



Общество с ограниченной ответственностью
«ГикБрейнс»

125167, г. Москва, Ленинградский пр-кт, д. 39, стр. 79, этаж 23,
пом. XXXIV, часть комнаты 1

ОГРН 1167746654569, ИНН 7726381870, КПП 771401001

Лицензия на осуществление образовательной деятельности № ЛО35-01298-77/00181496 от 03.12.2019 г.

Дополнительная профессиональная программа профессиональной
переподготовки «Искусственный интеллект» по заочной форме обучения с
использованием дистанционных образовательных технологий

**ДИПЛОМНАЯ РАБОТА
ПО ТЕМЕ:**

**“Изучение рынка продаж подержанных автомобилей:
на примере прогнозирования цены на подержанные автомобили Ford
с помощью построения регрессионных моделей и их анализа”**

Слушатель заочной формы обучения
МАЗУРОВА НАТАЛЬЯ НИКОЛАЕВНА

Санкт-Петербург

2024

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	3
ГЛАВА 1 ПРОЕКТ.....	6
1.1 Предложенный проект.....	6
1.2 Платформе Kaggle.....	7
1.3 Дата-сайентист.....	11
ГЛАВА 2 ДАННЫЕ.....	13
2.1 Обзор данных.....	13
2.2 План-алгоритм анализа данных.....	14
2.2.1 Загрузка библиотек.....	15
2.2.2 Загрузка данных.....	16
2.2.3 Загрузка данных для обучения.....	17
2.2.4 Обработка данных перед обучением модели.....	19
2.2.5 Обучение модели на обучающей выборке.....	28
2.2.6 Загрузка и предобработка данных для тестирования.....	31
2.2.7.Проанализирование модели на тестовой выборке.....	37
2.2.8 Загрузка данных для обучения.....	41
ГЛАВА 3 Google Colab.....	42
3.1. Google Colab. Настройка доступа. GitHub.....	42
ЗАКЛЮЧЕНИЕ.....	43
СПИСОК ЛИТЕРАТУРЫ.....	47

ВВЕДЕНИЕ

Ссылка на дипломную работу:

GitHub https://github.com/MazurovaNN/DIPLOM_2024

Тема проекта: “Изучение рынка продаж подержанных автомобилей: на примере прогнозирования цены на подержанные автомобили Ford с помощью построения регрессионных моделей и их анализа”

Цель: Изучение рынка продаж подержанных автомобилей, его особенностей в условиях конкуренции с применением построения регрессионных моделей и их анализа и как результат улучшение процесса продаж подержанных автомобилей на рынке продаж с учетом полученных рекомендаций по актуальным ценам.

Актуальной остается задача: Предсказание цены на подержанные автомобили (на примере марки) Ford

Задачи:

1. Изучить литературу, касающуюся темы исследования.
2. Рассмотреть основные виды, методы и применения построения регрессионных моделей и их анализа.
3. Ознакомиться с основными принципами составления и составить план-алгоритм анализа данных.
4. Составить план по работе и загрузить данные, а также загрузить данные для обучения.
5. Выполнить обработку данных перед обучением модели, обучить модель на обучающей выборке, загрузить и предобработать данные для тестирования, проанализировать модель на тестовой выборке.

6. Разработать предложения и рекомендации применения лучшей модели и рекомендации по улучшению ценообразования на подержанные автомобили для рынка продаж.

Бизнес-постановка задачи

Оценка подержанного автомобиля - это достаточно трудная задача, так как на стоимость влияют различные факторы, например:

возраст автомобиля, его состояние, пробег и даже личное отношение продавца.

Таким образом, цена подержанных автомобилей на рынке не является постоянной. И поскольку нет прозрачности в её образовании, а спрос растет ежегодно, у нечестных предпринимателей возникает стимул иррационально завышать цену.

Инструменты и методы: библиотека `pandas`, для корректной работы с графиками в `python` загружена специальная библиотека `matplotlib`, при импорте `matplotlib.pyplot`, программная библиотека на языке `python` для визуализации данных двумерной и трехмерной графикой. Графики используются для облегчения интерпретации полученных результатов, а также в качестве иллюстраций в презентациях и отчетах.

В работе применены два метода для построения моделей и произведено сравнение их между собой: Линейная регрессия *linear regression*, Лес решающих деревьев *random forest*

Для корректного подсчета метрик качества модели в `python` загружены из библиотеки `sklearn`, `sklearn.metrics`.

Основные **методы** для построения графиков:

1. `plot()` - графики
2. `semilogy()` - график логарифмический
3. `hist()` – гистограммы

Метрики качества модели:

1. `mean_absolute_error` - средняя абсолютная ошибка
2. `mean_squared_error` - средняя квадратичная ошибка
3. RMSE - корень из `mean_squared_error`

Структура: Дипломная работа состоит из введения, трех глав, первая разбита на три параграфа, вторая - на два, в третьей один, заключения и списка литературы. Общим объемом 48 страниц.

Для точных прогнозов цен требуется профессионализм таких специалистов, как например ДатаСейнтист.

Анализу рынка посвящена данная работа.

ГЛАВА 1 ПРОЕКТ

1.1 Предложенный проект

Предложенный проект – это

Модель для оценки стоимости подержанного автомобиля, которая помогла бы

1. покупателям не переплатить за желаемое авто, а
2. честным продавцам быстро устанавливать цену, соответствующую их предложениям.
3. Постановка задачи анализа данных

Целью данной задачи является прогнозирование цены на подержанные автомобили Ford с помощью **построения регрессионных моделей и их анализа.**

Следует отметить, что Набор данных состоит из информации о транспортных средствах, выставленных на продажу на сайте Craigslist.

1.2 Платформа Kaggle

Данные опубликованы в открытом доступе на платформе Kaggle.

Платформа Kaggle

Специалистам в области Data Science необходимо постоянно учиться и улучшать свои навыки.

Платформа Kaggle помогает начинающим дата-сайентистам практиковаться на реальных данных, а опытным — изучать работу коллег и соревноваться с ними.

Kaggle — это сообщество специалистов по Data Science. Здесь можно изучать машинное обучение, писать свои и разбирать чужие прогнозные модели, участвовать в соревнованиях и общаться с дата-сайентистами. Сервис полностью бесплатен.

Kaggle используют и начинающие, и опытные дата-сайентисты со всего мира.

Есть пользовательский рейтинг — очки в нем можно заработать за решение задач по машинному обучению, обсуждение на форуме, публикацию своего кода и наборов данных.

ВАЖНО!!! Многие компании при найме обращают внимание на место соискателя в рейтинге Kaggle.

Поможет освоить основные принципы Machine Learning и Data Science

В разделе Learn есть больше десятка полезных курсов: введение в SQL, введение в машинное обучение, Python, визуализация данных и другие. Курсы не объясняют математику, стоящую за алгоритмами машинного обучения, но научат принципам, необходимым для анализа данных. Это

поможет сэкономить время, которое обычно тратится на пассивное изучение материала.

Быстро погрузит в практику

Вместо того чтобы искать задачи по изученной теории, можно начать работать над проектом и уже в процессе «добирать» необходимые знания.

Так обучение Machine Learning и Data Science проходит увлекательнее и приносит больше пользы.

Поможет решать актуальные проблемы на реальных данных

Kaggle публикует соревнования, которые инициируют компании — они ищут решения актуальных проблем и дают участникам реальные наборы данных. Это дает возможность не только получить опыт в решении задач, но и начать взаимодействовать с компаниями и их запросами.

Шаг 1: получите базовые знания

Выберите язык программирования — например, Python или R — и изучите его основы. Затем перейти к Kaggle Learn, чтобы закрепить знания по выбранному языку программирования, начать погружение в машинное обучение и познакомиться с методами визуализации данных.

Шаг 2: найти интересный проект или набор данных

Для начала можно выбрать несложный конкурс и испытать себя. На этом этапе начинающим дата-сайентистам помогут Kernels («ядра») — онлайн-среда для программирования, которая работает на серверах Kaggle. В ней можно писать Python/R-скрипты и работать в Jupyter Notebooks.

Kernels бесплатны и отлично подходят для тестирования. Можно скопировать или изменить уже существующее «ядро» другого пользователя, а также поделиться своим с сообществом.

