Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ

КАФЕДРА СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

НА ТЕМУ:

Прогнозирование цен автомобилей

Студент ИУ5-33М _____ **Н.С. Анцифров**
(Группа) (Подпись, дата) (И.О. Фамилия)

Руководитель _____ **Ю.Е. Гапанюк**
(Подпись, дата) (И.О. Фамилия)

2024 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ

Заведующий кафедрой ИУ5
(Индекс)
_____ В.И. Терехов
(И.О.Фамилия)
«____» _____ ____ г.

З А Д А Н И Е
на выполнение научно-исследовательской работы

по теме Прогнозирование цен автомобилей

Студент группы ИУ5-33М

Анцифров Никита Сергеевич
(Фамилия, имя, отчество)

Направленность НИР (учебная, исследовательская, практическая, производственная, др.)

УЧЕБНАЯ

Источник тематики (кафедра, предприятие, НИР) КАФЕДРА

График выполнения работы: 25% к __ нед., 50% к __ нед., 75% к __ нед., 100% к __ нед.

Техническое задание Провести разведочный анализ данных. Решить задачу регрессии по прогнозированию цен автомобилей.

Оформление научно-исследовательской работы:

Расчетно-пояснительная записка на 27 листах формата А4.
Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)

Дата выдачи задания «__» _____ _____ г.

Руководитель НИР _____ **Ю.Е. Гапанюк**
(Подпись, дата) (И.О. Фамилия)

Студент _____ **Н.С. Анцифров**
(Подпись, дата) (И.О. Фамилия)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В качестве набора данных предметной области был выбран набор данных, содержащий данные об автомобилях, проданных за некоторый период на территории США [1].

Одной из основных задач можно считать предсказание цены автомобиля на основе нескольких факторов. Данная задача может быть актуальна для автомобильной компании, планирующей свой выход на автомобильный рынок США, открыв там своё производственное предприятие и производя автомобили локально, чтобы составить конкуренцию своим американским и европейским аналогам.

Решение этой задачи может быть использовано руководством автомобильной компании для понимания того, как именно цены изменяются в зависимости от характеристик автомобилей. С использованием этих данных, оно сможет более оптимально разрабатывать новые модели своих автомобилей, чтобы соответствовать определённым ценовым сегментам. Кроме того, модель регрессии может стать хорошим способом для понимания динамики ценообразования на новом рынке.

ОСНОВНАЯ ЧАСТЬ

Описание набора данных

Набор данных имеет следующие атрибуты:

– car_ID – порядковый номер строки;

– symboling – обозначение;

– CarName – марка + модель автомобиля;

– fueltype – тип топлива;

– aspiration – тип подачи воздуха в двигатель (атмосферный/турбированный);

– doornumber – число дверей;

– carbody – тип кузова;

– drivewheel – привод;

– enginelocation – расположение двигателя;

– wheelbase – длина колесной базы;

– carlength – длина автомобиля;

– carwidth – ширина автомобиля;

– carheight – высота автомобиля;

– curbweight – снаряженная масса;

– enginetype – тип двигателя;

– cylindernumber – число цилиндров;

– enginesize – объем двигателя;

– fuelsystem – тип топливной системы;

– boreratio – интерес для покупателя;

– stroke – поршни;

– compressionratio – компрессия;

– horsepower – лошадиные силы;

– peakrpm – обороты в минуты, при которых достигается максимальный момент;

– citympg – расход топлива по городу;

– highwaympg – расход по трассе;

– price – цена.

В качестве целевого признака выделим «цену».

Загрузка данных и вывод основных характеристик

Загрузим данные:

```
[2]: data = pd.read_csv('car.csv')
```

Выведем первые 5 строк датасета:

```
[3]: data.head()
```

```
[3]:    car_ID  symboling                   CarName fueltype aspiration doornumber  \
     0       1          3        alfa-romero giulia      gas        std        two
     1       2          3       alfa-romero stelvio      gas        std        two
     2       3          1  alfa-romero Quadrifoglio      gas        std        two
     3       4          2               audi 100 ls      gas        std       four
     4       5          2                audi 100ls      gas        std       four

         carbody drivewheel enginelocation  wheelbase  …  enginesize  \
     0  convertible        rwd          front       88.6  …         130
     1  convertible        rwd          front       88.6  …         130
     2    hatchback        rwd          front       94.5  …         152
     3        sedan        fwd          front       99.8  …         109
     4        sedan        4wd          front       99.4  …         136

        fuelsystem  boreratio  stroke compressionratio horsepower  peakrpm citympg  \
     0        mpfi       3.47    2.68              9.0        111     5000       21
     1        mpfi       3.47    2.68              9.0        111     5000       21
     2        mpfi       2.68    3.47              9.0        154     5000       19
     3        mpfi       3.19    3.40             10.0        102     5500       24
     4        mpfi       3.19    3.40              8.0        115     5500       18

        highwaympg    price
     0          27  13495.0
     1          27  16500.0
     2          26  16500.0
     3          30  13950.0
     4          22  17450.0

     [5 rows x 26 columns]
```

