МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

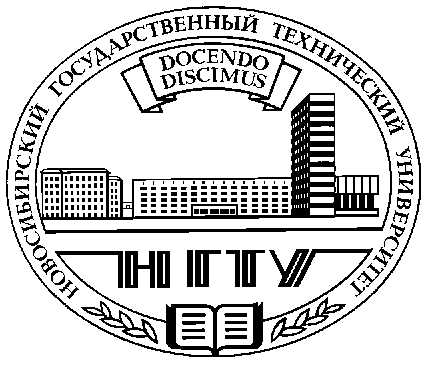
ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Факультет автоматики и вычислительной техники

Кафедра вычислительной техники



**ОТЧЁТ**

**по расчетно-графической работе**

по дисциплине: «Системы искусственного интеллекта и машинное обучение»

Выполнили:Проверил:

студенты гр. АВТ-918

Соловьев Н.Е Осипенко И.В.

Ванин К.Е

«\_\_\_» \_\_\_\_\_\_\_\_\_ 2023г.«\_\_\_» \_\_\_\_\_\_\_\_ 2023 г.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(оценка, подпись)

Новосибирск

2023

**Оглавление**

[Цель работы: 3](#_Toc122188332)

[Задание: 3](#_Toc122188333)

[Ход работы 3](#_Toc122188334)

[Вывод: 7](#_Toc122188335)

[Приложения А 8](#_Toc122188336)

# Цель работы:

Знакомство элементами технологии создания баз знаний интеллектуальных систем

# Задание:

* Освоение способа сохранения и загрузки обученных моделей машинного обучения;
* Разработка интерфейса доступа к сохраненным моделям;
* Организация произвольных запросов с использованием разработанного интерфейса.

Используемые программные инструменты:

* Язык программирования: python
* Реализация алгоритмов машинного обучения: tensorflow, sklearn
* Сериализация/десериализация: json, pickle, tensorflow
* web-интерфейс: streamlit

# Ход работы

1. Выполнил загрузку файла с подготовленным набором данных (очищенный от пропусков и выбросов для целевой и независимых переменных) в таблицу исходных данных df
2. Объявить список названий целевых (target) и независимой (features) переменных
3. Отделить X от Y исходной таблицы и сформировать новые таблицы dfX и dfY