Шаг 3: изучить открытые «ядра»

Анализ открытых «ядер» поможет сравнить свой код с кодом других пользователей и понять, какие разделы Machine Learning и Data Science следует изучить тщательнее. Это ускорит погружение в тему и сделает процесс более осознанным.

Шаг 4: опубликовать собственное «ядро»

Создайте собственный Kernel — даже если у вас еще нет уверенности в своих силах. «Ядро» лучше сделать публичным: так можно заработать больше очков на платформе и получить обратную связь.

Шаг 5: изучите новую информацию и снова опубликуйте «ядро»

Следуйте принципу «Learn, leap and repeat» (научись, сделай большой шаг вперед и начни снова): усовершенствуйте свои навыки и опять вернитесь к шагу 4.

Шаг 6: совершенствуйте анализ, изучая Kernels

На этой стадии у начинающего дата-сайентиста обычно уже есть свои методы работы с данными и прогнозирующие модели — поэтому еще раз изучите «ядра» других пользователей. Можно задать коллегам вопрос, начать дискуссию или просто дополнить свои наработки.

АПРОБАЦИЯ KAGGLE

Автор принял участие в соревновании ТИТАНИК

меню kaggle

ПОИСК

Титаник: машинное обучение на примере катастроф

Отправить прогноз ...

Обзор Данные Код Модели Обсуждение **Таблица лидеров** Правила Команда

Рейтинг	Имя участника	Скорость	Время
11077	Фунаки Сарина	0,76794	4 10 м
11078	Кодзи Фудзивара	0,76794	1 41 м
11079	НаталиМазур	0,76794	1 18 м
11080	ДаниилNxу	0,76555	3 2мес.

Ваша лучшая запись!
 Добро пожаловать в таблицу лидеров! Ваша оценка отражает точность вашего представления. Например, результат 0,7 в этом соревновании означает, что вы правильно предсказали выживание «Титаника» для 70% людей.

Что дальше? У вас есть несколько вариантов:

- Изучите навыки, которые помогут улучшить ваш результат, на нашем курсе Дана Беккера «Введение в машинное обучение».
- Посетите дискуссионный форум, чтобы найти множество руководств и идей от других конкурентов.
- Найдите новое задание, приняв участие в одном из наших открытых конкурсов или выполнив поиск в наших общедоступных наборах данных.

Активация Windows. Чтобы активировать Windows, перейдите в раздел "Параметры".

Работа в Kaggle, знакомство, изучение рубрики соревнования на примере Титаника
 проба и проверка рабочего кода
<https://www.kaggle.com/code/natalimazur/titanic?scriptVersionId=159640905>
 дубль в GitHub
https://github.com/MazurovaNN/artificial-intelligence/blob/main/dz_2_Titanic
 отправка и участие в конкурсе
<https://www.kaggle.com/code/natalimazur/notebook9d6b16ee78>

1.3 Дата-сайентист

Опытный дата-сайентист

Если у вас уже есть опыт, то вы сможете участвовать в соревнованиях по исследованию данных — в одиночку или командой решать задачи по машинному обучению.

Однако опытные специалисты соревнуются не только из интереса: призеры соревнований получают денежные призы, становятся известными в сообществе, их приглашают на престижные позиции.

Например, в конце 2020 года стартовал конкурс «Взлом почки».

Смысл Задачи для специалистов — находить ткани определенного типа на изображениях.

Это часть проекта Human BioMolecular Atlas Program (HuBMAP) по изучению работы человеческого организма на клеточном уровне.

Стимулом и бонусом для участников: Призовой фонд — \$60 000.

МИССИЯ DATA SCIENTISTA - решать по истине амбициозные задачи!

Они учатся

1. создавать искусственный интеллект
2. обучать нейронные сети
3. менять мир к лучшему
4. и конечно же отлично зарабатывать, занимаясь любимым делом!

ВКЛАД GeekBrains в направлении подготовки студентов по данному направлению неоспоримый!

GeekBrains не имеет себе равных.

Специалисты, наставники, преподаватели, кураторы - профессионалы с большой буквы в своем деле!

ТЕПЕРЬ К ДЕЛУ!

ГЛАВА 2 Данные

2.1 Обзор данных

Обзор доступных данных

В выборке 4913 наблюдений и 12 характеристик для каждого из объектов (штат продажи, год выпуска, технические характеристики автомобиля, цена транспортного средства и т.д.).

Пустые значения указывают на то, что о соответствующей характеристике нет информации.

Выборку разбиваю на две части:

1-я для обучения и

2-я для тестирования модели.

Итак, данные содержат два типа переменных:

Целевая: `price`

Остальные переменные: 11 переменных, которые могут использоваться для прогноза целевой переменной.

2.2 План-алгоритм анализа данных

План-алгоритм анализа данных (data mining):

1. Загрузить библиотеки
2. Загрузить данные
3. Загрузить данные для обучения
4. Обработать данные перед обучением модели
5. Обучить модель на обучающей выборке
6. Загрузить и предобработать данные для тестирования
7. Провалидировать модель на тестовой выборке

РЕАЛИЗАЦИЯ ПЛАНА

8. Загрузить данные для обучения

2.2.1 Загрузка библиотек

Загружаем библиотеки

Библиотека **warnings** отвечает за то, какие предупреждения (warnings) о работе будут выводиться пользователю.

FutureWarning - предупреждения о том, как изменится работа библиотек в будущих версиях.

Поэтому такие предупреждения следует игнорировать.

Чтобы включить режим игнорирования отбираем все предупреждения из категории FutureWarning и выбираем для них действия 'ignore'.

Данное действие делается вызовом функции simplefilter с задания двух атрибутов (ИЗ ПЛАНА-АЛГОРИТМА):

5. действия action и
6. категории предупреждений category.

In [1]:

```
import warnings warnings.simplefilter(action='ignore', category=FutureWarning)
```

Для корректной работы с данными в python требуется загрузить специальную библиотеку pandas, программную библиотеку на языке python для обработки и анализа данных.

In [2]:

```
import pandas as pd # загружаем библиотеку и для простоты обращения в коде называем её сокращенно pd
```

Для корректной работы с графиками в python требуется загрузить специальную библиотеку matplotlib, программную библиотеку на языке python для визуализации данных двумерной и трехмерной графикой.

Графики используются для облегчения интерпретации полученных результатов, а также в качестве иллюстраций в презентациях и отчетах.

Основные методы для построения:

7. plot() - графики
8. semilogy() - график логарифмический
9. hist() - гистограммы

In [3]:

```
import matplotlib.pyplot as plt # загружаем библиотеку и для простоты обращения в коде называем её сокращенно plt # позволяет отображать графики прямо в ноутбуке %matplotlib inline
```

2.2.2 Загрузка данных

Загрузим данные

Для решения задачи мы будем использовать данные.

Они состоят из двух частей:

1. часть для обучения и
2. часть для тестирования модели.

Загружаем данные с помощью команды `!wget`.