Размер датасета:

```
[4]: data.shape
```

```
[4]: (205, 26)
```

Столбцы:

```
[5]: data.columns
```

```
[5]: Index(['car_ID', 'symboling', 'CarName', 'fueltype', 'aspiration',
       'doornumber', 'carbody', 'drivewheel', 'enginelocation', 'wheelbase',
       'carlength', 'carwidth', 'carheight', 'curbweight', 'enginetype',
       'cylindernumber', 'enginesize', 'fuelsystem', 'boreratio', 'stroke',
       'compressionratio', 'horsepower', 'peakrpm', 'citympg', 'highwaympg',
       'price'],
      dtype='object')
```

Типы данных:

```
[6]: data.dtypes
```

```
[6]: car_ID              int64
     symboling           int64
     CarName            object
     fueltype           object
     aspiration         object
     doornumber         object
     carbody            object
     drivewheel         object
     enginelocation     object
     wheelbase         float64
     carlength         float64
     carwidth          float64
     carheight         float64
     curbweight          int64
     enginetype         object
     cylindernumber     object
     enginesize          int64
     fuelsystem         object
     boreratio         float64
     stroke            float64
     compressionratio  float64
     horsepower          int64
     peakrpm             int64
     citympg             int64
     highwaympg          int64
     price             float64
     dtype: object
```

Обработка данных с неинформативными признаками

В датасете присутствуют данные, которые не несут полезной информации для дальнейшего анализа. Аналитически посчитаем неинформативные признаки (у которых более 90% строк имеют одинаковое значение):

[7]:
```
num_rows = len(data.index)
low_information_cols = [] #

for col in data.columns:
    cnts = data[col].value_counts(dropna=False)
    top_pct = (cnts/num_rows).iloc[0]

    if top_pct > 0.90:
        low_information_cols.append(col)
        print('{0}: {1:.5f}%'.format(col, top_pct*100))
        print(cnts)
        print()
```

```
fueltype: 90.24390%
gas       185
diesel     20
Name: fueltype, dtype: int64

enginelocation: 98.53659%
front    202
rear       3
Name: enginelocation, dtype: int64
```

Удалим соответствующие столбцы:

[8]:
```
data.drop(['fueltype', 'enginelocation'], inplace=True, axis=1)
```

Некоторые столбцы также не представляют ценности для дальнейшего анализа. Также удалим их:

[9]:
```
data.drop(['car_ID', 'symboling', 'enginesize', 'stroke', 'compressionratio'],
↪inplace=True, axis=1)
```

Обработка пропусков

Определим столбцы с пропусками данных:

[11]:
```
data.isnull().sum()
```

[11]:
```
CarName           0
aspiration        0
doornumber        0
carbody           0
drivewheel        0
wheelbase         0
carlength         0
carwidth          0
carheight         0
curbweight        0
enginetype        0
cylindernumber    0
fuelsystem        0
boreratio         0
horsepower        0
peakrpm           0
citympg           0
highwaympg        0
price             0
dtype: int64
```

Видим, что в наборе данных отсутствуют пропуски.

Переименование столбцов

Для более удобной дальнейшей работы переименуем столбцы:

```
[12]: data.rename(columns = {'doornumber' : 'doors', 'carbody' : 'body', 'drivewheel' :
      ↪'drive', 'carlength' : 'length', 'carwidth' : 'width', 'carheight' : 'height',
      ↪'curbweight' : 'weight', 'cylindernumber' : 'cyl', 'boreratio' : 'bore'},
      ↪inplace = True)
```

```
[13]: data.head()
```

[13]:

	CarName	aspiration	doors	body	drive	wheelbase	\
0	alfa-romero giulia	std	two	convertible	rwd	88.6	
1	alfa-romero stelvio	std	two	convertible	rwd	88.6	
2	alfa-romero Quadrifoglio	std	two	hatchback	rwd	94.5	
3	audi 100 ls	std	four	sedan	fwd	99.8	
4	audi 100ls	std	four	sedan	4wd	99.4	

	length	width	height	weight	enginetype	cyl	fuelsystem	bore	\
0	168.8	64.1	48.8	2548	dohc	four	mpfi	3.47	
1	168.8	64.1	48.8	2548	dohc	four	mpfi	3.47	
2	171.2	65.5	52.4	2823	ohcv	six	mpfi	2.68	
3	176.6	66.2	54.3	2337	ohc	four	mpfi	3.19	
4	176.6	66.4	54.3	2824	ohc	five	mpfi	3.19	

	horsepower	peakrpm	citympg	highwaympg	price
0	111	5000	21	27	13495.0
1	111	5000	21	27	16500.0
2	154	5000	19	26	16500.0
3	102	5500	24	30	13950.0
4	115	5500	18	22	17450.0

Преобразование столбцов

Преобразуем столбец, содержащий информацию о марке и модели, к двум отдельным столбцам:

[14]:
```
data[['manuf', 'model']] = data['CarName'].str.split(' ', 1, expand=True)
data.drop(['CarName'],axis=1,inplace=True)
data = data[['manuf', 'model', 'aspiration', 'doors', 'body', 'drive',
 ↪'wheelbase', 'length', 'width', 'height', 'weight', 'enginetype', 'cyl',
 ↪'fuelsystem', 'bore', 'horsepower', 'peakrpm', 'citympg', 'highwaympg',
 ↪'price']]
data.head()
```

