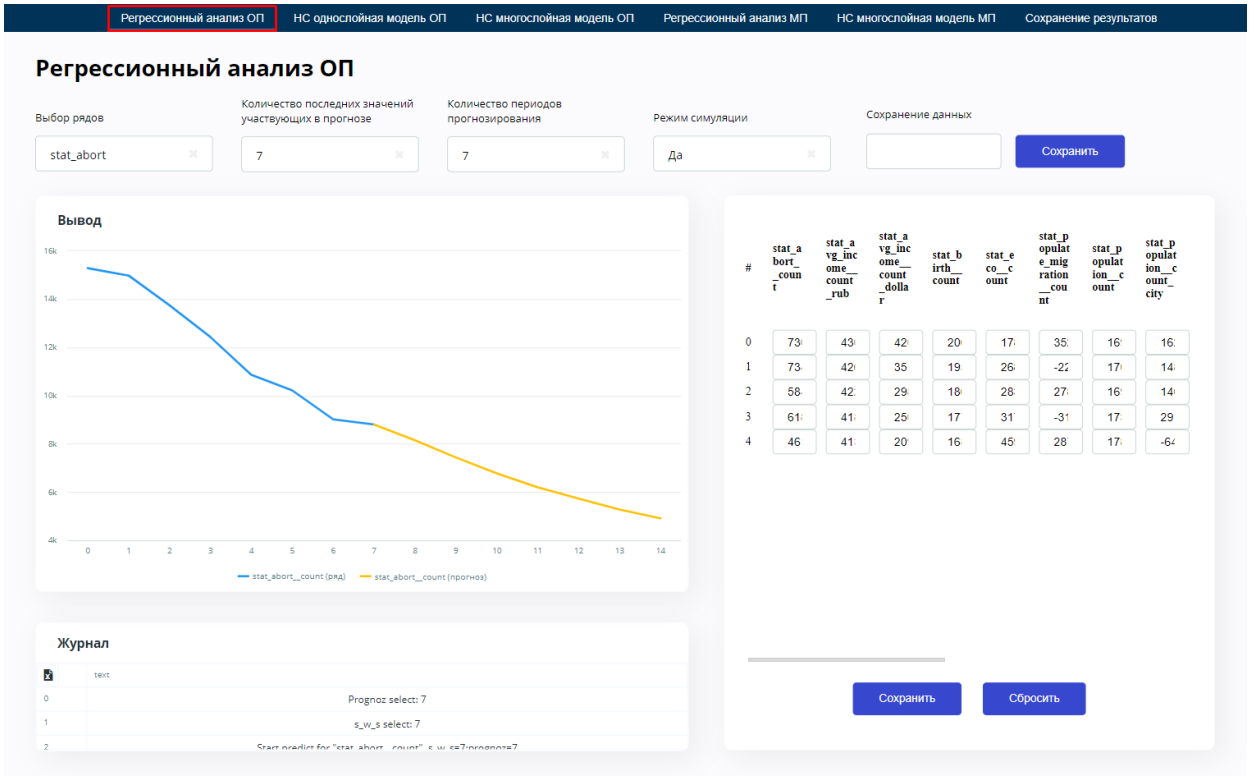


Рисунок 11 – Страница «Регрессионный анализ ОП»

Страница «НС однослойная модель ОП». На данной странице отображаются изменяемые поля для отображения данных, график, таблица изменений, позволяющая симулировать прогноз при определенных условиях, а также системный журнал.

Пользователь может построить прогноз по указанному временному ряду, используя метод линейной регрессии, при этом, в построении прогноза участвует только сам временной ряд.

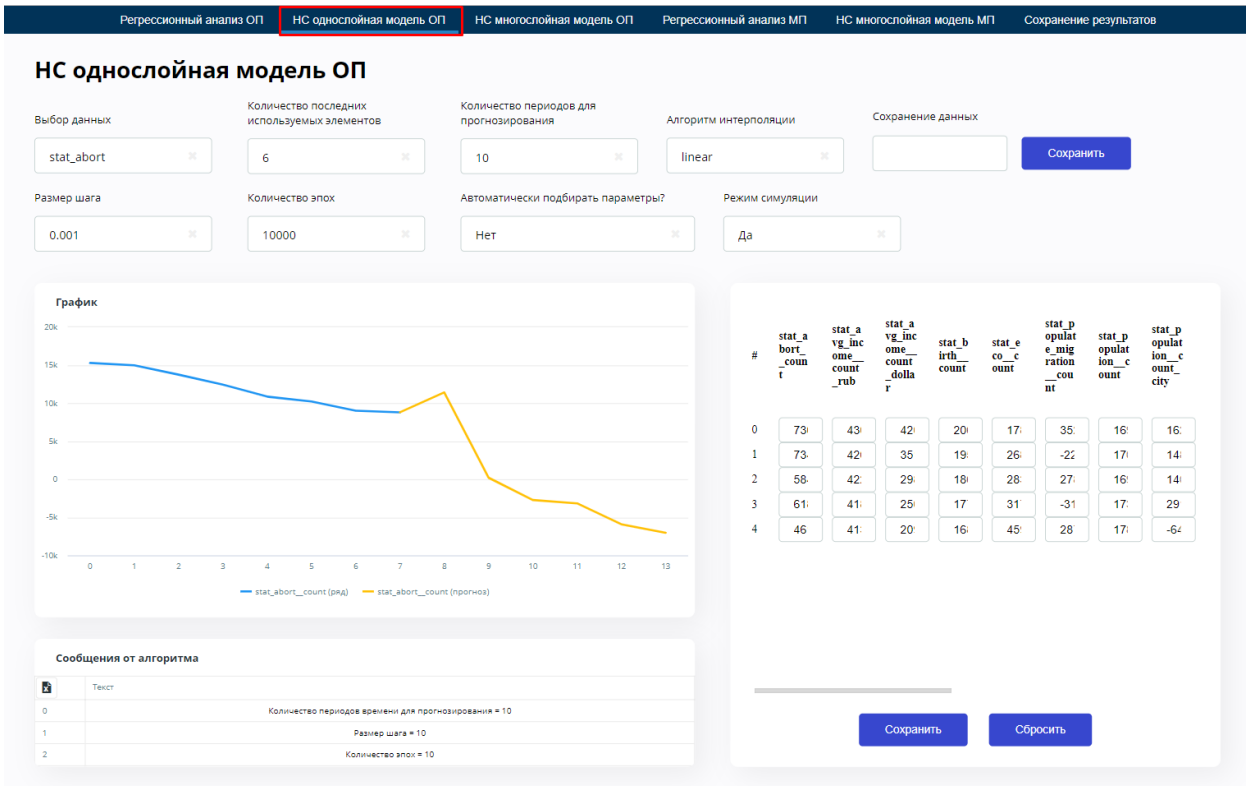


Рисунок 11 – Страница «НС однослойная модель ОП»

Страница «НС многослойная модель ОП» (рисунок 12). На данной странице отображаются изменяемые поля для отображения данных, график, таблица изменений, позволяющая симулировать прогноз при определенных условиях, а также системный журнал.

Пользователь может построить прогноз по указанному временному ряду, используя метод нейронной сети, при этом, в построении прогноза участвует только сам временной ряд.

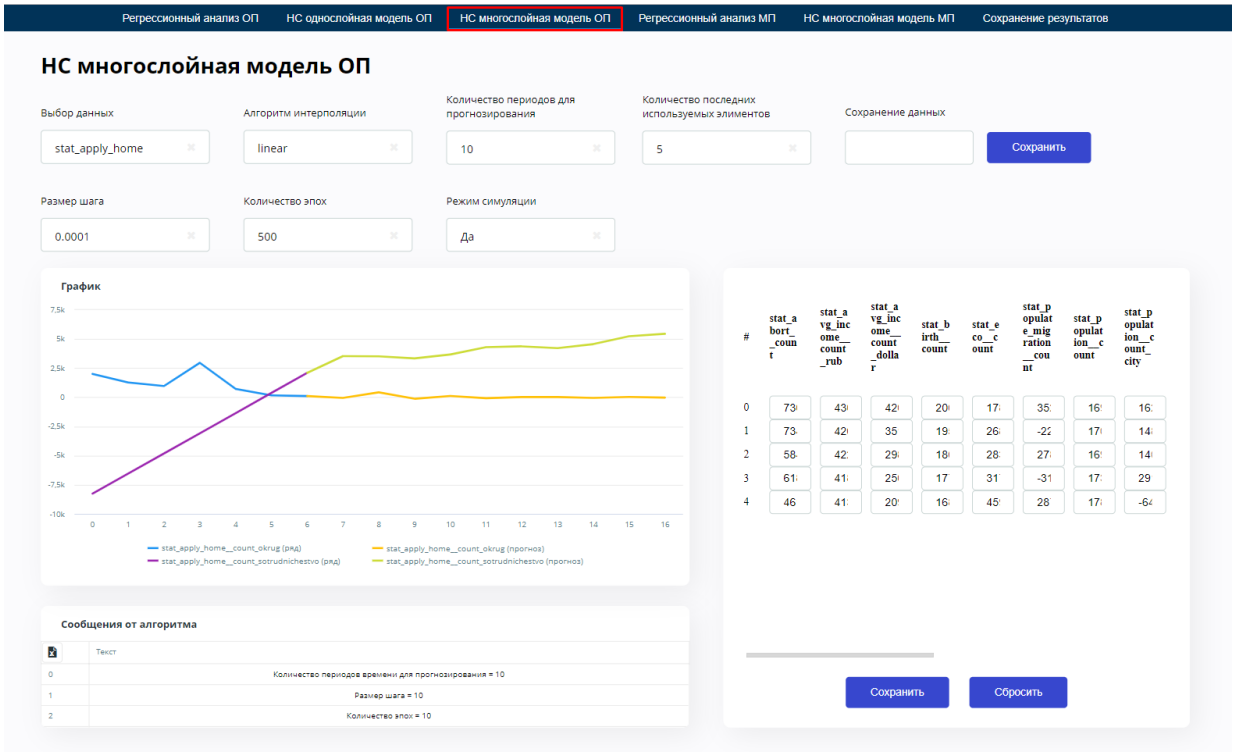


Рисунок 12 – Страница «НС многослойная модель ОП»

Страница «Регрессионный анализ МП» (рисунок 13). На данной странице отображаются изменяемые поля для отображения данных, график, таблица изменений, позволяющая симулировать прогноз при определенных условиях, а также системный журнал.

Пользователь может построить прогноз по указанному временному ряду, используя метод нейронной сети, при этом, в построении прогноза участвует только сам временной ряд.

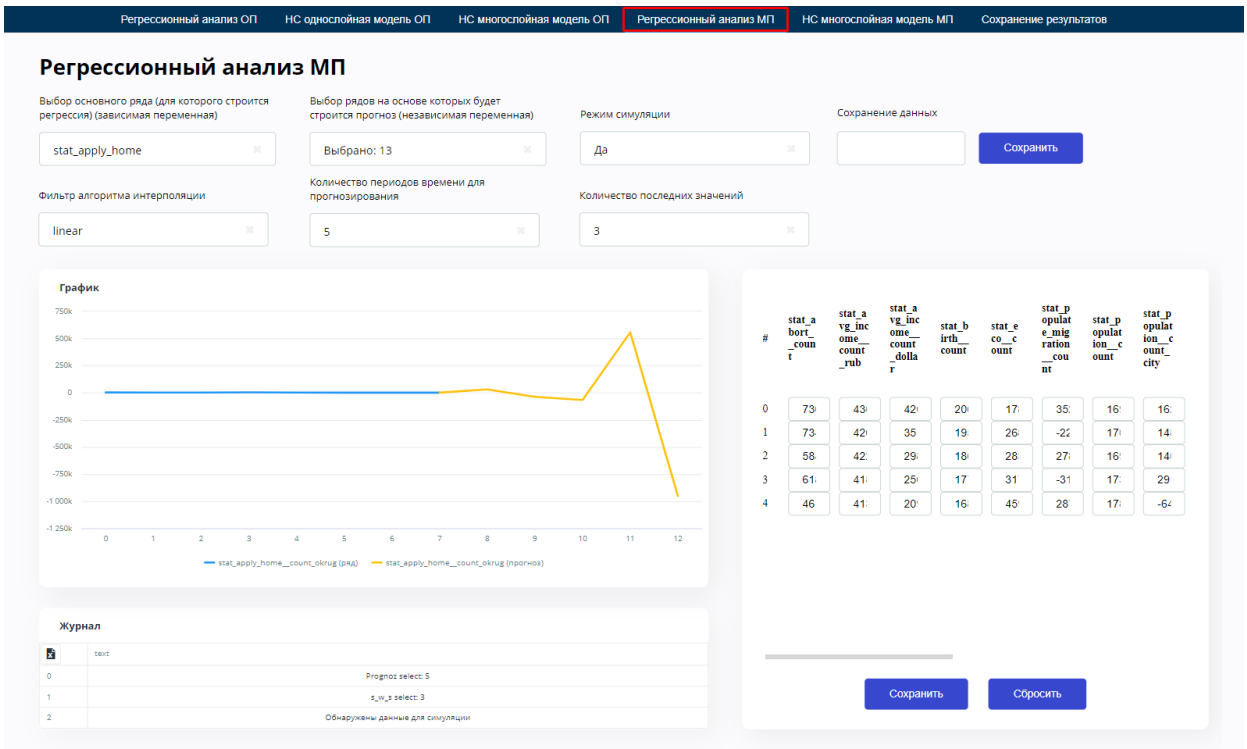


Рисунок 13 – Страница «Регрессионный анализ МП»

Страница «НС многослойная модель МП» (рисунок 14). На данной странице отображаются изменяемые поля для отображения данных, график, таблица изменений, позволяющая симулировать прогноз при определенных условиях, а также системный журнал.

Пользователь может построить прогноз на указанном временном ряду используя метод линейной регрессии, при этом, в построении прогноза участвует другие, выбранные пользователем временные ряды.