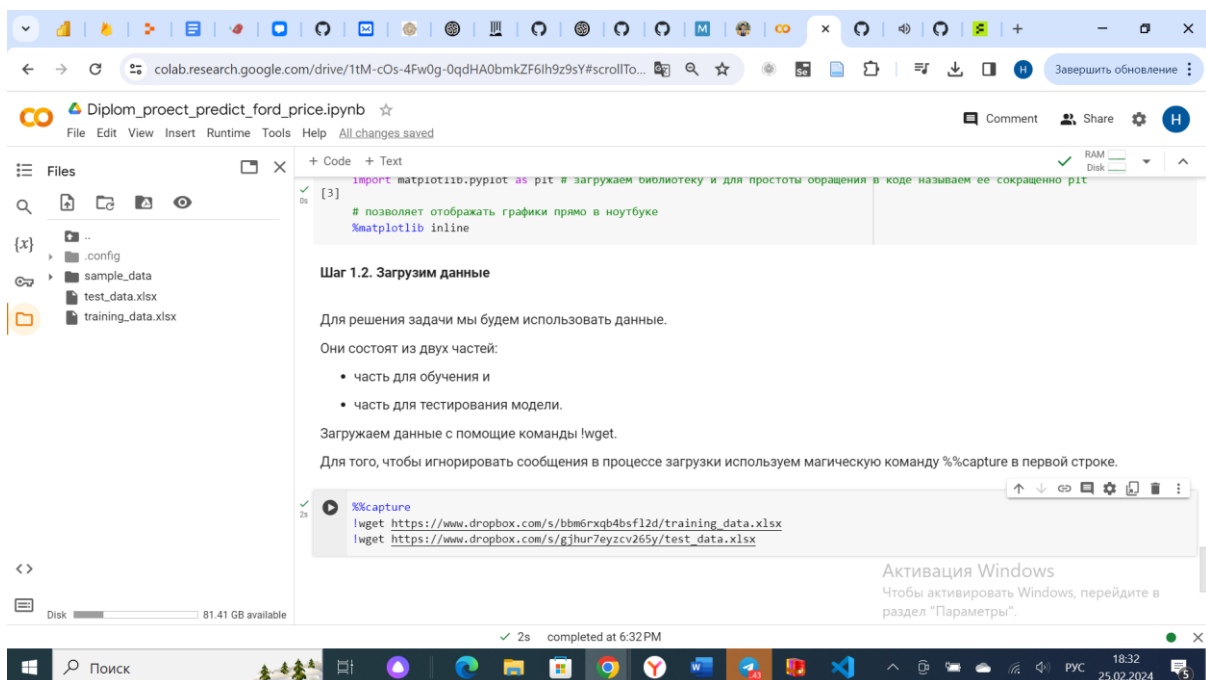
Для того, чтобы игнорировать сообщения в процессе загрузки используем магическую команду `%%capture` в первой строке.

In [4]:

```
%%capture !wget https://www.dropbox.com/s/bbm6rxqb4bsf12d/training_data.xlsx !wget
https://www.dropbox.com/s/gjhur7eyzcv265y/test_data.xlsx
```

Исходные данные скачались

Наблюдаем их в загруженных файлах:



Так как данные в формате `xlsx` (Excel), мы будем использовать специальную функцию из библиотеки `pandas` для загрузки таких данных `read_excel`.

В функции передаем один атрибут: название таблицы с данными.

In [5]:

```
training_data = pd.read_excel('training_data.xlsx', usecols=lambda x: 'Unnamed' not in x) # загружаем таблицу в переменную training_data
```

Команда `**head` выводит содержимое данных

In [6]:

```
training_data.head()
```

Out[6]:

	price	year	condition	cylinders	odometer	title_status	transmission	drive	size	lat	long	weather
0	43900	2016	4	6	43500	clean	automatic	4wd	full-size	36.471500	-82.483400	59.0
1	15490	2009	2	8	98131	clean	automatic	4wd	full-size	40.468826	-74.281734	52.0
2	2495	2002	2	8	201803	clean	automatic	4wd	full-size	42.477134	-82.949564	45.0
3	1300	2000	1	8	170305	rebuilt	automatic	4wd	full-size	40.764373	-82.349503	49.0
4	13865	2001	3	8	166062	clean	automatic	4wd	NaN	49.210949	-123.114720	NaN

Ниже в таблице представлено описание каждого из 12 полей.

Название поля	Описание	Название поля	Описание
price	Цена	transmission	Коробка передач
year	Год производства	drive	Привод
condition	Состояние	size	Полноразмер или нет
cylinders	Количество цилиндров	lat	Широта
odometer	Пробег	long	Долгота
title_status	Легальный статус авто (все документы в наличии)	weather	Среднегодовая температура в городе продажи

2.2.3 Загрузка данных для обучения

Посмотрим на размеры загруженной таблицы, у которой мы видели только первые 5 строк.

Для этого вызываем поле `shape` у нашей переменной `training_data`.

Поле вызывается также как метод, но в конце скобки не ставятся, так как для поля не предусмотрена передача аргументов.

```
training_data.shape
```

In [7]:
Out[7]:
(4913, 12)

Первое и второе числа в скобках означают следующее:

Итак, таблица содержит 4913 строк (объектов) и 12 столбцов (признаков), включая выходной (целевой) признак.

Таблицу проверили, теперь можно приступать к обработке данных.

Последнее действие очень важное и означает до 70 - 90 процентов всей работы специалиста при обучении ЛЮБОЙ МОДЕЛИ!

2.2.4 Обработка данных перед обучением модели

Шаг 1 Проверяем данные на наличие пропусков и типов переменных

Начнем с проверки общей информации о данных. Для того чтобы это сделать, нужно обратиться вызвать у переменной `training_data` метод `info()`.

Напомним, что в конце необходимо поставить скобочки.

In [8]:

```
training_data.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4913 entries, 0 to 4912
Data columns (total 12 columns):
#   Column          Non-Null Count  Dtype
---  -
0   price           4913 non-null   int64
1   year            4913 non-null   int64
2   condition       4913 non-null   int64
3   cylinders       4913 non-null   int64
4   odometer        4913 non-null   int64
5   title_status    4913 non-null   object
6   transmission    4913 non-null   object
7   drive          4651 non-null   object
8   size            3825 non-null   object
9   lat             4913 non-null   float64
10  long            4913 non-null   float64
11  weather         4801 non-null   float64
dtypes: float64(3), int64(5), object(4)
memory usage: 460.7+ KB
```

Анализируем результаты выполнения команды (ПРОДОЛЖЕНИЕ ПЛАНА-АЛГОРИТМА):

10. 4913 строк (entries)

11. 12 столбцов (Data columns)

В данных присутствует три типа dtypes:

12. int64 - целое число (5 столбцов)

13. `float64` - дробное число (3 столбца)
14. `object` - не число, обычно текст (4 столбца)

В нашем случае признаки с типом `object` имеют текстовые значения.

Цифры в каждой строчке обозначают количество заполненных (*non-null*) значений. Видно, что в данных содержатся пропуски, так как эти цифры не в каждой строчке совпадают с полным числом строк (4913).

Шаг 2 Удаляем пропуски

Как мы уже видели выше, в наших данных есть пропуски (значения NaN). Для удобства работы выкинем такие данные из нашего датасета, применив метод **`dropna()`** к *training_data*:

```
training_data = training_data.dropna()
```

Посмотрим на то, как изменились размеры таблички:

```
training_data.shape
```

In [9]:

In [10]:

Out[10]:

```
(3659, 12)
```

Также, после выкидывания строк с пропущенными значениями осталось 3659 строка из 4913. Нам повезло: наш набор данных был заполнен на 75%.