[14]:

	manuf	model	aspiration	doors	body	drive	wheelbase	\
0	alfa-romero	giulia	std	two	convertible	rwd	88.6	
1	alfa-romero	stelvio	std	two	convertible	rwd	88.6	
2	alfa-romero	Quadrifoglio	std	two	hatchback	rwd	94.5	
3	audi	100 ls	std	four	sedan	fwd	99.8	
4	audi	100ls	std	four	sedan	4wd	99.4	

	length	width	height	weight	enginetype	cyl	fuelsystem	bore	\
0	168.8	64.1	48.8	2548	dohc	four	mpfi	3.47	
1	168.8	64.1	48.8	2548	dohc	four	mpfi	3.47	
2	171.2	65.5	52.4	2823	ohcv	six	mpfi	2.68	
3	176.6	66.2	54.3	2337	ohc	four	mpfi	3.19	
4	176.6	66.4	54.3	2824	ohc	five	mpfi	3.19	

	horsepower	peakrpm	citympg	highwaympg	price
0	111	5000	21	27	13495.0
1	111	5000	21	27	16500.0
2	154	5000	19	26	16500.0
3	102	5500	24	30	13950.0
4	115	5500	18	22	17450.0

Исправление ошибок

Проверим наличие ошибок. В столбце автомобилей исправим некоторые ошибки:

[24]:
```
data.manuf = data.manuf.str.lower()

def replace_name(a,b):
    data.manuf.replace(a,b,inplace=True)

replace_name('maxda','mazda')
replace_name('porcshce','porsche')
replace_name('toyouta','toyota')
replace_name('vokswagen','volkswagen')
replace_name('vw','volkswagen')

data.manuf.unique()
```

[24]:
```
array(['alfa-romero', 'audi', 'bmw', 'chevrolet', 'dodge', 'honda',
       'isuzu', 'jaguar', 'mazda', 'buick', 'mercury', 'mitsubishi',
       'nissan', 'peugeot', 'plymouth', 'porsche', 'renault', 'saab',
       'subaru', 'toyota', 'volkswagen', 'volvo'], dtype=object)
```

```
[25]: data.head()
```

[25]:

	manuf	model	aspiration	doors	body	drive	wheelbase
0	alfa-romero	giulia	std	two	convertible	rwd	88.6
1	alfa-romero	stelvio	std	two	convertible	rwd	88.6
2	alfa-romero	Quadrifoglio	std	two	hatchback	rwd	94.5
3	audi	100 ls	std	four	sedan	fwd	99.8
4	audi	100ls	std	four	sedan	4wd	99.4

	length	width	height	weight	enginetype	cyl	fuelsystem	bore
0	168.8	64.1	48.8	2548	dohc	four	mpfi	3.47
1	168.8	64.1	48.8	2548	dohc	four	mpfi	3.47
2	171.2	65.5	52.4	2823	ohcv	six	mpfi	2.68
3	176.6	66.2	54.3	2337	ohc	four	mpfi	3.19
4	176.6	66.4	54.3	2824	ohc	five	mpfi	3.19

	horsepower	peakrpm	citympg	highwaympg	price
0	111	5000	21	27	13495.0
1	111	5000	21	27	16500.0
2	154	5000	19	26	16500.0
3	102	5500	24	30	13950.0
4	115	5500	18	22	17450.0

Замена данных

В столбцах «doors» и «cyl» – объекты типа Object, числовые данные записаны в виде набора символов. Преобразуем их в числа:

```
[26]: doors = {'two': 2, 'four': 4}
data['doors'] = data['doors'].replace(doors)
data['doors'] = data['doors'].astype({"doors":"int64"})

cyl = {'four': 4, 'six': 6, 'five': 5, 'three': 3, 'twelve': 12, 'two': 2,
 ↪'eight': 8}
data['cyl'] = data['cyl'].replace(cyl)
data['cyl'] = data['cyl'].astype({"cyl":"int64"})
data.head()
```

[26]:

	manuf	model	aspiration	doors	body	drive	wheelbase
0	alfa-romero	giulia	std	2	convertible	rwd	88.6
1	alfa-romero	stelvio	std	2	convertible	rwd	88.6
2	alfa-romero	Quadrifoglio	std	2	hatchback	rwd	94.5
3	audi	100 ls	std	4	sedan	fwd	99.8
4	audi	100ls	std	4	sedan	4wd	99.4

	length	width	height	weight	enginetype	cyl	fuelsystem	bore	horsepower
0	168.8	64.1	48.8	2548	dohc	4	mpfi	3.47	111
1	168.8	64.1	48.8	2548	dohc	4	mpfi	3.47	111
2	171.2	65.5	52.4	2823	ohcv	6	mpfi	2.68	154
3	176.6	66.2	54.3	2337	ohc	4	mpfi	3.19	102
4	176.6	66.4	54.3	2824	ohc	5	mpfi	3.19	115

	peakrpm	citympg	highwaympg	price
0	5000	21	27	13495.0
1	5000	21	27	16500.0
2	5000	19	26	16500.0
3	5500	24	30	13950.0
4	5500	18	22	17450.0