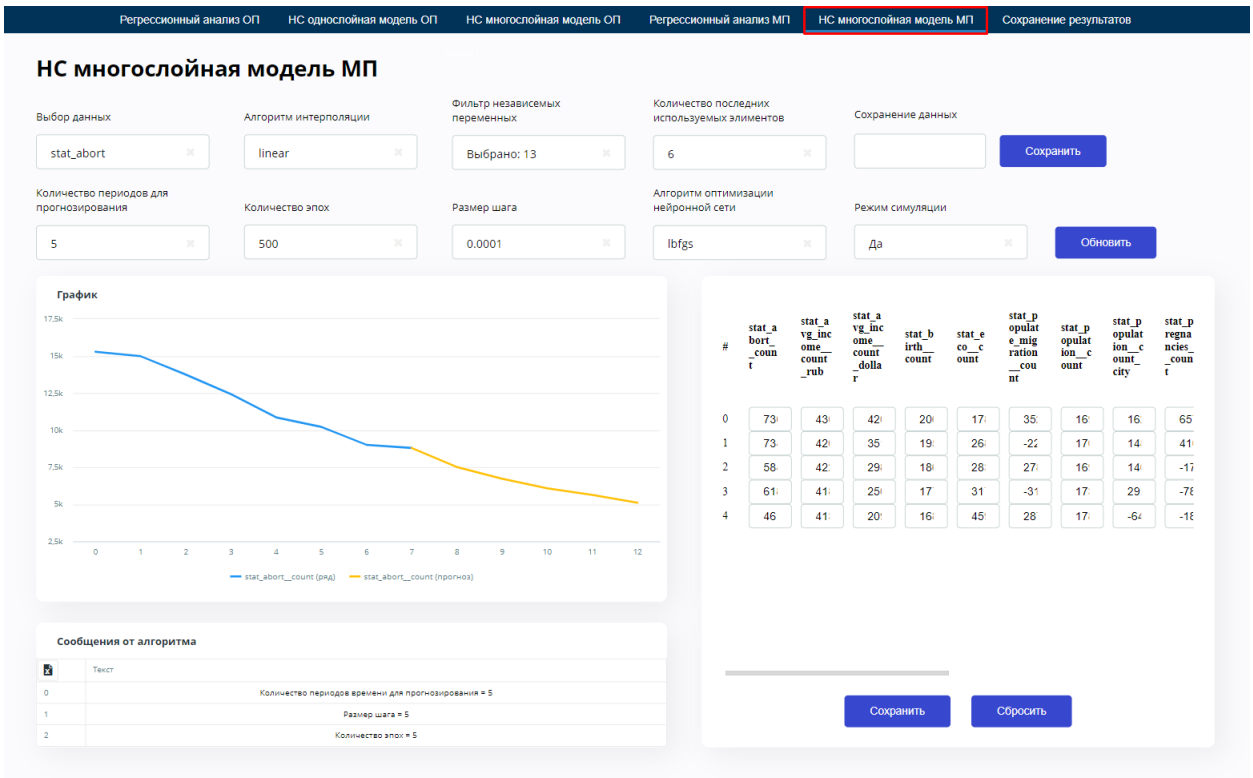


Рисунок 14 – Страница «НС многослойная модель МП»

Пользователь может построить прогноз на указанном временном ряду используя метод нейронные сети, при этом, в построении прогноза участвует другие, выбранные пользователем временные ряды.

Страница «Сохраненный прогноз» (рисунок 15).

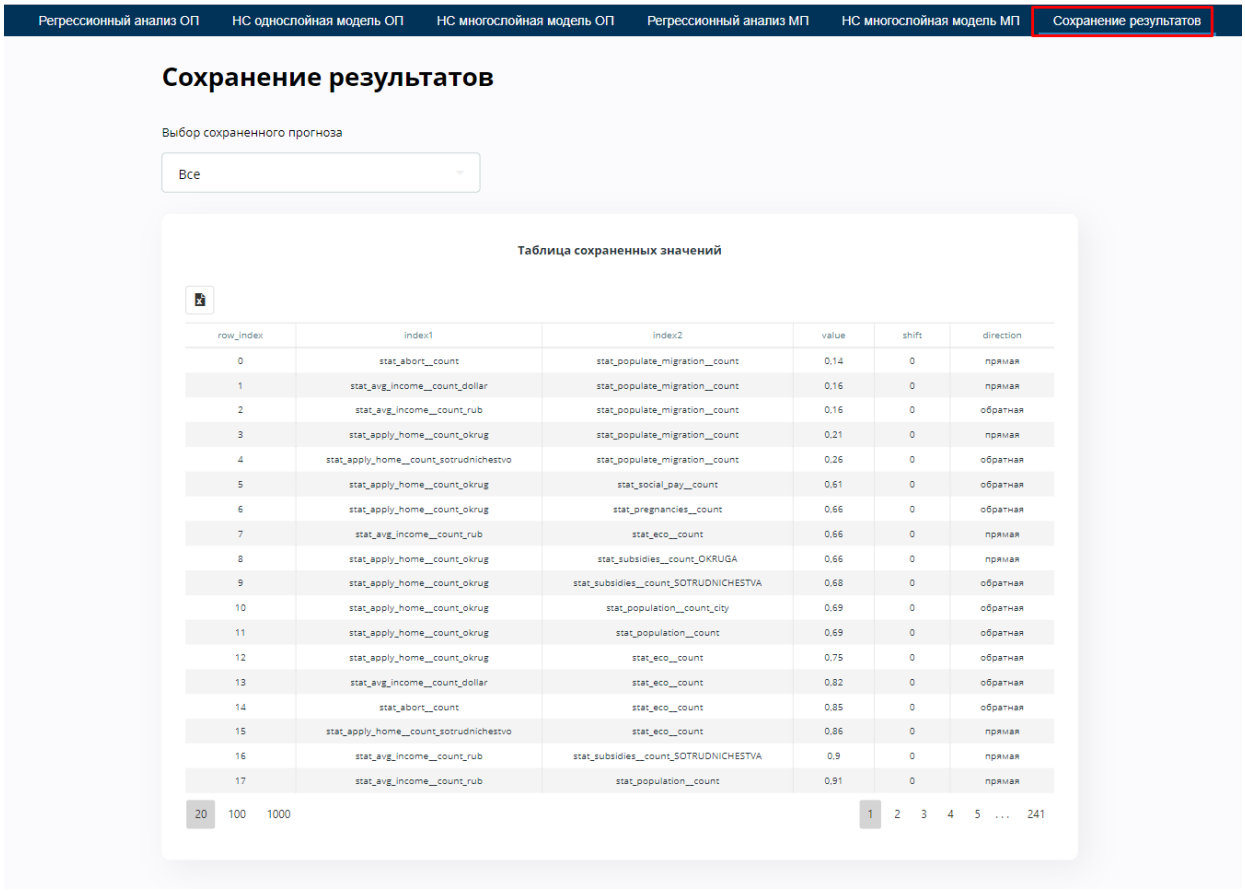


Рисунок 15 – Страница «Сохраненный прогноз»

В блоке «Выбор сохраненного прогноза» (рисунок 16) можно из списка выбрать ранее сохраненные данные, и посмотреть значения корреляции в табличном виде, а также экспортировать данные текущей таблицы в формате .xlsx, нажав на соответствующий значок (рисунок 17).

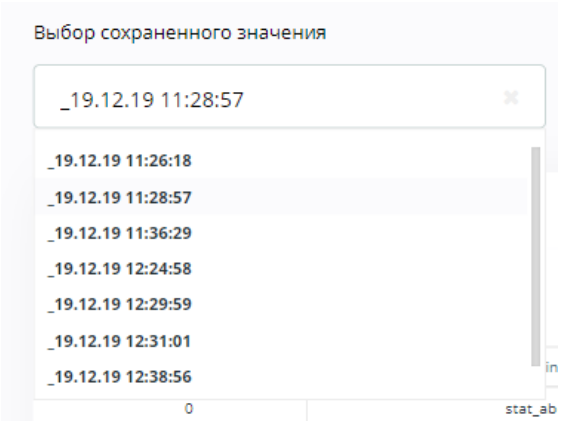
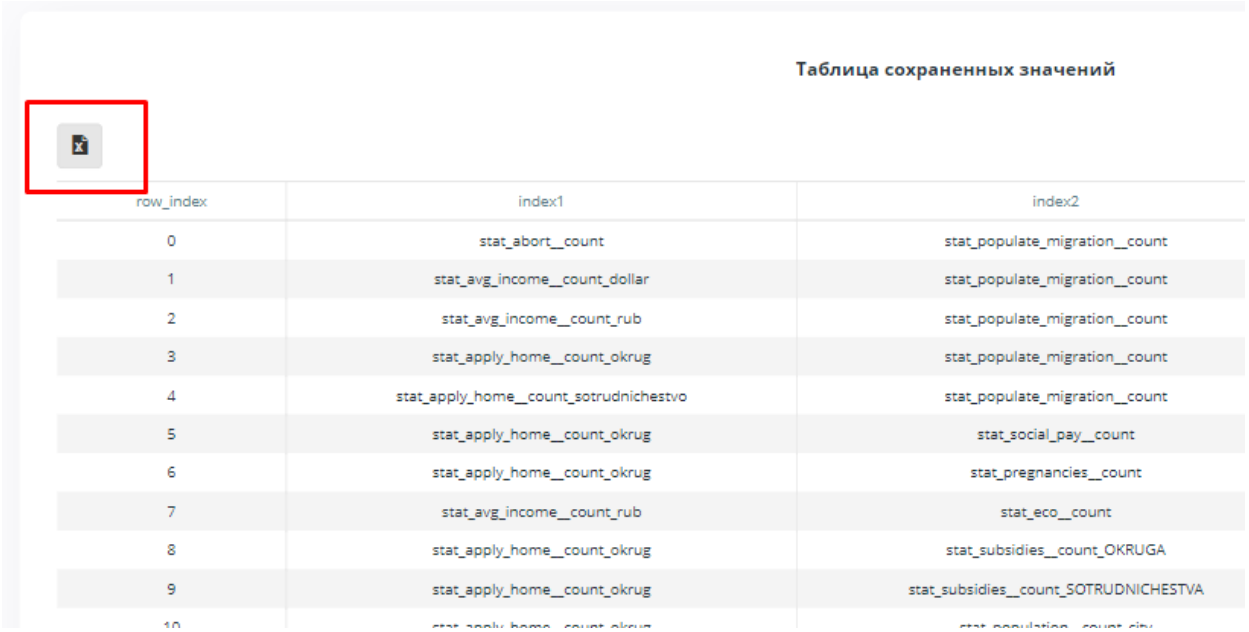


Рисунок 16 – Блок «Выбор сохраненного значения»



row_index	index1	index2
0	stat_abort__count	stat_populate_migration__count
1	stat_avg_income__count_dollar	stat_populate_migration__count
2	stat_avg_income__count_rub	stat_populate_migration__count
3	stat_apply_home__count_okrug	stat_populate_migration__count
4	stat_apply_home__count_sotrudnichestvo	stat_populate_migration__count
5	stat_apply_home__count_okrug	stat_social_pay__count
6	stat_apply_home__count_okrug	stat_pregnancies__count
7	stat_avg_income__count_rub	stat_eco__count
8	stat_apply_home__count_okrug	stat_subsidies__count_OKRUGA
9	stat_apply_home__count_okrug	stat_subsidies__count_SOTRUDNICHESTVA

Рисунок 17 – Экспортирование таблицы

Для того чтобы воспользоваться сервисом прогнозирования социально-экономического состояния региона, необходимо перейти по ссылке https://rms.admhmao.ru/viewer?dashboardGuid=978772b094184af3b1e7896e2a02ee5c&showNav=true&fit=true и авторизоваться (рисунок 18):

Рисунок 18 – Страница авторизации

После авторизации сразу попадаем на страницу «Строгие модели» (рисунок 19). На данном дашборде графически отображается модуль

прогнозирования социально-экономического состояния региона в сфере общего образования.

В блоке «Выбор статической модели» можно выбрать модель показателей, после чего на графике отобразится зависимость исследуемых параметров (рисунок 20).

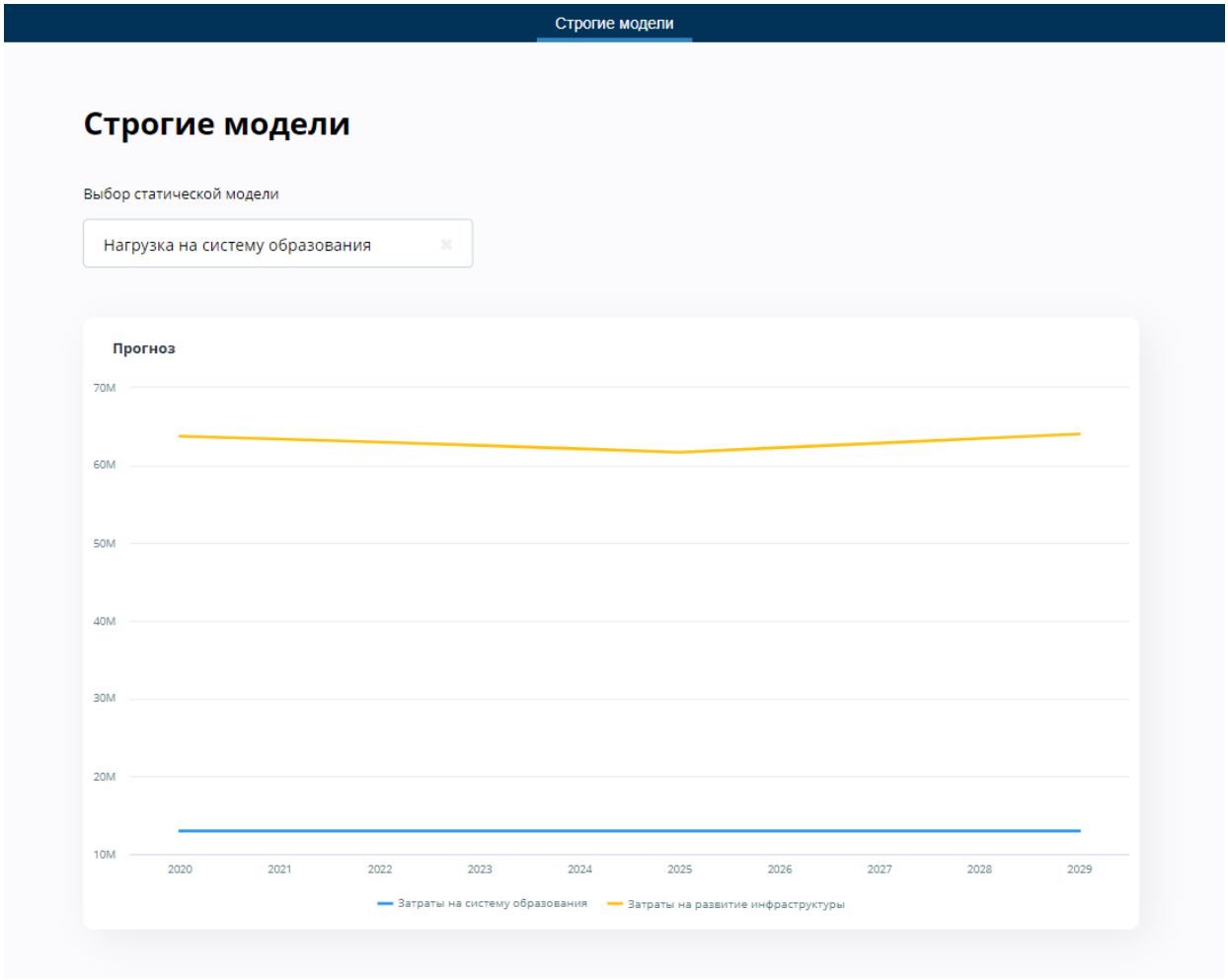


Рисунок 19 – Страница «Выбор модели»

