МИНОБРНАУКИ РОССИИ

РГУ НЕФТИ И ГАЗА (НИУ) ИМЕНИ И.М. ГУБКИНА

ФАКУЛЬТЕТ АВТОМАТИКИ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

КАФЕДРА ИНФОРМАТИКИ

ДИСЦИПЛИНА «ОСНОВЫ АНАЛИЗА БОЛЬШИХ ДАННЫХ И МАШИННОЕ ОБУЧЕНИЕ»

**О Т Ч Е Т**

**по Домашнему Заданию 3**

**«Анализ данных в среде Python»**

Выполнила: студентка группы АА-19-05

Данилова М.А.

Проверила: доцент Вишневская Е. А.

Москва 2022

**9. Регрессионный анализ**

**Регрессионный анализ: основные положения.**

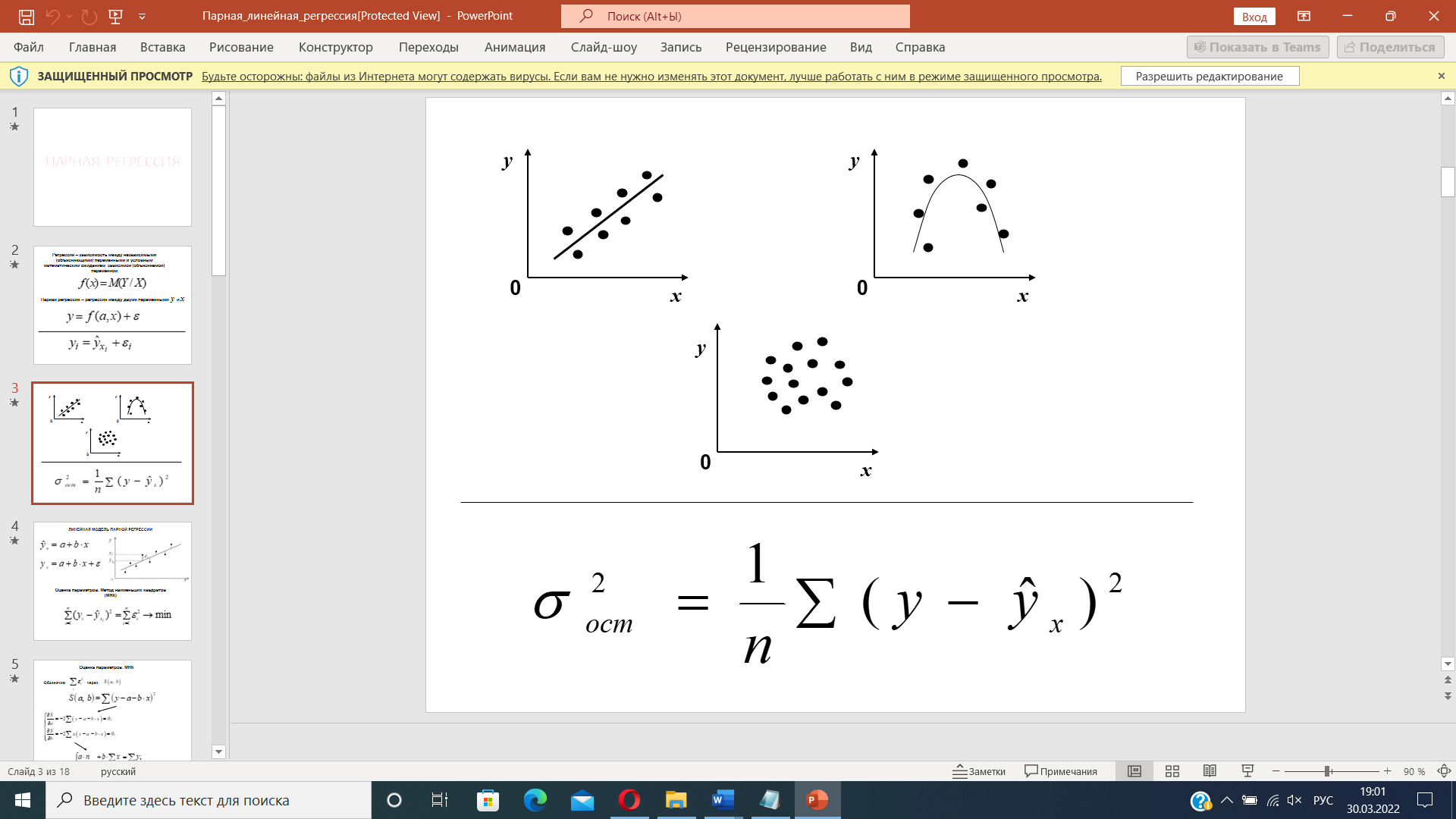
Регрессионные модели используются для прогнозирования непрерывных целевых значений (прогнозирования цен на жилье, прогноза погоды). Под регрессией понимается функциональная зависимость между объясняющими переменными xi и условным математическим ожиданием (средним значением) зависимой переменой y; модель строится с целью прогнозирования этого среднего значения при фиксированных значениях первых. Любая регрессионная модель позволяет обнаружить только количественные зависимости, которые не обязательно отражают причинные. В ходе регрессионного анализа определяют коэффициенты регрессии (β) - величины для каждой независимой переменной, которые представляют силу и тип взаимосвязи независимой переменной по отношению к зависимой.

В общем случае, предположим, что мы наблюдаем количественный отклик Y, и несколько разных предикторов X1, X2 , . . .,Xp . Предположим, что есть какая-то взаимосвязь между Y и X = (X1, X2 ,...,Xp ), которая в общей форме может быть записана в виде

Y = f(X) + є.

Здесь f – это некоторая фиксированная неизвестная функция переменных X1, X2 ,...,Xp , и є – случайная ошибка, не зависящая от X, с нулевым матожиданием.

Выбор формулы связи переменных называется **спецификацией** уравнения регрессии. В случае парной регрессии выбор формулы обычно осуществляется по графическому изображению реальных статистических данных.

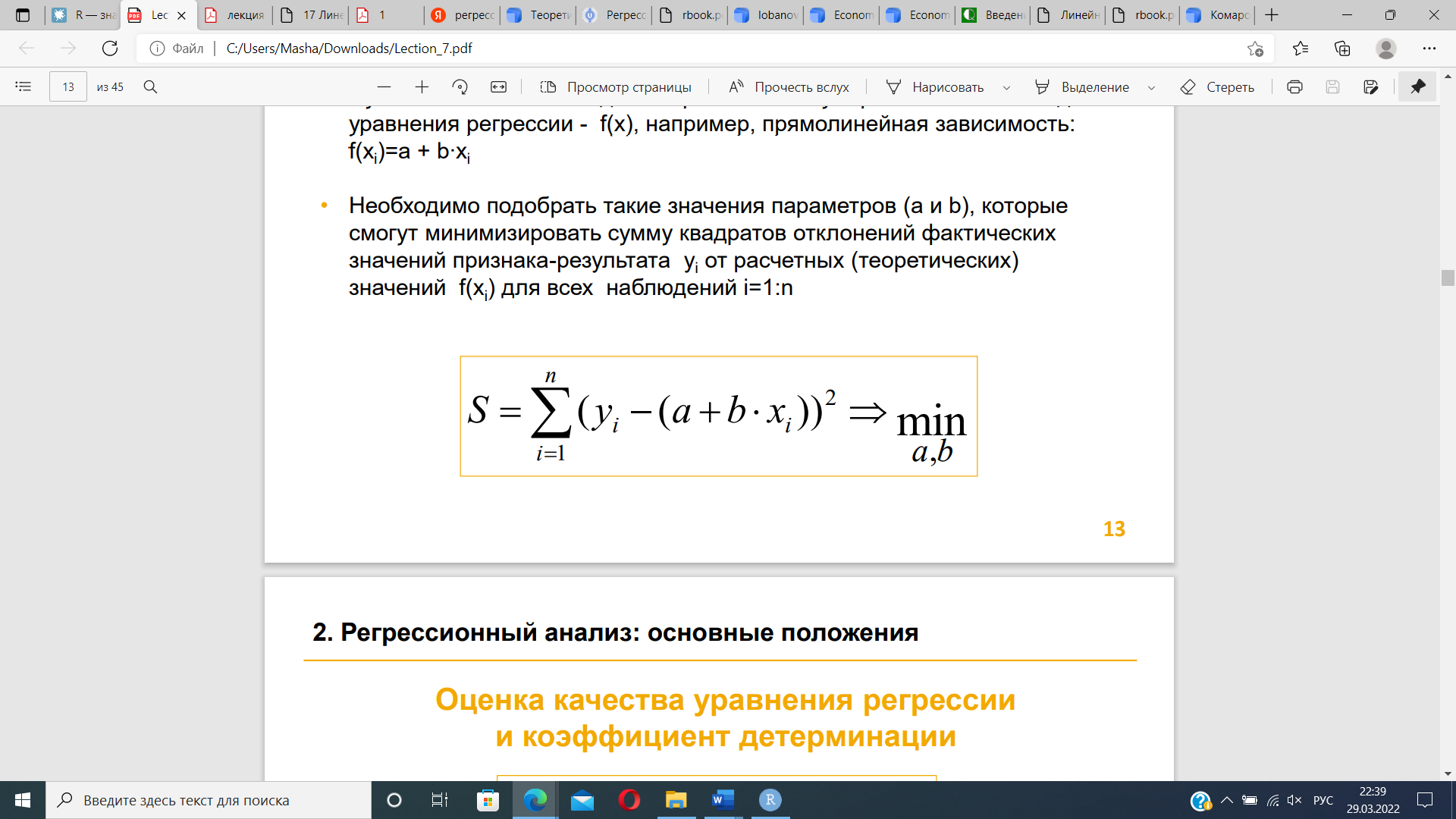
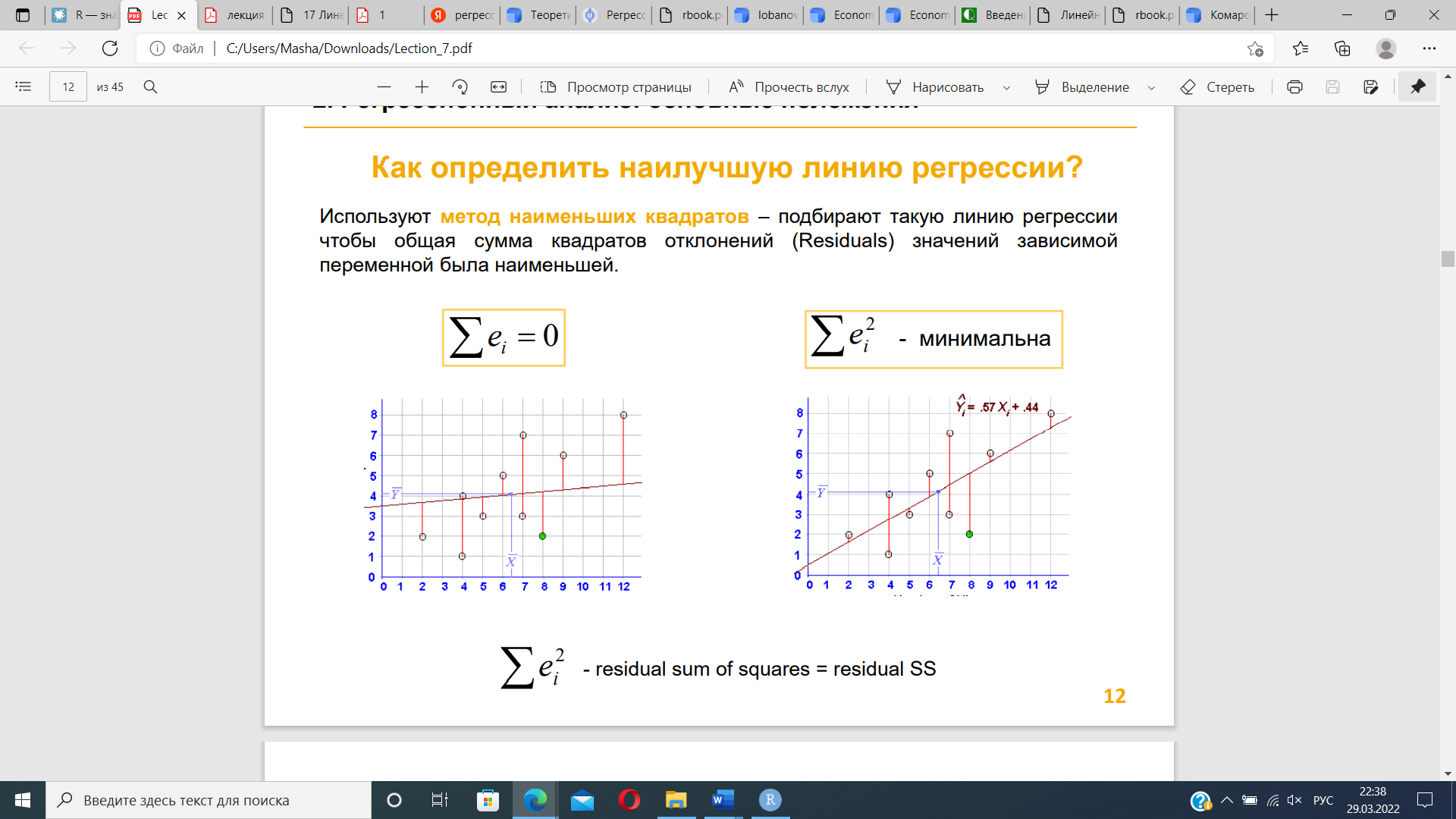