Структура данных

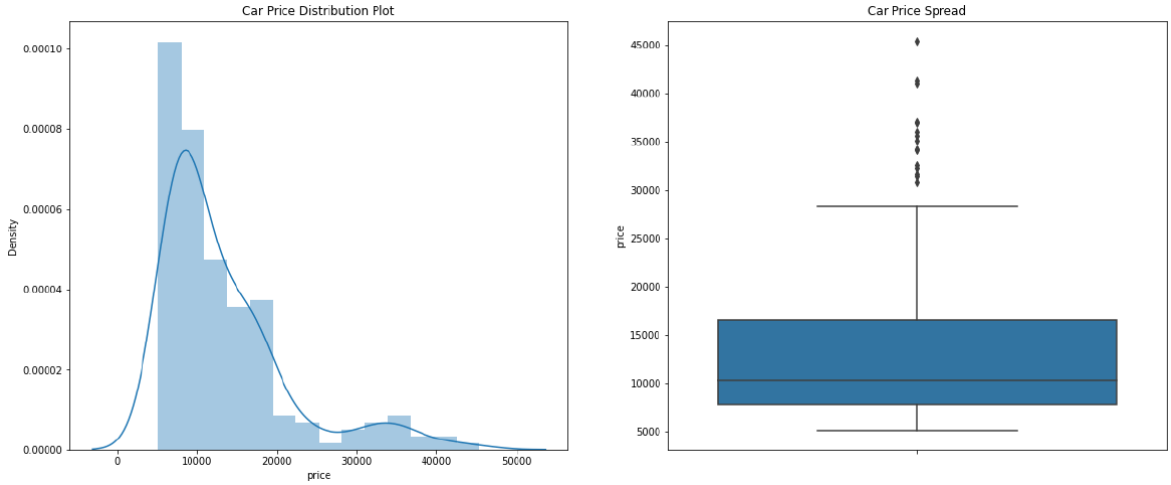
Построим графики распределения цен с помощью matplotlib [2]:

```
[28]: plt.figure(figsize=(20,8))

      plt.subplot(1,2,1)
      plt.title('Car Price Distribution Plot')
      sns.distplot(data.price)

      plt.subplot(1,2,2)
      plt.title('Car Price Spread')
      sns.boxplot(y=data.price)

      plt.show()
```



Кодирование категориальных признаков

Используя LabelEncoder из sckit-learn [3], закодируем некоторые столбцы типа Object в числовые значения:

```
[30]: from sklearn.preprocessing import LabelEncoder
```

[31]:
```
letypemanuf = LabelEncoder()
learrmanuf = letypemanuf.fit_transform(data["manuf"])
data["manuf"] = learrmanuf
data = data.astype({"manuf":"int64"})
```

[32]:
```
letypemodel = LabelEncoder()
learrmodel = letypemodel.fit_transform(data["model"])
data["model"] = learrmodel
data = data.astype({"model":"int64"})
```

[33]:
```
letypeasp = LabelEncoder()
learrasp = letypeasp.fit_transform(data["aspiration"])
data["aspiration"] = learrasp
data = data.astype({"aspiration":"int64"})
```

[34]:
```
letypebody = LabelEncoder()
learrbody = letypebody.fit_transform(data["body"])
data["body"] = learrbody
data = data.astype({"body":"int64"})
```

[35]:
```
letypedrive = LabelEncoder()
learrdrive = letypedrive.fit_transform(data["drive"])
data["drive"] = learrdrive
data = data.astype({"drive":"int64"})
```

[36]:
```
letypetype = LabelEncoder()
learrtype = letypetype.fit_transform(data["enginetype"])
data["enginetype"] = learrtype
data = data.astype({"enginetype":"int64"})
```

[37]:
```
letypefs = LabelEncoder()
learrfs = letypefs.fit_transform(data["fuelsystem"])
data["fuelsystem"] = learrfs
data = data.astype({"fuelsystem":"int64"})
```

[38]:
```
data.head()
```

[38]:

	manuf	model	aspiration	doors	body	drive	wheelbase	length	width	\
0	0	78	0	2	0	2	88.6	168.8	64.1	
1	0	122	0	2	0	2	88.6	168.8	64.1	
2	0	28	0	2	2	2	94.5	171.2	65.5	
3	1	0	0	4	3	1	99.8	176.6	66.2	
4	1	1	0	4	3	0	99.4	176.6	66.4	

	height	weight	enginetype	cyl	fuelsystem	bore	horsepower	peakrpm	\
0	48.8	2548	0	4	5	3.47	111	5000	
1	48.8	2548	0	4	5	3.47	111	5000	
2	52.4	2823	5	6	5	2.68	154	5000	
3	54.3	2337	3	4	5	3.19	102	5500	
4	54.3	2824	3	5	5	3.19	115	5500	

	citympg	highwaympg	price
0	21	27	13495.0
1	21	27	16500.0
2	19	26	16500.0
3	24	30	13950.0
4	18	22	17450.0

Масштабирование данных

Проведём масштабирование данных MinMax с помощью средств из sckit-learn:

[39]:
```
from sklearn.preprocessing import MinMaxScaler
```

[40]:
```
scaler = MinMaxScaler()
scaler_data = scaler.fit_transform(data[data.columns])
```

Сохраним масштабированные данные:

[41]:
```
data_scaled = pd.DataFrame()
```

[42]:
```
for i in range(len(data.columns)):
    col = data.columns[i]
    new_col_name = col + '_scaled'
    data_scaled[new_col_name] = scaler_data[:,i]
```

[43]:
```
data_scaled.head()
```

[43]:

	manuf_scaled	model_scaled	aspiration_scaled	doors_scaled	body_scaled
0	0.000000	0.553191	0.0	0.0	0.00
1	0.000000	0.865248	0.0	0.0	0.00
2	0.000000	0.198582	0.0	0.0	0.50
3	0.047619	0.000000	0.0	1.0	0.75
4	0.047619	0.007092	0.0	1.0	0.75

	drive_scaled	wheelbase_scaled	length_scaled	width_scaled	height_scaled
0	1.0	0.058309	0.413433	0.316667	0.083333
1	1.0	0.058309	0.413433	0.316667	0.083333
2	1.0	0.230321	0.449254	0.433333	0.383333
3	0.5	0.384840	0.529851	0.491667	0.541667
4	0.0	0.373178	0.529851	0.508333	0.541667

	weight_scaled	enginetype_scaled	cyl_scaled	fuelsystem_scaled
0	0.411171	0.000000	0.2	0.714286
1	0.411171	0.000000	0.2	0.714286
2	0.517843	0.833333	0.4	0.714286
3	0.329325	0.500000	0.2	0.714286
4	0.518231	0.500000	0.3	0.714286

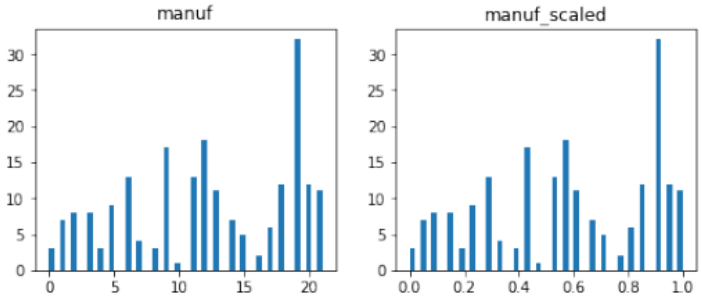
	bore_scaled	horsepower_scaled	peakrpm_scaled	citympg_scaled
0	0.664286	0.262500	0.346939	0.222222
1	0.664286	0.262500	0.346939	0.222222
2	0.100000	0.441667	0.346939	0.166667
3	0.464286	0.225000	0.551020	0.305556
4	0.464286	0.279167	0.551020	0.138889

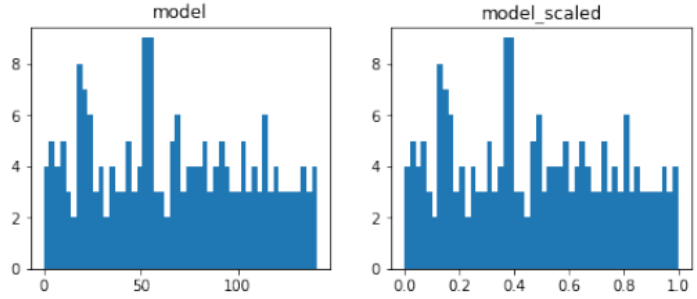
	highwaympg_scaled	price_scaled
0	0.289474	0.207959
1	0.289474	0.282558
2	0.263158	0.282558
3	0.368421	0.219254
4	0.157895	0.306142

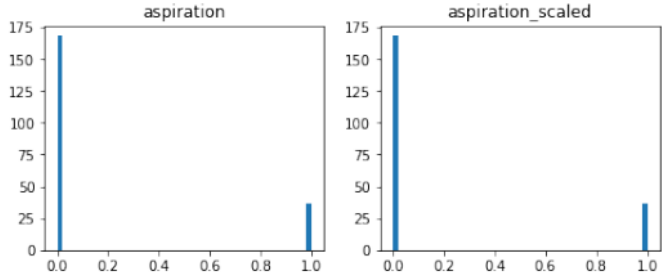
Масштабирование данных не повлияло на распределение данных:

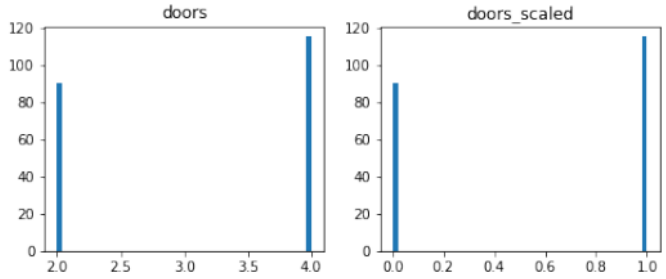
[44]:
```
for col in data.columns:
    col_scaled = col + '_scaled'

    fig, ax = plt.subplots(1, 2, figsize=(8,3))
    ax[0].hist(data[col], 50)
    ax[1].hist(data_scaled[col_scaled], 50)
    ax[0].title.set_text(col)
    ax[1].title.set_text(col_scaled)
    plt.show()
```