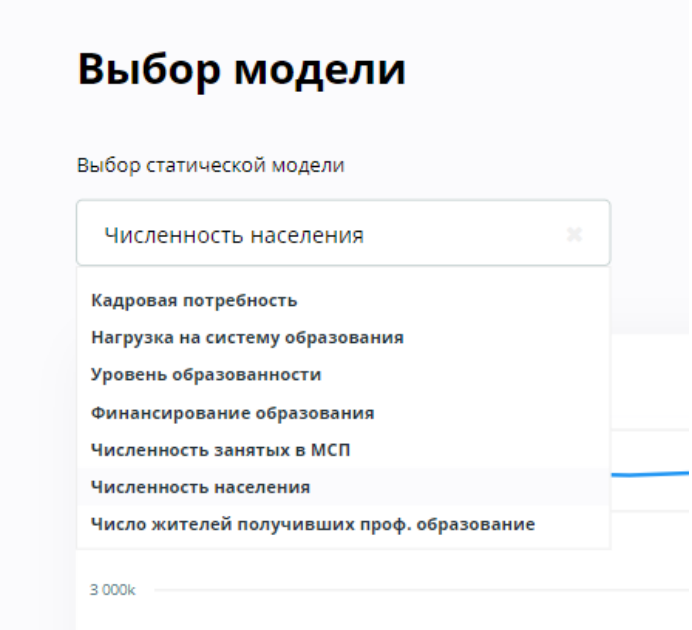


Рисунок 20 – Блок «Выбор статической модели»

Для того чтобы воспользоваться сервисом прогнозирования социально-экономического состояния региона, необходимо перейти по ссылке https://rms.admhmao.ru/viewer?dashboardGuid=66ba6b79959d40429e7401f275d6a96c&showNav=true&fit=true и авторизоваться (рисунок 21):

Рисунок 21 – Страница авторизации

После авторизации сразу попадаем на страницу «Скользящее окно на основе своего временного ряда» (рисунок 22). В блоке «Выбор параметра»

можно выбрать социально-экономический показатель, после чего на графике отобразится зависимость исследуемых параметров (рисунок 23).

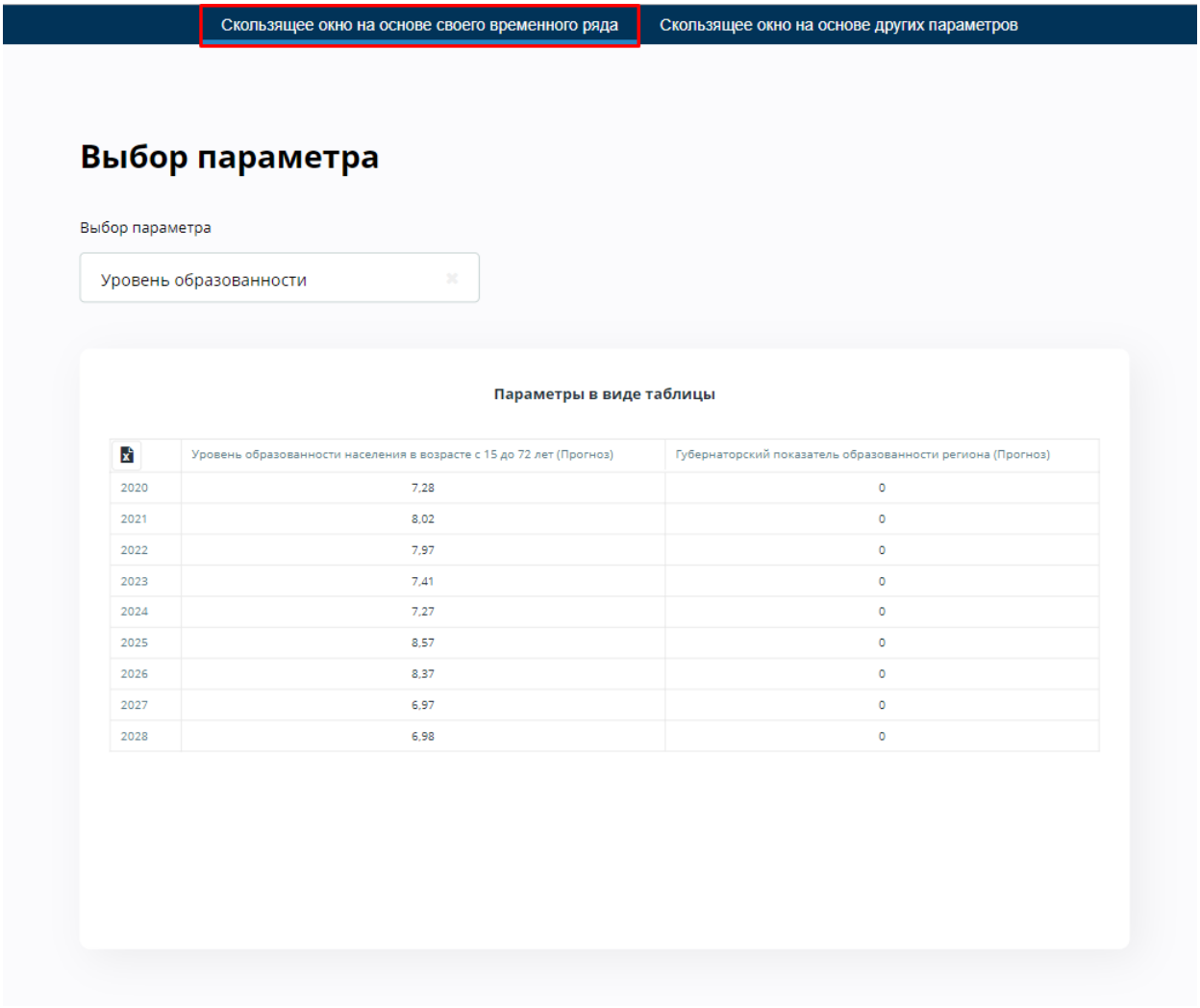


Рисунок 22 – Страница «Скользящее окно на основе своего временного ряда»

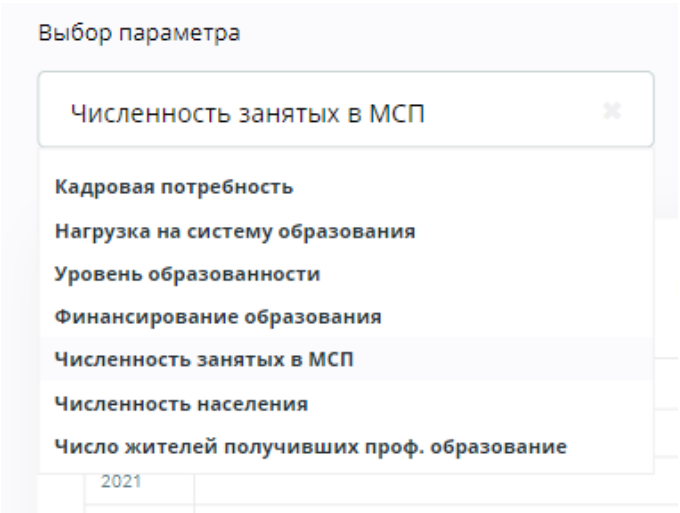
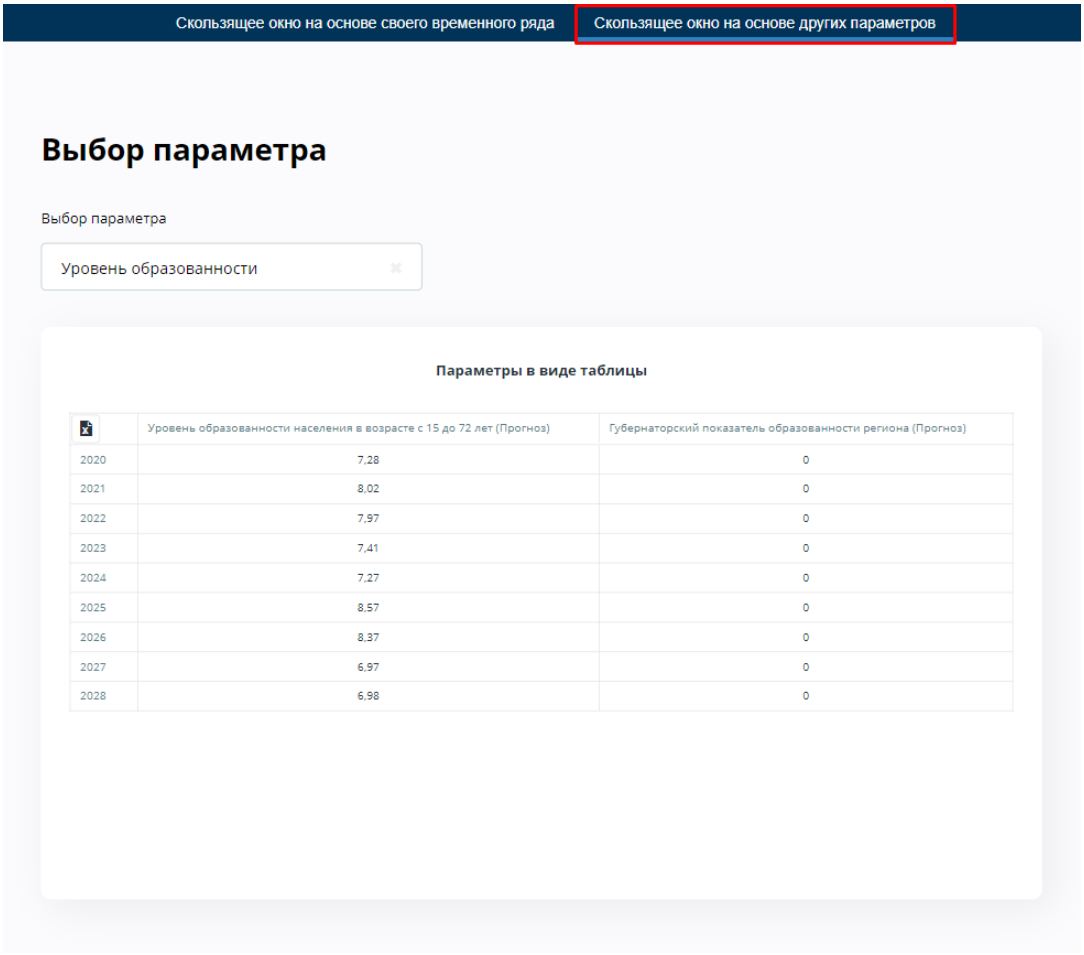


Рисунок 22 – Блок «Выбор параметра»

Страница «Скользящее окно на основе других параметров» (рисунок 23). В блоке «Выбор параметра» можно выбрать социально-экономический показатель, после чего на графике отобразится зависимость исследуемых параметров (рисунок 24).



	Уровень образованности населения в возрасте с 15 до 72 лет (Прогноз)	Губернаторский показатель образованности региона (Прогноз)
2020	7,28	0
2021	8,02	0
2022	7,97	0
2023	7,41	0
2024	7,27	0
2025	8,57	0
2026	8,37	0
2027	6,97	0
2028	6,98	0

Рисунок 23 – Страница «Скользящее окно на основе других параметров»

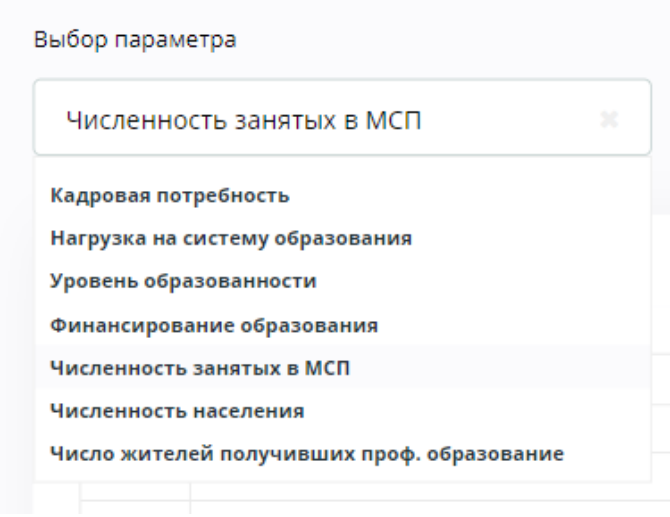


Рисунок 24 – Блок «Выбор параметра»

ЗАКЛЮЧЕНИЕ

В рамках проведенной научно-исследовательской работы получены следующие результаты:

1. Модуль регрессионного анализа, согласно пункту 1.1.3-а технического задания к договору № 05.5/19-ЮГУ-258 от 22.10.2019.
2. Модуль нейросетевого анализа, согласно пункту 1.1.3-б технического задания к договору № 05.5/19-ЮГУ-258 от 22.10.2019.
3. Модифицирован модуль нейросетевого анализа, согласно пункту 1.1.3-б технического задания к договору № 05.5/19-ЮГУ-258 от 22.10.2019.
4. Модуль модельного анализа данных, согласно пункту 1.1.3-в технические задания к договору № 05.5/19-ЮГУ-258 от 22.10.2019.
5. Программный модуль для работы с различными топологиями нейронной сети, согласно пункту 2 технического задания к договору № 05.5/19-ЮГУ-258 от 22.10.2019.
6. Реализован модуль формирования оптимальной архитектуры нейронной сети, согласно пункту 2 технического задания к договору № 05.5/19-ЮГУ-258 от 22.10.2019.
7. Реализован модуль анализа степени связанности данных, согласно пункту 1.1.2 технического задания к договору № 05.5/19-ЮГУ-258 от 22.10.2019.
8. Реализован модуль сбора данных, согласно пункту 1.1.1 технического задания к договору № 05.5/19-ЮГУ-258 от 22.10.2019.
9. Разработана и реализована информационная модель процесса прогнозирования социально-экономического состояния региона с точки зрения Пользователя.