**Парная (простая) линейная регрессия.** Этот подход для прогнозирования количественного отклика Y на основе единственной предикторной переменной X. Предполагается, что есть приблизительная линейная взаимосвязь между Y и X. Математически можно записать эту взаимосвязь следующим образом:

Yi = a + bXi, где

Yi – зависимая переменная и Xi – независимая переменная, a – константа, b – угловой коэффициент, характеризует наклон прямой и показывает, на какую величину в среднем изменится результативный признак Yi, если переменная Xi увеличится на единицу своего измерения.

Для определения наилучшей линии регрессии используют **метод наименьших квадратов**, те добиваются, чтобы сумма квадратов остатков e была минимальной. Под остатками понимается разность между очередным наблюдением и прогнозом модели.



В общем случае для n наблюдений решают систему уравнений:



Толковой интерпретации регрессионных коэффициентов мешает также различие в единицах измерения. Например, если предиктор измеряется в сантиметрах, его вес

будет в 100 раз отличаться по весу от предиктора, берущегося в метрах. Чтобы избежать такого, мы должны **стандартизировать** единицы измерения предикторных переменных перед тем, как проводить регрессионный анализ. Стандартизация – это выражение переменных в процентилях.

При использовании линейной регрессии в качестве показателем тесноты связи выступает линейный **коэффициент корреляции** (чем ближе коэффициент по модулю к единице, тем теснее связь).

**Множественная регрессия** является расширением простой линейной регрессии. Она исследует влияние двух и более предикторов на критерий (Y=B1\*X1+B2\*X2+B3\*X3+…+A).

Построение уравнения множественной регрессии начинается с решения вопроса о спецификации модели, который включает 2 круга вопросов: отбор факторов и выбор уравнения регрессии.

**Факторы, включаемые во множественную регрессию, должны отвечать следующим требованиям:**

1. Они должны быть **количественно измеримы**. Если необходимо включить в модель качественный фактор, не имеющий количественного измерения, то ему нужно придать количественную определенность.

2. Каждый **фактор должен быть достаточно тесно связан с результатом** (т.е. коэффициент парной линейной корреляции между фактором и результатом должен быть существенным).

3. **Факторы не должны быть сильно коррелированы друг с другом** или находиться в строгой функциональной связи (т.е. они не должны быть интеркоррелированы). Мультиколлинеарность может привести к нежелательным последствиям. Существуют различные подходы преодоления сильной межфакторной корреляции. Простейший из них – исключение из модели факторов, в наибольшей степени ответственных за мультиколлинеарность. Определение факторов, ответственных за мультиколлинеарность, может быть основано на анализе матрицы межфакторной корреляции. При этом определяют пару признаков-факторов, которые сильнее всего связаны между собой (коэффициент линейной парной корреляции максимален по модулю). Из этой пары в наибольшей степени ответственным за мультиколлинеарность будет тот признак, который теснее связан с другими факторами модели (имеет более высокие по модулю значения коэффициентов парной линейной корреляции).

**Коэффициенты VIF** (variance inflation factor) показывают, насколько сильно связаны друг с другом регрессоры модели. Если коэффициенты VIF для всех регрессоров оказались меньше 10 (иногда используют 5), это значит, что существенной мультиколлинеарности в модели не наблюдается. В противном случае стоит сделать вывод о том, что в модели есть мультиколлинеарность.

4. **Отсутствие автокорреляции** – отсутствие независимости остатков. Выявляется с помощью теста Дурбина-Уотсона (обнаруживает автокорреляцию первого порядка).

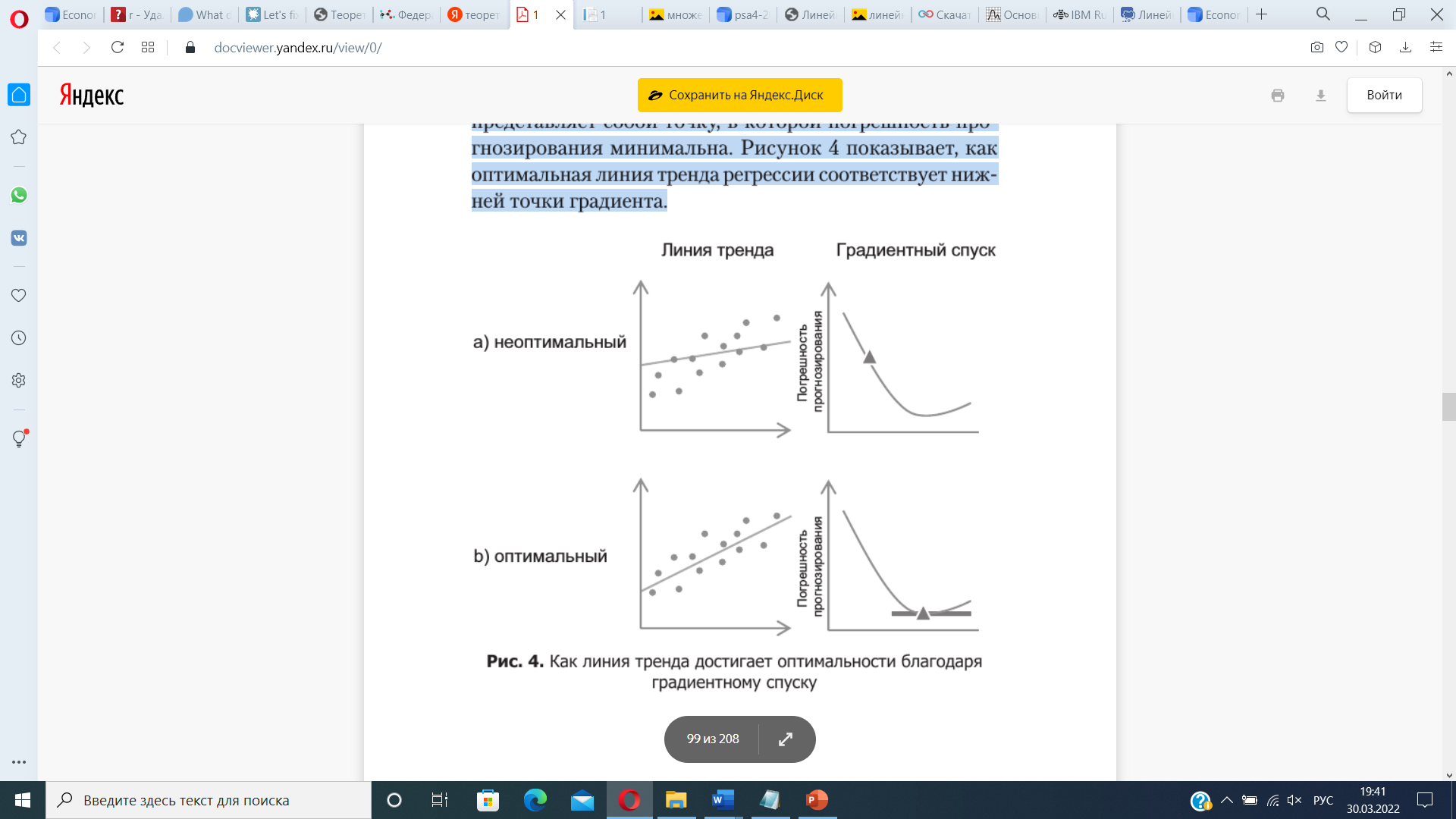
‒ Если d=2 – отсутствие автокорреляции.

При выборе формы уравнения множественной регрессии предпочтение отдается линейной функции в виду четкой интерпретации параметров. Параметры уравнения множественной регрессии можно также оценить методом наименьших квадратов, составив и решив систему нормальных линейных уравнений.

**Градиентный спуск (gradient descent)** используется в случаях, когда параметры уравнения нельзя получить путем решения систем уравнений. Алгоритм градиентного спуска делает первоначальное предположение о наборе весовых составляющих, после чего начинается итеративный процесс их применения к каждому элементу данных для прогнозирования, а затем они перенастраиваются для снижения общей ошибки прогнозирования.

Этот процесс можно сравнивать с пошаговым спуском в овраг в поисках дна. На каждом этапе алгоритм определяет, какое направление даст наиболее крутой спуск,

и пересчитывает весовые составляющие. В конечном итоге мы достигнем самой нижней позиции, которая представляет собой точку, в которой погрешность прогнозирования минимальна. Рисунок показывает, как оптимальная линия тренда регрессии соответствует нижней точки градиента.



Кроме регрессии градиентный спуск может также использоваться для оптимизации параметров в других моделях, таких как метод опорных векторов или в нейронных сетях.

**Оценка качества уравнения регрессии.**

Коэффициент детерминации рассматривают в качестве основного показателя, отражающего меру качества регрессионной модели. Он показывает, какая доля вариации объясняемой переменной y учтена в модели и обусловлена влиянием на нее факторов, включенных в модель:

- значения наблюдаемой переменной, – среднее значение по наблюдаемым данным, – модельные значения, построенные по оцененным параметрам.

Чем ближе R-квадрат к 1, тем выше качество регрессионной модели (факторы сильнее влияют на результат).

**Значимость уравнения регрессии и отдельных параметров.**

Проверить значимость уравнения регрессии – значит установить, соответствует ли

аналитическая модель экспериментальным данным, и достаточно ли включенных в уравнение объясняющих переменных для описания зависимой переменной. Оценка значимости уравнения регрессии в целом производится на основе **F-критерия Фишера** (чем больше значение параметра — тем лучше).

