2. G`. Nasriddinov “Iqtisodiy – matematikmodellar va usullar”, Toshkent-2011.
3. Q. Safaeva , F. Mansurov “Iqtisodiyotda matematika” , Toshkent – 2010.
4. Internet, Wikipedia.com.
5. Internet, ziyonet.com.

ПРИМЕНЕНИЕ РЕГРЕССИОННЫХ МОДЕЛЕЙ В ЭКОНОМИЧЕСКОМ АНАЛИЗЕ

[1]А.Х. Хожамкулов, [1]А.А. Мирзаев, [2]Ч.Х. Сайдуллаев
[1]Национальный университет Узбекистана имени Мирзо Улушбека, [2] Ташкентский химико-технологический институт
E-mail: *abdulazizxojamqulov47@gmail.com*, *akmalmirzaev9505@gmail.com*

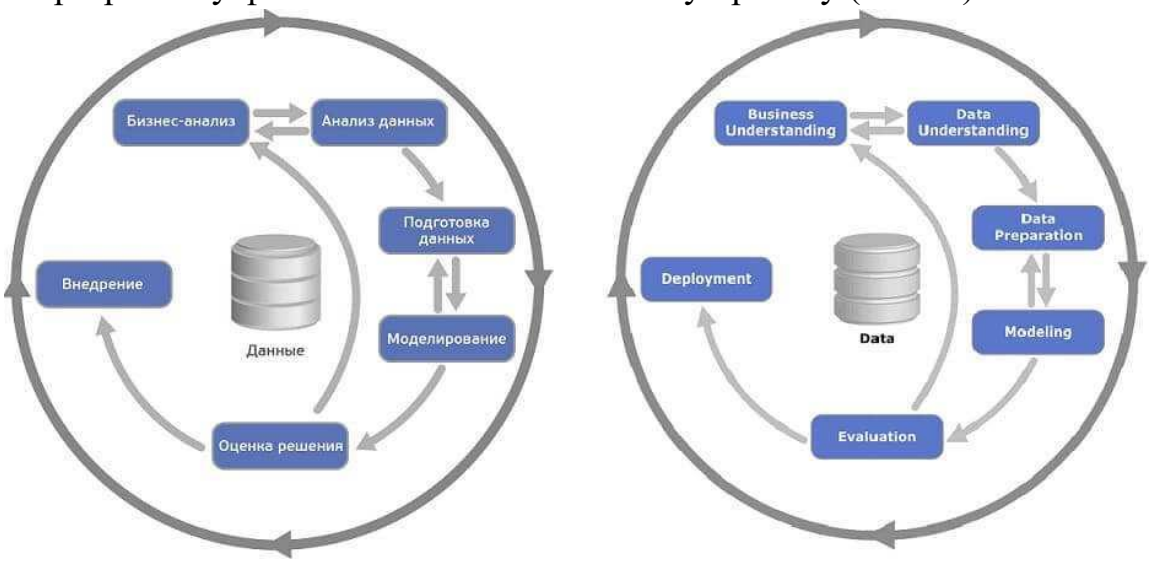
Математический анализ экономических процессов, получение точных результатов, автоматизация процессов с помощью вычислительной техники-одна из актуальных проблем сегодняшнего дня. Подход к проблеме с использованием четкой методологии для решения этих проблем облегчает решение проблемы и повышает точность результата. Ниже мы попытаемся найти решение проблемы с помощью одной из таких методологий. Данная методология является методологией CRISP-DM (Cross-Industry Standard Process for Data Mining) — это наиболее распространенная на практике методология выполнения Data Science проектов, которую принято называть межотраслевым стандартным процессом исследования данных. Он описывает жизненный цикл Data Science проектов в следующих 6 фазах, каждая из которых включает ряд задач:

1. ***Понимание бизнеса (Business Understanding)***, где через оценку текущей ситуации определяются бизнес-цели и требования, а также разрабатывается предварительный план проекта;
2. ***Начальное изучение данных (Data Understanding)***, включая их сбор, описание, исследование (поиск закономерностей, формирование гипотез) и проверку качества;
3. **Подготовка данных (Data Preparation)**, когда из исходного набора данных формируется датасет для работы с моделями машинного обучения (Machine Learning) путем выполнения соответствующих операций Data Preparation – выборка очистка, генерация признаков, интеграция, форматирование.

4. ***Моделирование (Modeling)***, где выбираются алгоритмы, пишутся тесты, строятся и обучаются модели Machine Learning, а также выполняется настройка их параметров и оценка качества;

5. ***Оценка решения (Solution Evaluation)***, когда качество ML-моделей анализируется с точки зрения достижения поставленных бизнес-целей и определяются дальнейшие шаги по улучшению результатов;

6. ***Внедрение (Deployment)***, которое предполагает развертывание полученных ML-моделей в промышленную эксплуатацию (production), включая разработку финальных отчетов по всему проекту (review).



Для решения проблемы CRISP-DM ставит перед собой задачу на последовательный поиск решений следующих 10 вопросов:

От проблемы к подходу

1) Какой именно проблеме вы ищите решение?

2) Каким путём можно будет воспользоваться имеющимися данными для решение задачи?

Работа с данными

3) Какие данные необходимы для поиска решения?

4) Откудого поступает информация (источиники) и как мы собираемся их загружать?

5) Имеет ли отношение собранные нами сведения для решения данной проблемы?

6) Что необходимо выполнить для преобразования данных по вашему требованию?

Поиск решений

7) Как можно визуализировать данные для того чтобы получить полезное сообщение из ссылки?

8) Решает ли созданная нами модель поставленную задачу или необходимо устронить недостатки?