Оценка полноты решений поставленных задач:

Работа выполнена согласно техническому заданию. Претензий со стороны Заказчика отсутствуют.

Рекомендации по конкретному использованию результатов НИР:

Полученные результаты НИР можно использовать в государственном и муниципальном управлении, а также при подготовке специалистов по направлению 09.03.01- «Информатика и вычислительная техника».

Оценка технико-экономической эффективности внедрения:

Разработанная технология позволила в 3 раза сократить ресурсные издержки органов региональной власти при решении задачи прогнозирования социально-экономических параметров региона.

Оценка научно-технического уровня выполненной НИР в сравнении с лучшими достижениями в этой области:

Результаты, полученные в ходе выполнения НИР, превосходят существующие аналоги.

Приложение 1. Программная реализация разработанных модулей

```
function alf=Adaptive_step(Architecture_NN,kk,k,Y,E_hidden_layer,F_A)

                                aaa=1;
                                aaa2=0;
                                aaa3=0;
                                for ii=1:Architecture_NN(kk+1)
                                   aaa=aaa+Y(ii,kk+1,k)*Y(ii,kk+1,k);
                                end
                                aaa=1/aaa;
                                switch F_A
                                case 1
                                    for jj=1:Architecture_NN(kk)
                                    aaa2=aaa2+Y(jj,kk,k)*(1-
Y(jj,kk,k))*E_hidden_layer(jj,kk)^2;
                                    aaa3=aaa3+Y(jj,kk,k)^2*(1-
Y(jj,kk,k))^2*E_hidden_layer(jj,kk)^2;
                                    end
                                alf=4*aaa*aaa2/aaa3;
                                case 2
                                    for jj=1:Architecture_NN(kk)
                                    aaa2=aaa2+Y(jj,kk,k)*(1-
Y(jj,kk,k))*E_hidden_layer(jj,kk)^2;
                                    aaa3=aaa3+Y(jj,kk,k)^2*(1-
Y(jj,kk,k))^2*E_hidden_layer(jj,kk)^2;
                                    end
                                alf=aaa*aaa2/aaa3;
                                case 3
                                alf=aaa;
                                end


function
alf=Adaptive_step_deep_NN(number_neurons,kk,k,Y,E_hidden_layer,F_A,A)
if kk==1
                                aaa=0;
                                aaa2=0;
                                aaa22=0;
                                aaa3=0;
                                for ii=1:number_neurons(kk+1)
                                   aaa=aaa+Y(ii,k,kk)*Y(ii,k,kk);
                                   aaa2=aaa2+E_hidden_layer(ii,k,kk)^2;
                                end
                                aaa=aaa+1;
                                aaa=1/aaa;
                                switch F_A
                                case 1
                                    for jj=1:number_neurons(kk)
                                    aaa22=aaa22+A(jj,k)*(1-A(jj,k));
                                    aaa3=aaa3+A(jj,k)^2*(1-A(jj,k))^2;
                                    end
                                aaa2=aaa2*aaa22;
                                aaa3=aaa3*aaa2;
                                alf=4*aaa*aaa2/aaa3;
                                case 2
                                    for jj=1:number_neurons(kk)
                                    aaa22=aaa22+A(jj,k)*(1-A(jj,k));
                                    aaa3=aaa3+A(jj,k)^2*(1-A(jj,k))^2;
                                    end
```

```
                                aaa2=aaa2*aaa22;
                                aaa3=aaa3*aaa2;
                                alf=aaa*aaa2/aaa3;
                                case 3
                                alf=aaa;
                                end

else
                                    aaa=0;
                                aaa2=0;
                                aaa22=0;
                                aaa3=0;
                                for ii=1:number_neurons(kk+1)
                                   aaa=aaa+Y(ii,k,kk)*Y(ii,k,kk);
                                   aaa2=aaa2+E_hidden_layer(ii,k,kk)^2;
                                end
                                aaa=aaa+1;
                                aaa=1/aaa;
                                switch F_A
                                case 1
                                    for jj=1:number_neurons(kk)
                                    aaa22=aaa22+Y(jj,k,kk)*(1-Y(jj,k,kk));
                                    aaa3=aaa3+Y(jj,k,kk)^2*(1-Y(jj,k,kk))^2;
                                    end
                                 aaa2=aaa2*aaa22;
                                 aaa3=aaa3*aaa2;
                                alf=4*aaa*aaa2/aaa3;
                                case 2
                                    for jj=1:number_neurons(kk)
                                    aaa22=aaa22+Y(jj,k,kk)*(1-Y(jj,k,kk));
                                    aaa3=aaa3+Y(jj,k,kk)^2*(1-Y(jj,k,kk))^2;
                                    end
                                aaa2=aaa2*aaa22;
                                aaa3=aaa3*aaa2;
                                alf=aaa*aaa2/aaa3;
                                case 3
                                alf=aaa;
                                end

end
if alf>1
    alf=1;
end

function
[F,F_E,F_W,F_T,Arhitecture_NN]=deep_neural_network_adap_FA(A,I,number_neurons
,value,F_A,time_or_iteration)
% A - обучающая выборка
% I - эталонные значения
% Iteration - количество итераций при финальном обучении
% number_neurons - количество нейроннов в сети
% alf - шаг обучения.
% F_A - функция активации;
% F - значения полученне обученной нейронной сетью
% Influence_matrix - матрица влияния входных данных на выходные данные
% F_E - средняя ошибка обучения нейронной сети в процентах
[~,m]=size(A);
F(1,1)=0;
[nn,~]=size(I);
alf=0.05;
[Arhitecture_NN,W,T]=Training_NN_Deep_belief_FA(A,number_neurons,I,alf,F_A);
if time_or_iteration==1
```

```
[Y,F_E,F_W,F_T]=Training_finish_time_adap_FA(W,T,A,I,value,Arhitecture_NN,F_A
);
else
[Y,F_E,F_W,F_T]=Training_finish_adap_FA(W,T,A,I,value,Arhitecture_NN,F_A);
end
[~,~,l]=size(Y);
for i=1:m
F(1:nn,i)=Y(1:nn,i,l);
end

function
[F,F_E,F_W,F_T,Arhitecture_NN]=deep_neural_network_FA(A,I,number_neurons,Iter
ation,alf,F_A)
% A - обучающая выборка
% I - эталонные значения
% Iteration - количество итераций при финальном обучении
% number_neurons - количество нейроннов в сети
% alf - шаг обучения.
% F_A - функция активации;
% F - значения полученне обученной нейронной сетью
% Influence_matrix - матрица влияния входных данных на выходные данные
% F_E - средняя ошибка обучения нейронной сети в процентах
[~,m]=size(A);
F(1,1)=0;
[nn,~]=size(I);
[Arhitecture_NN,W,T]=Training_NN_Deep_belief_FA(A,number_neurons,I,alf,F_A);
[Y,F_E,F_W,F_T]=Training_finish_FA(W,T,A,I,Iteration,Arhitecture_NN,alf,F_A);
[~,~,l]=size(Y);
for i=1:m
F(1:nn,i)=Y(1:nn,i,l);
end

function
[F,F_E,F_W,F_T,Arhitecture_NN]=deep_neural_network_time_FA(A,I,number_neurons
,time,alf,F_A)
% A - обучающая выборка
% I - эталонные значения
% Iteration - количество итераций при финальном обучении
% number_neurons - количество нейроннов в сети
% alf - шаг обучения.
% F_A - функция активации;
% F - значения полученне обученной нейронной сетью
% Influence_matrix - матрица влияния входных данных на выходные данные
% F_E - средняя ошибка обучения нейронной сети в процентах
[~,m]=size(A);
F(1,1)=0;
[nn,~]=size(I);
[Arhitecture_NN,W,T]=Training_NN_Deep_belief_FA(A,number_neurons,I,alf,F_A);
[Y,F_E,F_W,F_T]=Training_finish_time_FA(W,T,A,I,time,Arhitecture_NN,alf,F_A);
[~,~,l]=size(Y);
for i=1:m
F(1:nn,i)=Y(1:nn,i,l);
end

function
[A,I,Conversion_rate]=forecast_data_processing(Input_matrix_data,Output_matri
x_data,activation_function)
% forecast_data_processing - функция обрабатывает данные для дальнейшего
прогнозирования/моделирования с использованием нейронных сетей
% Input_matrix_data - матрица влияющих параметров
% Input_vektor_data - вектор исследуемых параметров
```

```
% activation_function - функция активации. 1 - сигмоидная, 2 - биполярная,
сигмоидная, 3 - гиперболический тангенс, 4 - линейная.

[n,m]=size(Input_matrix_data);
[n1,~]=size(Output_matrix_data);
A(n,m)=0;
I(n1,m)=0;
Conversion_rate(max(n,n1),2)=0;
                    for i=1:n

Conversion_rate(i,1)=4*max(abs(Input_matrix_data(i,:)))/3;
                    A(i,:)=Input_matrix_data(i,:)/Conversion_rate(i,1);
                    end
switch activation_function
              case 1
                    for i=1:n1
                    min_c_r=min(Output_matrix_data(i,:));
                    if   min_c_r<0
                    d_c_r=2*abs(min_c_r);
                    else
                    d_c_r=0;
                    end
                    Conversion_rate(i,2)=d_c_r;

Conversion_rate(i,3)=4*max(Output_matrix_data(i,:)+d_c_r)/3;

I(i,:)=(Output_matrix_data(i,:)+d_c_r)/Conversion_rate(i,3);
                    end
              case 2
                    for i=1:n1

Conversion_rate(i,3)=4*max(abs(Output_matrix_data(i,:)))/3;
                    Conversion_rate(i,2)=0;
                    I(i,:)=Output_matrix_data(i,:)/Conversion_rate(i,3);
                    end
              case 3
                    for i=1:n1
                    min_c_r=min(Output_matrix_data(i,:));
                    if   min_c_r<0
                    d_c_r=2*abs(min_c_r);
                    else
                    d_c_r=0;
                    end
                    Conversion_rate(i,2)=d_c_r;

Conversion_rate(i,3)=4*max(Output_matrix_data(i,:)+d_c_r)/3;

I(i,:)=(Output_matrix_data(i,:)+d_c_r)/Conversion_rate(i,3);
                    end
end

function
Forecast_data=forward_phase_deep_neural_network_(A,W,T,Arhitecture_NN,F_A,Con
version_rate)
[n,m]=size(A);
A=A./Conversion_rate(1:n,1);
Y=W.*0;
[~,l]=size(Arhitecture_NN);
b=Arhitecture_NN(2,l);
Forecast_data(b,m)=0;
F(b,m)=0;
 switch F_A
```

```
                case 1
                      for k=1:m
                          for sloy=1:l
                             for i=1:Arhitecture_NN(2,sloy)
                                 Y(i,k,sloy)=0;
                                 if sloy==1
                                      for j=1:Arhitecture_NN(1,sloy)

Y(i,k,sloy)=Y(i,k,sloy)+A(j,k)*W(j,i,sloy);
                                      end
                                  Y(i,k,sloy)=Y(i,k,sloy)-T(i,sloy);
                                  Y(i,k,sloy)=1/(1+exp(-Y(i,k,sloy)));
                                 else
                                     for j=1:Arhitecture_NN(1,sloy)
                                          Y(i,k,sloy)=Y(i,k,sloy)+Y(j,k,sloy-
1)*W(j,i,sloy);
                                      end
                                  Y(i,k,sloy)=Y(i,k,sloy)-T(i,sloy);
                                  Y(i,k,sloy)=1/(1+exp(-Y(i,k,sloy)));

                                 end
                             end
                          end
                          F(:,k)=Y(1:Arhitecture_NN(2,l),k,sloy);

Forecast_data(1:b,k)=(F(:,k).*Conversion_rate(1:b,3))-Conversion_rate(1:b,2);
                      end
                 case 2
                       for k=1:m
                          for sloy=1:l
                             for i=1:Arhitecture_NN(2,sloy)
                                 Y(i,k,sloy)=0;