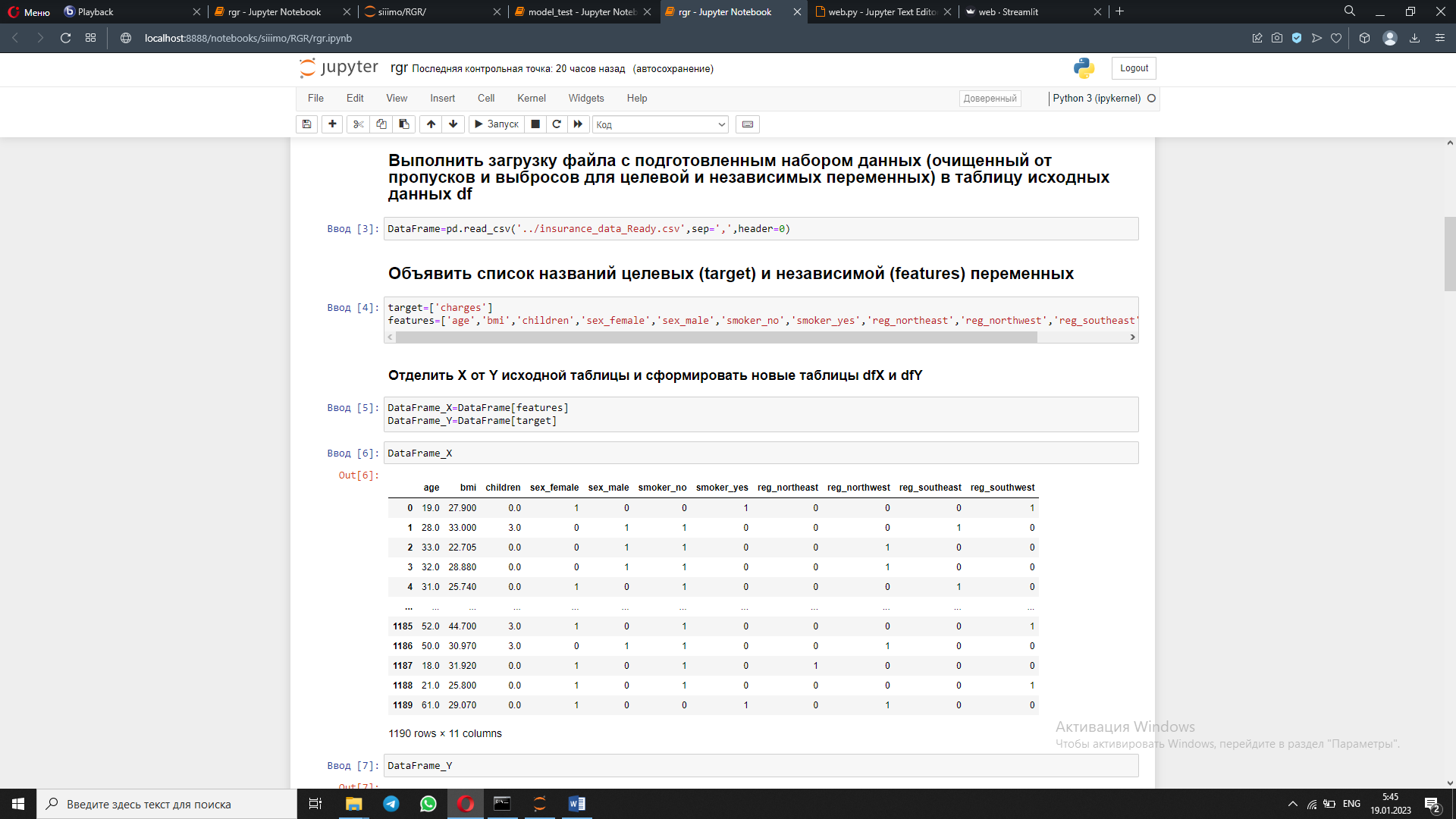


Рис.1 – Результат выполнения первых трёх пунктов

1. Выполнить процедуру НОРМАЛИЗАЦИИ данных и сохранить в отдельную таблиц

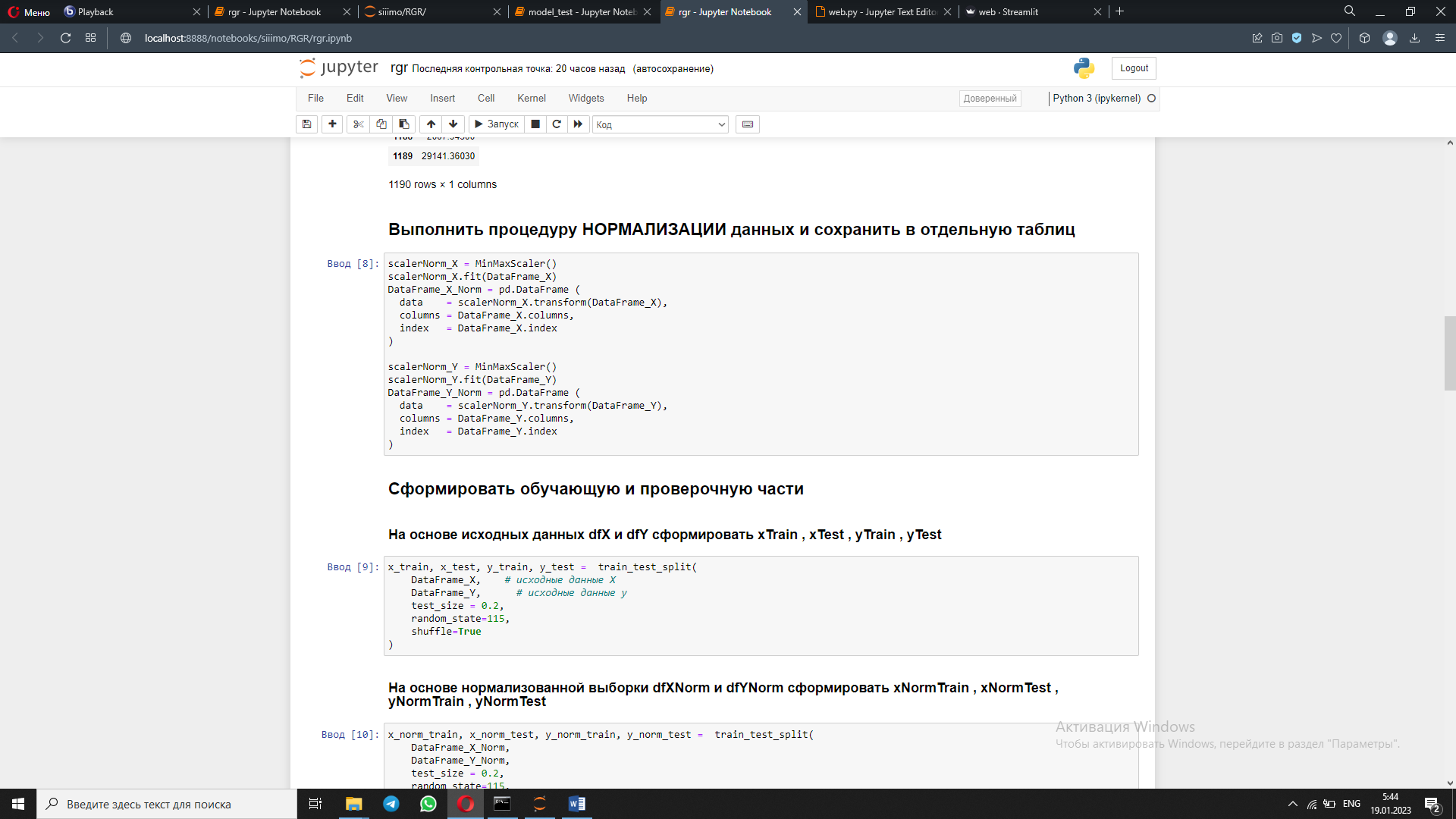


Рис.2 – Результат выполнения нормализации

1. Сформировать обучающую и проверочную части