9) Возможно ли использование данной модели в практике?

10) Можете ли вы принять конструктивные идеи, чтобы ответить на вопрос?

Чтобы лучше понять эту методологию, мы ищем решение следующей проблемы с помощью CRISP-DM: необходимо создать модель, прогнозирующую стоимость домов для продажи в Ташкенте. Для построения модели мы будем использовать информацию, полученную с сайта uybor.uz. Для обработки данных мы используем язык программирования Python и среду Google Colaboratory.

Вызываем необходимые библиотеки Python:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Загружаем данные:

```
df=pd.read_csv("https://github.com/abdulazizdatasc/mohirdev_kursi/blob/main/Uybor.uz.csv?raw=true")
df.head(10)
```

1 to 10 of 10 entries Filter

index	location	district	rooms	size	level	max_levels	price
0	город Ташкент, Юнусабадский район, Юнусабад 8-й квартал	Юнусабадский	3	57	4	4	52000
1	город Ташкент, Яккасарайский район, 1-й тупик Шота Руставели	Яккасарайский	2	52	4	5	56000
2	город Ташкент, Чиланзарский район, Чиланзар 2-й квартал	Чиланзарский	2	42	4	4	37000
3	город Ташкент, Чиланзарский район, Чиланзар 9-й квартал	Чиланзарский	3	65	1	4	49500
4	город Ташкент, Чиланзарский район, площадь Актепа	Чиланзарский	3	70	3	5	55000
5	город Ташкент, Чиланзарский район, Чиланзар 6-й квартал	Чиланзарский	1	28	1	4	25500
6	город Ташкент, Чиланзарский район, Чиланзар-16	Чиланзарский	1	30	2	4	21200
7	город Ташкент, Яккасарайский район, Саламатина	Яккасарайский	2	32	5	5	20000
8	город Ташкент, Учтепинский район, Чиланзар-21	Учтепинский	2	51	3	4	26200
9	город Ташкент, Чиланзарский район, Чиланзар-8	Чиланзарский	1	30	1	4	22200

Show 25 per page

Like what you see? Visit the data table notebook to learn more about interactive tables.

В данных столбцах отражается следующая информация:

- location – местоположение квартиры
- district – район местонохождения дома
- rooms – количесвто комнат квартиры
- size – квалратура квартиры (кв.м.)
- level – на каком этаже расположена квартира
- max_levels – этажжность дома
- price – цена квартиры

Модель, которую мы собираемся построить, должна прогнозировать price (цену), используя данные из столбцов location, district, rooms, size,

level, max_level. Работу начнём с получения более расширенной информации данных:

df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7565 entries, 0 to 7564
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   location    7565 non-null   object
 1   district    7565 non-null   object
 2   rooms       7565 non-null   int64 
 3   size        7565 non-null   object
 4   level       7565 non-null   int64 
 5   max_levels  7565 non-null   int64 
 6   price       7565 non-null   object
dtypes: int64(3), object(4)
memory usage: 413.8+ KB
```

Это означает, что информация хранит в себе 7 столбцов, а каждый столбец содержит 7565 строк. Из столбцов столбцы rooms, level, max_level состоят из целых чисел, а столбцы location, district, size, price состоят из текстов. Учитывая, что модель, которую мы строим, работает только исходя из чисел, мы можем преобразовать текстовые столбцы в числовое представление. Для этого мы получаем информацию о значениях, расположенных в каждом текстовом столбце:

df['price'].unique()