                                 if sloy==1
                                      for j=1:Arhitecture_NN(1,sloy)

Y(i,k,sloy)=Y(i,k,sloy)+A(j,k)*W(j,i,sloy);
                                      end
                                  Y(i,k,sloy)=Y(i,k,sloy)-T(i,sloy);
                                  Y(i,k,sloy)=2/(1+exp(-Y(i,k,sloy)))-1;
                                 else
                                     for j=1:Arhitecture_NN(1,sloy)
                                          Y(i,k,sloy)=Y(i,k,sloy)+Y(j,k,sloy-
1)*W(j,i,sloy);
                                      end
                                  Y(i,k,sloy)=Y(i,k,sloy)-T(i,sloy);
                                  Y(i,k,sloy)=2/(1+exp(-Y(i,k,sloy)))-1;

                                 end
                             end
                          end
                          F(:,k)=Y(1:Arhitecture_NN(2,l),k,sloy);

Forecast_data(1:b,k)=(F(:,k).*Conversion_rate(1:b,3))-Conversion_rate(1:b,2);
                      end
                case 3
                       for k=1:m
                          for sloy=1:l
                             for i=1:Arhitecture_NN(2,sloy)
                                 Y(i,k,sloy)=0;
                                 if sloy==1
                                      for j=1:Arhitecture_NN(1,sloy)
```

```
Y(i,k,sloy)=Y(i,k,sloy)+A(j,k)*W(j,i,sloy);
                                    end
                                Y(i,k,sloy)=Y(i,k,sloy)-T(i,sloy);
                                Y(i,k,sloy)=(exp(Y(i,k,sloy))-exp(-
Y(i,k,sloy)))/(exp(Y(i,k,sloy))+exp(-Y(i,k,sloy)));
                               else
                                   for j=1:Arhitecture_NN(1,sloy)
                                        Y(i,k,sloy)=Y(i,k,sloy)+Y(j,k,sloy-
1)*W(j,i,sloy);
                                    end
                                Y(i,k,sloy)=Y(i,k,sloy)-T(i,sloy);
                                Y(i,k,sloy)=(exp(Y(i,k,sloy))-exp(-
Y(i,k,sloy)))/(exp(Y(i,k,sloy))+exp(-Y(i,k,sloy)));

                               end
                           end
                       end
                       F(:,k)=Y(1:Arhitecture_NN(2,l),k,sloy);

Forecast_data(1:b,k)=(F(:,k).*Conversion_rate(1:b,3))-Conversion_rate(1:b,2);
                   end
 end

function [Average_error,N_o_P,E_output_layer,W,T]=learning_iteration(W,T,T_S,
R_V,Architecture_NN,alf,F_A)
% learning_iteration - одна итерация обучения всей выборки
% F_A - функция активации. 1 - сигмоидная, 2 - биполярная сигмоидная, 3 -
гиперболический тангенс.
% average_error - средняя ошибка
layer=length(Architecture_NN);
mm=max(Architecture_NN);
nn=Architecture_NN(1);
[nnn,m]=size(R_V);
Y(mm,layer,m)=0;
E_output_layer(nnn,m)=0;
E_hidden_layer(mm,layer)=0;
N_o_P(nnn,m)=0;
Average_error(nnn,1)=0;
for k=1:m
    Y(1:nn,1,k)=T_S(:,k);
    for kk=2:layer
        a=Architecture_NN(kk-1);
        b=Architecture_NN(kk);
        for i=1:b
                for j=1:a
                    Y(i,kk,k)=Y(i,kk,k)+Y(j,kk-1,k)*W(j,i,kk-1);
                end
                    Y(i,kk,k)=Y(i,kk,k)-T(i,kk-1);
              switch F_A
              case 1
                 Y(i,kk,k)=1/(1+exp(-Y(i,kk,k)));
              case 2
                 Y(i,kk,k)=2/(1+exp(-Y(i,kk,k)))-1;
              case 3
                 Y(i,kk,k)=(exp(Y(i,kk,k))-exp(-
Y(i,kk,k)))/(exp(Y(i,kk,k))+exp(-Y(i,kk,k)));
              end
        end
    end

                for j=1:b
```

```
                E_output_layer(j,k)=Y(j,layer,k)-R_V(j,k);
                end
                E_hidden_layer(1:b,layer)=E_output_layer(:,k);
                if layer>2
                for kk=layer-1:-1:2
                    a=Architecture_NN(kk);
                    b=Architecture_NN(kk+1);
                for i=1:a
                        for j=1:b
                             switch F_A
                             case 1

E_hidden_layer(i,kk)=E_hidden_layer(i,kk)+E_hidden_layer(j,kk+1)*Y(j,kk+1,k)*
(1-Y(j,kk+1,k))*W(i,j,kk);
                             case 2

E_hidden_layer(i,kk)=E_hidden_layer(i,kk)+E_hidden_layer(j,kk+1)*2*Y(j,kk+1,k
)*(1-Y(j,kk+1,k))*W(i,j,kk);
                             case 3

E_hidden_layer(i,kk)=E_hidden_layer(i,kk)+E_hidden_layer(j,kk+1)*(1-
Y(j,kk+1,k)*Y(j,kk+1,k))*W(i,j,kk);
                             end

                        end
                end
                end
                end

                for kk=layer:-1:2
                    a=Architecture_NN(kk-1);
                    b=Architecture_NN(kk);
                    for i=1:a
                             switch F_A
                             case 1
                             T(i,kk-1)=T(i,kk-
1)+alf*E_hidden_layer(i,kk)*Y(i,kk,k)*(1-Y(i,kk,k));
                             case 2
                             T(i,kk-1)=T(i,kk-
1)+alf*E_hidden_layer(i,kk)*2*Y(i,kk,k)*(1-Y(i,kk,k));
                             case 3
                             T(i,kk-1)=T(i,kk-1)+alf*E_hidden_layer(i,kk)*(1-
Y(i,kk,k)*Y(i,kk,k));
                             end
                    for j=1:b
                             switch F_A
                             case 1
                             W(i,j,kk-1)=W(i,j,kk-1)-
alf*E_hidden_layer(j,kk)*Y(j,kk,k)*(1-Y(j,kk,k))*Y(i,kk-1,k);
                             case 2
                             W(i,j,kk-1)=W(i,j,kk-1)-
alf*E_hidden_layer(j,kk)*2*Y(j,kk,k)*(1-Y(j,kk,k))*Y(i,kk-1,k);
                             case 3
                             W(i,j,kk-1)=W(i,j,kk-1)-
alf*E_hidden_layer(j,kk)*(1-Y(j,kk,k)*Y(j,kk,k))*Y(i,kk-1,k);
                             end

                    end
                    end
                end

            E_hidden_layer=E_hidden_layer.*0;
```

```
            N_o_P(:,k)=Y(1:nnn,layer,k);
end
            Average_error(1:nnn,1)=sum(abs(E_output_layer'))/m;

function
[F,F_E,F_W,F_T,Arhitecture_NN]=NSGD(A,I,number_neurons,time_limit,alf)
% A - обучающая выборка
% I - эталонные значения
% time_limit - планируемое время обучения НС
% number_neurons - количество нейроннов в скрытом слое
% alf - шаг обучения.
% II=I;
[~,m]=size(A);
F(1,1)=0;
[n,~]=size(A);
[nn,~]=size(I);
number_neurons=[n,number_neurons,nn];
[Arhitecture_NN,W,T]=Training_NN_Deep_belief(A,number_neurons,I,alf);
[Y,F_E,F_W,F_T]=Training_finish(W,T,A,I,time_limit,Arhitecture_NN,alf);
[~,~,l]=size(Y);
for i=1:m
F(1:nn,i)=Y(1:nn,i,l);
end
% EEE=0;
% % EE=round(F);
% % EEE=sum(abs(EE-II));
% for i=1:m
% if F(i)<0.25
%    EE(i)=0;
% else
%     if F(i)<0.75
%     EE(i)=0.5;
%     else
%     EE(i)=1;
%     end
% end
% if EE(i)==II(i)
% else
%     EEE=EEE+1;
% end
% end


function
[Influence_matrix,N_o_P_f]=Opposite_data_conversion(R_V,F,Conversion_rate,W_f
inish,T_finish,number_neurons,F_A)
% функция делает обратное преобразовние данных и определяет матрицу влияния
% F - выходные значения нейронной сети
% N_o_P_f - преобразованные значения нейронной сети


n=number_neurons(1);
[m,~]=size(R_V);
T_S_one_signal(n,1)=0;
Influence_matrix(n,m)=0;
for i=1:n
T_S_one_signal(i,1)=1;
[~,N_o_P,~,~,~]=learning_iteration(W_finish,T_finish,T_S_one_signal,R_V(:,1)
,number_neurons,0.05,F_A);
T_S_one_signal(i,1)=T_S_one_signal(i,1)*0;
Influence_matrix(i,1:m)=N_o_P;
end
```

```
Influence_matrix=Influence_matrix*100;
% преобразуем выходные значения нейронной сети
N_o_P_f=(F.*Conversion_rate(1:m,3))-Conversion_rate(1:m,2);

function
[Forecast_data,Influence_matrix,Conversion_rate,W_finish,T_finish,N_o_P_f,A_e
_P,Arhitecture_NN]=Prognoz_deep_NN_adap(Input_matrix_data,Output_matrix_data,
Forecast_data_matrix,F_A, number_neurons,value, time_or_iteration)
% Prognoz_neyro_multilayer_time_no_adap - прогнозирование с использованием
многослойной нейронной сети, постоянным шагом обучения и критерий остановки
алгоритма - время
% Training_sample(T_S) - обучающая выборка
% Reference_values(R_V) - эталонные значения
% Activation function (F_A) - функция активации
% network output pattern (N_o_P)- выходные значения нейронной сети
% average_error_percentage (A_e_P) - средняя ошибка обучения нейронной сети в
процентах
% number_of_iterations(N_o_I) - количество итераций
% Influence_matrix - матрица влияния входных данных на выходные данные
% Forecast_data_matrix - матрица данных для прогнозирования
% Conversion_rate - данные для обратного преобразования
% forward_phase - фаза прямого прохождения сигнала

% преобразование входных данных для прогнозирования
[T_S,R_V,Conversion_rate]=forecast_data_processing(Input_matrix_data,Output_m
atrix_data,F_A);

% обучение нейронной сети
[F,A_e_P,W_finish,T_finish,Arhitecture_NN]=deep_neural_network_adap_FA(T_S,R_
V,number_neurons,value,F_A,time_or_iteration);
[Influence_matrix,N_o_P_f]=Opposite_data_conversion(R_V,F,Conversion_rate,W_f
inish,T_finish,number_neurons,F_A);

% фаза прямого прохождение данных по обученной нейронной сети
Forecast_data=forward_phase_deep_neural_network_(Forecast_data_matrix,W_finis
h,T_finish,Arhitecture_NN,F_A,Conversion_rate);

function
[Forecast_data,Influence_matrix,Conversion_rate,W_finish,T_finish,N_o_P_f,A_e
_P,Arhitecture_NN]=Prognoz_deep_NN_iterations_no_adap(Input_matrix_data,Outpu
t_matrix_data,Forecast_data_matrix,F_A, number_neurons,Iteration,alf)
% Prognoz_neyro_multilayer_time_no_adap - прогнозирование с использованием
многослойной нейронной сети, постоянным шагом обучения и критерий остановки
алгоритма - время
% Training_sample(T_S) - обучающая выборка
% Reference_values(R_V) - эталонные значения
% Activation function (F_A) - функция активации
% network output pattern (N_o_P)- выходные значения нейронной сети
% average_error_percentage (A_e_P) - средняя ошибка обучения нейронной сети в
процентах
% number_of_iterations(N_o_I) - количество итераций
% Influence_matrix - матрица влияния входных данных на выходные данные
% Forecast_data_matrix - матрица данных для прогнозирования
% Conversion_rate - данные для обратного преобразования
% forward_phase - фаза прямого прохождения сигнала

% преобразование входных данных для прогнозирования
[T_S,R_V,Conversion_rate]=forecast_data_processing(Input_matrix_data,Output_m
atrix_data,F_A);

% обучение нейронной сети
```

```
[F,A_e_P,W_finish,T_finish,Arhitecture_NN]=deep_neural_network_FA(T_S,R_V,num
ber_neurons,Iteration,alf,F_A);
[Influence_matrix,N_o_P_f]=Opposite_data_conversion(R_V,F,Conversion_rate,W_f
inish,T_finish,number_neurons,F_A);


% фаза прямого прохождение данных по обученной нейронной сети
Forecast_data=forward_phase_deep_neural_network_(Forecast_data_matrix,W_finis
h,T_finish,Arhitecture_NN,F_A,Conversion_rate);



function
[Forecast_data,Influence_matrix,Conversion_rate,W_finish,T_finish,N_o_P_f,A_e
_P,Arhitecture_NN]=Prognoz_deep_NN_time_no_adap(Input_matrix_data,Output_matr
ix_data,Forecast_data_matrix,F_A, number_neurons,time,alf)
% Prognoz_neyro_multilayer_time_no_adap - прогнозирование с использованием
многослойной нейронной сети, постоянным шагом обучения и критерий остановки
алгоритма - время
% Training_sample(T_S) - обучающая выборка
% Reference_values(R_V) - эталонные значения
% Activation function (F_A) - функция активации
% network output pattern (N_o_P)- выходные значения нейронной сети
% average_error_percentage (A_e_P) - средняя ошибка обучения нейронной сети в
процентах
% number_of_iterations(N_o_I) - количество итераций
% Influence_matrix - матрица влияния входных данных на выходные данные
% Forecast_data_matrix - матрица данных для прогнозирования
% Conversion_rate - данные для обратного преобразования