Для проверки значимости коэффициента регрессии **применяется t -распределение Стьюдента** (если есть основания считать, что между величинами Y и X нет линейной зависимости, то коэффициент статистически незначим-слишком близок к 0).

Если между изучаемыми явлениями существуют нелинейные соотношения, то

они выражаются с помощью соответствующих нелинейных функций (полиномы различных степеней, гипербола, степенная, показательная, экспоненциальная регрессии и тд).

**Для сравнения регрессионных моделей по степени точности предсказаний используются метрики оценки.**

***MSE (Mean Squared Error)*** *и*змеряет среднюю сумму квадратной разности между фактическим значением и прогнозируемым значением для всех точек данных. Самая популярная метрика, используемая для задач регрессии. Усиливается влияние ошибок по квадратуре от исходного значения.

MSE = , где

Чем меньше MSE, тем точнее наше предсказание. Оптимум достигается в точке 0. Является дифференцируемой, что позволяет более эффективно использовать для поиска экстремумов с помощью математических методов.

***Root Mean Squared Error (RMSE)*** *- к*орень от квадратной ошибки. Ее легко интерпретировать, поскольку она имеет те же единицы, что и исходные значения (в отличие от MSE). Также она оперирует меньшими величинами по абсолютному значению, что может быть полезно для вычисления на компьютере.

RMSE =

Итак, **основными этапами регрессионного анализа являются:**

1. Выбор вида уравнения регрессии (спецификация модели).

2. Выбор независимых переменных, оказывающих существенное влияние на

зависимую переменную.

3. Оценка параметров уравнения регрессии (параметризация модели).

4. Оценка статистической надежности регрессионной модели (верификация).

**Данные с подходящей структурой для выбранного метода**.

Для анализа используем данные автомобильной компании Geely Auto, представленные на сайте Kaggle: <https://www.kaggle.com/datasets/hellbuoy/car-price-prediction>

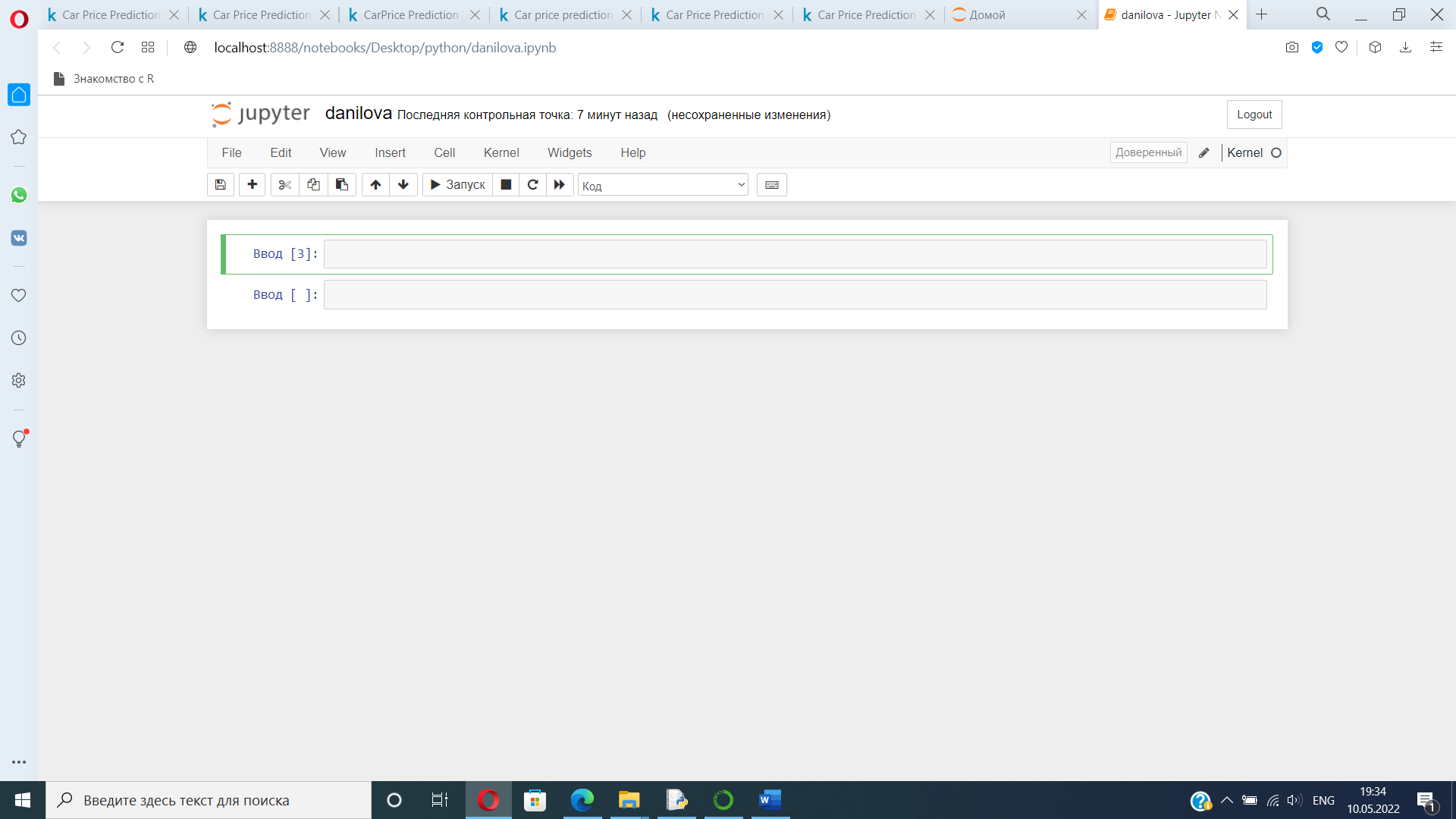
Задача будет состоять в том, чтобы определить взаимосвязь между различными параметрами автомобилей и их ценой на рынке (модель поможет руководству понять динамику ценообразования на рынке).

Столбцы таблицы:

|  |  |
| --- | --- |
| **Car\_ID** | Уникальный идентификатор |
| **wheelbase** | Колесная база автомобиля |
| **carlength** | Длина автомобиля |
| **carwidth** | Ширина автомобиля |
| **carheight** | Высота автомобиля |
| **curbweight** | Вес автомобиля без пассажиров и багажа |
| **enginesize** | Размер двигателя |
| **boreratio** | Борератио автомобиля |
| **stroke** | Ход или объем внутри двигателя |
| **compressionratio** | Степень сжатия автомобиля |
| **horsepower** | Лошадиная сила |
| **peakrpm** | Пиковые обороты автомобиля |
| **citympg** | Пробег в городе |
| **highwaympg** | Пробег по трассе |
| **price(Dependent variable)** | Цена автомобиля – целевая ячейка |

Для написания кода на Python используем среду разработки Anaconda Navigator - Jupiter Notebook.

Рабочее поле выглядит следующим образом:



Этап 1. Чтение и очистка данных.

Для чтения данных из .csv воспользуемся библиотекой pandas. Импортируем pandas:

import pandas as pd

Если в конце инструкции с import есть as pd, Jupyter Notebook понимает, что в будущем при вводе pd подразумевается именно библиотека pandas.

Обозначим директорию до файла:

file = 'CarPrice\_Assignment.csv'

Для загрузки .csv файла с данными в pandas используется функция read\_csv(), в которую передается строковый путь с данными

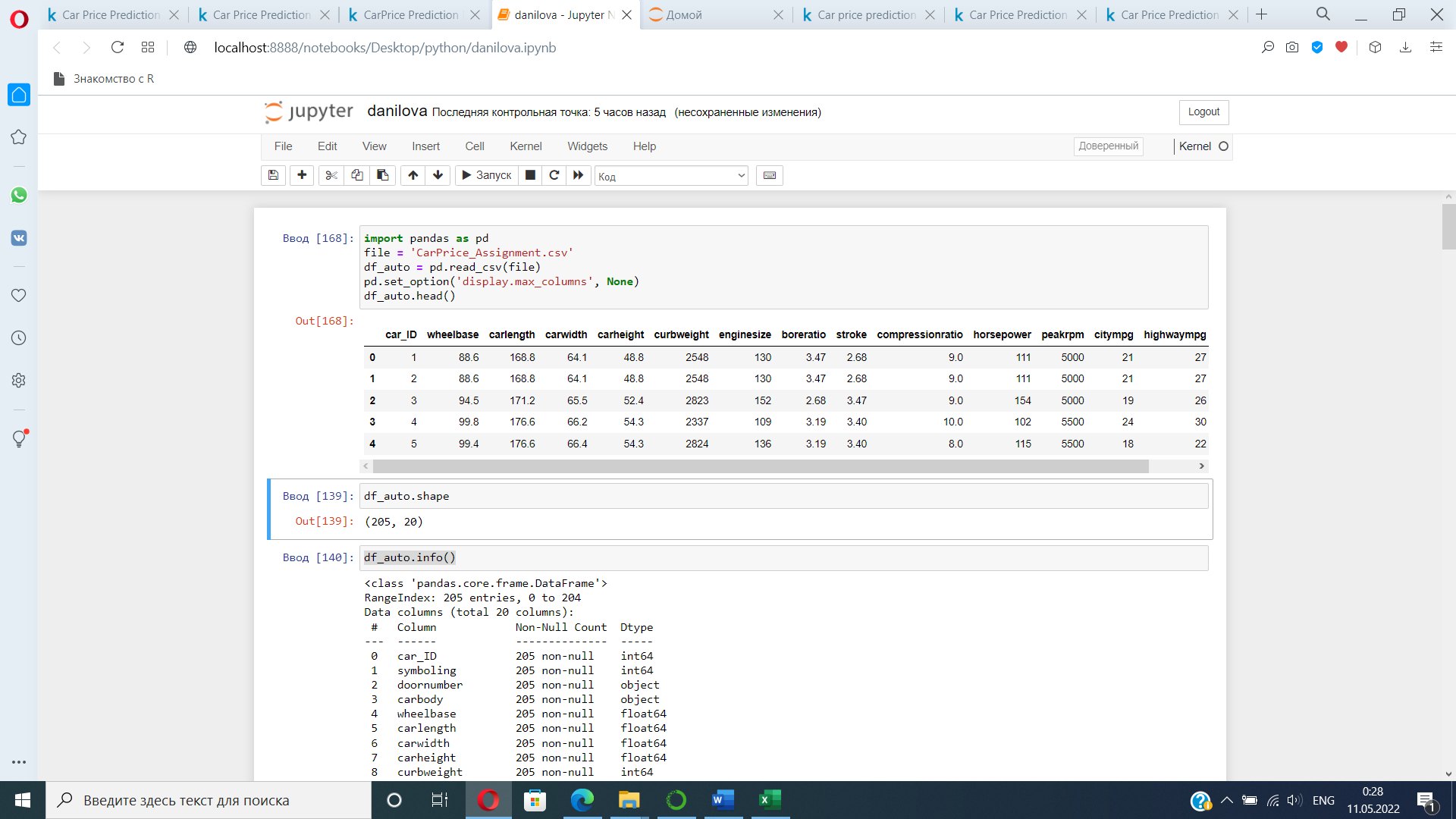
df\_auto = pd.read\_csv(file)

При установке значений опций ‘display.max\_rows’, ‘display.max\_columns’, ‘display.max\_colwidth’ как None, можно сбросить ограничения на число выводимых рядов, столбцов таблицы, а также на длину записи в таблице

pd.set\_option('display.max\_columns', None)

head() функция возвращает первые строки для объекта. Это полезно для быстрой проверки того, есть ли в вашем объекте правильный тип данных.

df\_auto.head()



Все считалось корректно.