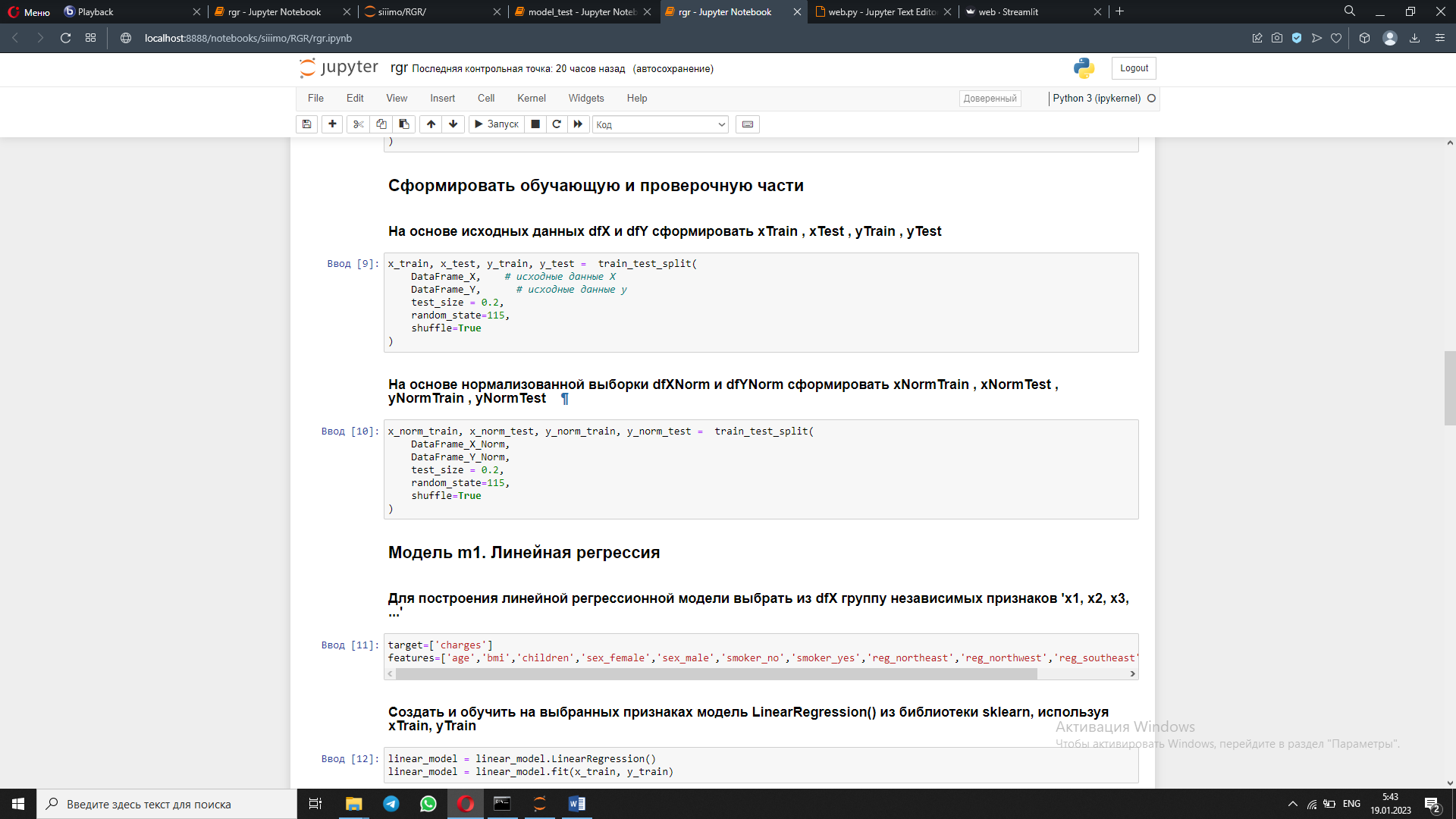


Рис.3 – Формирование обучающей и проверочной частей

1. Модель m1. Линейная регрессия

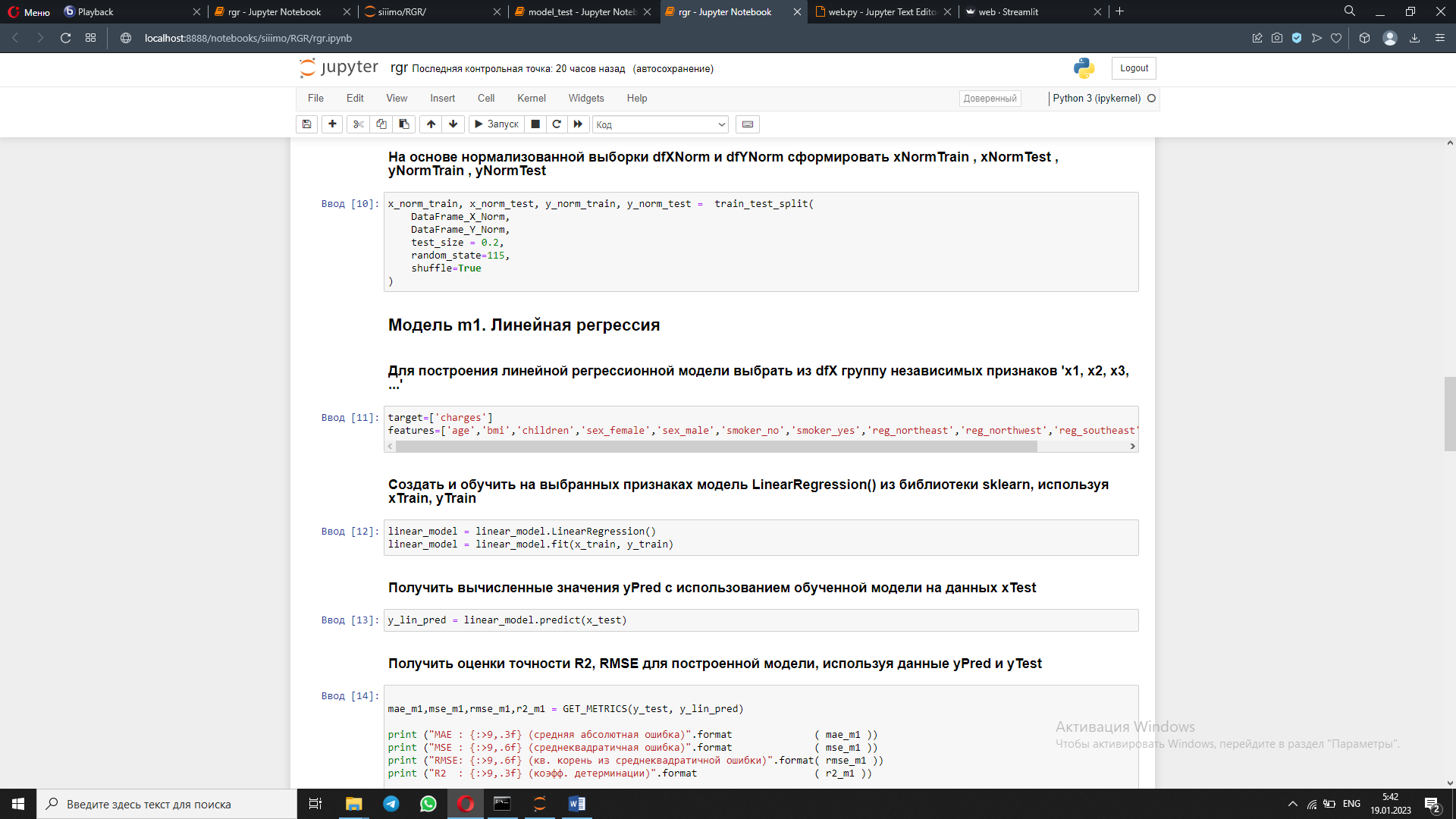


Рис.4 – Инициализируем модель линейной регрессии и обучим её

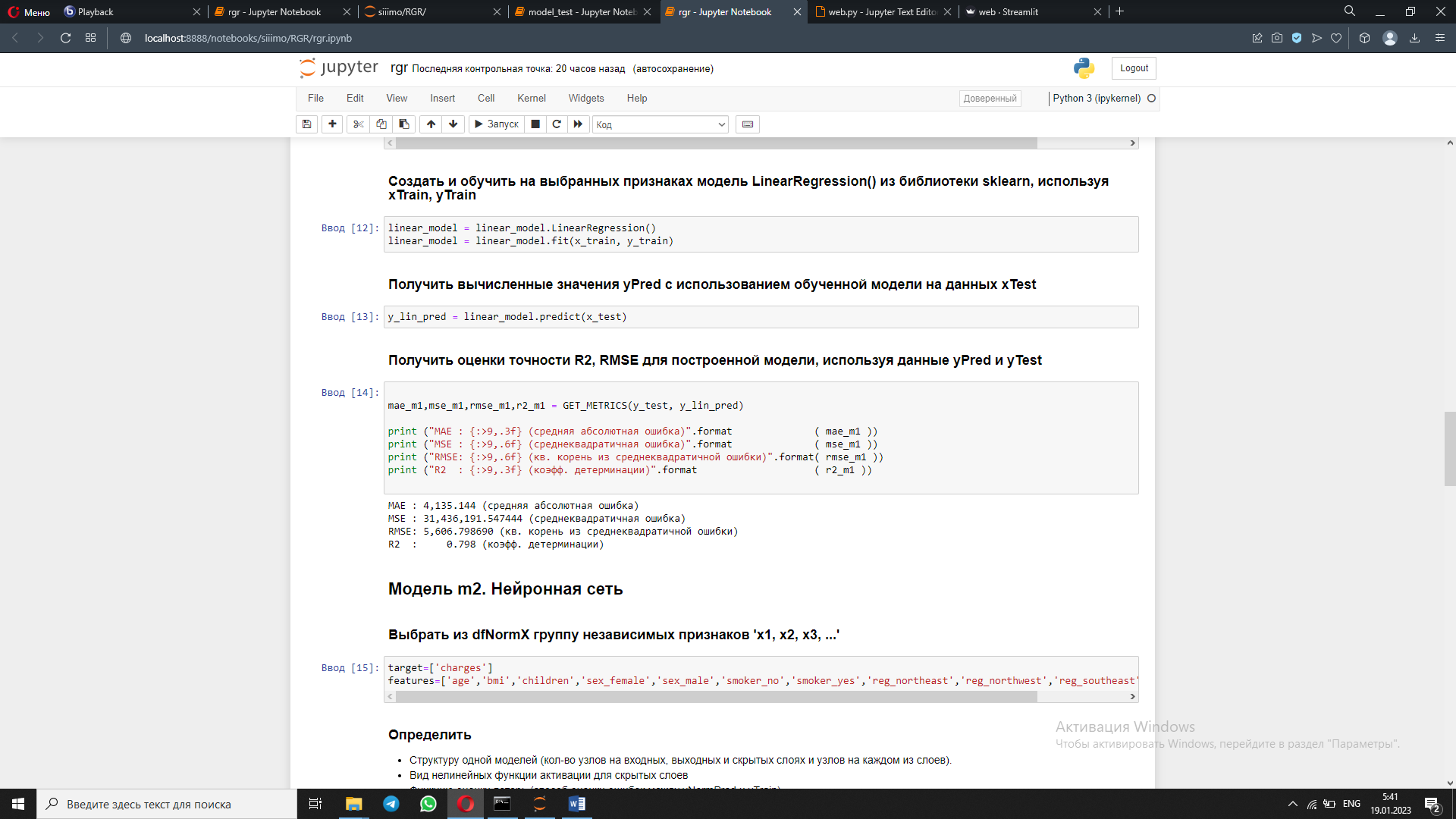


Рис.5 – Получим необходимые оценки модели

1. Модель m2. Нейронная сеть

Инициализируем нейронную сеть, пропишем её структуру с входными, скрытыми и выходным слоями. Обучим её, загрузив нашу обучающую выборку. После чего проверим результат обучения и выведем необходимые оценки работы сети.