```
array(['52000', '56000', '37000', '49500', '55000', '25500', '21200',
       '20000', '26200', '22200', '24200', '30200', '22500', '32500',
       '45000', '47000', '49900', '76000', '65000', '47500', '30000',
       '32000', '44000', '56500', '35000', '23500', '60500', '68000',
       '41500', '52500', '43000', '80000', '42000', '23000', '88784',
       '97000', '28500', '34500', '51000', '48000', '40000', '67000',
       '40500', '54000', '36000', '63000', '63500', '70000', '91000',
       '83000', '19765', '50000', '86000', '31500', '26500', '14500',
       '27000', '59500', '47299', '38000', '29500', '61500', '46500',
       '58400', '26000', '105000', '53500', '12500', '107000', '39414',
       '60000', '49000', '18000', '42500', '45800', '24000', '58000',
       '55500', '31000', '41000', '50500', '13071', '45188', '53000',
       '25600', '35226', '36500', '142000', '85000', '110000', '69000',
       '28900', '34000', '25000', '39000', '125000', '28000', '43500',
       '39999', '24500', '210000', '7500', '95000', 'Договорная', '70500',
       '170000', '30500', '37500', '46000', '38500', '81772', '50533',
       '119000', '93000', '23800', '32900', '29800', '62000', '36999',
       '36200', '45500', '120000', '84000', '77000', '260000', '75000',
       '54900', '54500', '1000', '31300', '33500', '64500', '13075',
```

Из приведенных выше значений видно, что между price (ценой) выпадает текст «договорная». Определяем количество таких строк:

df[df['price']=='Договорная']

	location	district	rooms	size	level	max_levels	price
202	город Ташкент, Яккасарайский район, Баходыра	Яккасарайский	3	119	3	9	Договорная
411	город Ташкент, Яккасарайский район, Баходыра	Яккасарайский	4	160	4	9	Договорная
439	город Ташкент, Мирзо-Улугбекский район, улица ...	Мирзо-Улугбекский	3	105	5	6	Договорная
460	город Ташкент, Чиланзарский район, Чиланзар 1-...	Чиланзарский	3	90	6	8	Договорная
507	город Ташкент, Яшнободский район, 1-й проезд А...	Яшнободский	2	48	4	4	Договорная
...	...	...	...	...	...	...	...
7039	город Ташкент, Яшнободский район, Городок Авиа...	Яшнободский	1	38.70	3	8	Договорная
7196	город Ташкент, Чиланзарский район, Чиланзар-16	Чиланзарский	2	51	3	4	Договорная
7323	город Ташкент, Мирзо-Улугбекский район, жилой ...	Мирзо-Улугбекский	6	208	1	7	Договорная
7403	город Ташкент, Учтепинский район, Чиланзар 14-...	Учтепинский	2	35	2	9	Договорная
7404	город Ташкент, Учтепинский район, Чиланзар 14-...	Учтепинский	2	35	2	9	Договорная

99 rows × 7 columns

Следовательно, количество таких строк равно 99. Теперь мы можем работать с такими строками 2 различными способами: Первый - мы можем пропустить эти строки, а второй – мы можем заполнить эти строки другими данными (средние или медианные значения для этого столбца). Например, для заполнения средними значениями достаточно ввести следующий код:

df_price_num=df[df['price']!='Договорная']

df['price'].replace('Договорная', np.mean(df_price_num['price']), inplace= True)

Этот код принимает все столбцы, кроме столбцов с текстом «договорная» в столбце “price”, и вычисляет среднее значение для них, заменяя каждый текст «договорная» средним значением который высчитывается по следующей формуле

$$\frac{\sum_{i=1}^{n} x_i}{n}.$$

Теперь мы можем преобразовать все значения в столбце price and size в число в виде десятичной дроби:

йdf['price']=df['price'].astype(np.float64)
df['size']=df['size'].astype(np.float64)

Оказывается, что данные также содержат строки со значением infimum среди элементов столбца price.

df[df['price']==np.inf]

	location	district	rooms	size	level	max_levels	price
202	город Ташкент, Яккасарайский район, Баходыра	Яккасарайский	3	119.0	3	9	inf
411	город Ташкент, Яккасарайский район, Баходыра	Яккасарайский	4	160.0	4	9	inf
439	город Ташкент, Мирзо-Улугбекский район, улица ...	Мирзо-Улугбекский	3	105.0	5	6	inf
460	город Ташкент, Чиланзарский район, Чиланзар 1-...	Чиланзарский	3	90.0	6	8	inf
507	город Ташкент, Яшнободский район, 1-й проезд А...	Яшнободский	2	48.0	4	4	inf
...	...	...	...	...	...	...	...
7039	город Ташкент, Яшнободский район, Городок Авиа...	Яшнободский	1	38.7	3	8	inf
7196	город Ташкент, Чиланзарский район, Чиланзар-16	Чиланзарский	2	51.0	3	4	inf
7323	город Ташкент, Мирзо-Улугбекский район, жилой ...	Мирзо-Улугбекский	6	208.0	1	7	inf
7403	город Ташкент, Учтепинский район, Чиланзар 14-...	Учтепинский	2	35.0	2	9	inf
7404	город Ташкент, Учтепинский район, Чиланзар 14-...	Учтепинский	2	35.0	2	9	inf

99 rows × 7 columns

Имеется 2 разных способа работы с такими строками, как указано выше:

Первый - мы можем пропустить эти строки, а второй – мы можем заполнить эти строки другими данными (средние или медианные значения для этого столбца или стандартные значения, которые мы считаем логически правильными). Учитывая, что у нас есть данные в 7 столбцах,

7565 строк и размер данных для построения модели, мы можем заменить значения infimum в строке price другими значениями. Подходим к этому процессу логично. Стоимость дома в большинстве случаев определяется исходя из его площади. Следовательно, мы также заменим значения *infimum* другим значением, соответствующим площади квартиры:

```
df_price1=df[(df['size']>0)&(df['size']<50)].price.replace(np.inf, 40000)
df_price2=df[(df['size']>=50)&(df['size']<100)].price.replace(np.inf, 75000)
df_price3=df[(df['size']>=100)&(df['size']<150)].price.replace(np.inf, 120000)
df_price4=df[(df['size']>=150)&(df['size']<250)].price.replace(np.inf, 140000)
df_price5=df[(df['size']>=250)].price.replace(np.inf, 170000)
df_price=pd.concat([df_price1, df_price2, df_price3, df_price4, df_price5])
df['price']=df_price
```

Если обратить внимание на размер площади домов - столбец “size", то можно заметить, что среди них попадаются необычные значения:

```
df[df['size']<25]
```

	location	district	rooms	size	level	max_levels	price
81	город Ташкент, Учтепинский район, Гулистон	Учтепинский	1	20.0	3	9	14500.0
110	город Ташкент, Яшнободский район, Ташсельмаш	Яшнободский	1	20.0	3	3	12500.0
119	город Ташкент, Мирзо-Улугбекский район, Феруза	Мирзо-Улугбекский	1	22.0	1	2	18000.0
195	город Ташкент, Мирзо-Улугбекский район, Массив...	Мирзо-Улугбекский	1	11.0	3	9	7500.0
298	город Ташкент, Чиланзарский район, Чиланзар 7-...	Чиланзарский	1	24.0	3	4	20000.0
...	...	...	...	...	...	...	...
7407	город Ташкент, Яшнободский район, Тузель 3-й к...	Яшнободский	3	15.0	7	9	50000.0
7474	город Ташкент, Мирабадский район, Фидоийлар	Мирабадский	1	18.0	1	4	10500.0
7505	город Ташкент, Учтепинский район, Чиланзар 25-...	Учтепинский	1	18.0	3	4	16000.0
7522	город Ташкент, Чиланзарский район, проспект Бу...	Чиланзарский	1	20.0	5	5	18500.0
7563	город Ташкент, Мирзо-Улугбекский район, Буюк И...	Мирзо-Улугбекский	1	18.0	1	4	11000.0

88 rows × 7 columns

Обычно в Ташкенте площадь квартир составляет не меньше 25 кв.м. Количество квартир площадь которых меньше 25 кв.м. домов составляет 88 строк. Выбросим их.