Оценим размер датафрейма с помощью функции shape

df\_auto.shape

(205, 15)

То есть 205 строк и 15 столбцов

Выведем также сведения о датафрейме с помощью фии info()

df\_auto.info()



Видно, что пропущенных и нулевых значений нет

Для вывода описательной статистики (мин/макс/среднее по каждому столбцу) можно воспользоваться функцией describe()

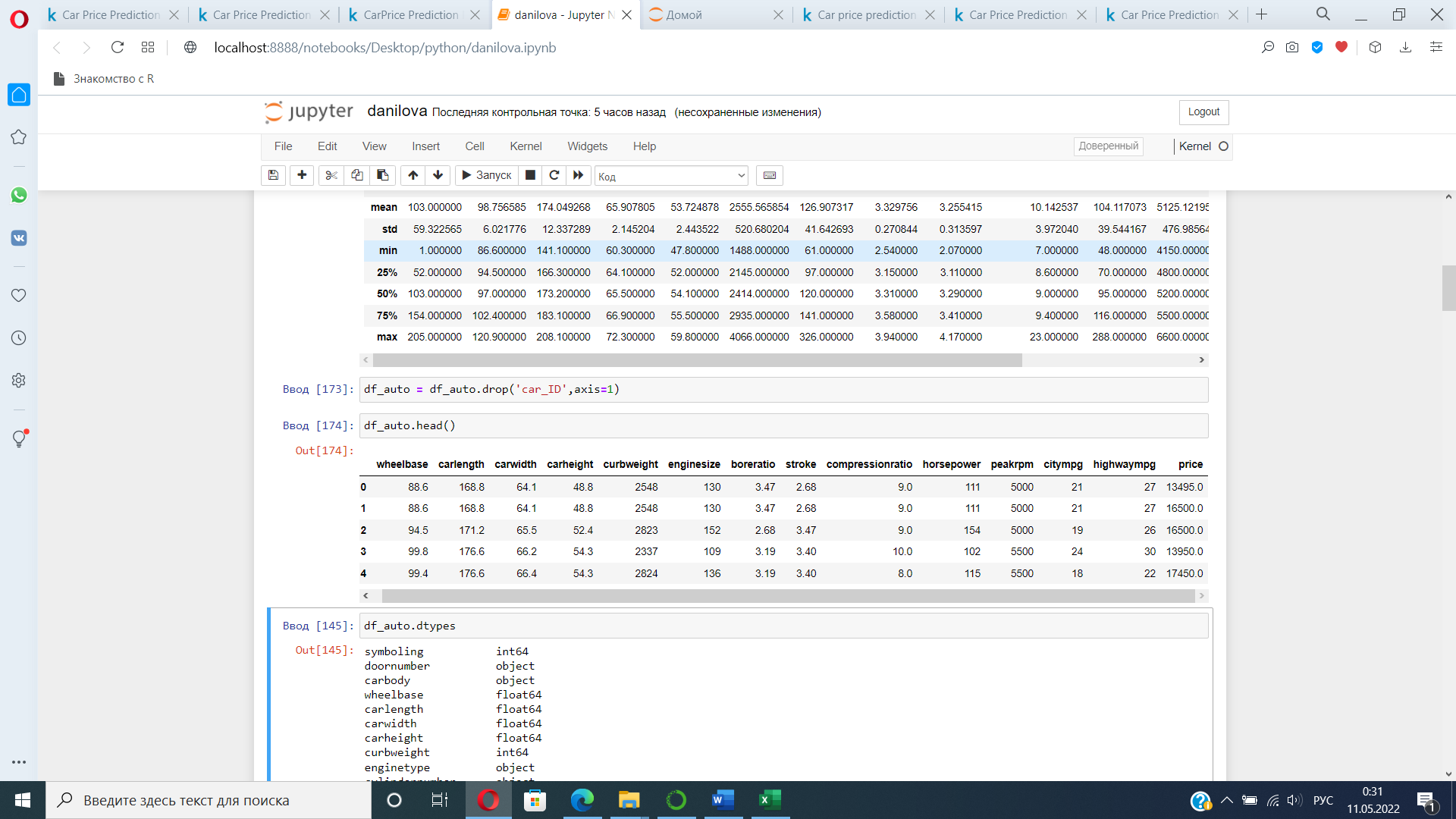
df\_auto.describe()



Удалим столбец с идентификатором, поскольку этот фактор не должен влиять на цену авто, параметр axis=1 задает удаление всего столбца

df\_auto = df\_auto.drop('car\_ID',axis=1)

Перепроверим



Этап 2. Визуализация данных

Здесь мы определим, имеют ли некоторые предикторы прямую связь с ценой.

Визуализируем распределение цен на автомобили, для этого воспользуемся библиотеками matplotlib.pyplot и seaborn

import matplotlib.pyplot as plt

import seaborn as sns

Установим тему для отображения и размер фигуры

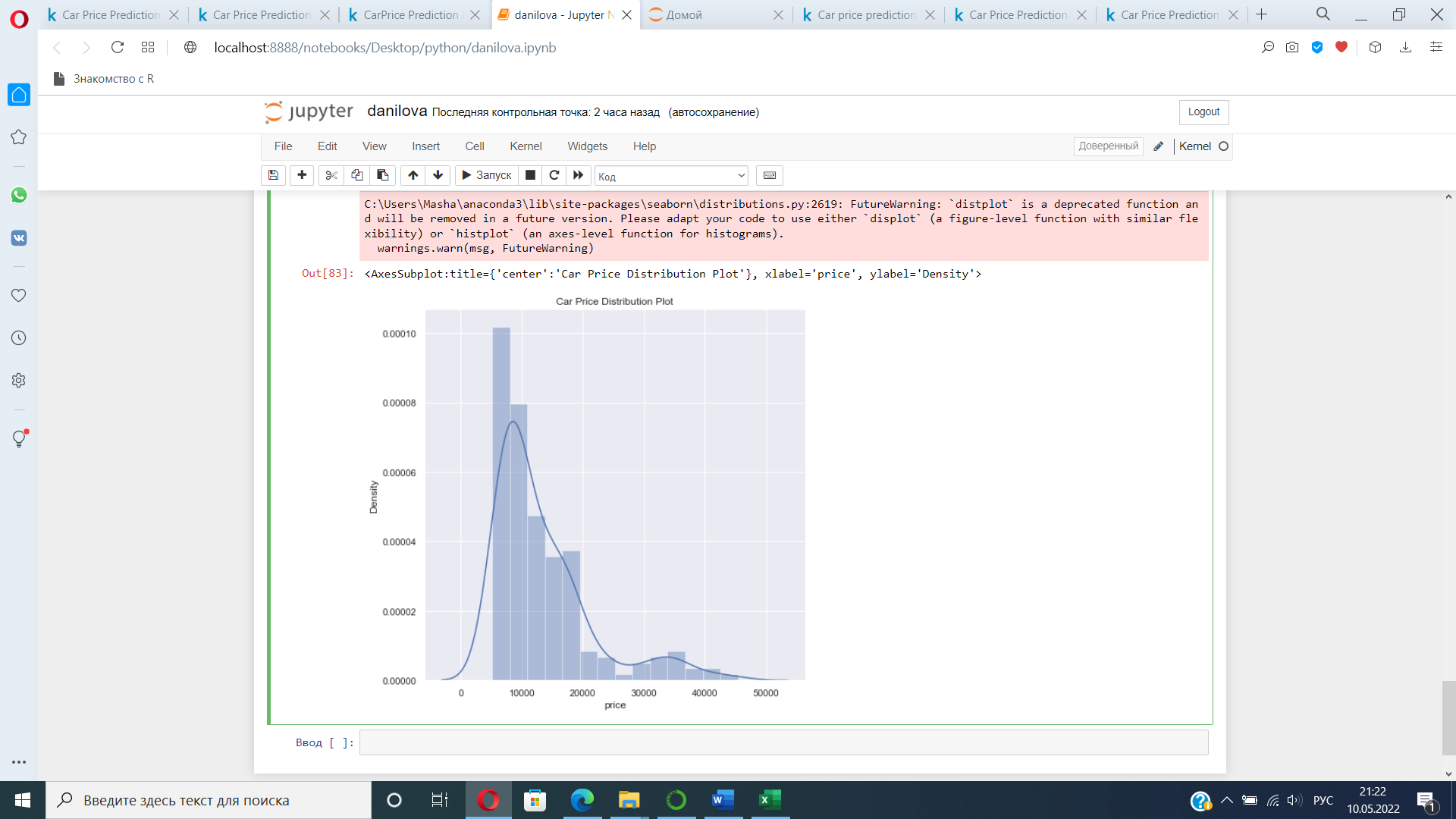
sns.set(style='darkgrid')

plt.rcParams['figure.figsize'] = [20,20]

Введем название графика и отрисуем график распределения по столбцу цены

plt.title('Car Price Distribution Plot')

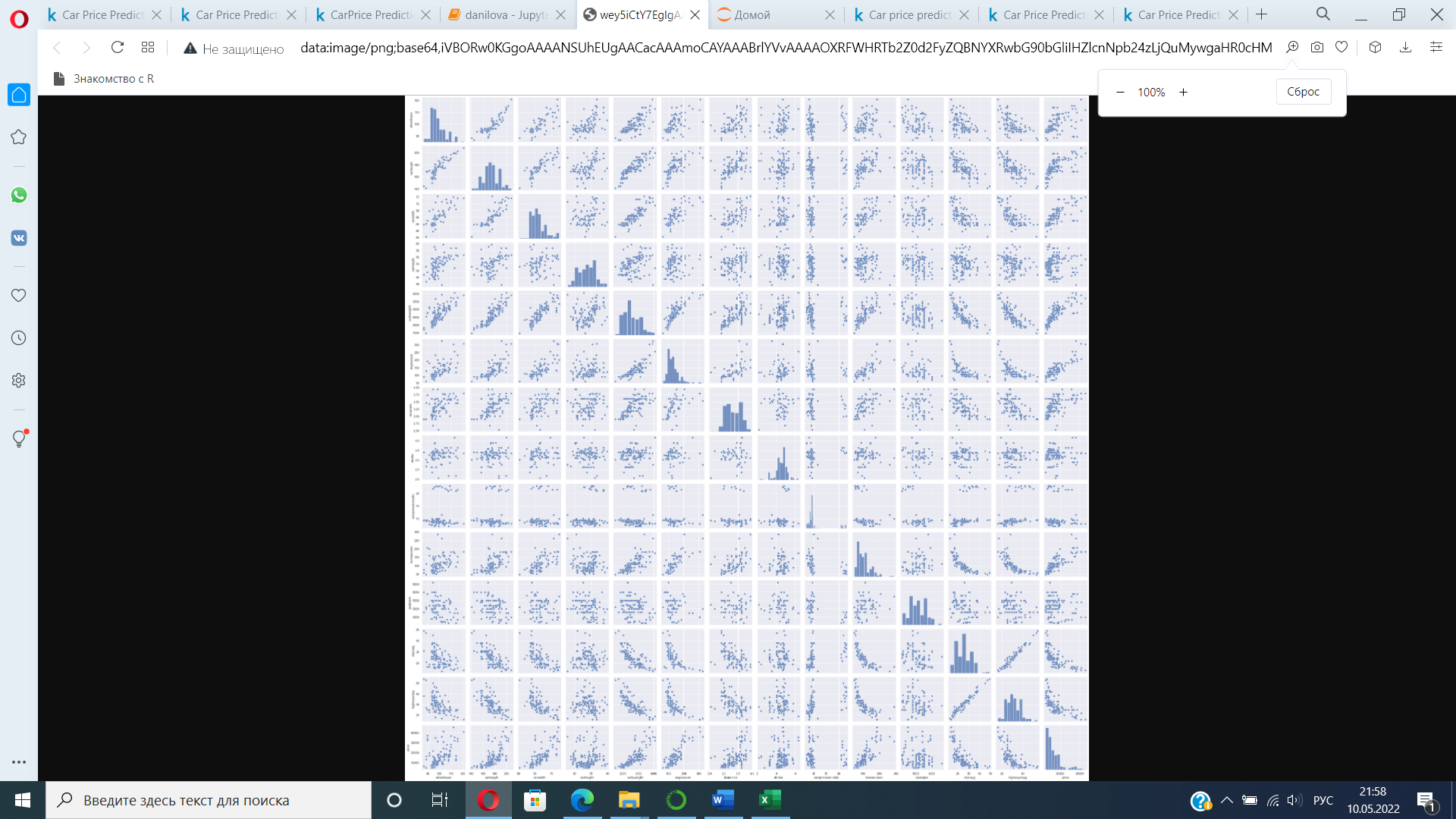
sns.distplot(df\_auto['price'])



Цена находится в диапазоне 10000-50000 долларов.