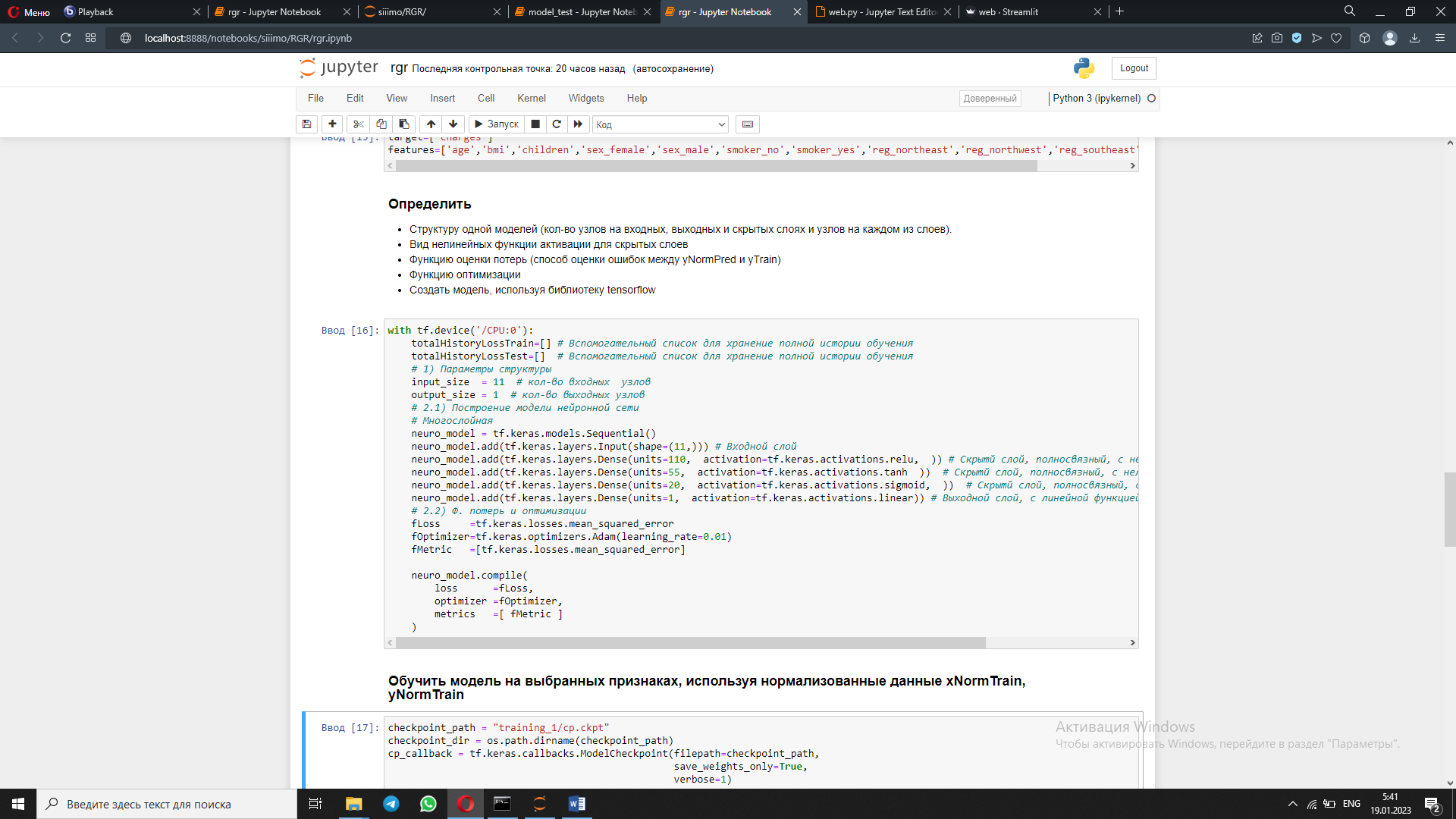


Рис.6 – Инициализация нейронной сети

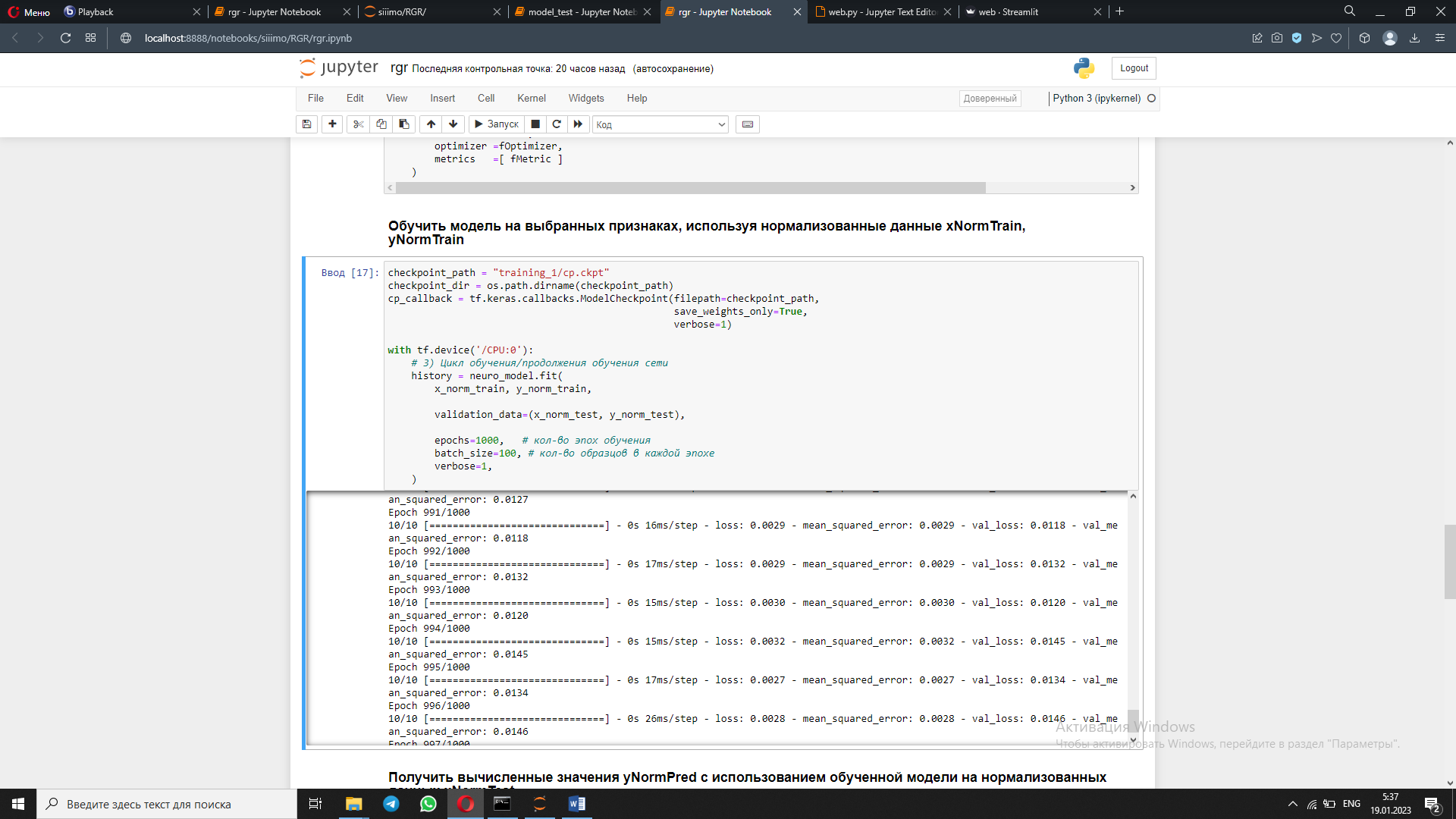


Рис.7 – Обучение модели на нормализованных данных

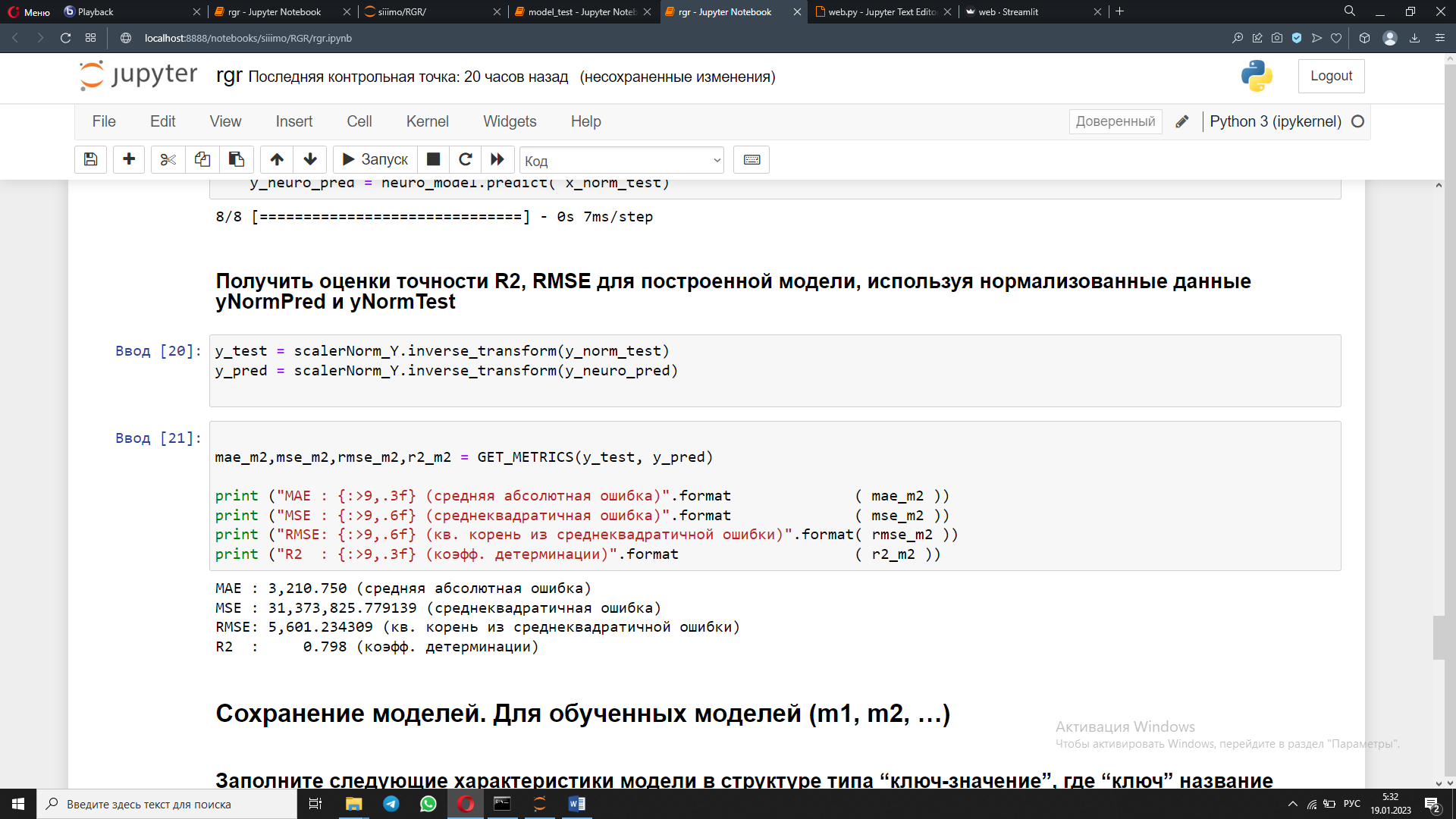


Рис.8 – Результат обучения сети в виде необходимых оценок точности

1. Сохранение моделей. Для обученных моделей (m1, m2, …):

Далее сохраним наши полученные модели и всю информацию о них. Для этого воспользуемся командой json.dump. Также, сохраним шкалеры наших нормализованных данных, чтобы в дальнейшем правильно