% forward_phase - фаза прямого прохождения сигнала

% преобразование входных данных для прогнозирования
[T_S,R_V,Conversion_rate]=forecast_data_processing(Input_matrix_data,Output_m
atrix_data,F_A);

% обучение нейронной сети
[F,A_e_P,W_finish,T_finish,Arhitecture_NN]=deep_neural_network_time_FA(T_S,R_
V,number_neurons,time,alf,F_A);
[Influence_matrix,N_o_P_f]=Opposite_data_conversion(R_V,F,Conversion_rate,W_f
inish,T_finish,number_neurons,F_A);


% фаза прямого прохождение данных по обученной нейронной сети
Forecast_data=forward_phase_deep_neural_network_(Forecast_data_matrix,W_finis
h,T_finish,Arhitecture_NN,F_A,Conversion_rate);



function
[min_A_e_P,Architecture_NN_F,F_A_f,alf_f]=Selection_architecture_deep_NN(Inpu
t_matrix_data,Output_matrix_data)
[n,~]=size(Input_matrix_data);
[n1,~]=size(Output_matrix_data);
Architecture_NN(1)=n;
iterations=20;
min_A_e_P=10^6;
for jj=1:3
F_A=jj;
[T_S,R_V,~]=forecast_data_processing(Input_matrix_data,Output_matrix_data,F_A
);
for ii=1:10
    alf=ii/50;
```

```
                    for i=n+1:2*n+1
                    Architecture_NN(2)=i;
                    Architecture_NN(3)=n1;
                    [~,A_e_P,~,~,~]=deep_neural_network_FA(T_S,
R_V,Architecture_NN,iterations,alf,F_A);
                    if A_e_P<min_A_e_P
                    min_A_e_P=A_e_P;
                    Architecture_NN_F=Architecture_NN;
                    F_A_f=F_A;
                    alf_f=alf;
                    end
                    end
                    if n>3
                        for i=n+1:2*n+1
                            for j=i-1:-1:2
                        Architecture_NN(2)=i;
                        Architecture_NN(3)=j;
                        Architecture_NN(4)=n1;
                        [~,A_e_P,~,~,~]=deep_neural_network_FA(T_S,
R_V,Architecture_NN,iterations,alf,F_A);
                            if A_e_P<min_A_e_P
                            min_A_e_P=A_e_P;
                            Architecture_NN_F=Architecture_NN;
                            F_A_f=F_A;
                            alf_f=alf;
                            end
                            end
                        end
                        for i=n+1:2*n+1
                            for j=i-1:-1:2
                                for ji=j-1:-1:2
                                    Architecture_NN(2)=i;
                                    Architecture_NN(3)=j;
                                    Architecture_NN(4)=ji;
                                    Architecture_NN(5)=n1;

[~,A_e_P,~,~,~]=deep_neural_network_FA(T_S,
R_V,Architecture_NN,iterations,alf,F_A);
                                    if A_e_P<min_A_e_P
                                    min_A_e_P=A_e_P;
                                    Architecture_NN_F=Architecture_NN;
                                    F_A_f=F_A;
                                    alf_f=alf;
                                    end
                                end
                            end
                        end
                    end




end
end

function [W,T,AA]=Training_Avto(A,a,alf)
[n,m]=size(A);

w1=2*rand(n,a)-1;
w2=2*rand(a,n)-1;
```

```
T1=2*rand(a,1)-1;
T2=2*rand(n,1)-1;

y1(a,m)=0;
y2(n,m)=0;
E_vih_sloy(n,m)=0;
E_skrt_sloy(a,m)=0;

for sss=1:100

for k=1:m
            for i=1:a
                y1(i,k)=0;
                for j=1:n
                    y1(i,k)=y1(i,k)+A(j,k)*w1(j,i);
                end
            y1(i,k)=y1(i,k)-T1(i);
            y1(i,k)=1/(1+exp(-y1(i,k)));
            end

                for j=1:n
                for i=1:a
                    y2(j,k)=y2(j,k)+w2(i,j)*y1(i,k);
                end

                y2(j,k)=y2(j,k)-T2(j);
                y2(j,k)=1/(1+exp(-y2(j,k)));
                E_vih_sloy(j,k)=y2(j,k)-A(j,k);
                end

            for i=1:a
                E_skrt_sloy(i,k)=0;
                for j=1:n

E_skrt_sloy(i,k)=E_skrt_sloy(i,k)+E_vih_sloy(j,k)*y2(j,k)*(1-
y2(j,k))*w2(i,j);
                end
            end
            for i=1:n
                for j=1:a
                    w1(i,j)=w1(i,j)-alf*E_skrt_sloy(j,k)*y1(j,k)*(1-
y1(j,k))*A(i,k);
                    T1(j)=T1(j)+alf*E_skrt_sloy(j,k)*y1(j,k)*(1-y1(j,k));
                end
            end

            for j=1:n
            for i=1:a
                    w2(i,j)=w2(i,j)-alf*E_vih_sloy(j,k)*y2(j,k)*(1-
y2(j,k))*y1(i,k);
                     T2(j)=T2(j)+alf*E_vih_sloy(j,k)*y2(j,k)*(1-y2(j,k));
            end
            end
end

end
AA(a,m)=0;
for k=1:m
for i=1:a

                for j=1:n
```

```
                    AA(i,k)=AA(i,k)+A(j,k)*w1(j,i);
                end
            AA(i,k)=AA(i,k)-T1(i);
            AA(i,k)=1/(1+exp(-AA(i,k)));
end
end

W=w1;
T=T1;

function [W,T]=Training_Avto_end(A,gauge,alf)
[n,m]=size(A);
[a,~]=size(gauge);
% w1(n,a)=0;
% T1(a,1)=0;


w1=2*rand(n,a)-1;
T1=2*rand(a,1)-1;

% for i=1:n
%     for j=1:a
%     w1(i,j)=0.9*i*j/(n*a);
%     end
% end
% for j=1:a
%     T1(j,1)=0.9*j/a;
% end


y1(a,m)=0;
E_vih_sloy(a,m)=0;


for sss=1:100

for k=1:m
            for i=1:a
                y1(i,k)=0;
                for j=1:n
                    y1(i,k)=y1(i,k)+A(j,k)*w1(j,i);
                end
            y1(i,k)=y1(i,k)-T1(i);
            y1(i,k)=1/(1+exp(-y1(i,k)));
            E_vih_sloy(i,k)=y1(i,k)-gauge(i,k);
            end

            for i=1:n
                for j=1:a
                    w1(i,j)=w1(i,j)-alf*E_vih_sloy(j,k)*y1(j,k)*(1-
y1(j,k))*A(i,k);
                    T1(j)=T1(j)+alf*E_vih_sloy(j,k)*y1(j,k)*(1-y1(j,k));
                end
            end

end

end
W=w1;
T=T1;
```

```
function [W,T]=Training_Avto_end_FA(A,gauge,alf,F_A)
[n,m]=size(A);
[a,~]=size(gauge);

 w1=2*rand(n,a)-1;
T1=2*rand(a,1)-1;

y1(a,m)=0;
E_vih_sloy(a,m)=0;

for sss=1:100

for k=1:m
            for i=1:a
                y1(i,k)=0;
                for j=1:n
                    y1(i,k)=y1(i,k)+A(j,k)*w1(j,i);
                end
                switch F_A
                case 1
                y1(i,k)=1/(1+exp(-y1(i,k)));
                E_vih_sloy(i,k)=y1(i,k)-gauge(i,k);
                case 2
                y1(i,k)=2/(1+exp(-y1(i,k)))-1;
                E_vih_sloy(i,k)=y1(i,k)-gauge(i,k);
                case 3
                y1(i,k)=(exp(y1(i,k))-exp(-y1(i,k)))/(exp(y1(i,k))+exp(-
y1(i,k)));
                E_vih_sloy(i,k)=y1(i,k)-gauge(i,k);
                end
            end


             for j=1:a

                                switch F_A
                                case 1
                                T1(j)=T1(j)+alf*E_vih_sloy(j,k)*y1(j,k)*(1-
y1(j,k));
                                case 2
                                T1(j)=T1(j)+alf*E_vih_sloy(j,k)*2*y1(j,k)*(1-
y1(j,k));
                                case 3
                                T1(j)=T1(j)+alf*E_vih_sloy(j,k)*(1-
y1(j,k)*y1(j,k));
                                end
                 for i=1:n

                                switch F_A
                                case 1
                                w1(i,j)=w1(i,j)-alf*E_vih_sloy(j,k)*y1(j,k)*(1-
y1(j,k))*A(i,k);
                                case 2
                                w1(i,j)=w1(i,j)-
alf*E_vih_sloy(j,k)*2*y1(j,k)*(1-y1(j,k))*A(i,k);
                                case 3
                                w1(i,j)=w1(i,j)-alf*E_vih_sloy(j,k)*(1-
y1(j,k)*y1(j,k))*A(i,k);
                                end
```

```
            end
        end

end

end


W=w1;
T=T1;

function [W,T,AA]=Training_Avto_FA(A,a,alf,F_A)
[n,m]=size(A);

w1=2*rand(n,a)-1;
w2=2*rand(a,n)-1;
T1=2*rand(a,1)-1;
T2=2*rand(n,1)-1;

y1(a,m)=0;
y2(n,m)=0;
E_vih_sloy(n,m)=0;
E_skrt_sloy(a,m)=0;

for sss=1:100

for k=1:m
        for i=1:a
            y1(i,k)=0;
            for j=1:n
                y1(i,k)=y1(i,k)+A(j,k)*w1(j,i);
            end
            y1(i,k)=y1(i,k)-T1(i);
            switch F_A
            case 1
            y1(i,k)=1/(1+exp(-y1(i,k)));
            case 2
            y1(i,k)=2/(1+exp(-y1(i,k)))-1;
            case 3
            y1(i,k)=(exp(y1(i,k))-exp(-y1(i,k)))/(exp(y1(i,k))+exp(-
y1(i,k)));
            end
        end

            for j=1:n
            for i=1:a
                y2(j,k)=y2(j,k)+w2(i,j)*y1(i,k);
            end
            y2(j,k)=y2(j,k)-T2(j);
            switch F_A
            case 1
            y2(j,k)=1/(1+exp(-y2(j,k)));
            E_vih_sloy(j,k)=y2(j,k)-A(j,k);
            case 2
            y2(j,k)=2/(1+exp(-y2(j,k)))-1;
            E_vih_sloy(j,k)=y2(j,k)-A(j,k);
            case 3
            y2(j,k)=(exp(y2(j,k))-exp(-y2(j,k)))/(exp(y2(j,k))+exp(-
y2(j,k)));
            E_vih_sloy(j,k)=y2(j,k)-A(j,k);
            end
```

```
                end

            for i=1:a
                E_skrt_sloy(i,k)=0;
                for j=1:n
                        switch F_A
                        case 1

E_skrt_sloy(i,k)=E_skrt_sloy(i,k)+E_vih_sloy(j,k)*y2(j,k)*(1-
y2(j,k))*w2(i,j);
%
E_hidden_layer(i,kk)=E_hidden_layer(i,kk)+E_hidden_layer(j,kk+1)*Y(j,kk+1,k)*
(1-Y(j,kk+1,k))*W(i,j,kk);
                        case 2

E_skrt_sloy(i,k)=E_skrt_sloy(i,k)+E_vih_sloy(j,k)*2*y2(j,k)*(1-
y2(j,k))*w2(i,j);
%
E_hidden_layer(i,kk)=E_hidden_layer(i,kk)+E_hidden_layer(j,kk+1)*2*Y(j,kk+1,k
)*(1-Y(j,kk+1,k))*W(i,j,kk);
                        case 3
                        E_skrt_sloy(i,k)=E_skrt_sloy(i,k)+E_vih_sloy(j,k)*(1-
y2(j,k)*y2(j,k))*w2(i,j);
%
E_hidden_layer(i,kk)=E_hidden_layer(i,kk)+E_hidden_layer(j,kk+1)*(1-
Y(j,kk+1,k)*Y(j,kk+1,k))*W(i,j,kk);
                        end



                end
            end
            for j=1:a
                                switch F_A
                                case 1
                                T1(j)=T1(j)+alf*E_skrt_sloy(j,k)*y1(j,k)*(1-
y1(j,k));
                                case 2
                                T1(j)=T1(j)+alf*E_skrt_sloy(j,k)*2*y1(j,k)*(1-
y1(j,k));
                                case 3
                                T1(j)=T1(j)+alf*E_skrt_sloy(j,k)*(1-
y1(j,k)*y1(j,k));
                                end
                for i=1:n
                                switch F_A
                                case 1
                                w1(i,j)=w1(i,j)-alf*E_skrt_sloy(j,k)*y1(j,k)*(1-
y1(j,k))*A(i,k);
                                case 2
                                w1(i,j)=w1(i,j)-
alf*E_skrt_sloy(j,k)*2*y1(j,k)*(1-y1(j,k))*A(i,k);
                                case 3
                                w1(i,j)=w1(i,j)-alf*E_skrt_sloy(j,k)*(1-
y1(j,k)*y1(j,k))*A(i,k);
                                end
                end
            end

            for j=1:n
                    switch F_A
                    case 1
                    T2(j)=T2(j)+alf*E_vih_sloy(j,k)*y2(j,k)*(1-y2(j,k));
```

```
                    case 2
                    T2(j)=T2(j)+alf*E_vih_sloy(j,k)*2*y2(j,k)*(1-y2(j,k));
                    case 3
                    T2(j)=T2(j)+alf*E_vih_sloy(j,k)*(1-y2(j,k)*y2(j,k));
                    end
            for i=1:a
                    switch F_A
                    case 1
                    w2(i,j)=w2(i,j)-alf*E_vih_sloy(j,k)*y2(j,k)*(1-
y2(j,k))*y1(i,k);
                    case 2
                    w2(i,j)=w2(i,j)-alf*E_vih_sloy(j,k)*2*y2(j,k)*(1-
y2(j,k))*y1(i,k);
                    case 3
                    w2(i,j)=w2(i,j)-alf*E_vih_sloy(j,k)*(1-
y2(j,k)*y2(j,k))*y1(i,k);
                    end
            end