Поскольку в таблице много числовых столбцов, для выявления менее значимых факторов проанализируем матрицу взаимных корреляций числовых переменных между собой.

sns.pairplot(df\_auto)



В таком представлении трудно разобраться. В данном случае нас будет интересовать последний столбец – корреляция цены с другими параметрами, поэтому можно вывести корреляцию только со столбцом цены. Функция heatmap строит тепловую карту, первый ее параметр получает набор данных, в котором можно предварительно вычислить корреляционную матрицу столбцов, параметр annot отвечает за отображение текстовой информации, cmap – цветовое пространство

sns.heatmap(df\_auto.corr()[['price']], annot = True, cmap = 'coolwarm')



Чем ближе значение к 1, тем ближе цвет ячейки к красному, и чем ближе к -1, тем ближе к синему.

Видно, что carwidth, carlength, curbweight, enginesize, horsepower имеют положительную корреляцию с price.

citympg , highwaympg имеют отрицательную корреляцию с price.

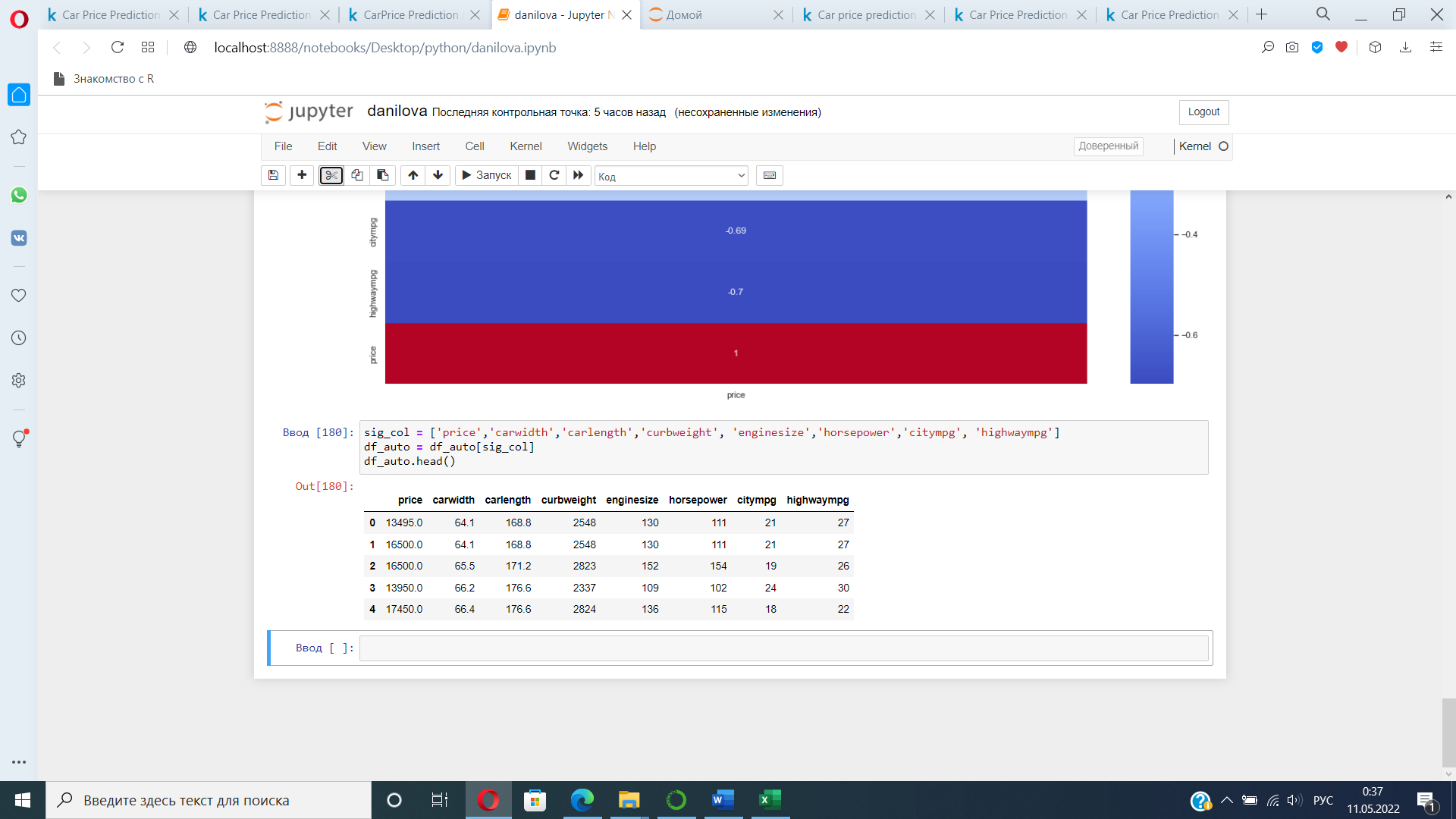
У всех остальных числовых столбцов низкая корреляция с ценой.

Запишем значимые переменные в отдельный массив и оставим в датафрейме данные только по этим переменным.

sig\_col = ['price','carwidth','carlength','curbweight', 'enginesize','horsepower','citympg', 'highwaympg']

df\_auto = df\_auto[sig\_col]

df\_auto.head()



Этап 3. Разделение данных на наборы для обучения и тестирования.

Используем для построения модели 70% представленных данных, и для тестирования качества модели в дальнейшем оставим 30%. Для этого используем функцию train\_test\_split из библиотеки sklearn.model\_selection. В качестве 1 параметра передается набор данных, в train\_size передается пропорция набора данных, которую нужно включить для построения модели, в test\_size – пропорция набора данных, которую нужно оставить для тестирования и параметр random\_state отвечает за перетасовку, применяемую к данным (нужно передать int). Функция random.seed() в Python используется для инициализации случайных чисел, для ее использования необходима библиотека numpy.

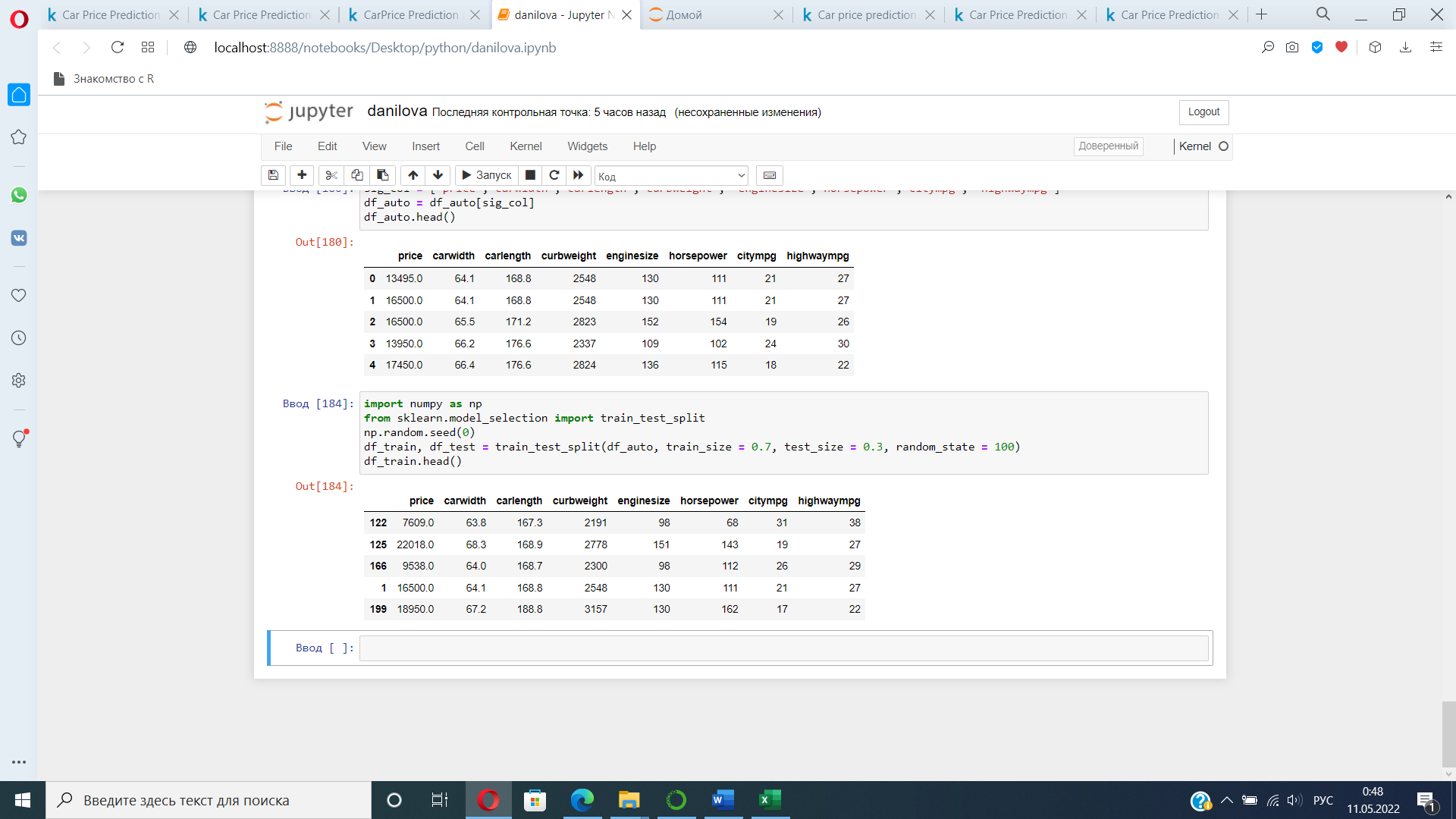
import numpy as np

from sklearn.model\_selection import train\_test\_split

np.random.seed(0)

df\_train, df\_test = train\_test\_split(df\_auto, train\_size = 0.7, test\_size = 0.3, random\_state = 100)

df\_train.head()



Этап 4. Построение линейной модели.