```
idx=df[df['size']<25].index
df.drop(index=idx, inplace=True)
idx=df[df['size']>400].index
df.drop(index=idx, inplace=True
```

Существуют строки в которых наблюдается большой дисбаланс цены квартиры по отношению к площади:

```
df[df['price']>500000]
```

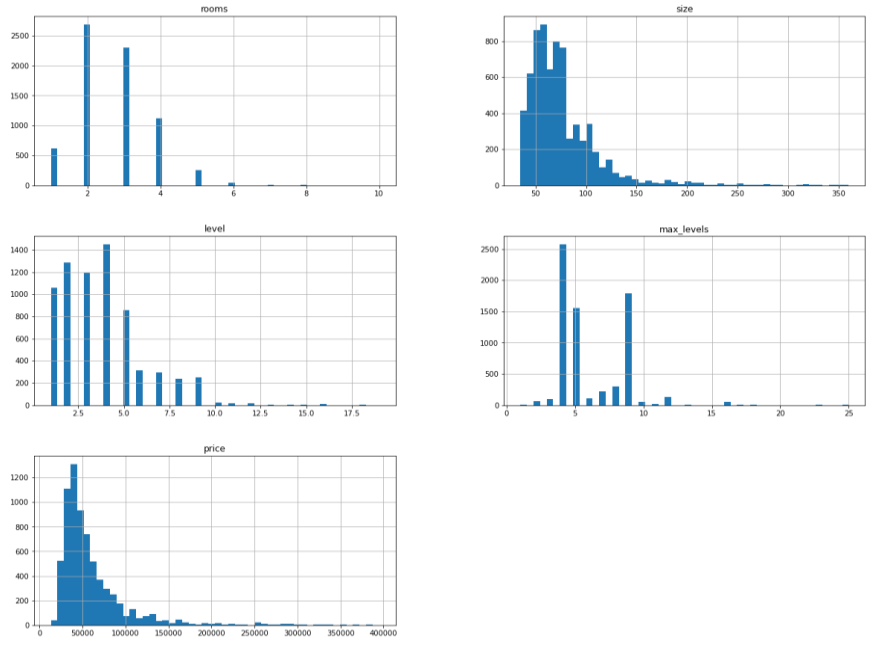
	location	district	rooms	size	level	max_levels	price
330	город Ташкент, Яшнободский район, 1-й переулок...	Яшнободский	2	68.0	5	8	1666000.0
1744	город Ташкент, Олмазорский район, Toshmi	Олмазорский	3	84.0	1	5	3780000.0
3625	город Ташкент, Мирабадский район, Тараса Шевченко	Мирабадский	5	225.0	5	8	530000.0
3656	город Ташкент, Мирзо-Улугбекский район, Дархан	Мирзо-Улугбекский	5	336.0	8	9	504000.0
4935	город Ташкент, Чиланзарский район, Катта козир...	Чиланзарский	1	28.0	3	4	644000.0
5903	город Ташкент, Учтепинский район, Chtepa Fozil...	Учтепинский	4	72.0	2	5	52000000.0
6133	город Ташкент, Юнусабадский район, город Ташке...	Юнусабадский	3	42.0	3	4	1344000.0
6517	город Ташкент, Юнусабадский район, Юнусабад 5к...	Юнусабадский	3	80.0	8	9	4240000.0
6952	город Ташкент, Шайхантахурский район, Алишера ...	Шайхантахурский	3	114.0	5	7	15504000.0
7081	город Ташкент, Чиланзарский район, Тирсакобод	Чиланзарский	2	85.0	2	3	6630000.0
7296	город Ташкент, Яшнободский район, Садыка Азимо...	Яшнободский	4	96.0	2	5	5568000.0
7466	город Ташкент, Чиланзарский район, 1-й проезд ...	Чиланзарский	2	42.0	1	3	1470000.0
7478	город Ташкент, Чиланзарский район, Катта козир...	Чиланзарский	1	28.0	3	4	644000.0

Выбрасываем эксцентрические значения:

```
idx=df[df['price']>500000].index
df.drop(index=idx, inplace=True)
idx=df[df['price']<5000].index
df.drop(index=idx, inplace=True)
```

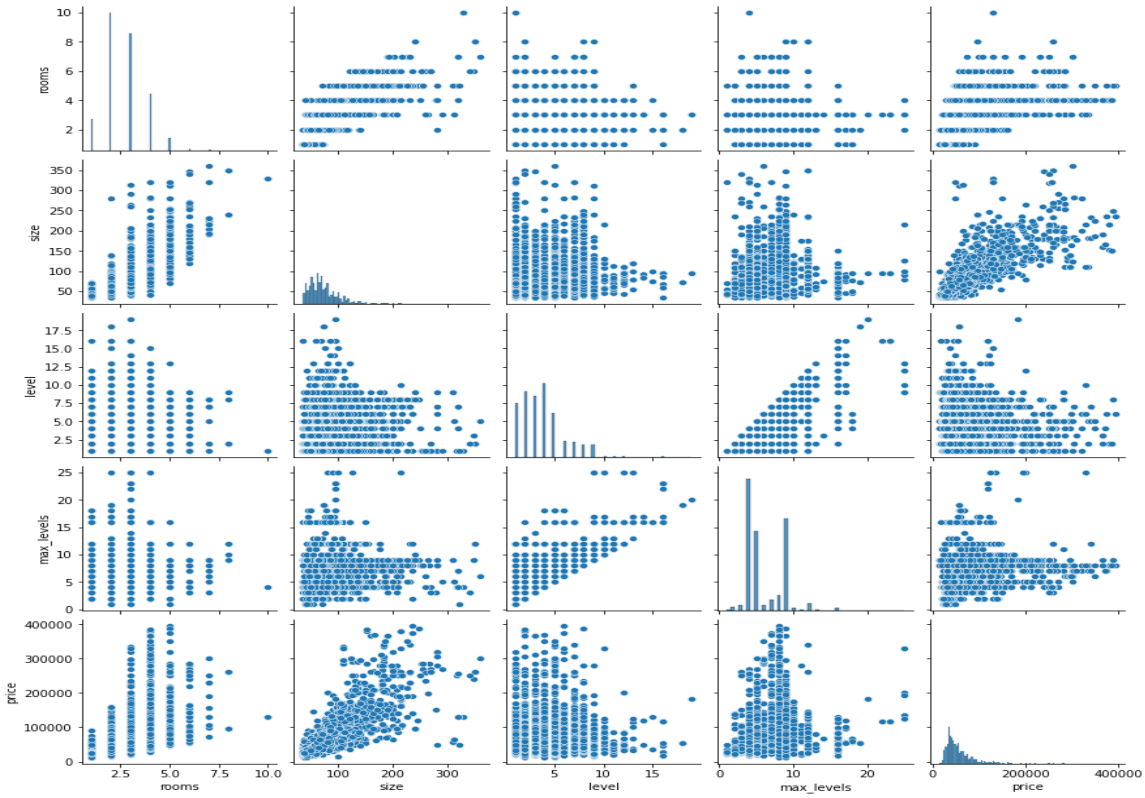
Интерпретируем данные в графиках:

```
df.hist(bins=50, figsize=(20,16))
plt.show()
```



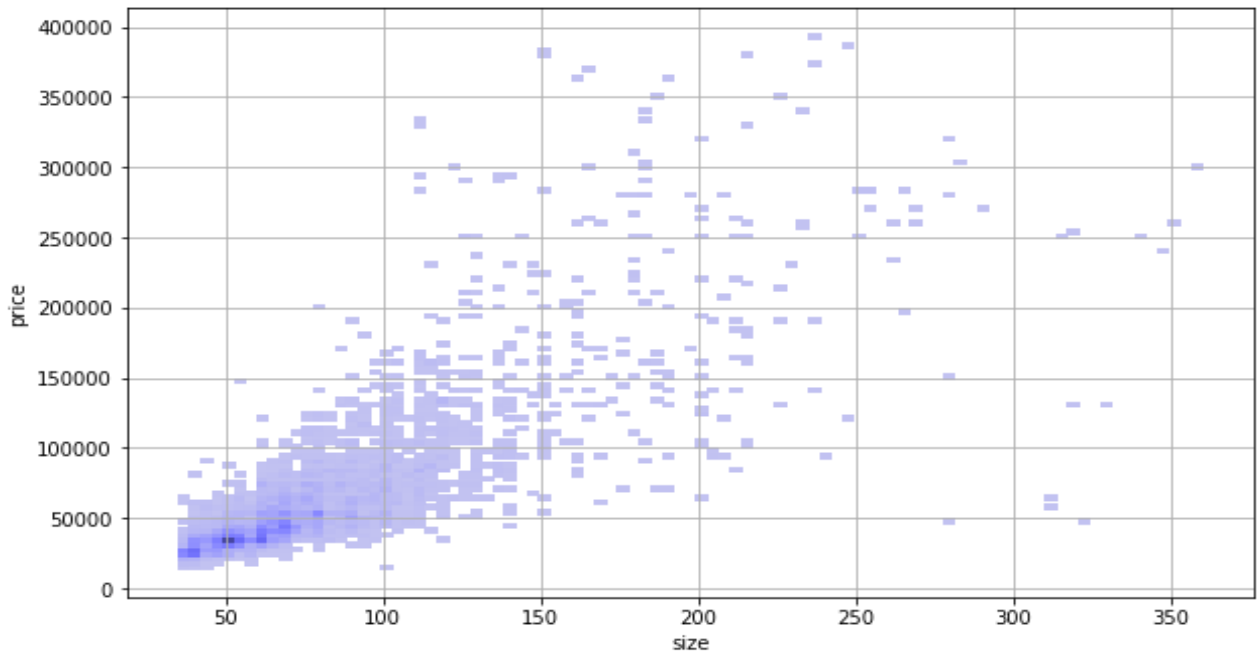
Из приведенного выше графика наиболее распространенным типом квартир являются 2-комнатные (более 2500 строк), квартиры площадью от 50 кв.м. до 80 кв.м., что дома с этажностью до 5 этажей составляют основную часть домов в продаже и что цены на дома в основном варьируются от 30 000 до 80 000 долларов США.

Теперь следует интерпретировать связь между каждым столбцом графически:



Из приведенного выше графика видно, что между столбцами size и price существует корреляционная зависимость.

Выделим график отдельно отражающий связь между столбцами size и price:



Из приведенного выше графика можно сделать вывод, что по мере того, как цены на квартиры растут более 300 000 долларов США и их площадь увеличивается более чем 300 кв.м., связь между столбцами price и size начинает исчезать. Выбрасиваем такие значения.

После обработки данных в таблице осталось 7440 строк данных.

	location	district	rooms	size	level	max_levels	price
0	город Ташкент, Юнусабадский район, Юнусабад 8-...	Юнусабадский	3	57.0	4	4	52000.0
1	город Ташкент, Яккасарайский район, 1-й тупик ...	Яккасарайский	2	52.0	4	5	56000.0
2	город Ташкент, Чиланзарский район, Чиланзар 2-...	Чиланзарский	2	42.0	4	4	37000.0
3	город Ташкент, Чиланзарский район, Чиланзар 9-...	Чиланзарский	3	65.0	1	4	49500.0
4	город Ташкент, Чиланзарский район, площадь Актепа	Чиланзарский	3	70.0	3	5	55000.0
...	...	...	...	...	...	...	...
7559	город Ташкент, Шайхантахурский район, Алишера ...	Шайхантахурский	1	27.5	2	4	23000.0
7560	город Ташкент, Яшнободский район, Городок Авиа...	Яшнободский	1	38.0	5	5	24500.0
7561	город Ташкент, Яшнободский район, 1-й проезд А...	Яшнободский	2	49.0	1	4	32000.0
7562	город Ташкент, Шайхантахурский район, Зульфиях...	Шайхантахурский	2	64.0	3	9	40000.0
7564	город Ташкент, Чиланзарский район, Чиланзар 6-...	Чиланзарский	1	30.0	2	4	22914.0

7440 rows × 7 columns

Рассчитаем зависимость значений столбца цена (price) между остальными значениями таблицы которую требуется спрогнозировать по формуле коэффициента корреляции:

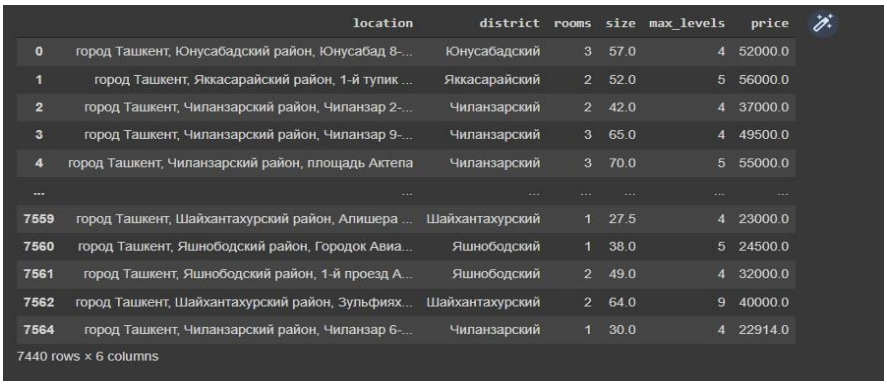
$$r = \frac{\sum_{i=1}^{n} x_i \left(\left(x_i - \bar{x} \right) \left(y_i - \bar{y} \right) \right)}{\sqrt{\sum_{i=1}^{n} \left(x_i - \bar{x} \right)^2 \sum_{i=1}^{n} \left(y_i - \bar{y} \right)^2}}$$

df.corrwith(df["price"])