интерпретировать результаты, полученные нашей нейронной сетью. Сохранять модели мы будем с помощью команд pickle.dump и tensorflow.save. Мы сохраним нейронную сеть полностью, но если она слишком большая, то разумнее сохранять лишь её веса для дальнейшей загрузки.

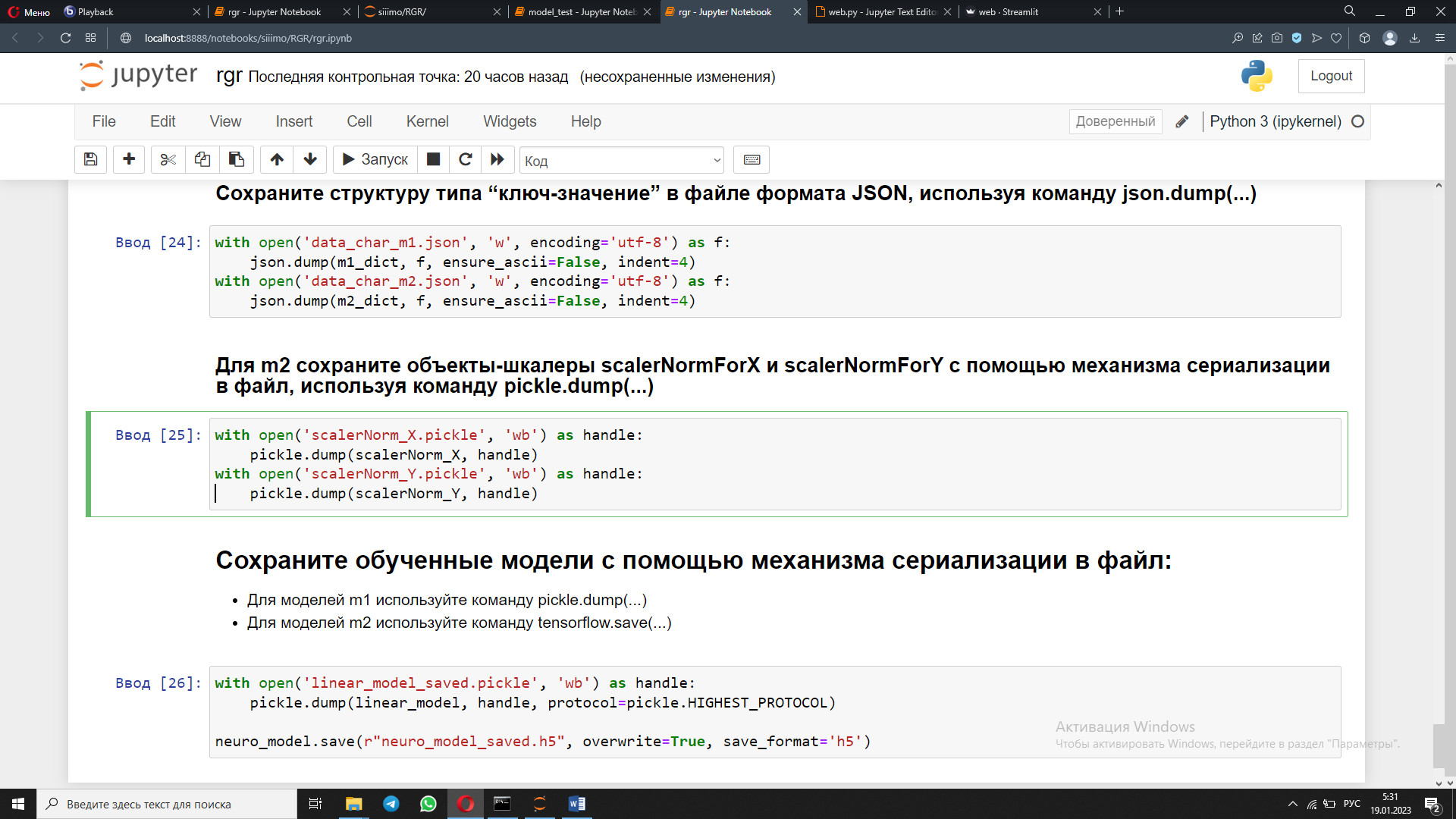


Рис.9 – Сохранения всех наших полученных данных

1. Разработка web-интерфейса

В отдельном файле пропишем наш интерфейс для работы с нашими моделями с помощью библиотеки streamlir. Загрузим наши модели и информацию о них.