В качестве модели выберем линейную регрессию. Возьмем параметр с наибольшим коэффициентом корреляции с ценой. Также можно построить точечную диаграмму для каждой переменной и проанализировать, для какой из них хорошо прослеживается линейная зависимость (1 параметр-независимая переменная, 2 – зависимая, 3 – набор данных). Subplot делит область построения на несколько секций.

fig,axes = plt.subplots(2,4,figsize=(18,15))

plt.subplot(2,4,1)

sns.scatterplot(x='carwidth', y='price' ,data=df\_auto)

plt.subplot(2,4,2)

sns.scatterplot(x='carlength', y='price' ,data=df\_auto)

plt.subplot(2,4,3)

sns.scatterplot(x='curbweight', y='price' ,data=df\_auto)

plt.subplot(2,4,4)

sns.scatterplot(x='enginesize', y='price' ,data=df\_auto)

plt.subplot(2,4,5)

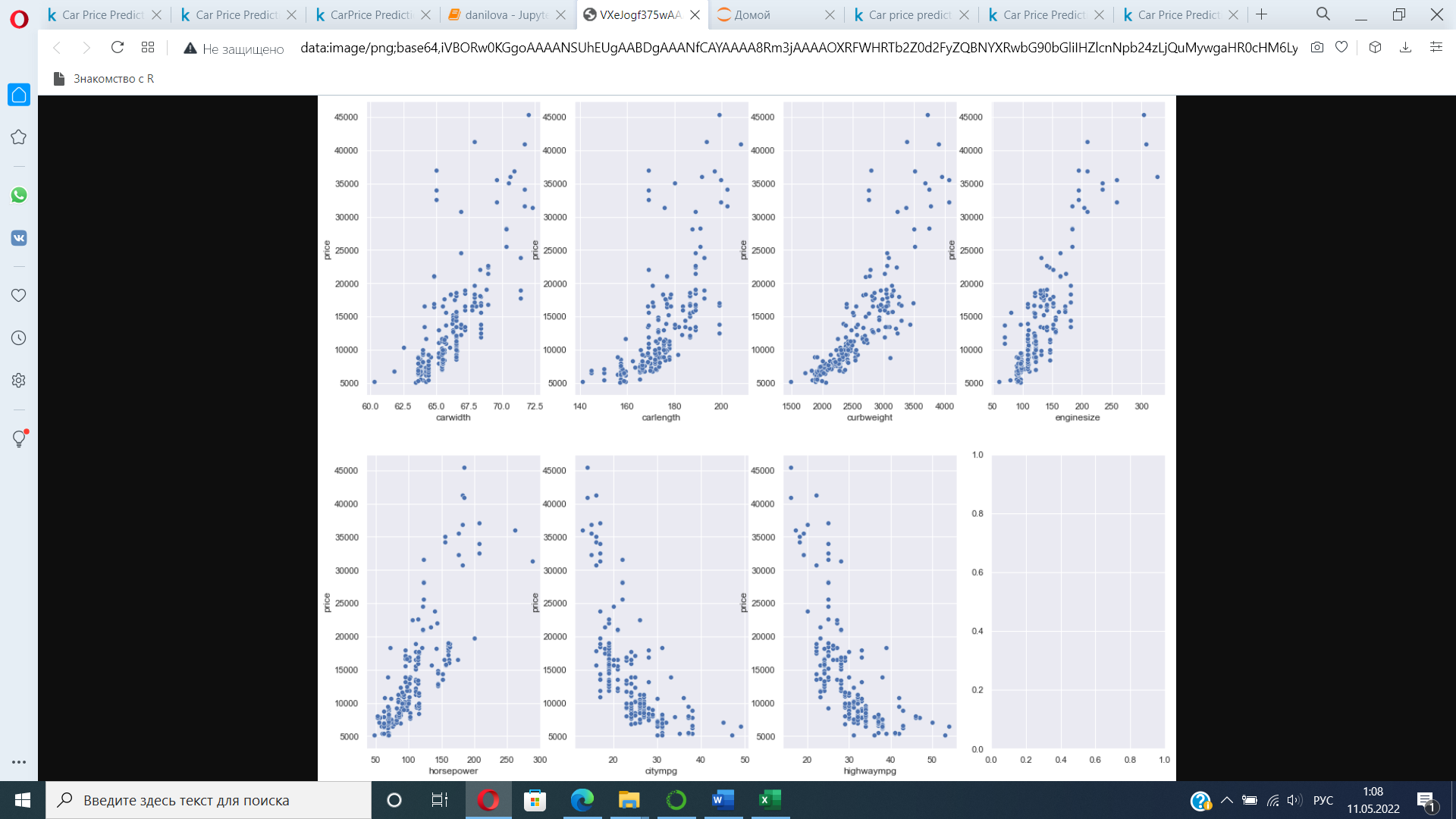
sns.scatterplot(x='horsepower', y='price' ,data=df\_auto)

plt.subplot(2,4,6)

sns.scatterplot(x='citympg', y='price' ,data=df\_auto)

plt.subplot(2,4,7)

sns.scatterplot(x='highwaympg', y='price' ,data=df\_auto)



Выберем в качестве фактора enginesize, тк ее коэффициент корреляции больше остальных (0.87) и для нее же лучше прослеживается линейная зависимость.

y\_train = df\_train['price']

x\_train = df\_train['enginesize']

Для построения регрессионной модели потребуется statsmodels - модуль Python, который предоставляет классы и функции для оценки множества различных статистических моделей, а также для проведения статистических тестов и исследования статистических данных.

import statsmodels.api as sm

Также добавим константу в нашу модель

x\_train1 = sm.add\_constant(x\_train)

Подгонка линейной модели с помощью наименьших квадратов.

lr\_1 = sm.OLS(y\_train, x\_train1).fit()

Выводим коэффициенты при переменных

lr\_1.params

Вывод:

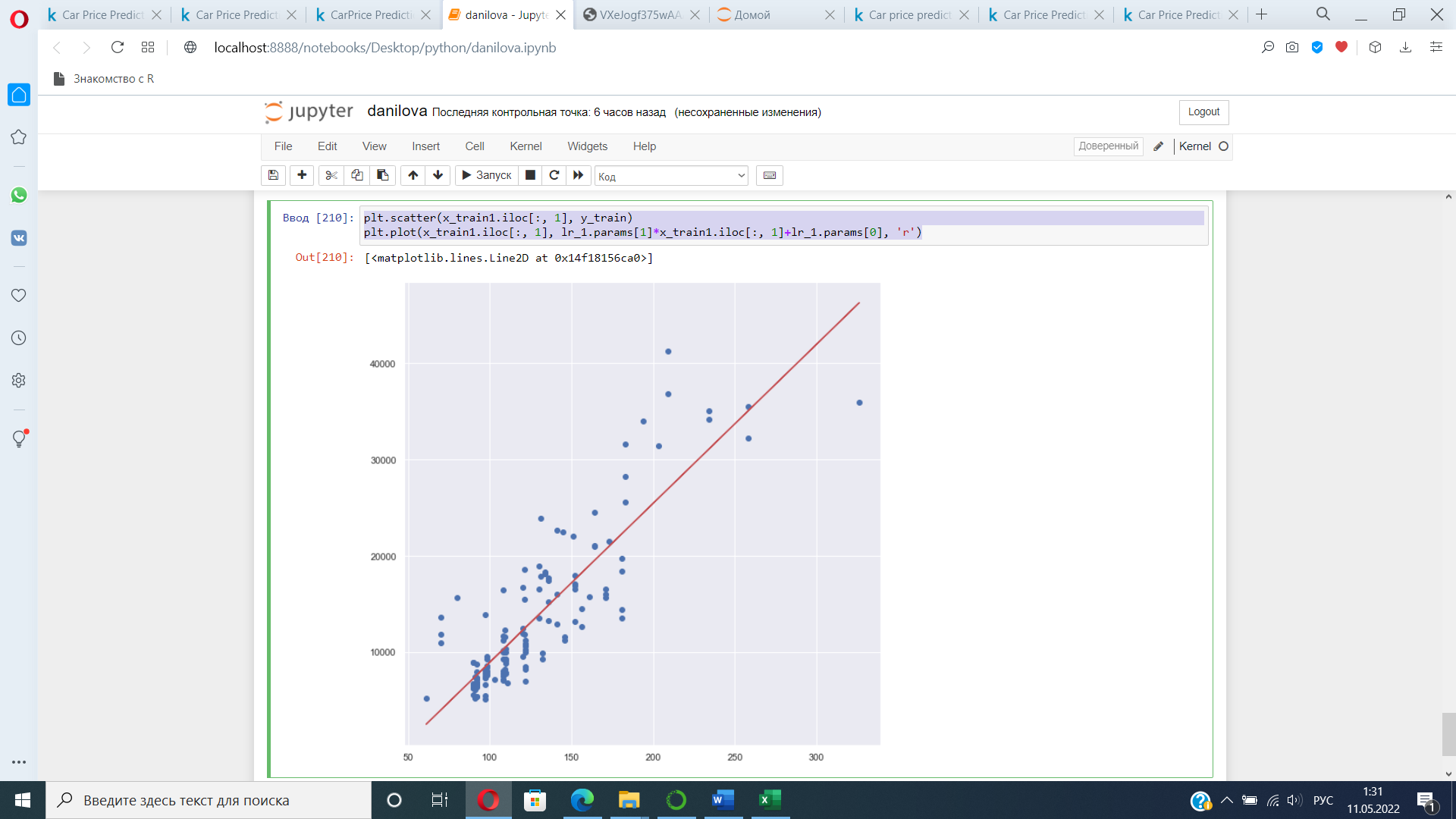
const -7607.862243

enginesize 165.369185

Далее снова построим точечную диаграмму и подобранную линию регрессии (iloc используется для того, чтобы выбрать необходимые столбцы/строки)