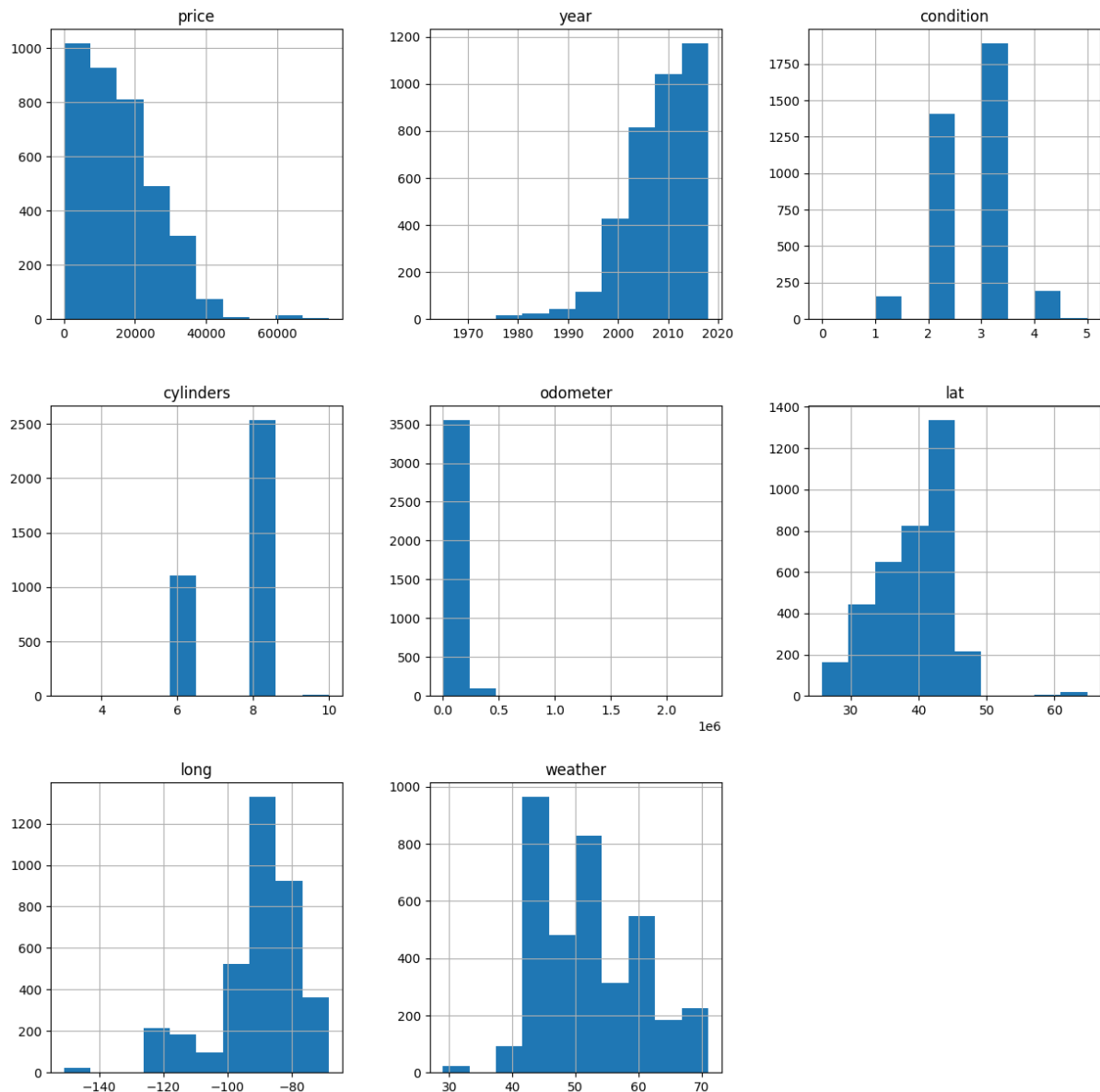
Для числовых признаков можно построить гистограмму. Гистограмма - это способ графического представления табличных данных, благодаря которому можно увидеть распределение значений признака.

Для построения гистограммы необходимо вызвать метод **`hist()`** у объекта *training_data*. Желательно указать аргумент *figsize*, который устанавливает ожидаемый размер изображения. В нашем случае это (15,15).

Заметим, что название переменной, по которой строится гистограмма, указано в названии графика.

```
training_data.hist(figsize=(15, 15));
```

In [11]:



Например, рассмотрим признак `cylinders`. Из гистограммы видно, что у нас очень мало машин с четырьмя и десятью цилиндрами. Машин с шестью цилиндрами - 1500, и около 3500 машин с восемью цилиндрами.

Шаг 3 Кодировем категориальные признаки


Категориальный признак - это такой признак, который может принимать одно значение из ограниченного числа возможных.

В наших данных есть два числовых категориальных признаков: `condition`, `cylinders`

И несколько текстовых категориальных признаков: `title_status`, `transmission`, `drive`, `size`.

Машине сложно обрабатывать текстовые признаки, поэтому нам необходимо закодировать их, то есть преобразовать в числовые. Пример кодирования для категориального признака `Category`, принимающего одно из четырех возможных значений ['Human', 'Penguin', 'Octopus', 'Alien'].

Sample	Category	Numerical				
1	Human	1				
2	Human	1				
3	Penguin	2				
4	Octopus	3				
5	Alien	4				
6	Octopus	3				
7	Alien	4				



Sample	is_Human	is_Penguin	is_Octopus	is_Alien
1	1	0	0	0
2	1	0	0	0
3	0	1	0	0
4	0	0	1	0
5	0	0	0	1
6	0	0	1	0
7	0	0	0	1

Предположим, что некоторый признак может принимать n разных значений. Применив One Hot Encoding, мы создадим n признаков, все из которых для каждой строчки равны нулю за исключением одного. На позицию, соответствующую значению категории признака, мы помещаем 1.

Рассмотрим на уже знакомом примере. Пусть имеется категориальный признак `Category`, принимающий одно из четырех возможных значений ['Human', 'Penguin', 'Octopus', 'Alien']. После применения One Hot Encoding мы получим четыре новых признака (по количеству возможных значений) `is_Human`, `is_Penguin`, `is_Octopus`, `is_Alien`. Для той строчки, у которой в исходных данных стояла категория `Human`, в столбце `is_Human` будет стоять 1, в остальных столбцах 0. Аналогично для другого значения.

Посмотрим, как изменится качество модели с кодированием признаков с помощью One Hot Encoding

```
In [12]: training_data = pd.concat([training_data, pd.get_dummies(training_data['title_status'], prefix="title_status"),
pd.get_dummies(training_data['transmission'], prefix="transmission"), pd.get_dummies(training_data['drive'],
prefix="drive"), pd.get_dummies(training_data['size'], prefix="size")], axis=1)
```

```
In [13]: training_data.drop(['title_status', 'transmission', 'drive', 'size'], axis=1, inplace=True)
```

```
In [14]:
```

```
training_data.head()
```

```
Out[14]:
```

	price	year	condition	cylinders	displacement	horsepower	weight	title_status_clean	title_status_links	transmission_automatic	transmission_manual	transmission_other	drive_wheels_4wd	drive_wheels_fwd	drive_wheels_rwd	size_compact	size_fullsize	size_subcompact	
0	43900	2001	4	6	4.5	82	5000	1	0	1	0	0	1	0	0	0	1	0	0
1	15490	2000	2	8	1.8	74	2800	1	0	1	0	0	1	0	0	0	1	0	0
2	24950	2000	2	8	1.8	74	2800	1	0	1	0	0	1	0	0	0	1	0	0
3	33000	2000	1	8	3.0	82	4000	0	0	1	0	0	1	0	0	0	1	0	0

```

      4
      1 5. -
5 6 2      6 5 12 5
  9 0      7 1 2. 0
  9 0 3 8 6 8 57 .      1 0 .      1 0 0 1 0 0 0 1 0 0
  5 3      6 0 87 0      .
      2 3 52
      1
5 rows x 23 columns

```

Шаг 4 Работаем с целевой переменной

Какая переменная целевая?

В данном случае по условию задачи мы должны прогнозировать стоимость автомобиля, поэтому целевая переменная - это price.

Нам нужно выделить в отдельную переменную *training_values* столбец из нашей таблицы, который соответствует определенной выше целевой переменной.

In [15]:

```
training_values = training_data['price']
```

Отделим входные переменные от выходной (целевой), чтобы можно было построить модель предсказания целевой переменной по входным.

Для это нужно у переменной *training_data* вызвать метод *drop()*. *axis=1* - означает, что мы удаляем столбец, а в случае *axis=0* - означает, что мы удаляем строку

In [16]:

```
training_points = training_data.drop('price', axis=1)
```

Можно посмотреть результаты этих действий, вызвав метод *head()* и поле *shape*, которыми мы пользовались ранее, но сейчас нужно вызывать их от новой переменной *training_points*.

In [17]:

```
training_points.head()
```

Out[17]:

```

   c   c   o   w   title   title   title_   transm   trans   trans   dr   d   d   siz   si   si   si
   o   y   d   e   _sta   _st   statu   ission_   missio   missi   iv   ri   ri   e_   z   z   z
   n   li   o   a   tus_   atu   s_   ission_   n_   on_o   e   v   v   co   e   e   e
   d   n   m   t   ng   t   clea   s_li   autom   n_ma   ther   _   _   _   m   _   _   _
   r   it   d   e   h   n   en   sing   atic   nual   _   _   _   _   f   m   s

```


	i e t			e			w w w			pa	u	i	u
	o r e			r			d	d	d	ct	ll	d	b
	n s r												
													- - -
													si si c
													z z o
													e e m
													p a c t
				3									
				6.	-								
				4	4	82	5						
				3	7	.4	9						
0	2	4	6	5	1	83	.	1	0	0	.	1	0
	0			0	5	40	0				.		
	1			0	0	0							
	6			0	0	0							
				4									
				0.	-								
				9	4	74	5						
				8	6	.2	2						
1	2	8		1	8	81	.	1	0	0	.	1	0
	0			3	8	73	0				.		
	9			1	2	4							
				6									
				4									
				2	2.	-							
				0	4	82	4						
				1	7	.9	5						
2	2	8		8	7	49	.	1	0	0	.	1	0
	0			0	1	56	0				.		
	2			3	3	4							
				4									
				4									
				1	0.	-							
				7	7	82	4						
				0	6	.3	9						
3	1	8		3	4	49	.	0	0	0	.	1	0
	0			0	3	50	0				.		
	0			5	7	3							
				3									
				4									
				1	5.	-							
				6	5	12	5						
				7	1	2.	0						
5	3	8		6	8	57	.	1	0	0	.	1	0
	0			6	0	87	0				.		
	3			2	3	52							
				1									