```
rooms         0.555916
size          0.794604
level         0.055512
max_levels    0.232720
price         1.000000
dtype: float64
```

Как видно из приведенной выше таблицы, корреляция между ценой дома (price) и площадью квартиры (size) является сильной. Также корреляция между ценой квартиры (price) и количеством комнат (rooms) достаточно велика. Но корреляция между этажем (level) и ценой квартиры (price) на котором находится квартира очень мала. Из этого можно сделать вывод, что этаж (level), на котором расположена квартира, практически не оказывает положительного влияния на модель, которую мы хотим построить. Вот почему мы пропускаем этот столбец.

df.drop("level", axis=1, inplace=True)



	location	district	rooms	size	max_levels	price
0	город Ташкент, Юнусабадский район, Юнусабад 8-...	Юнусабадский	3	57.0	4	52000.0
1	город Ташкент, Яккасарайский район, 1-й тупик ...	Яккасарайский	2	52.0	5	56000.0
2	город Ташкент, Чиланзарский район, Чиланзар 2-...	Чиланзарский	2	42.0	4	37000.0
3	город Ташкент, Чиланзарский район, Чиланзар 9-...	Чиланзарский	3	65.0	4	49500.0
4	город Ташкент, Чиланзарский район, площадь Актепа	Чиланзарский	3	70.0	5	55000.0
...	...	...	...	...	...	...
7559	город Ташкент, Шайхантахурский район, Алишера ...	Шайхантахурский	1	27.5	4	23000.0
7560	город Ташкент, Яшнободский район, Городок Авиа...	Яшнободский	1	38.0	5	24500.0
7561	город Ташкент, Яшнободский район, 1-й проезд А...	Яшнободский	2	49.0	4	32000.0
7562	город Ташкент, Шайхантахурский район, Зульфиях...	Шайхантахурский	2	64.0	9	40000.0
7564	город Ташкент, Чиланзарский район, Чиланзар 6-...	Чиланзарский	1	30.0	4	22914.0

7440 rows × 6 columns

Из таблицы видно, что исчезает столбец “level” (этаж), на котором находится квартира, и остаётся 6 столбцов. Далее необходимо разделить данные на 2 части для построения и тестирования модели. Соотношение 80% к 20%.

```
from sklearn.model_selection import train_test_split

train_set, test_set = train_test_split(df, test_size=0.2, random_state=42)
```

При помощи команды (train_set) строим модель с данными первой части, а при помощи команды (test_set) тестируем модель с данными второй части, используя 20% данных. Поскольку наша цель - предсказать цену квартиры (price), следовательно, из таблицы необходимо отделить столбец “price” (цена квартиры).