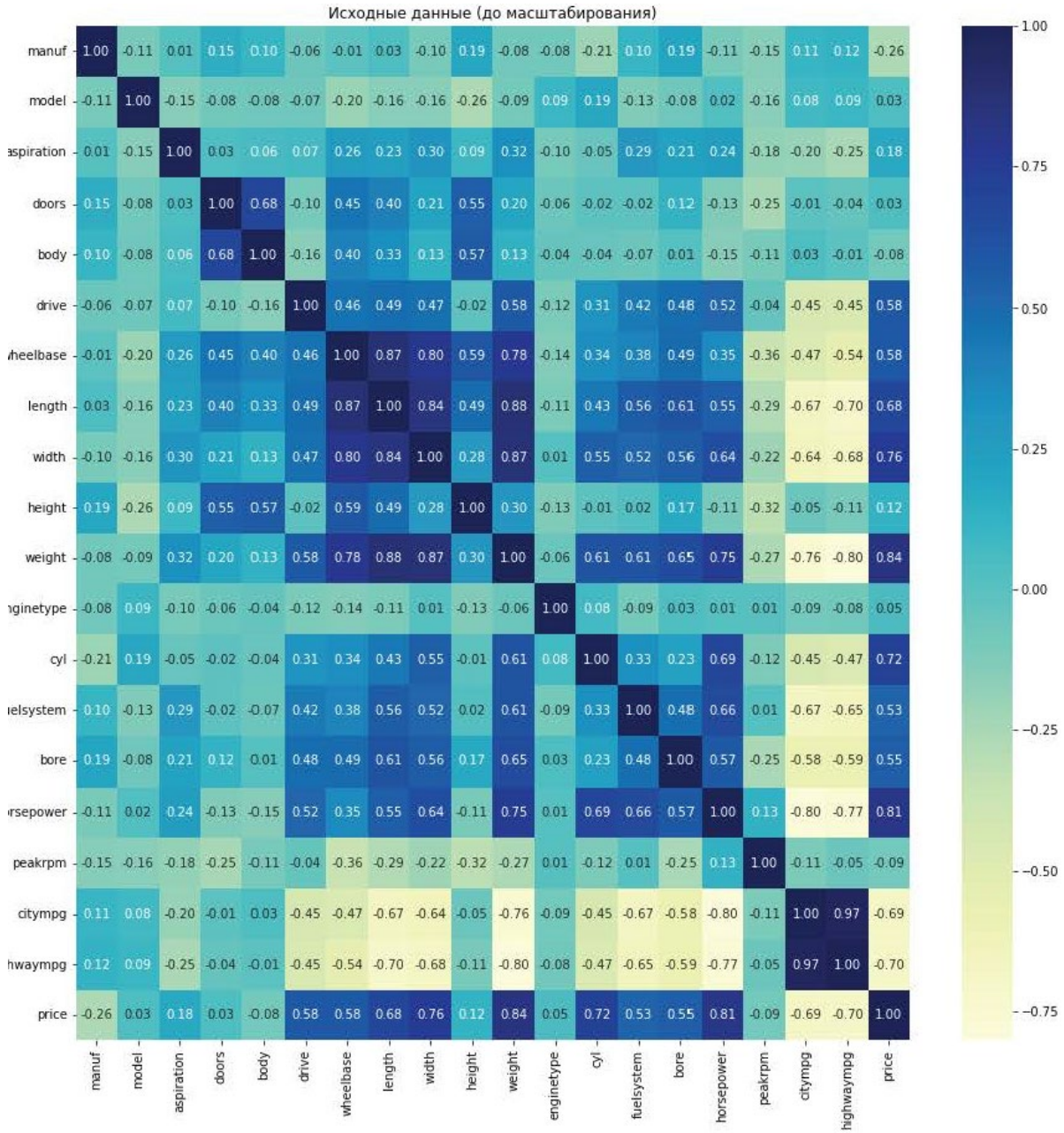


…

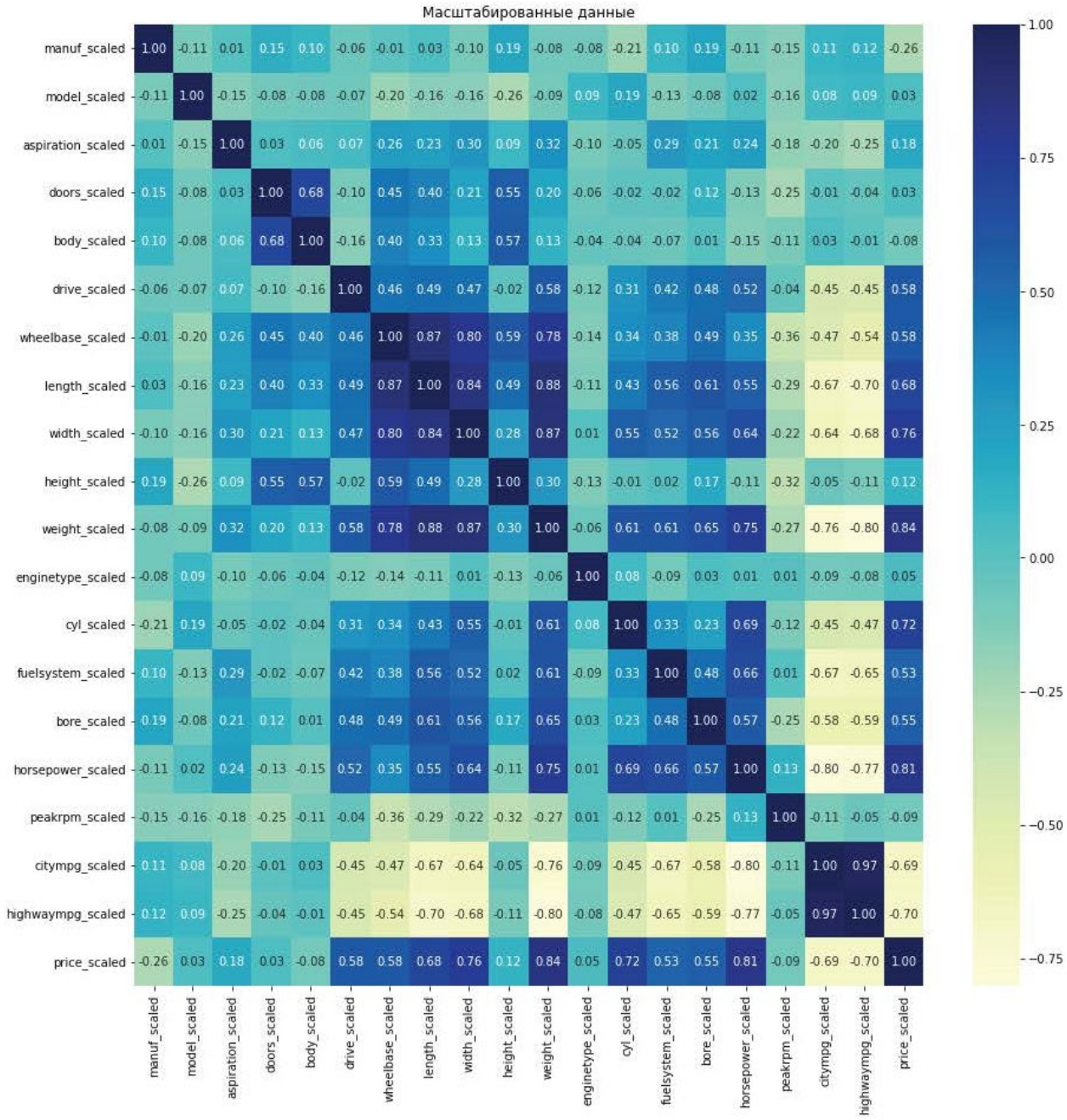
Корреляционный анализ данных

Построим корреляционные матрицы:

[45]:
```
fig, ax = plt.subplots(figsize=(15,15))
sns.heatmap(data[data.columns].corr(), annot=True, fmt='.2f', cmap="YlGnBu")
ax.set_title('Исходные данные (до масштабирования)')
plt.show()
```



[46]:
```
fig, ax = plt.subplots(figsize=(15,15))
sns.heatmap(data_scaled[data_scaled.columns].corr(), annot=True, fmt='.2f',
↪cmap="YlGnBu")
ax.set_title('Масштабированные данные')
plt.show()
```



На основании корреляционных матрицы можно сделать следующие выводы:

– Корреляционные матрицы для исходных и масштабированных данных идентичны.

– Целевой признак регрессии «price» наиболее сильно коррелирует с «drive» (0.58), «wheelbase» (0.58), «length» (0.68), «width» (0.76), «weight» (0.84), «cyl» (0.72) и «horsepower» (0.81). Эти