            end
end

end
AA(a,m)=0;
for k=1:m
for i=1:a
                for j=1:n
                    AA(i,k)=AA(i,k)+A(j,k)*w1(j,i);
                end
                    AA(i,k)=AA(i,k)-T1(i);
                    switch F_A
                    case 1
                    AA(i,k)=1/(1+exp(-AA(i,k)));
                    case 2
                    AA(i,k)=2/(1+exp(-AA(i,k)))-1;
                    case 3
                    AA(i,k)=(exp(AA(i,k))-exp(-AA(i,k)))/(exp(AA(i,k))+exp(-
AA(i,k)));
                    end
end
end

W=w1;
T=T1;

function
[Influence_matrix,W_finish,T_finish,N_o_P_f,A_e_P,Arhitecture_NN_f,Conversion
_rate]=training_deep_neural_network_iterations_no_adap(Input_matrix_data,
Output_matrix_data, Architecture_NN,iterations,alf,F_A)
% training_neyro_multilayer_time_no_adap - обучение многослойной нейронной
сети, постоянным шагом обучения и критерий остановки алгоритма - время
% Training_sample(T_S) - обучаюшая выборка
% Reference_values(R_V) - эталонные значения
% Activation function (F_A) - функция активации
% network output pattern (N_o_P)- выходные значения нейронной сети
% average_error_percentage (A_e_P) - средняя ошибка обучения нейронной сети в
процентах
% number_of_iterations(N_o_I) - количество итераций
% E_NN - ошибка обучения нейронной сети в процентах
% преобразование входных данных для прогнозирования
[T_S,R_V,Conversion_rate]=forecast_data_processing(Input_matrix_data,Output_m
atrix_data,F_A);
[~,m]=size(T_S);
```

```
[nn,~]=size(R_V);
N_o_P_f(nn,m)=0;
number_neurons=Architecture_NN;
switch F_A
    case 1
[Arhitecture_NN_f,W,T]=Training_NN_Deep_belief(T_S,number_neurons,R_V,alf);
[Y,~,W_finish,T_finish]=Training_finish(W,T,T_S,R_V,iterations,Arhitecture_NN
_f,alf);
[~,~,l]=size(Y);
for i=1:m
N_o_P_f(1:nn,i)=Y(1:nn,i,l);
end
b=nn;
N_o_P_f=(N_o_P_f.*Conversion_rate(1:b,3))-Conversion_rate(1:b,2);
R_V=(R_V.*Conversion_rate(1:b,3))-Conversion_rate(1:b,2);
A_e_P(nn,1)=0;
% нормируем ошибку к 100 процентам.
    for j=1:nn
    A_e_P(j,1)=sum(abs((N_o_P_f-R_V)./R_V)*100);
    end
A_e_P= A_e_P/m;

n=Architecture_NN(1);
T_S_one_signal(n,1)=0;
Influence_matrix(n,1)=0;
Conversion_rate_ones=ones(n,3);
Conversion_rate_ones(1:n,2)=Conversion_rate_ones(1:n,2)*0;
for i=1:n
T_S_one_signal(i,1)=1;
N_o_P=forward_phase_deep_neural_network_(T_S_one_signal(:,1),W_finish,T_finis
h,Arhitecture_NN_f,F_A,Conversion_rate_ones);
T_S_one_signal(n,1)=T_S_one_signal(n,1)*0;
Influence_matrix(i,1:b)=N_o_P;
end
Influence_matrix=Influence_matrix*100;
    case 2

    case 3
end

function
[YY,F_E,F_W,F_T]=Training_finish(W,T,A,gauge,Iteration,Arhitecture_NN,alf)

[~,m]=size(A);
Y=W.*0;
Error_sloy=Y;
[~,l]=size(Arhitecture_NN);
E1=1000000000;
F_E(1,1)=0;
tt=1;
% tic
% while  toc<time_limit
ttt=0;
while  ttt<Iteration
ttt=ttt+1;
for k=1:m
    for sloy=1:l
       for i=1:Arhitecture_NN(2,sloy)
           Y(i,k,sloy)=0;
           if sloy==1
                for j=1:Arhitecture_NN(1,sloy)
                    Y(i,k,sloy)=Y(i,k,sloy)+A(j,k)*W(j,i,sloy);
```

```
                end
            Y(i,k,sloy)=Y(i,k,sloy)-T(i,sloy);
            Y(i,k,sloy)=1/(1+exp(-Y(i,k,sloy)));
           else
               for j=1:Arhitecture_NN(1,sloy)
                    Y(i,k,sloy)=Y(i,k,sloy)+Y(j,k,sloy-1)*W(j,i,sloy);
                end
            Y(i,k,sloy)=Y(i,k,sloy)-T(i,sloy);
            Y(i,k,sloy)=1/(1+exp(-Y(i,k,sloy)));

           end
       end
   end
        for i=1:Arhitecture_NN(2,l)
           Error_sloy(i,k,l)=Y(i,k,l)-gauge(i,k);
        end

        for sloy=l-1:-1:1

             for i=1:Arhitecture_NN(1,sloy+1)
                Error_sloy(i,k,sloy)=0;
                for j=1:Arhitecture_NN(2,sloy+1)

Error_sloy(i,k,sloy)=Error_sloy(i,k,sloy)+Error_sloy(j,k,sloy+1)*Y(j,k,sloy+1
)*(1-Y(j,k,sloy+1))*W(i,j,sloy+1);
                end
             end

        end

        for sloy=1:l
            for i=1:Arhitecture_NN(1,sloy)
                for j=1:Arhitecture_NN(2,sloy)
                    if sloy==1
                    W(i,j,sloy)=W(i,j,sloy)-
alf*Error_sloy(j,k,sloy)*Y(j,k,sloy)*(1-Y(j,k,sloy))*A(i,k);

T(j,sloy)=T(j,sloy)+alf*Error_sloy(j,k,sloy)*Y(j,k,sloy)*(1-Y(j,k,sloy));
                    else
                    W(i,j,sloy)=W(i,j,sloy)-
alf*Error_sloy(j,k,sloy)*Y(j,k,sloy)*(1-Y(j,k,sloy))*Y(i,k,sloy-1);

T(j,sloy)=T(j,sloy)+alf*Error_sloy(j,k,sloy)*Y(j,k,sloy)*(1-Y(j,k,sloy));
                    end
                end
            end
        end
end
            E=0;
            for i=1:m
                for j=1:Arhitecture_NN(2,l)
                    E=E+(Y(j,i,l)-gauge(j,i))^2;
                end
            end
            F_E(tt)=E;
            tt=tt+1;
            if E<E1
            E1=E;
            F_W=W;
            F_T=T;
            YY=Y;
            end
```

```
end

function
[YY,A_e_P,F_W,F_T]=Training_finish_adap_FA(W,T,A,gauge,Iteration,Arhitecture_
NN,F_A)

[~,m]=size(A);
Y=W.*0;
Error_sloy=Y;
[~,l]=size(Arhitecture_NN);
alf=ones(1,l)*0.05;
A_e_P=1000000000;
number_neurons=Arhitecture_NN(1,:);
number_neurons(l+1)=Arhitecture_NN(2,l);
ttt=0;
while  ttt<Iteration
ttt=ttt+1;
for k=1:m
    for sloy=1:l
       for i=1:Arhitecture_NN(2,sloy)
           Y(i,k,sloy)=0;
           if sloy==1
                for j=1:Arhitecture_NN(1,sloy)
                    Y(i,k,sloy)=Y(i,k,sloy)+A(j,k)*W(j,i,sloy);
                end
            Y(i,k,sloy)=Y(i,k,sloy)-T(i,sloy);
                switch F_A
                case 1
                Y(i,k,sloy)=1/(1+exp(-Y(i,k,sloy)));
                case 2
                Y(i,k,sloy)=2/(1+exp(-Y(i,k,sloy)))-1;
                case 3
                Y(i,k,sloy)=(exp(Y(i,k,sloy))-exp(-
Y(i,k,sloy)))/(exp(Y(i,k,sloy))+exp(-Y(i,k,sloy)));
                end

           else
               for j=1:Arhitecture_NN(1,sloy)
                    Y(i,k,sloy)=Y(i,k,sloy)+Y(j,k,sloy-1)*W(j,i,sloy);
                end
            Y(i,k,sloy)=Y(i,k,sloy)-T(i,sloy);
                switch F_A
                case 1
                Y(i,k,sloy)=1/(1+exp(-Y(i,k,sloy)));
                case 2
                Y(i,k,sloy)=2/(1+exp(-Y(i,k,sloy)))-1;
                case 3
                Y(i,k,sloy)=(exp(Y(i,k,sloy))-exp(-
Y(i,k,sloy)))/(exp(Y(i,k,sloy))+exp(-Y(i,k,sloy)));
                end
           end
       end
    end
        for i=1:Arhitecture_NN(2,l)
           Error_sloy(i,k,l)=Y(i,k,l)-gauge(i,k);
        end

        for sloy=l-1:-1:1

             for i=1:Arhitecture_NN(1,sloy+1)
                Error_sloy(i,k,sloy)=0;
                for j=1:Arhitecture_NN(2,sloy+1)
```

```
                    switch F_A
                    case 1

Error_sloy(i,k,sloy)=Error_sloy(i,k,sloy)+Error_sloy(j,k,sloy+1)*Y(j,k,sloy+1
)*(1-Y(j,k,sloy+1))*W(i,j,sloy+1);
                    case 2

Error_sloy(i,k,sloy)=Error_sloy(i,k,sloy)+Error_sloy(j,k,sloy+1)*2*Y(j,k,sloy
+1)*(1-Y(j,k,sloy+1))*W(i,j,sloy+1);
                    case 3

Error_sloy(i,k,sloy)=Error_sloy(i,k,sloy)+Error_sloy(j,k,sloy+1)*(1-
Y(j,k,sloy+1)*Y(j,k,sloy+1))*W(i,j,sloy+1);
                    end
                end
            end

        end

        for sloy=1:l


alf(1,sloy)=Adaptive_step_deep_NN(number_neurons,sloy,k,Y,Error_sloy,F_A,A);
            for j=1:Arhitecture_NN(2,sloy)
                    switch F_A
                    case 1

T(j,sloy)=T(j,sloy)+alf(1,sloy)*Error_sloy(j,k,sloy)*Y(j,k,sloy)*(1-
Y(j,k,sloy));
                    case 2

T(j,sloy)=T(j,sloy)+alf(1,sloy)*Error_sloy(j,k,sloy)*2*Y(j,k,sloy)*(1-
Y(j,k,sloy));
                    case 3
                    T(j,sloy)=T(j,sloy)+alf*Error_sloy(j,k,sloy)*(1-
Y(j,k,sloy)*Y(j,k,sloy));
                    end
                for i=1:Arhitecture_NN(1,sloy)
                    if sloy==1
                    switch F_A
                    case 1
                    W(i,j,sloy)=W(i,j,sloy)-
alf(1,sloy)*Error_sloy(j,k,sloy)*Y(j,k,sloy)*(1-Y(j,k,sloy))*A(i,k);
                    case 2
                    W(i,j,sloy)=W(i,j,sloy)-
alf(1,sloy)*Error_sloy(j,k,sloy)*2*Y(j,k,sloy)*(1-Y(j,k,sloy))*A(i,k);
                    case 3
                    W(i,j,sloy)=W(i,j,sloy)-
alf(1,sloy)*Error_sloy(j,k,sloy)*(1-Y(j,k,sloy)*Y(j,k,sloy))*A(i,k);
                    end
                    else
                    switch F_A
                    case 1
                    W(i,j,sloy)=W(i,j,sloy)-
alf(1,sloy)*Error_sloy(j,k,sloy)*Y(j,k,sloy)*(1-Y(j,k,sloy))*Y(i,k,sloy-1);
                    case 2
                    W(i,j,sloy)=W(i,j,sloy)-
alf(1,sloy)*Error_sloy(j,k,sloy)*2*Y(j,k,sloy)*(1-Y(j,k,sloy))*Y(i,k,sloy-1);
                    case 3
                    W(i,j,sloy)=W(i,j,sloy)-
alf(1,sloy)*Error_sloy(j,k,sloy)*(1-Y(j,k,sloy)*Y(j,k,sloy))*Y(i,k,sloy-1);
                    end
```

```
                    end
                end
            end
        end
end
            E=0;
            for i=1:m
                for j=1:Arhitecture_NN(2,l)
                    E=E+abs((Y(j,i,l)-gauge(j,i))/gauge(j,i));
                end
            end
            E=E*100/(m*Arhitecture_NN(2,l));
           % F_E(tt)=E;
           % tt=tt+1;
            if E<A_e_P
            A_e_P=E;
            F_W=W;
            F_T=T;
            YY=Y;
            end
end

function
[YY,A_e_P,F_W,F_T]=Training_finish_FA(W,T,A,gauge,Iteration,Arhitecture_NN,al
f,F_A)

[~,m]=size(A);
Y=W.*0;
Error_sloy=Y;
[~,l]=size(Arhitecture_NN);
A_e_P=1000000000;
%F_E(1,1)=0;
%tt=1;
% tic
% while  toc<time_limit
ttt=0;
while  ttt<Iteration
ttt=ttt+1;
for k=1:m
    for sloy=1:l
       for i=1:Arhitecture_NN(2,sloy)
           Y(i,k,sloy)=0;
           if sloy==1
                for j=1:Arhitecture_NN(1,sloy)
                    Y(i,k,sloy)=Y(i,k,sloy)+A(j,k)*W(j,i,sloy);
                end
            Y(i,k,sloy)=Y(i,k,sloy)-T(i,sloy);
                switch F_A
                case 1
                Y(i,k,sloy)=1/(1+exp(-Y(i,k,sloy)));
                case 2
                Y(i,k,sloy)=2/(1+exp(-Y(i,k,sloy)))-1;
                case 3
                Y(i,k,sloy)=(exp(Y(i,k,sloy))-exp(-
Y(i,k,sloy)))/(exp(Y(i,k,sloy))+exp(-Y(i,k,sloy)));
                end

           else
               for j=1:Arhitecture_NN(1,sloy)
                    Y(i,k,sloy)=Y(i,k,sloy)+Y(j,k,sloy-1)*W(j,i,sloy);
                end
            Y(i,k,sloy)=Y(i,k,sloy)-T(i,sloy);
```

```
                switch F_A
                case 1
                Y(i,k,sloy)=1/(1+exp(-Y(i,k,sloy)));
                case 2
                Y(i,k,sloy)=2/(1+exp(-Y(i,k,sloy)))-1;
                case 3
                Y(i,k,sloy)=(exp(Y(i,k,sloy))-exp(-
Y(i,k,sloy)))/(exp(Y(i,k,sloy))+exp(-Y(i,k,sloy)));
                end
            end
        end
    end
        for i=1:Arhitecture_NN(2,l)
           Error_sloy(i,k,l)=Y(i,k,l)-gauge(i,k);
        end

        for sloy=l-1:-1:1

             for i=1:Arhitecture_NN(1,sloy+1)
                Error_sloy(i,k,sloy)=0;
                for j=1:Arhitecture_NN(2,sloy+1)
                    switch F_A
                    case 1

Error_sloy(i,k,sloy)=Error_sloy(i,k,sloy)+Error_sloy(j,k,sloy+1)*Y(j,k,sloy+1
)*(1-Y(j,k,sloy+1))*W(i,j,sloy+1);
                    case 2

Error_sloy(i,k,sloy)=Error_sloy(i,k,sloy)+Error_sloy(j,k,sloy+1)*2*Y(j,k,sloy
+1)*(1-Y(j,k,sloy+1))*W(i,j,sloy+1);
                    case 3

Error_sloy(i,k,sloy)=Error_sloy(i,k,sloy)+Error_sloy(j,k,sloy+1)*(1-
Y(j,k,sloy+1)*Y(j,k,sloy+1))*W(i,j,sloy+1);
                    end
                end
             end

        end