5 rows × 22 columns

```
training_points.shape
```

In [18]:

(3659, 22)

Out[18]:

```
training_points['size_sub-compact'].value_counts(dropna=False)
```

In [19]:

0 3658

1 1

Name: size_sub-compact, dtype: int64

Out[19]:

```
training_points = training_points.drop('size_sub-compact', axis=1)
```

In [20]:

```
training_points.head()
```

In [21]:

Out[21]:

	year	country	document	language	weight	title_status_clean	title_statistics_lien	title_statistics_missing	title_statistics_salvage	transmission_automatic	transmission_manual	transmission_other	drive_4wd	drive_4wd	drive_4wd	size_compact	size_fullsize	size_mini
0	2016	4	6	364500	82.483400	1	0	0	0	1	0	0	1	0	0	0	1	0
1	2009	2	8	981264	74.281734	1	0	0	0	1	0	0	1	0	0	0	1	0
2	2002	2	8	208803	82.949564	1	0	0	0	1	0	0	1	0	0	0	1	0

Данные в 5 первых строках такие же, как были ранее.

2.2.5 Обучение модели на обучающей выборке



Шаг 1 Выбираем метод, который будем использовать

Проще всего начать с простых методов. Мы воспользуемся двумя методами для построения моделей и сравним их между собой (ДАЛЕЕ ПО ПЛАНУ-АЛГОРИТМУ):

15. Линейная регрессия *linear regression*
16. Лес решающих деревьев *random forest*

На выбор метода для построения модели влияет набор признаков, размер выборки, интуиция про то, какая связь между входными переменными и целевой. Но часто решение принимается исходя из того, какая модель сработала лучше.

Для корректной работы с методами построения моделей в python требуется загрузить специальную библиотеку **sklearn**, программную библиотеку на языке python для машинного обучения и анализа данных.

Мы импортируем два модуля из этой библиотеки (ПО ПЛАНУ-АЛГОРИТМУ):

17. *linear_model* - тут находятся все линейные модели
18. *ensemble* - тут находятся модели на основе ансамблей

```
from sklearn import linear_model, ensemble
```

Прежде чем начать делать ремонт, нужно подготовить инструменты для работы. Аналогично в нашем случае, прежде чем обучать модели, нужно создать их прототипы.

Чтобы создать модель линейной регрессии, пишем имя модуля 'linear_model', затем точку, затем название модели.

In [23]:

```
linear_regression_model = linear_model.LinearRegression() # создаем модель
```

Чтобы создать модель случайного леса, пишем имя модуля ensemble, затем точку, затем название модели.

Обратите внимание, что для воспроизводимости результата на разных компьютерах необходимо для всех зафиксировать один параметр random_state. Например, можно установить для него значение 123.

In [24]:

```
random_forest_model = ensemble.RandomForestRegressor(random_state=123)
```

У модели на основе случайного леса больше параметров. Рассмотрим наиболее важные (В СООТВЕТСТВИИ ПЛАНА-АЛГОРИТМА):

19. параметр *n_estimators* определяет, сколько деревьев в лесу,
20. в параметре *max_depth* устанавливается, какая максимальная глубина у дерева,
21. в параметре *min_samples_leaf* задается, какое максимальное число объектов может попасть в лист дерева.

Так как у модели на основе случайного решающего леса больше параметров, такая модель обычно обучается медленнее. Кроме этого, на время обучения влияют значения параметров модели. Например, чем больше деревьев в лесе - тем дольше модель будет учиться.

Шаг 2 Обучить модель

Теперь, когда мы создали прототипы обеих моделей, можем их обучить с помощью обучающей выборки.

Для этого вызываем метод **fit()** у каждой модели и передаем ему на вход два аргумента: таблицу входных признаков и столбец значений целевой переменной - (training_points, training_values).

In [25]:

```
linear_regression_model.fit(training_points, training_values)
```

Out[25]:

LinearRegression()**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.**

On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

Делаем тоже самое для модели решающего леса.

In [27]:

```
random_forest_model.fit(training_points, training_values)
```

Out[27]:

RandomForestRegressor(random_state=123)**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.**

On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

(В ПРОДОЛЖЕНИЕ ПЛАНА-АЛГОРИТМА):

22. Для двух разных моделей в sklearn методы для обучения модели не отличаются.
23. Мы получили две обученные модели.
24. Теперь необходимо провалидировать модели на новых тестовых данных.

2.2.6 Загрузка и предобработка данных для тестирования

Шаг 1 Загрузим и проанализируем тестовые данные.

Так как данные в формате `xlsx` (Excel), мы будем использовать специальную функцию из библиотеки `pandas` для загрузки таких данных `read_excel`.

В функции передаем один атрибут: название файла, в котором находится таблица с данными.

In [28]:

```
test_data = pd.read_excel('test_data.xlsx', usecols=lambda x: 'Unnamed' not in x)
```

Что важно посмотреть, после того, как мы загрузили данные? (ПО ПЛАНУ-АЛГОРИТМУ):

25. проверить, что данные действительно загрузились

26. посмотреть на данные, чтобы удостовериться, что они правильные: колонки имеют те же названия, что и в таблице и т.д.

In [29]:

```
test_data.head()
```

Out[29]:

	price	year	condition	cylinders	odometer	title_status	transmission	drive	size	lat	long	weather
0	5990	2004	4	8	210445	clean	automatic	4wd	full-size	38.731803	90.073678	48.0
1	18995	2001	2	6	142282	clean	automatic	4wd	full-size	42.504823	92.405569	47.0
2	23500	2002	3	6	100710	clean	automatic	4wd	full-size	36.060541	95.795447	57.0

	1											
	2											
	2											
3	24	0	4	6	85572	clean	automati	4	NaN	30.457	-	
	98	1					c	w		703	84.347	65.0
	8	2					d	d			448	
		2										
	20	0										
4	99	1	3	6	15755	clean	automati	Na	full-	42.701	-	
	5	4			6		c	N	size	772	89.008	45.0
											858	

Посмотрим на размеры загруженной таблицы, так как мы видели только 5 строк

Для этого вызываем поле `shape` у нашей переменной `test_data`. Поле вызывается также как метод, но в конце скобки не ставятся (!), так как для поля не предусмотрена передача аргументов.

In [30]:

```
test_data.shape
```

Out[30]:

```
(2104, 12)
```

Что означает первое и второе число? Таблица содержит 2104 строк (объектов) и 12 столбцов (признаков), включая выходной (целевой) признак. Также как в учебных данных до обучения.

Таблицу проверили, теперь можно приступать к обработке данных. Действуем аналогично тому, как делали с данными для обучения.

Проверим, есть ли в данных пропуски. Для того чтобы это сделать, нужно обратиться вызвать у переменной `test_data` метод **`info()`**.

In [31]:

```
test_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2104 entries, 0 to 2103
Data columns (total 12 columns):
#   Column          Non-Null Count  Dtype
---  -
0   price           2104 non-null  int64
1   year            2104 non-null  int64
2   condition       2104 non-null  int64
3   cylinders       2104 non-null  int64
4   odometer        2104 non-null  int64
5   title_status    2104 non-null  object
6   transmission    2104 non-null  object
7   drive           1975 non-null  object
8   size            1628 non-null  object
9   lat             2104 non-null  float64
10  long            2104 non-null  float64
11  weather         2036 non-null  float64
dtypes: float64(3), int64(5), object(4)
```



```
memory usage: 197.4+ KB
```

Цифры в каждой строчке обозначают количество заполненных (*non-null*) значений.

Видно, что в данных содержатся пропуски, так как эти цифры не в каждой строчке совпадают с полным числом строк (2106).