признаки в модели регрессии оставляем. Признаки «citympg» и «highwaympg» имеют корреляцию, близкую по модулю к 1, поэтому оставим только один из них – «citympg».

– Данные позволяют построить модель машинного обучения.

Удалим ненужный признак:

[47]:
```
data.drop(['highwaympg'], inplace=True, axis=1)
data_scaled.drop(['highwaympg_scaled'], inplace=True, axis=1)
```

[48]:
```
data.head()
```

[48]:

	manuf	model	aspiration	doors	body	drive	wheelbase	length	width	\
0	0	78	0	2	0	2	88.6	168.8	64.1	
1	0	122	0	2	0	2	88.6	168.8	64.1	
2	0	28	0	2	2	2	94.5	171.2	65.5	
3	1	0	0	4	3	1	99.8	176.6	66.2	
4	1	1	0	4	3	0	99.4	176.6	66.4	

	height	weight	enginetype	cyl	fuelsystem	bore	horsepower	peakrpm	\
0	48.8	2548	0	4	5	3.47	111	5000	
1	48.8	2548	0	4	5	3.47	111	5000	
2	52.4	2823	5	6	5	2.68	154	5000	
3	54.3	2337	3	4	5	3.19	102	5500	
4	54.3	2824	3	5	5	3.19	115	5500	