Полный код работы находится в приложении А.

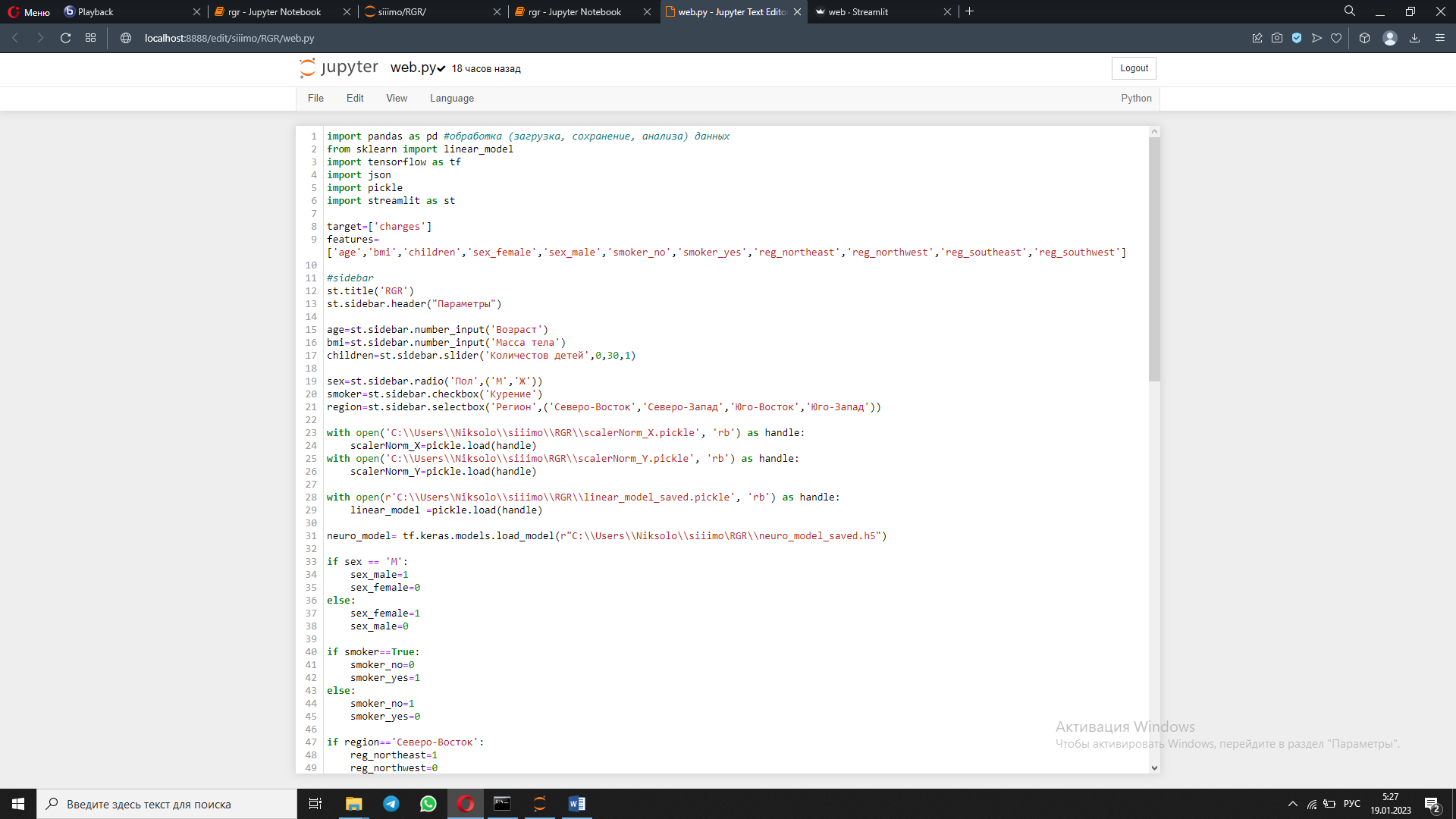


Рис.10 – Создание интерфейса и загрузка наших моделей

1. Проверка работоспособности

Проверим работы написанного интерфейса и получим результаты работы обученных моделей. Выберем данные для проверки работоспособности и выведем их в таблицу для сверки.

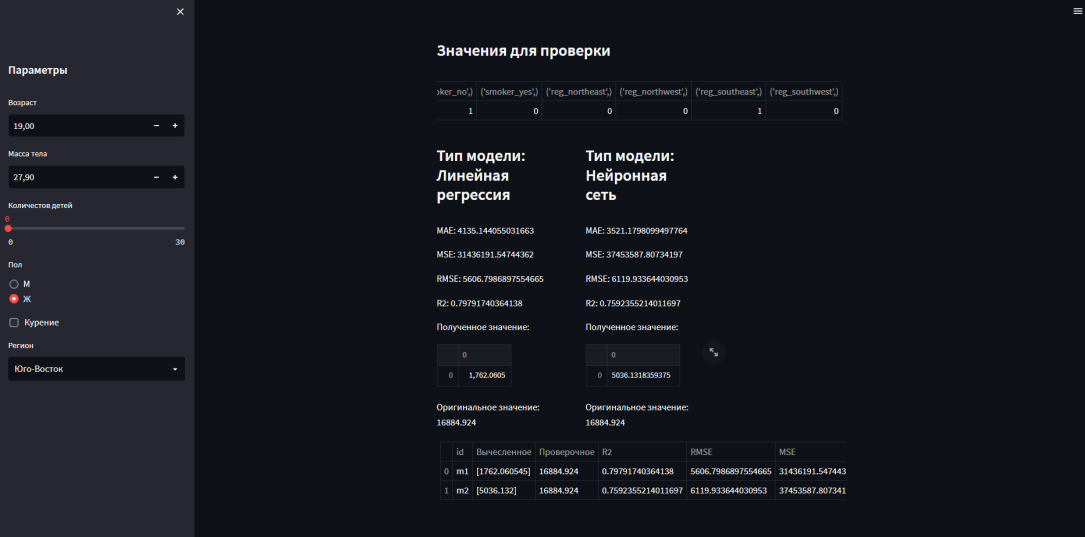


Рис.11 –Результат выполнения работы

Таблица.1 – Таблица сравнения данных и оценок работы моделей



Таблица.2 Сравнение данных на основе представленных

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Возраст | bmi | | дети | пол | курит | регион | Yправ | Ym1 | Ym2 |
| 28 | 33 | 3 | | 1 | 1 | Юг | 4449.46200 | 29866.3089 | 35553.921875 |
| 33 | 22,71 | 0 | | 1 | 0 | сев | 21984.47061 | 3.368,969500667891 | 3530.3115234375 |
| 53 | 38.06 | 3 | | 0 | 0 | югВ | 20462.99766 | 15.060,095676508412 | 23313 |

**Вывод**:

В ходе выполнения данной работы были закреплены ранее полученные навыки работы с данными, работы с моделями. Также, были изучена и реализована работа с сохранением \ загрузкой моделей и данных для них в разных форматах. Была затронута работа с интерфейсами. Обе модели выдают похожие результаты, каждая модель выдает хороший результат в определенном случае. Цель знакомства с элементами и технологиями создания баз знаний интеллектуальных систем можно считать выполненной.