Нам необходимо удалить пропуски. Для этого применяем метод `dropna()` к `test_data`:

```
test_data = test_data.dropna()
```

Кодируем нечисловые признаки таким же образом

In [33]:

```
test_data = pd.concat([test_data, pd.get_dummies(test_data['title_status'], prefix="title_status"),
pd.get_dummies(test_data['transmission'], prefix="transmission"), pd.get_dummies(test_data['drive'], prefix="drive"),
pd.get_dummies(test_data['size'], prefix="size")], axis=1)
```

```
test_data.drop(['title_status', 'transmission', 'drive', 'size'], axis=1, inplace=True)
```

test_data.head()

In [35]:

Out[35]:

	price	condition	color	label	weight	title_status_clean	title_status_dirty	title_status_salvage	transmission_automatic	transmission_manual	transmission_other	drive_4wd	drive_2wd	drive_3wd	size_compact	size_fullsize	size_mini
0	5990	2	4	8	2104	37185	03103	0	1	0	0	1	0	0	0	1	0
1	189	2	2	6	422	425	092	0	1	0	0	1	0	0	0	1	0

[illegible]

Шаг 2 Отделяем целевую переменную

Нам нужно выделить в отдельную переменную *test_values* столбец из нашей таблицы, который соответствует определенной выше целевой переменной.

Для этого мы у таблицы `test_data` в квадратных скобках указываем имя нужного столбца.

В нашем случае это имя записано в переменной *target_variable_name*.

```
test_values = test_data['price']
```

Отделим входные переменные от выходной (целевой), чтобы можно было построить модель предсказания целевой переменной по входным.

Для это нужно у переменной `test_data` вызвать метод `drop()`.

Результат мы записываем в новую переменную `test_points`.

После выполнения запроса `test_points` будет содержать исходную таблицу без целевого столбца.

Обратите внимание, что в данном случае мы передаем два аргумента:

- `target_variable_name` - название столбца цены, который мы ранее записали в эту переменную и теперь хотим удалить из `training_data`
- `axis=1` - означает, что мы удаляем столбец, а в случае `axis=0` - означает, что мы удаляем строку

In [37]:

```
test_points = test_data.drop('price', axis=1)
```

И проверяем результат записанный в `test_points`:

In [38]:

```
test_points.head()
```

Out[38]:

	c	c	o		w	e	title	titl	title_	title_	trans	trans	trans	v	v	v	siz	e_	-	-	s	i	si
	y	n	l	i	o	a	_sta	e_s	statu	statu	missio	missi	missi	e	e	e	co	m	u	m	i	d	e
	a	it	d	e	t	lo	tus_	tat	s_mi	s_sal	n_aut	on_m	on_o	4	f	r	pa	ct	-	-	s	i	z
	r	i	e	r	e	n	clea	us_	ssing	vage	omatic	anual	ther	w	w	w							
	n	s	r		r		n	lien						d	d	d							
0	2	4	8	0	4	8	0	7	0	4	3	7	0	5	1	3	2	3	-	1	8	9	4
	0	0	0	4	8	0	0	8	1	0	0	0	0	1	0	0	0	1	0	0	0	1	0

```

      8 6
      0 7
      3 8
      4 -
      2 9
      1
      4 . 2. 4
      0 2 6 2 5 4 7
      1 1 2 2 0 0 0 . 1 0 0 1 0 0 0 1 0
      5 8 8 5 0
      2 8 8 5
      2 2 6
      3 9
      3 -
      6 9
      1
      0 . 5. 5
      0 0 7 7
      2 3 6 7 6 9 7
      1 1 0 0 . 1 0 0 1 0 0 0 1 0
      2 1 7 0 5 .
      2 1 5 4 0
      0 4 4
      1 7
      4 -
      2 8
      1
      0 . 3. 4
      8 0 8 6 0 5
      1 3 5
      3 5 2 4 .
      3 0 4 5 0
      0 0 0
      0 0
      3 -
      2 9
      6 . 7. 6
      9 8 1 7
      0 3 6 7
      0 2 9 .
      0 2 0 0
      9 1
      6 2

```

5 rows × 21 columns

test_points.shape

(1543, 21)

In [39]:

Out[39]:

Сравним наборы признаков в тестовой и обучающей выборке

`list(test_points)==list(training_points)`

In [40]:

Out[40]:

True

2.2.7. Проанализирование модели на тестовой выборке

Шаг 1 Сравнение моделей.

Теперь мы готовы сравнить качество двух моделей! □

1. Какая модель лучше?

Получим прогнозы целевой переменной на тестовых данных для модели линейной регрессии и модели случайного леса.

Для этого вызовем у каждой модели метод **predict()**, в качестве аргумента передадим *test_points*.

```
In [41]:
test_predictions_linear = linear_regression_model.predict(test_points)

In [42]:
test_predictions_random_forest = random_forest_model.predict(test_points)
```

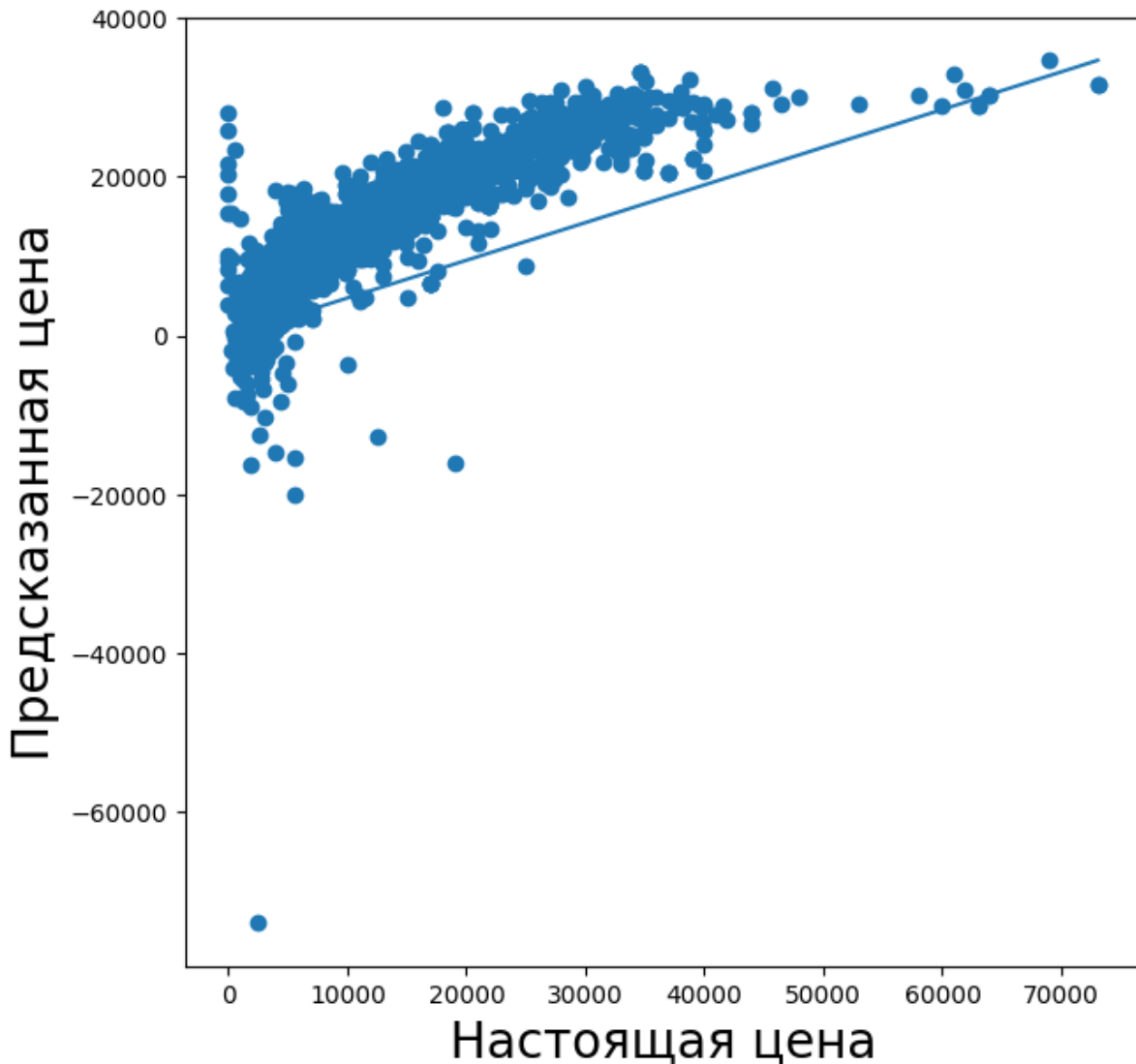
Качество регрессионных моделей оценим двумя способами:

1. Сравним визуально прогнозы с настоящими ценами (тестовые с предсказанием)
2. Сравним метрики качества

Визуализируем прогноз линейной модели и настоящие значения из тестовой выборки

```
In [43]:
plt.figure(figsize=(7, 7)) plt.scatter(test_values, test_predictions_linear) # рисуем точки, соответствующие парам
настоящее значение - прогноз plt.plot([0, max(test_values)], [0, max(test_predictions_linear)]) # рисуем прямую,
```

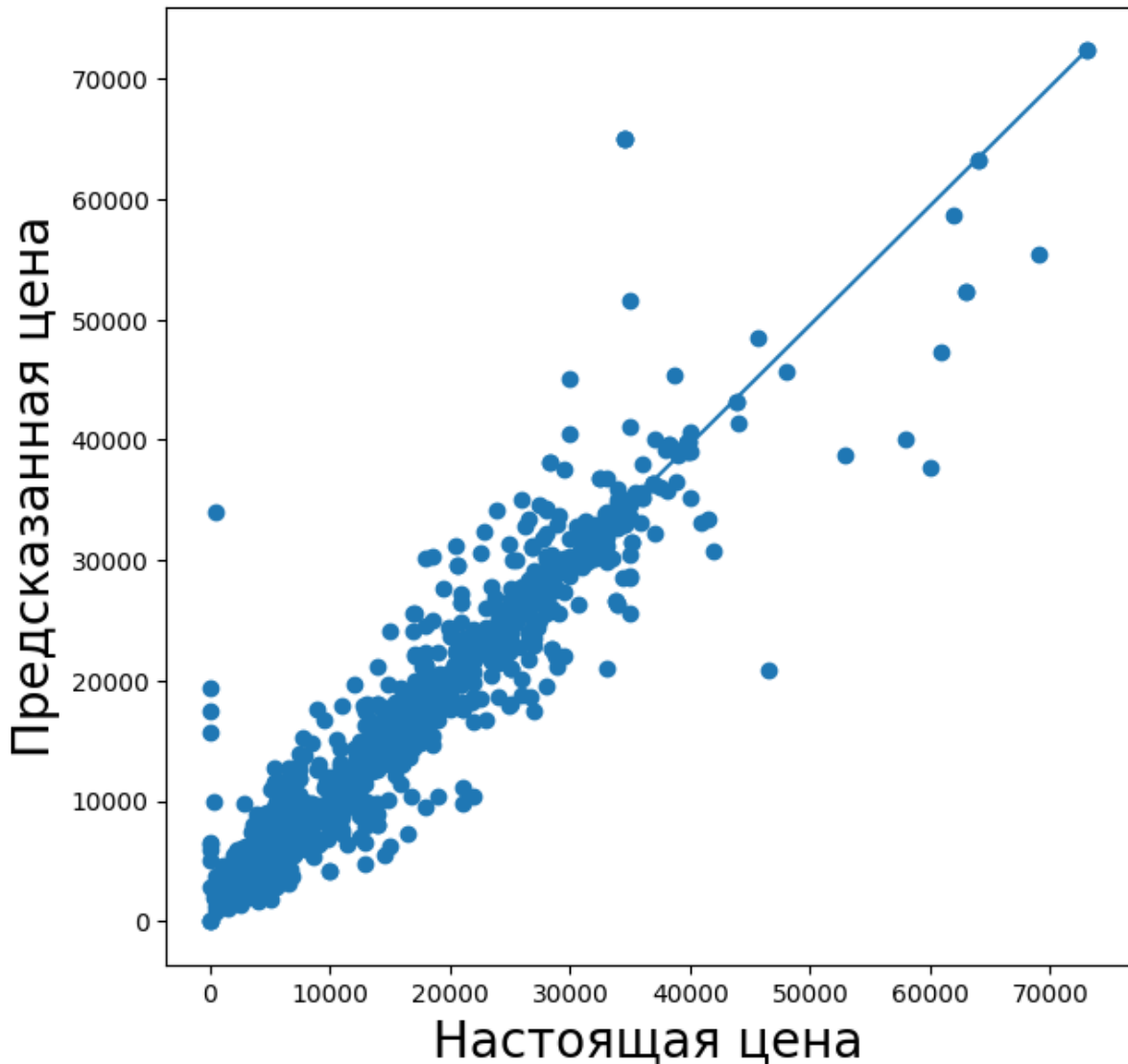
на которой предсказания и настоящие значения совпадают plt.xlabel('Настоящая цена', fontsize=20)
plt.ylabel('Предсказанная цена', fontsize=20);



Визуализируем прогноз модели случайного леса и настоящие значения из тестовой выборки

In [44]:

```
plt.figure(figsize=(7, 7)) plt.scatter(test_values, test_predictions_random_forest) # рисуем точки, соответствующие
парам настоящее значение - прогноз plt.plot([0, max(test_values)], [0, max(test_predictions_random_forest)]) #
рисуем прямую, на которой предсказания и настоящие значения совпадают plt.xlabel('Настоящая цена',
fontsize=20) plt.ylabel('Предсказанная цена', fontsize=20);
```



Кажется, что лучше сработала модель случайного леса, так как точки на втором изображении расположены вдоль диагонали. На первом изображении видно, что для высоких настоящих цен модель линейной регрессии дает существенно заниженный результат.

Проверим, так ли это с помощью **метрик качества регрессионной модели**.

Для корректного подсчета метрик качества модели в python требуется загрузить их из библиотеки **sklearn**.

Мы используем две метрики качества (ПО ПЛАНУ-АЛГОРИТМУ):

27. *mean_absolute_error* - средняя абсолютная ошибка / ~~??~~ ~~??~~ ~~??~~

28. *mean_squared_error* - средняя квадратичная ошибка ($\sqrt{\frac{1}{n} \sum (y_i - \hat{y}_i)^2}$)

29. *RMSE* - корень из *mean_squared_error*

In [45]:

```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

Подсчитаем ошибки для линейной модели.

Для этого вызовем методы *mean_absolute_error()* и *mean_squared_error()*.

На вход им передается столбец настоящих значений *test_values* и столбец значений, предсказанных моделью линейной регрессии *test_predictions_linear*.

In [46]:

```
mean_absolute_error_linear_model = mean_absolute_error(test_values, test_predictions_linear)
mean_squared_error_linear_model = mean_squared_error(test_values, test_predictions_linear)
r2_score_linear_model = r2_score(test_values, test_predictions_linear)
```

Подсчитаем ошибки для модели случайного леса.

Для этого вызовем методы *mean_absolute_error()* и *mean_squared_error()*.

2.2.8 Загрузка данных для обучения

На вход им передается столбец настоящих значений `test_values` и столбец значений, предсказанных моделью линейной регрессии `test_predictions_random_forest`.

In [47]:

```
mean_absolute_error_random_forest_model = mean_absolute_error(test_values, test_predictions_random_forest)
mean_squared_error_random_forest_model = mean_squared_error(test_values, test_predictions_random_forest)
r2_score_random_forest_model = r2_score(test_values, test_predictions_random_forest)
Теперь напечатаем полученные ошибки.
```

In [48]:

```
print("MAE: {0:7.2f}, RMSE: {1:7.2f}, R2: {2:7.2f} for linear model".format(mean_absolute_error(test_values,
test_predictions_linear), mean_squared_error(test_values, test_predictions_linear)**0.5, r2_score_linear_model))
print("MAE: {0:7.2f}, RMSE: {1:7.2f}, R2: {2:7.2f} for random forest model".format(
mean_absolute_error(test_values, test_predictions_random_forest), mean_squared_error(test_values,
test_predictions_random_forest)**0.5, r2_score_random_forest_model))
```