```
X_train=train_set.drop('price', axis=1)

y_train=train_set['price'].copy()
```

Как уже говорилось выше, модель, которую мы собираемся построить, работает только с числами. Для этого первые два столбца в начале таблицы загружаем отдельно, а столбцы с числовыми значениями в отдельные переменные.

```
X_train_cat=X_train[['location', 'district']]

X_train_num=X_train[['rooms', 'size', 'max_levels']]
```

При помощи **“Transformer”**, который добавляет еще один столбец положительно влияющий на данные столбца “price” (цена квартиры). “Transformer” (далее трансормер) выполняет математические операции над данными в таблице и изменяет ее форму.

```
from sklearn.base import BaseEstimator, TransformerMixin

rooms_ix, size_ix, = 0, 1
```

```
class CombinedAttributesAdder(BaseEstimator, TransformerMixin):
def __init__(self, add_size_rooms=True):
self.add_size_rooms=add_size_rooms
def fit(self, X, y=None):
return self
def transform(self, X):
if self.add_size_rooms:
size_rooms = X[:, size_ix] / X[:, rooms_ix]
return np.c_[X, size_rooms]
else:
return np.c_[X]
```

Этот трансормер рассчитывает соотношение всех элементов друг к другу столбца “size” (площадь квартиры) к столбцу “rooms” (количество комнат). В результате образуется столбец средней площади квартиры, соответствующий количеству комнат квартиры.

Чтобы не всегда выполнять процесс подготовки данных вручную, мы автоматизируем этот процесс. Для этого задаем следующую команду ***“Pipeline”:***

```
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OrdinalEncoder, StandardScaler
num_pipeline=Pipeline([
        ('imputer', SimpleImputer(strategy="median")),
        ('attribs_adder', CombinedAttributesAdder(add_size_rooms
=True)),
        ('std_scaler', StandardScaler())
])
num_pipeline.fit_transform(X_train_num)
```

```
array([[-0.6022353 , -0.38600497, -0.80252968,  0.15991495],
       [-0.6022353 , -0.32923049, -0.80252968,  0.28927823],
       [-0.6022353 , -0.49955392, -0.80252968, -0.0988116 ],
       ...,
       [-0.6022353 , -0.49955392, -0.80252968, -0.0988116 ],
       [-1.55101893, -1.2943966 , -0.42015243, -0.0988116 ],
       [-0.6022353 , -0.38600497, -0.42015243,  0.15991495]])
```

При помощи 3-х трансформеров в выше указанном **“*Pipeline*”** преобразуется формы данных путем обработки данных:

— вместо пустых пробелов в таблице выставляются медианные значения элементов этого столбца с помощью трансформера ***“SimpleImputer”***.

— вычисляется отношение всех элементов в столбце площадь квартиры (size) и количество комнат (rooms) друг к другу и полученный из них столбец добавляется в таблицу с помощью трансформера ***“CombinedAttributesAdder”***.

— выводится стандартный интервал для числовых данных, прошедших вышеуказанные этапы при помощи трансформера ***“StandardScaler”*** с применением формулы

$$z = \frac{x - \mu}{\sigma}$$

Здесь σ -стандартное отклонение, μ - среднее значение для каждого столбца, x - элементы каждого столбца.

Теперь мы обрабатываем столбцы с текстовыми значениями при помощи другой “Pipeline”:

```
from sklearn.compose import ColumnTransformer

num_attribs=list(X_train_num)

cat_attribs=['location', 'district']

full_pipeline=ColumnTransformer([

                ("num", num_pipeline, num_attribs),

                ("cat", OrdinalEncoder(), cat_attribs)

])

X_prepar=full_pipeline.fit_transform(X_train)

X_prepared=X_prepar.toarray()
```

X_prepared

```
array([[-0.6022353 , -0.38600497, -0.80252968, ...,  0.        ,
         0.        ,  0.        ],
       [-0.6022353 , -0.32923049, -0.80252968, ...,  0.        ,
         0.        ,  0.        ],
       [-0.6022353 , -0.49955392, -0.80252968, ...,  0.        ,
         0.        ,  0.        ],
       ...,
       [-0.6022353 , -0.49955392, -0.80252968, ...,  0.        ,
         0.        ,  0.        ],
       [-1.55101893, -1.2943966 , -0.42015243, ...,  0.        ,
         0.        ,  0.        ],
       [-0.6022353 , -0.38600497, -0.42015243, ...,  0.        ,
         0.        ,  0.        ]])
```

В приведенной выше ***"Pipeline"***, наряду с работой с числовыми данными, выполненной на предыдущем шаге, осуществляется перевод текстов в числовое представление с помощью трансформера ***"OrdinalEncoder"***. Этот процесс осуществляется следующим образом:

Трансформер сначала создает уникальную с невторяющимися значениями таблицу из значений столбца "location" (местоположение квартиры) и столбца "district" район в котором находится дом. Затем помещает числа от $(1, n)$ в каждый элемент таблицы.

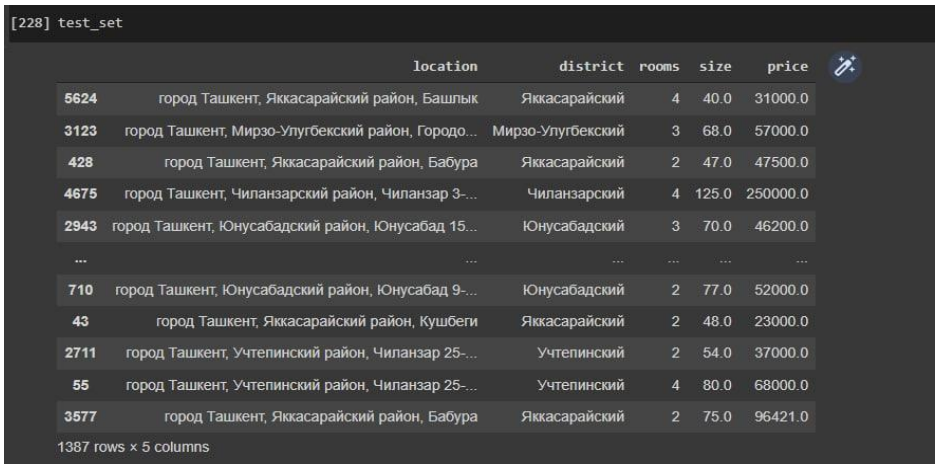
Теперь приступим к построению модели. Для этого мы вызываем Трансформер *"LinearRegression"* из библиотеки *sklearn* и передаем Трансформеру подготовленные данные **(X_prepared**) в качестве аргументов, а цену квартиры "price" (**y_train**) в качестве значений, которые необходимо вычислить.