        for sloy=1:l
            for j=1:Arhitecture_NN(2,sloy)
                    switch F_A
                    case 1

T(j,sloy)=T(j,sloy)+alf*Error_sloy(j,k,sloy)*Y(j,k,sloy)*(1-Y(j,k,sloy));
                    case 2

T(j,sloy)=T(j,sloy)+alf*Error_sloy(j,k,sloy)*2*Y(j,k,sloy)*(1-Y(j,k,sloy));
                    case 3
                    T(j,sloy)=T(j,sloy)+alf*Error_sloy(j,k,sloy)*(1-
Y(j,k,sloy)*Y(j,k,sloy));
                    end
                for i=1:Arhitecture_NN(1,sloy)
                    if sloy==1
                    switch F_A
                    case 1
                    W(i,j,sloy)=W(i,j,sloy)-
alf*Error_sloy(j,k,sloy)*Y(j,k,sloy)*(1-Y(j,k,sloy))*A(i,k);
                    case 2
                    W(i,j,sloy)=W(i,j,sloy)-
alf*Error_sloy(j,k,sloy)*2*Y(j,k,sloy)*(1-Y(j,k,sloy))*A(i,k);
```

```
                    case 3
                    W(i,j,sloy)=W(i,j,sloy)-alf*Error_sloy(j,k,sloy)*(1-
Y(j,k,sloy)*Y(j,k,sloy))*A(i,k);
                    end
                    else
                    switch F_A
                    case 1
                    W(i,j,sloy)=W(i,j,sloy)-
alf*Error_sloy(j,k,sloy)*Y(j,k,sloy)*(1-Y(j,k,sloy))*Y(i,k,sloy-1);
                    case 2
                    W(i,j,sloy)=W(i,j,sloy)-
alf*Error_sloy(j,k,sloy)*2*Y(j,k,sloy)*(1-Y(j,k,sloy))*Y(i,k,sloy-1);
                    case 3
                    W(i,j,sloy)=W(i,j,sloy)-alf*Error_sloy(j,k,sloy)*(1-
Y(j,k,sloy)*Y(j,k,sloy))*Y(i,k,sloy-1);
                    end
                    end
                end
            end
        end
end
            E=0;
            for i=1:m
                for j=1:Arhitecture_NN(2,l)
                    E=E+abs((Y(j,i,l)-gauge(j,i))/gauge(j,i));
                end
            end
            E=E*100/(m*Arhitecture_NN(2,l));
           % F_E(tt)=E;
           % tt=tt+1;
            if E<A_e_P
            A_e_P=E;
            F_W=W;
            F_T=T;
            YY=Y;
            end
end

function
[YY,A_e_P,F_W,F_T]=Training_finish_time_adap_FA(W,T,A,gauge,time,Arhitecture_
NN,F_A)

[~,m]=size(A);
Y=W.*0;
Error_sloy=Y;
[~,l]=size(Arhitecture_NN);
alf=ones(1,l)*0.05;
A_e_P=1000000000;
ttt=0;
number_neurons=Arhitecture_NN(1,:);
number_neurons(l+1)=Arhitecture_NN(2,l);
tic
while  toc<time
ttt=ttt+1;
for k=1:m
    for sloy=1:l
       for i=1:Arhitecture_NN(2,sloy)
           Y(i,k,sloy)=0;
           if sloy==1
                for j=1:Arhitecture_NN(1,sloy)
                    Y(i,k,sloy)=Y(i,k,sloy)+A(j,k)*W(j,i,sloy);
                end
```

```
            Y(i,k,sloy)=Y(i,k,sloy)-T(i,sloy);
                switch F_A
                case 1
                Y(i,k,sloy)=1/(1+exp(-Y(i,k,sloy)));
                case 2
                Y(i,k,sloy)=2/(1+exp(-Y(i,k,sloy)))-1;
                case 3
                Y(i,k,sloy)=(exp(Y(i,k,sloy))-exp(-
Y(i,k,sloy)))/(exp(Y(i,k,sloy))+exp(-Y(i,k,sloy)));
                end

           else
               for j=1:Arhitecture_NN(1,sloy)
                    Y(i,k,sloy)=Y(i,k,sloy)+Y(j,k,sloy-1)*W(j,i,sloy);
                end
            Y(i,k,sloy)=Y(i,k,sloy)-T(i,sloy);
                switch F_A
                case 1
                Y(i,k,sloy)=1/(1+exp(-Y(i,k,sloy)));
                case 2
                Y(i,k,sloy)=2/(1+exp(-Y(i,k,sloy)))-1;
                case 3
                Y(i,k,sloy)=(exp(Y(i,k,sloy))-exp(-
Y(i,k,sloy)))/(exp(Y(i,k,sloy))+exp(-Y(i,k,sloy)));
                end
           end
       end
    end
        for i=1:Arhitecture_NN(2,l)
           Error_sloy(i,k,l)=Y(i,k,l)-gauge(i,k);
        end

        for sloy=l-1:-1:1

             for i=1:Arhitecture_NN(1,sloy+1)
                Error_sloy(i,k,sloy)=0;
                for j=1:Arhitecture_NN(2,sloy+1)
                    switch F_A
                    case 1

Error_sloy(i,k,sloy)=Error_sloy(i,k,sloy)+Error_sloy(j,k,sloy+1)*Y(j,k,sloy+1
)*(1-Y(j,k,sloy+1))*W(i,j,sloy+1);
                    case 2

Error_sloy(i,k,sloy)=Error_sloy(i,k,sloy)+Error_sloy(j,k,sloy+1)*2*Y(j,k,sloy
+1)*(1-Y(j,k,sloy+1))*W(i,j,sloy+1);
                    case 3

Error_sloy(i,k,sloy)=Error_sloy(i,k,sloy)+Error_sloy(j,k,sloy+1)*(1-
Y(j,k,sloy+1)*Y(j,k,sloy+1))*W(i,j,sloy+1);
                    end
                end
             end

        end

        for sloy=1:l


alf(1,sloy)=Adaptive_step_deep_NN(number_neurons,sloy,k,Y,Error_sloy,F_A,A);
            for j=1:Arhitecture_NN(2,sloy)
                    switch F_A
```

```
                    case 1

T(j,sloy)=T(j,sloy)+alf(1,sloy)*Error_sloy(j,k,sloy)*Y(j,k,sloy)*(1-
Y(j,k,sloy));
                    case 2

T(j,sloy)=T(j,sloy)+alf(1,sloy)*Error_sloy(j,k,sloy)*2*Y(j,k,sloy)*(1-
Y(j,k,sloy));
                    case 3
                    T(j,sloy)=T(j,sloy)+alf*Error_sloy(j,k,sloy)*(1-
Y(j,k,sloy)*Y(j,k,sloy));
                    end
                for i=1:Arhitecture_NN(1,sloy)
                    if sloy==1
                    switch F_A
                    case 1
                    W(i,j,sloy)=W(i,j,sloy)-
alf(1,sloy)*Error_sloy(j,k,sloy)*Y(j,k,sloy)*(1-Y(j,k,sloy))*A(i,k);
                    case 2
                    W(i,j,sloy)=W(i,j,sloy)-
alf(1,sloy)*Error_sloy(j,k,sloy)*2*Y(j,k,sloy)*(1-Y(j,k,sloy))*A(i,k);
                    case 3
                    W(i,j,sloy)=W(i,j,sloy)-
alf(1,sloy)*Error_sloy(j,k,sloy)*(1-Y(j,k,sloy)*Y(j,k,sloy))*A(i,k);
                    end
                    else
                    switch F_A
                    case 1
                    W(i,j,sloy)=W(i,j,sloy)-
alf(1,sloy)*Error_sloy(j,k,sloy)*Y(j,k,sloy)*(1-Y(j,k,sloy))*Y(i,k,sloy-1);
                    case 2
                    W(i,j,sloy)=W(i,j,sloy)-
alf(1,sloy)*Error_sloy(j,k,sloy)*2*Y(j,k,sloy)*(1-Y(j,k,sloy))*Y(i,k,sloy-1);
                    case 3
                    W(i,j,sloy)=W(i,j,sloy)-
alf(1,sloy)*Error_sloy(j,k,sloy)*(1-Y(j,k,sloy)*Y(j,k,sloy))*Y(i,k,sloy-1);
                    end
                    end
                end
            end
        end
end
            E=0;
            for i=1:m
                for j=1:Arhitecture_NN(2,l)
                    E=E+abs((Y(j,i,l)-gauge(j,i))/gauge(j,i));
                end
            end
            E=E*100/(m*Arhitecture_NN(2,l));
           % F_E(tt)=E;
           % tt=tt+1;
            if E<A_e_P
            A_e_P=E;
            F_W=W;
            F_T=T;
            YY=Y;
            end
end
```

```
function
[YY,A_e_P,F_W,F_T]=Training_finish_time_FA(W,T,A,gauge,time,Arhitecture_NN,al
f,F_A)

[~,m]=size(A);
Y=W.*0;
Error_sloy=Y;
[~,l]=size(Arhitecture_NN);
A_e_P=1000000000;
ttt=0;
tic
while  toc<time
ttt=ttt+1;
for k=1:m
    for sloy=1:l
       for i=1:Arhitecture_NN(2,sloy)
           Y(i,k,sloy)=0;
           if sloy==1
                for j=1:Arhitecture_NN(1,sloy)
                    Y(i,k,sloy)=Y(i,k,sloy)+A(j,k)*W(j,i,sloy);
                end
            Y(i,k,sloy)=Y(i,k,sloy)-T(i,sloy);
                switch F_A
                case 1
                Y(i,k,sloy)=1/(1+exp(-Y(i,k,sloy)));
                case 2
                Y(i,k,sloy)=2/(1+exp(-Y(i,k,sloy)))-1;
                case 3
                Y(i,k,sloy)=(exp(Y(i,k,sloy))-exp(-
Y(i,k,sloy)))/(exp(Y(i,k,sloy))+exp(-Y(i,k,sloy)));
                end

           else
               for j=1:Arhitecture_NN(1,sloy)
                    Y(i,k,sloy)=Y(i,k,sloy)+Y(j,k,sloy-1)*W(j,i,sloy);
                end
            Y(i,k,sloy)=Y(i,k,sloy)-T(i,sloy);
                switch F_A
                case 1
                Y(i,k,sloy)=1/(1+exp(-Y(i,k,sloy)));
                case 2
                Y(i,k,sloy)=2/(1+exp(-Y(i,k,sloy)))-1;
                case 3
                Y(i,k,sloy)=(exp(Y(i,k,sloy))-exp(-
Y(i,k,sloy)))/(exp(Y(i,k,sloy))+exp(-Y(i,k,sloy)));
                end
           end
       end
    end
        for i=1:Arhitecture_NN(2,l)
           Error_sloy(i,k,l)=Y(i,k,l)-gauge(i,k);
        end

        for sloy=l-1:-1:1

             for i=1:Arhitecture_NN(1,sloy+1)
                Error_sloy(i,k,sloy)=0;
                for j=1:Arhitecture_NN(2,sloy+1)
                    switch F_A
                    case 1
```

```
Error_sloy(i,k,sloy)=Error_sloy(i,k,sloy)+Error_sloy(j,k,sloy+1)*Y(j,k,sloy+1
)*(1-Y(j,k,sloy+1))*W(i,j,sloy+1);
                    case 2

Error_sloy(i,k,sloy)=Error_sloy(i,k,sloy)+Error_sloy(j,k,sloy+1)*2*Y(j,k,sloy
+1)*(1-Y(j,k,sloy+1))*W(i,j,sloy+1);
                    case 3

Error_sloy(i,k,sloy)=Error_sloy(i,k,sloy)+Error_sloy(j,k,sloy+1)*(1-
Y(j,k,sloy+1)*Y(j,k,sloy+1))*W(i,j,sloy+1);
                    end
                end
            end

        end

        for sloy=1:l
            for j=1:Arhitecture_NN(2,sloy)
                    switch F_A
                    case 1

T(j,sloy)=T(j,sloy)+alf*Error_sloy(j,k,sloy)*Y(j,k,sloy)*(1-Y(j,k,sloy));
                    case 2

T(j,sloy)=T(j,sloy)+alf*Error_sloy(j,k,sloy)*2*Y(j,k,sloy)*(1-Y(j,k,sloy));
                    case 3
                    T(j,sloy)=T(j,sloy)+alf*Error_sloy(j,k,sloy)*(1-
Y(j,k,sloy)*Y(j,k,sloy));
                    end
                for i=1:Arhitecture_NN(1,sloy)
                    if sloy==1
                    switch F_A
                    case 1
                    W(i,j,sloy)=W(i,j,sloy)-
alf*Error_sloy(j,k,sloy)*Y(j,k,sloy)*(1-Y(j,k,sloy))*A(i,k);
                    case 2
                    W(i,j,sloy)=W(i,j,sloy)-
alf*Error_sloy(j,k,sloy)*2*Y(j,k,sloy)*(1-Y(j,k,sloy))*A(i,k);
                    case 3
                    W(i,j,sloy)=W(i,j,sloy)-alf*Error_sloy(j,k,sloy)*(1-
Y(j,k,sloy)*Y(j,k,sloy))*A(i,k);
                    end
                    else
                    switch F_A
                    case 1
                    W(i,j,sloy)=W(i,j,sloy)-
alf*Error_sloy(j,k,sloy)*Y(j,k,sloy)*(1-Y(j,k,sloy))*Y(i,k,sloy-1);
                    case 2
                    W(i,j,sloy)=W(i,j,sloy)-
alf*Error_sloy(j,k,sloy)*2*Y(j,k,sloy)*(1-Y(j,k,sloy))*Y(i,k,sloy-1);
                    case 3
                    W(i,j,sloy)=W(i,j,sloy)-alf*Error_sloy(j,k,sloy)*(1-
Y(j,k,sloy)*Y(j,k,sloy))*Y(i,k,sloy-1);
                    end
                    end
                end
            end
        end
end
            E=0;
            for i=1:m
```

```
                for j=1:Arhitecture_NN(2,l)
                    E=E+abs((Y(j,i,l)-gauge(j,i))/gauge(j,i));
                end
            end
            E=E*100/(m*Arhitecture_NN(2,l));
           % F_E(tt)=E;
           % tt=tt+1;
            if E<A_e_P
            A_e_P=E;
            F_W=W;
            F_T=T;
            YY=Y;
            end
end

function
[Arhitecture_NN,W,T]=Training_NN_Deep_belief(A,Arhitecture,gauge,alf)
l=length(Arhitecture);
AA=A;
W(1,1,1)=0;
n(l-1)=0;
m(l-1)=0;
T(1,1)=0;
for i=1:l-2
    [WW,TT,AA]=Training_Avto(AA,Arhitecture(i+1),alf);
    [n(i),m(i)]=size(WW);
    W(1:n(i),1:m(i),i)=WW;
    T(1:m(i),i)=TT;
end
[WW,TT]=Training_Avto_end(AA,gauge,alf);
[n(i+1),m(i+1)]=size(WW);
W(1:n(i+1),1:m(i+1),l-1)=WW;
T(1:m(i+1),l-1)=TT;

Arhitecture_NN=[n;m];

function
[Arhitecture_NN,W,T]=Training_NN_Deep_belief_FA(A,Arhitecture,gauge,alf,F_A)
l=length(Arhitecture);
AA=A;
W(1,1,1)=0;
n(l-1)=0;
m(l-1)=0;
T(1,1)=0;
for i=1:l-2
    [WW,TT,AA]=Training_Avto_FA(AA,Arhitecture(i+1),alf,F_A);
    [n(i),m(i)]=size(WW);
    W(1:n(i),1:m(i),i)=WW;
    T(1:m(i),i)=TT;
end
[WW,TT]=Training_Avto_end_FA(AA,gauge,alf,F_A);
[n(i+1),m(i+1)]=size(WW);
W(1:n(i+1),1:m(i+1),l-1)=WW;
T(1:m(i+1),l-1)=TT;

Arhitecture_NN=[n;m];
```