**Приложения А**

import pandas as pd #обработка (загрузка, сохранение, анализа) данных

from sklearn import linear\_model

import tensorflow as tf

import json

import pickle

import streamlit as st

target=['charges']

features=['age','bmi','children','sex\_female','sex\_male','smoker\_no','smoker\_yes','reg\_northeast','reg\_northwest','reg\_southeast','reg\_southwest']

#sidebar

st.title('RGR')

st.sidebar.header("Параметры")

age=st.sidebar.number\_input('Возраст')

bmi=st.sidebar.number\_input('Масса тела')

children=st.sidebar.slider('Количестов детей',0,30,1)

sex=st.sidebar.radio('Пол',('М','Ж'))

smoker=st.sidebar.checkbox('Курение')

region=st.sidebar.selectbox('Регион',('Северо-Восток','Северо-Запад','Юго-Восток','Юго-Запад'))

with open('C:\\Users\\Niksolo\\siiimo\\RGR\\scalerNorm\_X.pickle', 'rb') as handle:

scalerNorm\_X=pickle.load(handle)

with open('C:\\Users\\Niksolo\\siiimo\RGR\\scalerNorm\_Y.pickle', 'rb') as handle:

scalerNorm\_Y=pickle.load(handle)

with open(r'C:\\Users\Niksolo\\siiimo\\RGR\\linear\_model\_saved.pickle', 'rb') as handle:

linear\_model =pickle.load(handle)

neuro\_model= tf.keras.models.load\_model(r"C:\\Users\\Niksolo\\siiimo\RGR\\neuro\_model\_saved.h5")

if sex == 'М':

sex\_male=1

sex\_female=0

else:

sex\_female=1

sex\_male=0

if smoker==True:

smoker\_no=0

smoker\_yes=1

else:

smoker\_no=1

smoker\_yes=0

if region=='Северо-Восток':

reg\_northeast=1

reg\_northwest=0

reg\_southeast=0

reg\_southwest=0

if region=='Северо-Запад':

reg\_northeast=0

reg\_northwest=1

reg\_southeast=0

reg\_southwest=0

if region=='Юго-Восток':

reg\_northeast=0

reg\_northwest=0

reg\_southeast=1

reg\_southwest=0

if region=='Юго-Запад':

reg\_northeast=0

reg\_northwest=0

reg\_southeast=0

reg\_southwest=1

features=['age','bmi','children','sex\_female','sex\_male','smoker\_no','smoker\_yes','reg\_northeast','reg\_northwest','reg\_southeast','reg\_southwest']

DataFrame\_Custom=pd.DataFrame(data=[[age, bmi, children, sex\_female, sex\_male, smoker\_no, smoker\_yes, reg\_northeast, reg\_northwest, reg\_southeast, reg\_southwest]],

columns=[features])

DataFrame\_Valid=pd.DataFrame(data=[[19.0, 27.9, 0.0, 1, 0, 1, 0, 0, 0, 1, 0]], columns=[features])

DataFrame\_Custom\_Norm = pd.DataFrame (

data = scalerNorm\_X.transform(DataFrame\_Custom),

columns = DataFrame\_Custom.columns,

index = DataFrame\_Custom.index

)

with open('C:\\Users\\Niksolo\\siiimo\\RGR\\data\_char\_m1.json', 'r', encoding='utf-8') as f:

m1\_dict= json.load(f)

with open('C:\\Users\\Niksolo\\siiimo\\RGR\\data\_char\_m2.json', 'r', encoding='utf-8') as f:

m2\_dict= json.load(f)

y\_pred\_lin=linear\_model.predict(DataFrame\_Custom)

st.subheader("Ваши Данные")

st.table(DataFrame\_Custom)

st.subheader("Значения для проверки")

st.table(DataFrame\_Valid)

col1, col2, col3 = st.columns(3,gap='large')

with col1:

st.subheader('Тип модели: Линейная регрессия')

st.write('MAE:', str(m1\_dict['MAE']))

st.write('MSE:', str(m1\_dict['MSE']))

st.write('RMSE:', str(m1\_dict['RMSE']))

st.write('R2:', str(m1\_dict['R2']))

st.write('Полученное значение:',y\_pred\_lin)

st.write('Оригинальное значение:',m1\_dict['y\_test'])

with tf.device('/CPU:0'):

y\_neuro\_pred\_norm = neuro\_model.predict( DataFrame\_Custom\_Norm)

y\_neuro\_pred = scalerNorm\_Y.inverse\_transform(y\_neuro\_pred\_norm)

with col2:

st.subheader('Тип модели: Нейронная сеть')

st.write('MAE:', str(m2\_dict['MAE']))

st.write('MSE:', str(m2\_dict['MSE']))

st.write('RMSE:', str(m2\_dict['RMSE']))

st.write('R2:', str(m2\_dict['R2']))

st.write('Полученное значение:',y\_neuro\_pred[0])

st.write('Оригинальное значение:',m2\_dict['y\_test'])

DataResults = pd.DataFrame(

[

['m1', str(y\_pred\_lin[0]), str(m1\_dict['y\_test']), str(m1\_dict['R2']), str(m1\_dict['RMSE']), str(m1\_dict['MSE']), str(m1\_dict['MAE'])],

["m2", str(y\_neuro\_pred[0]), str(m2\_dict['y\_test']), str(m2\_dict['R2']), str(m2\_dict['RMSE']), str(m2\_dict['MSE']), str(m2\_dict['MAE'])],

],

columns= ["id", "Вычесленное", "Проверочное", "R2", "RMSE", "MSE", "MAE" ],

)

st.table(DataResults)