```
from sklearn.linear_model import LinearRegression

model=LinearRegression()

model.fit(X_prepared, y_train)
```

Трансормер *"LinearRegression"* вычисляет коэффициенты $a_0, a_1, a_2, ..., a_n$ для X_prepared и y_train, а затем составляет формулу линейной регрессии.

$$y = a_0 + a_1 x + a_2 x + ... + a_n x$$

Используя "*Pipeline"* для обработки "*test_set"* можно спрогнозировать модель.



[228] test_set

	location	district	rooms	size	price
5624	город Ташкент, Яккасарайский район, Башлык	Яккасарайский	4	40.0	31000.0
3123	город Ташкент, Мирзо-Улугбекский район, Городо...	Мирзо-Улугбекский	3	68.0	57000.0
428	город Ташкент, Яккасарайский район, Бабура	Яккасарайский	2	47.0	47500.0
4675	город Ташкент, Чиланзарский район, Чиланзар 3-...	Чиланзарский	4	125.0	250000.0
2943	город Ташкент, Юнусабадский район, Юнусабад 15...	Юнусабадский	3	70.0	46200.0
...	...	...	...	...	...
710	город Ташкент, Юнусабадский район, Юнусабад 9-...	Юнусабадский	2	77.0	52000.0
43	город Ташкент, Яккасарайский район, Кушбеги	Яккасарайский	2	48.0	23000.0
2711	город Ташкент, Учтепинский район, Чиланзар 25-...	Учтепинский	2	54.0	37000.0
55	город Ташкент, Учтепинский район, Чиланзар 25-...	Учтепинский	4	80.0	68000.0
3577	город Ташкент, Яккасарайский район, Бабура	Яккасарайский	2	75.0	96421.0

1387 rows × 5 columns

X_test=test_set.drop(‘price’, axis=1)

y_test=test_set[‘price’].copy()

X_test_cat=X_test[[‘location’, ‘district’]]

X_test_num=X_test[[‘rooms’, ‘size’]]

from sklearn.compose import ColumnTransformer

num_attribs=list(X_test_num)

cat_attribs=[“location”, “district”]

full_pipeline=ColumnTransformer([

(“num”, num_pipeline, num_attribs),

(“cat”, OrdinalEncoder(), cat_attribs)

])

X_test_prepared=num_pipeline.fit_transform(X_test_num)

X_test_prepared

```
array([[ 1.29648542, -1.09874807, -2.58311279],
       [ 0.29858016, -0.19393356, -0.79116866],
       [-0.6993251 , -0.87254444, -0.67327759],
       ...,
       [-0.6993251 , -0.64634081, -0.17813514],
       [ 1.29648542,  0.19384409, -1.16842005],
       [-0.6993251 ,  0.03227007,  1.30729224]])
```

Прогнозируем с помощью модели:

y_predicted=model.predict(X_test_prepared)

y_predicted

```
array([30006.91703295, 45126.30573305, 33977.1457005 , ...,
       39093.15124785, 52167.4598737 , 54620.84563566])
```

Для оценки модели рассчитаем среднюю квадратичную ошибку (Root mean squared error-RMSE) и среднюю абсолютную ошибку (Mean absolute error-MAE:

$$\text{RSMSE}(\mathbf{X}, h) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(h(x^{(i)} - y^{(i)})\right)^2}$$

$$MAE(\mathbf{X}, h) = \frac{1}{m}\sum_{i=1}^{m}\left|h(x^{(i)} - y^{(i)})\right|$$

```
from sklearn.metrics import mean_absolute_error, mean_squared_error

mae=mean_absolute_error(y_test, y_predicted)

rmse=np.sqrt(mean_squared_error(y_test, y_predicted))

print(f"MAE: {mae}")

print(f"RMSE: {rmse}")
```

MAE: 3230,948468296183

RSME: 4534,54482317707

Подводя итог проделанной работы, недостаток модели заключается в том, что при прогнозировании цены на дом она в среднем отклоняется на 3230 долларов США. Преимущество модели заключается в том, что она может спрогнозировать цены на жилье за очень короткое время независимо от объёма информации.

Использованные литературы

1. Н.В.Манюкова, Л.Ю.Уразаева, Р.Е.Мамедли – «Математическое моделирование в преподавании информационных технологий». Математические структуры и моделирование 2019 №4(52). С: 118-133. Санкт-Петербург, Россия.

2. . Anvar Narzullayev - «Python dasturlash asoslari». Akademnashr, Toshkent 2021.

3. William P.Fox, Robert E. Burks – «Advanced Mathematical Modeling with technology». CRC Preti, 2021