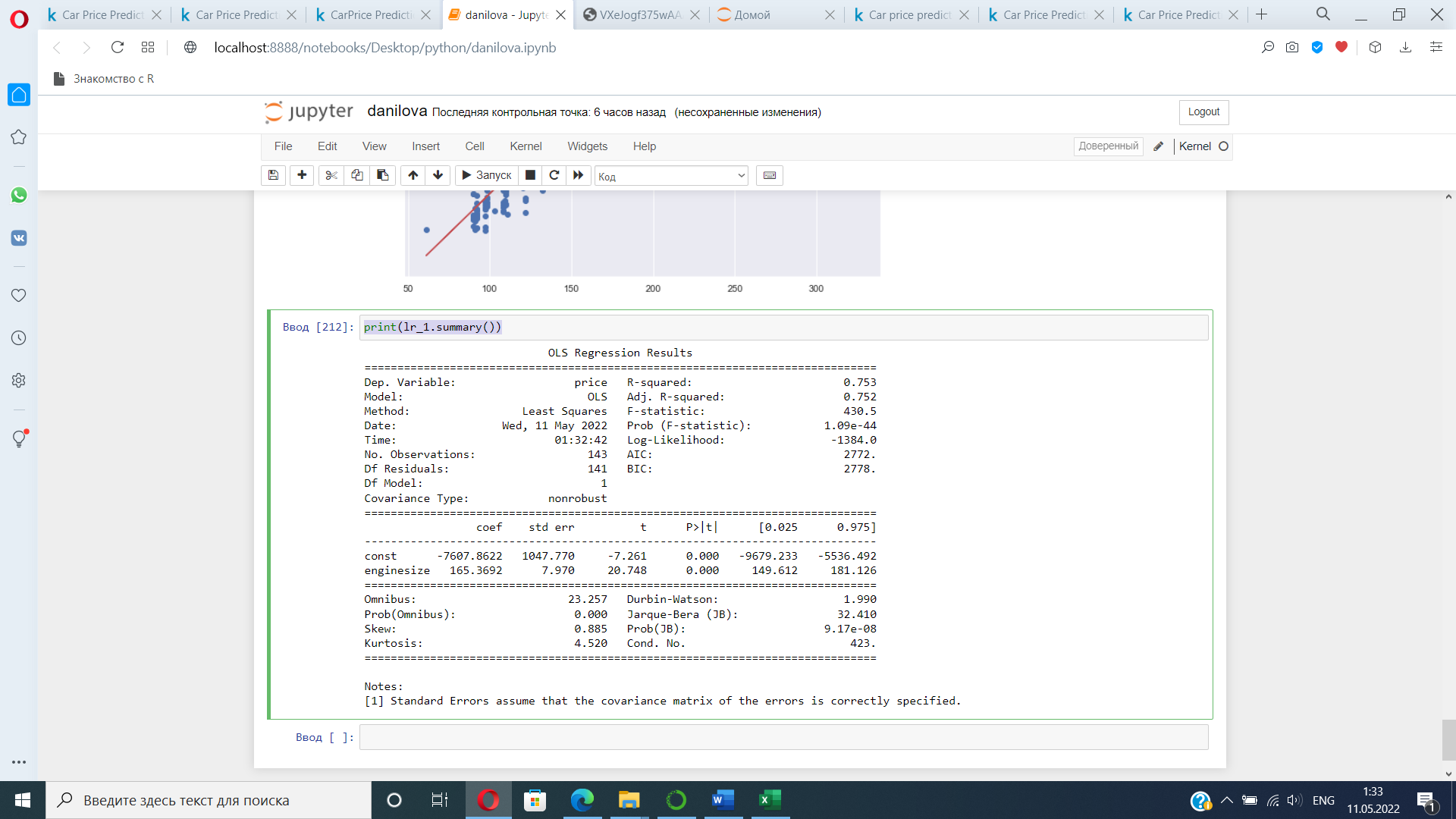
plt.scatter(x\_train1.iloc[:, 1], y\_train)

plt.plot(x\_train1.iloc[:, 1], lr\_1.params[1]\*x\_train1.iloc[:, 1]+lr\_1.params[0], 'r')



Чтобы посмотреть сведения о линейной аппроксимации используется функция summary

print(lr\_1.summary())



Интересующие нас обозначения:

coef – коэффициенты линейной регрессии.

std err – стандартная ошибка

R-squared — коэффициент детерминации; указывает, насколько тесной является связь между факторами регрессии и зависимой переменной. Чем ближе к 1, тем ярче выражена зависимость. В данном случае равен 0.753.

F-statistic — используется для оценки значимости модели регрессии в целом (чем больше значение параметра, тем лучше).

t — критерий, основанный на t распределении Стьюдента. Значение параметра в линейной регрессии указывает на значимость фактора, можно считать, что при |t|>2 фактор является значимым для модели.

P>|t| — вероятность истинности нуль гипотезы, которая гласит, что независимые переменные не объясняют динамику зависимой переменной. Если значение p-value ниже порогового уровня (0.05), то нуль гипотеза ложная. Чем ниже — тем лучше.

Durbin-Watson – тест, проверяющие наличие автокорреляции. Чем ближе к 2, тем меньше автокорреляция.

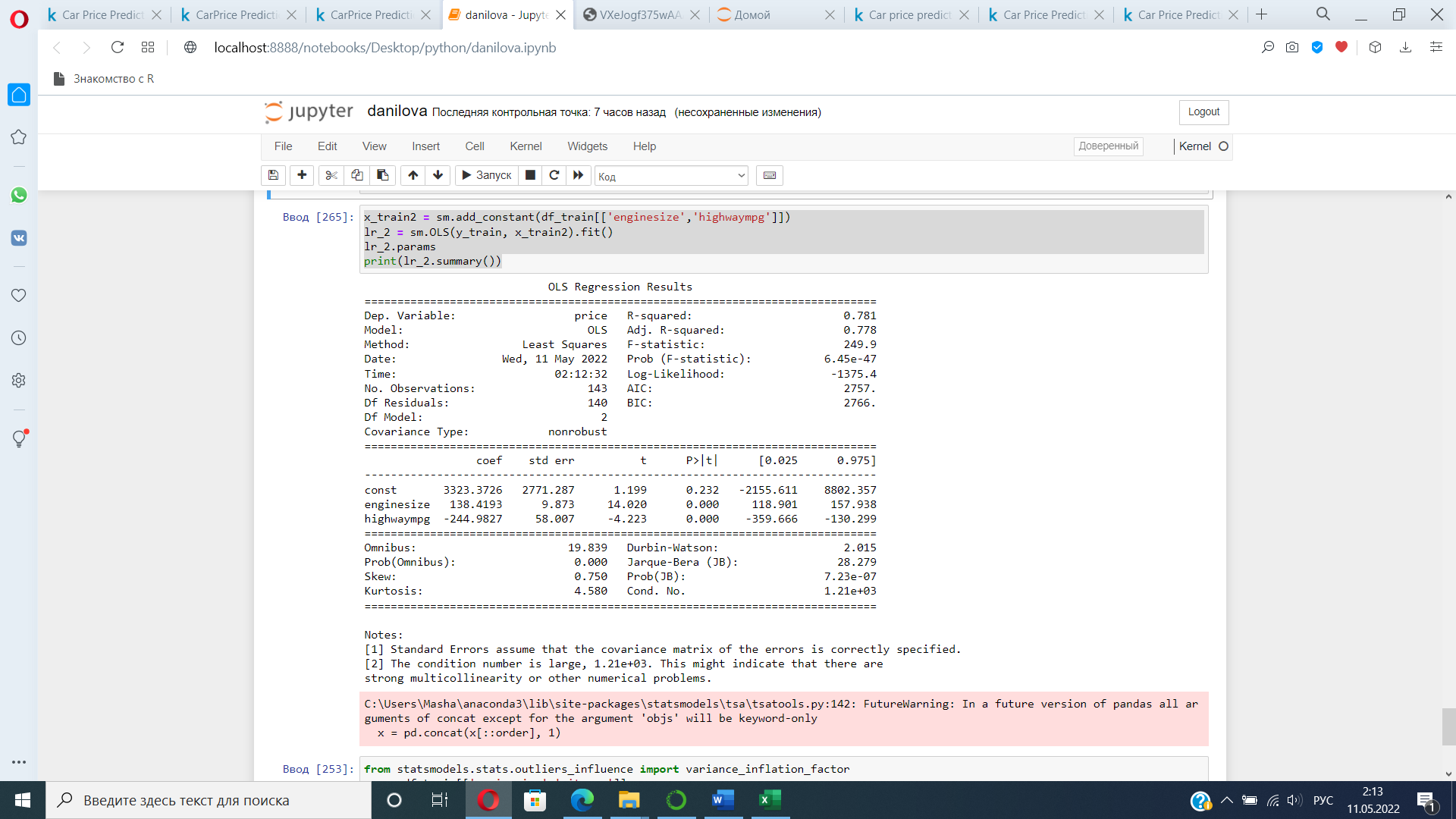
Можно попытаться улучшить полученный результат, добавив еще одну переменную, сильно коррелирующую с ценой, например highwaympg (у нее наибольшая отрицательная корреляция с ценой)

x\_train2 = sm.add\_constant(df\_train[['enginesize','highwaympg']])

lr\_2 = sm.OLS(y\_train, x\_train2).fit()

lr\_2.params

print(lr\_2.summary())



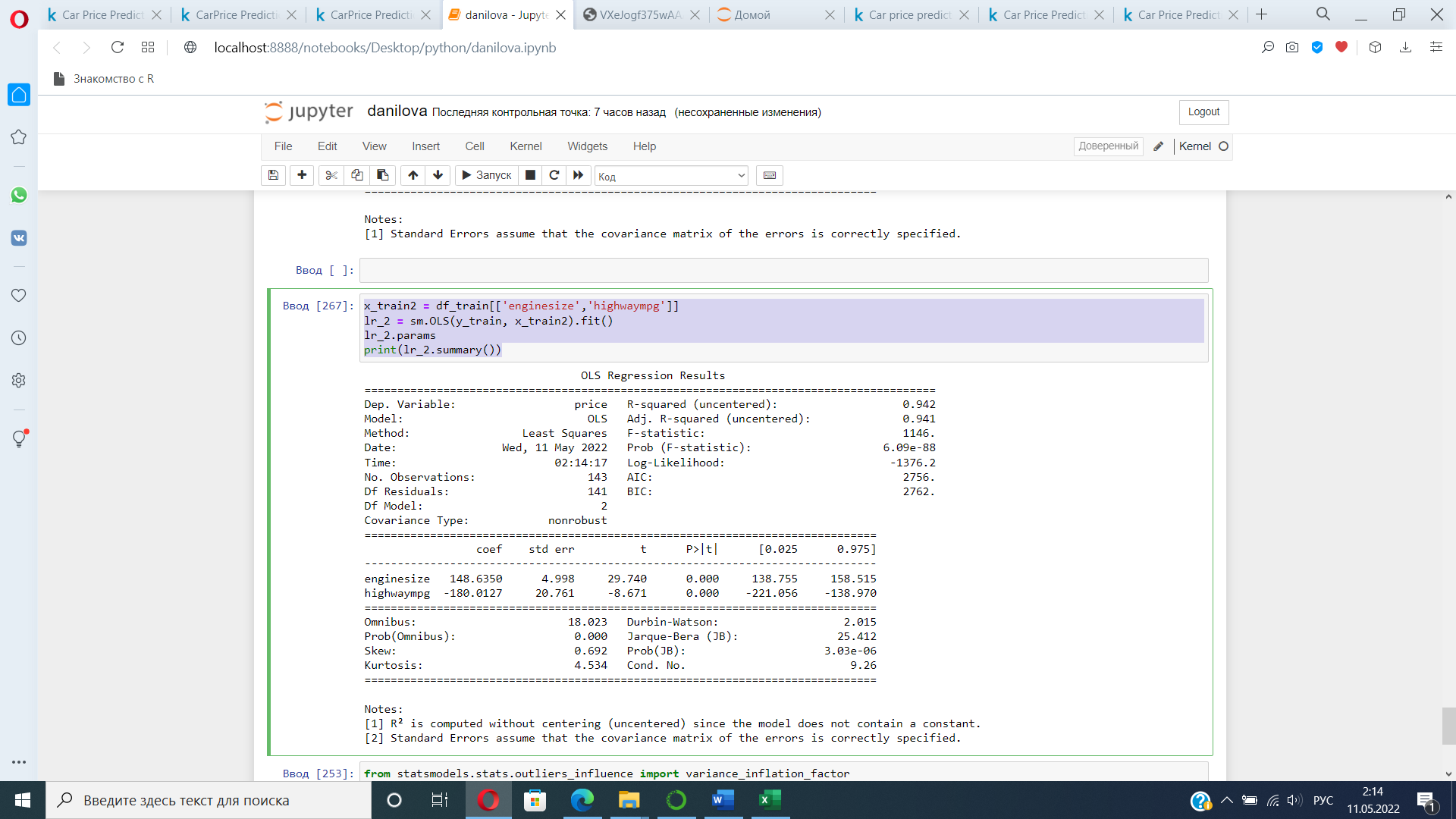
Видим, что R-статистика улучшилась, однако t-value константы меньше 2, поэтому уберем ее из рассмотрения:

x\_train2 = df\_train[['enginesize','highwaympg']]

lr\_2 = sm.OLS(y\_train, x\_train2).fit()

lr\_2.params

print(lr\_2.summary())



R-squared значительно возросло до 0.942, что является неплохим результатом; также все переменные являются значимыми.