```
MAE: 4263.96, RMSE: 6350.82, R2: 0.66 for linear model
MAE: 1673.39, RMSE: 3316.56, R2: 0.91 for random forest model
```

Модель случайного леса работает лучше и визуально, и потому, что абсолютная и средне квадратичная ошибка меньше для линейной регрессии.

Мы получили значения метрик ошибок наших моделей. Чтобы понять, насколько это нас утсраивает, важно взглянуть на исходный порядок цен на автомобили. Видно, что средняя цена имеет порядок 20 000 долларов, что означает, что полученная ошибка может удовлетворять предъявляемым требованиям к модели регрессии.

ГЛАВА 3 Google Colab

3.1 Google Colab. Настройки доступа

РАБОТА ВЫПОЛНЕНА НА

Google Colab — это бесплатная среда для разработки и выполнения программного кода в облаке.

Она предоставляет возможность писать и запускать код на языке Python, используя только браузер, без установки специальных программ на компьютер.

НАСТРОЙКА ДОСТУПА

Войдите в аккаунт администратора (он не заканчивается на @gmail.com).

Дополнительные сервисы Google.

Нажмите Google Colab.

Нажмите Статус сервиса.

ДОСТУП

После запуска команды Colab предложит ввести код авторизации.

Открыв URL, вы должны предоставить сервису доступ к своему аккаунту.

Тогда он выдаст код, который нужно будет вставить в поле, нажать ВВОД, и Google Colab подключится к хранилищу.

НАСТРОЙКА с GitHub Save a copy in GitHub

Выбираем репозиторий Diplom_2024

https://github.com/MazurovaNN/DIPLOM_2024/tree/main

ЗАКЛЮЧЕНИЕ

Обзор результатов и выводы

В данном дипломном проекте в подготовительной и основной частях:

1. Загружены данные
2. Предобработаны данные, определены наличие пропусков в данных
3. Избавились от пропусков в данных, выделили целевую переменную
4. Выборка разделена на две части: 1-я для обучения и 2-я для тестирования модели.
5. Построены гистограммы для возможных значений признаков
6. Обучена модель на обучающей выборке
7. Провалидирована модель на тестовой выборке
8. Дан анализ полученных результатов.

Предложенный проект продемонстрировал Модель для оценки стоимости подержанного автомобиля, которая может помочь:

1. покупателям не переплатить за желаемое авто, а
2. честным продавцам быстро устанавливать цену, соответствующую их предложениям.

Поставленная задача анализа данных выполнена и продемонстрирована

Цель данной задачи выполнена: прогнозирование цены на подержанные автомобили Ford осуществилось с помощью **построения регрессионных моделей и их анализа.**

В результате проведенного анализа, выявлено, что Модель случайного леса работает лучше и визуально, и потому, что абсолютная и средне квадратичная ошибка меньше, чем для линейной регрессии.

Мы получили значения метрик ошибок наших моделей. Это нас устраивает и важно взглянуть на исходный порядок цен на автомобили. Видно, что

средняя цена имеет порядок 20 000 долларов, что означает, что полученная ошибка может удовлетворять предъявляемым требованиям к модели регрессии и как следствие, приводит к правильному определению цены на подержанные автомобили для рынка продаж подержанных автомобилей.

Корректность подсчета метрик качества регрессионной модели в python обеспечивается загрузка их из библиотеки **sklearn**.

Таким образом, применены метрики, осуществлена визуализация, выбранная модель попадает в точку, она протестирована и автор ее рекомендует к применению при определении цен.

Отмечу, что Набор данных состоит из информации о транспортных средствах, выставленных на продажу на сайте Craigslist.

В заключение следует отметить, что представленная работа содержит элементы искусственного интеллекта.

Искусственный интеллект — это технология, которая позволяет поручить творческую работу компьютерным алгоритмам. С помощью ИИ можно анализировать большие объемы данных, генерировать тексты, изображения, музыку и видео. Бизнес может применять ИИ, чтобы сделать работу более эффективной или сэкономить на оплате труда, но только не на зарплате инженера искусственного интеллекта.

Современный искусственный интеллект — это система, которая способна воспринимать свою среду и принимать меры, чтобы максимизировать шансы на успешное достижение своих целей, а также интерпретировать и анализировать данные таким образом, чтобы они обучались и адаптировались по мере развития.

Искусственный интеллект может обрабатывать большие объемы данных, выявлять закономерности, идентифицировать информацию и давать ответы.

Также с его помощью можно решать проблемы в различных областях, например, обнаруживать мошенничество, проводить медицинскую диагностику и бизнес-анализ, как например представленный в работе анализ цен на подержанные автомобили Ford/

Следует отметить и проблемы ИИ: ИИ не может проявлять эмоции, что может быть проблемой во взаимодействии с людьми в некоторых случаях. Необходимость большого объема данных для обучения. ИИ требует большого количества данных для обучения и настройки своих алгоритмов, что может быть проблемой, если данных недостаточно. А также он обучается, при этом не следует исключать, что возможно искажение реальности.

В абсолютном выражении около 70% потенциала ИИ приходится на шесть ключевых для российской экономики отраслей: транспорт и логистика, банкинг, ретейл, добывающая промышленность, производство потребительских товаров, ИТ.

Отмечу, что Искусственный интеллект — это широкий термин, который охватывает любую систему, способную выполнять задачи, которые обычно требуют человеческого интеллекта, в то время как нейронные сети — это конкретный тип искусственного интеллекта, который используется для обработки сложных наборов данных.

Спасибо за термин «искусственный интеллект» («artificial intelligence»), он был изобретен и впервые озвучен в 1956 году американским ученым Джоном Маккарти.

В ближайшие годы нейросети кардинально изменят области распознавания речи и изображений, поиска по социальным медиа и анализа больших данных, а также повлияют на медицину, безопасность и финансы, сообщил в своем докладе в мае 2016 года Крис Роуэн, технический директор отдела интеллектуальной собственности американской корпорации Cadence Design Systems.

ВКЛАД GeekBrains в направлении подготовки студентов по данному направлению (Инженер ИИ) неоспоримый!

GeekBrains не имеет себе равных!

Специалисты, наставники, преподаватели, кураторы - профессионалы с большой буквы в своем деле!

СПИСОК ЛИТЕРАТУРЫ

1. Акинин, М. В. Нейросетевые системы искусственного интеллекта в задачах обработки изображений / М.В. Акинин, М.Б. Никифоров, А.И. Таганов. - М.: РиС, 2016. - 152 с.
2. Акинин, М.В. Нейросетевые системы искусственного интеллекта в задачах обработки изображений / М.В. Акинин, М.Б. Никифоров, А.И. Таганов. - М.: ГЛТ, 2016. - 152 с.
3. Астахова, И. Системы искусственного интеллекта Практический курс: Учебное пособие / И. Астахова. - М.: Бином. Лаборатория знаний, 2009. - 292 с.
4. Болотова, Л.С. Системы искусственного интеллекта: модели и технологии, основанные на знаниях: Учебник / Л.С. Болотова. - М.: Финансы и статистика, 2012. - 664 с.
5. Гаврилова, А.Н. Системы искусственного интеллекта / А.Н. Гаврилова, А.А. Попов. - М.: КноРус, 2011. - 248 с.
6. Евменов, В.П. Интеллектуальные системы управления: превосходство искусственного интеллекта над естественным интеллектом? / В.П. Евменов. - М.: КД Либроком, 2016. - 304 с.
7. Сидоркина, И.Г. Системы искусственного интеллекта: Учебное пособие / И.Г. Сидоркина. - М.: КноРус, 2011. - 248 с.
8. Черняк, В.З. Системы искусственного интеллекта: модели и технологии, основанные на знаниях: Учебник / В.З. Черняк. - М.: Финансы и статистика, 2012. - 664 с.
9. Сидоркина, И.Г. Системы искусственного интеллекта / И.Г. Сидоркина. - М.: КноРус, 2016. - 167 с.
10. Лекции и семинары обучающей платформы GeekBrains

ОСТАЛОСЬ СДЕЛАТЬ И ПРИКРЕПИТЬ ПРЕЗЕНТАЦИЮ

СПАСИБО GeekBrains за великолепное обучение на протяжении 2-х лет и настрой на дальнейшее постоянное изучение и освоение нового в профессии инженера искусственного интеллекта!