Также вычислим значение vif (отвечает за мультиколлинеарность, если больше 5-10), для этого импортируем variance\_inflation\_factor и обозначим данные, для которх будем вычислять vif

from statsmodels.stats.outliers\_influence import variance\_inflation\_factor

xxx = x\_train2

Создадим фрейм данных, который будет содержать имена переменных (columns) и их соответствующие VIF

vif = pd.DataFrame()

vif['Features'] = xxx.columns

Рассчитываем VIF для каждой переменной (shape возвращает размер датафрейма)

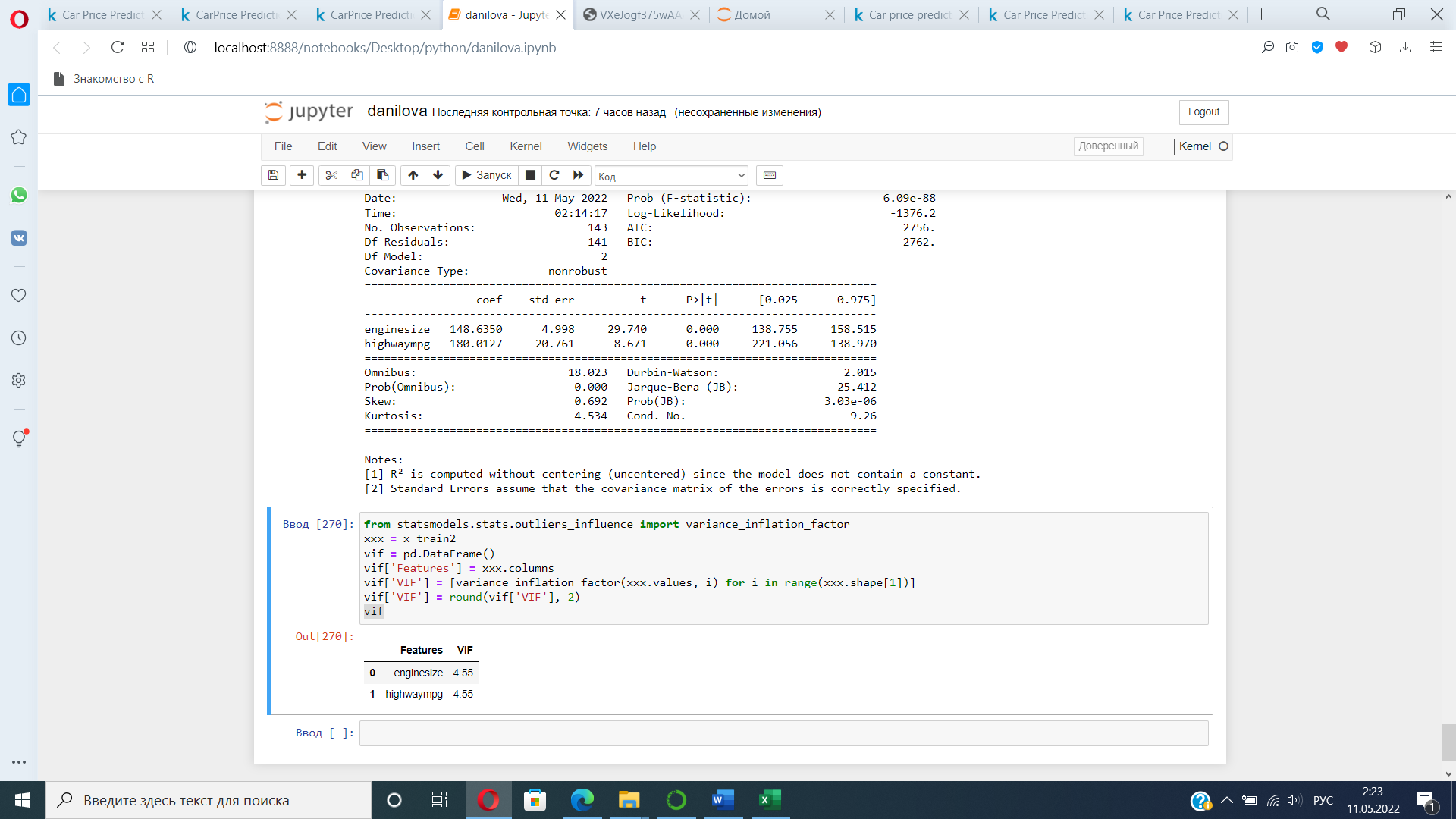
vif['VIF'] = [variance\_inflation\_factor(xxx.values, i) for i in range(xxx.shape[1])]

Округлим полученные значения

vif['VIF'] = round(vif['VIF'], 2)

И выведем их

vif



Мультиколлинеарности в модели нет.

Построим гистограмму, чтобы проверить, нормально ли распределены ошибки. Для этого сначала сделаем прогноз с помощью полученной модели, затем по вычисленной между предсказанными и исходными значениями разнице построим гистограмму с помощью distplot (bins задает количество корзин разбиения)

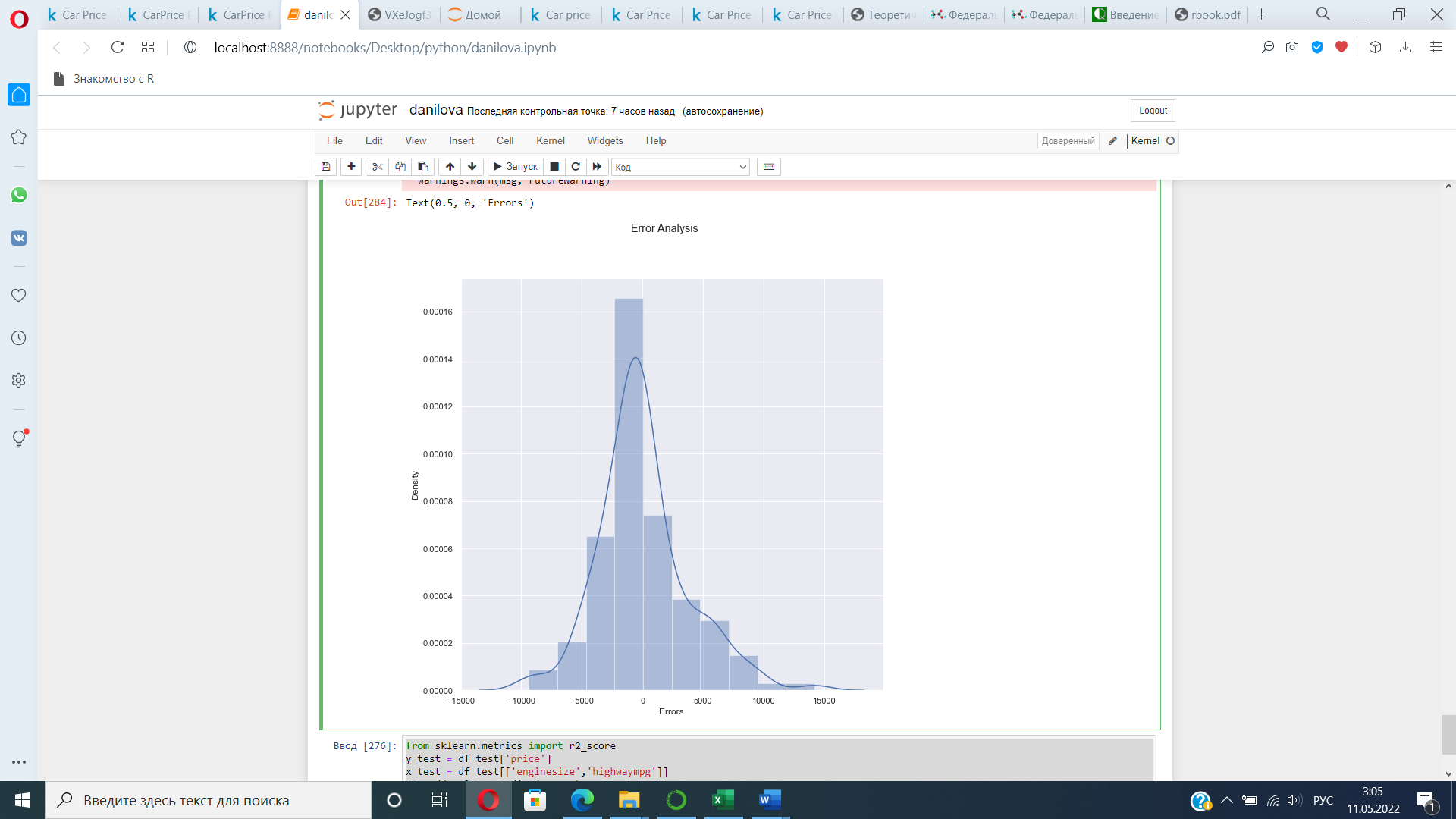
y\_pred = lr\_2.predict(x\_train2)

fig = plt.figure()

sns.distplot((y\_train - y\_pred), bins = 10)

fig.suptitle('Error Analysis')

plt.xlabel('Errors')



Видно, что ошибки имеют нормальное распределение, что соответствует требованиям линейной регрессии.

Теперь с помощью полученной модели сделаем прогноз по тестовым данным и вычислим R-статистику.

Импортируем необходимый модуль для вычисления статистики

from sklearn.metrics import r2\_score

Обозначим зависимые и независимые переменные из тестового набора данных

y\_test = df\_test['price']

x\_test = df\_test[['enginesize','highwaympg']]

Сделаем прогноз

y\_predd = lr\_2.predict(x\_test)

Посчитаем R-квадрат

r2\_score(y\_test, y\_predd)

Результат:

0.89982151

достаточно близок к 0.9

Таким образом, полученная регрессионная модель:

price = enginesize\*148.6350 - highwaympg\*180.0127,

enginesize – размер двигателя, highwaympg – пробег по трассе.

Полученную модель можно видоизменять, выбирая другие зависимости (нелинейные) или переменные.

**Список использованных источников:**

1. Marsden Eric. Regression analysis using Python. [Электронный ресурс]. – URL: https://risk-engineering.org/static/PDF/slides-linear-regression.pdf (дата обращения: 10.05.2022).

2. Баранова Т. А. Многомерные статистические методы. Регрессионный анализ. [Электронный ресурс]. – URL: http://main.isuct.ru/files/publ/PUBL\_ALL/167.pdf (дата обращения: 10.05.2022).

3. Любимцев О. В. Линейные регрессионные модели в эконометрике. [Электронный ресурс]. – URL: https://bibl.nngasu.ru/electronicresources/uch-metod/economic\_statistics/859984.pdf?ysclid=l30u09b04g (дата обращения: 10.05.2022).

4. Рындина С. В. Бизнес-аналитика на основе больших данных: обучение с учителем на языках Python и R. – URL: https://elib.pnzgu.ru/files/eb/lX6js0tINm8c.pdf (дата обращения: 10.05.2022).

Данные: <https://www.kaggle.com/datasets/hellbuoy/car-price-prediction>