	citympg	price
0	21	13495.0
1	21	16500.0
2	19	16500.0
3	24	13950.0
4	18	17450.0

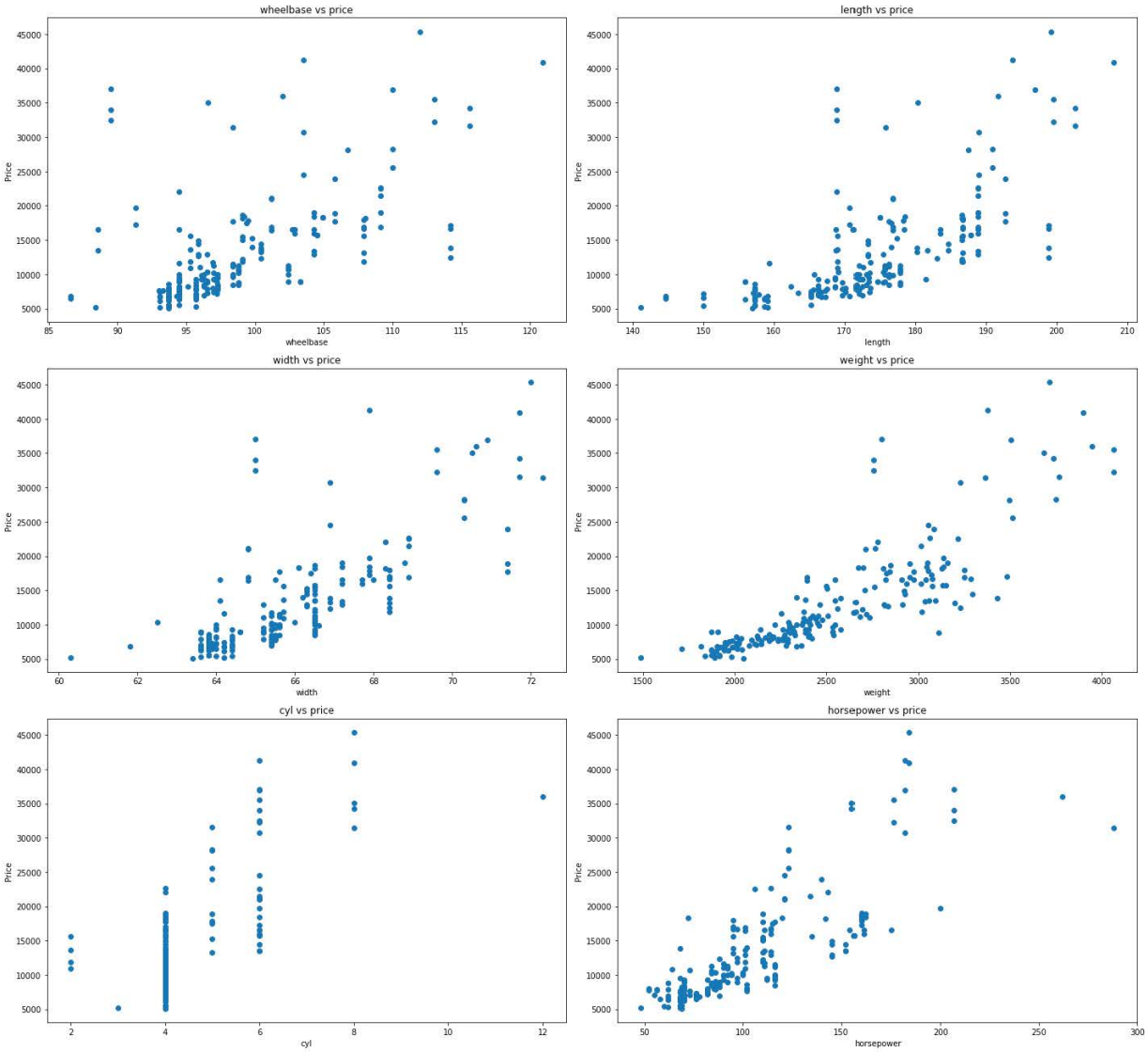
Построим графики зависимостей признаков с сильной корреляцией:

[49]:
```
def scatter(x,fig):
    plt.subplot(5,2,fig)
    plt.scatter(data[x],data['price'])
    plt.title(x+' vs price')
    plt.ylabel('Price')
    plt.xlabel(x)

plt.figure(figsize=(20,30))

scatter('wheelbase', 1)
scatter('length', 2)
scatter('width', 3)
scatter('weight', 4)
scatter('cyl', 5)
scatter('horsepower', 6)

plt.tight_layout()
```



Модели линейной регрессии и метрики

Для решения задачи регрессии будем использовать следующие модели:

- Линейная регрессия.
- Модель ближайших соседей.
- Модель опорных векторов.
- Дерево решений.
- Случайный лес.
- Градиентный бустинг.

В качестве метрик для решения задачи регрессии будем использовать метрики:

– Mean absolute error (средняя абсолютная ошибка).
– Mean squared error (средняя квадратичная ошибка).
– R2-score (коэффициент детерминации).

Формирование обучающей и тестовой выборок

Разделим выборку:

[51]:
```
from sklearn.model_selection import train_test_split
```

[52]:
```
X_train, X_test, y_train, y_test = train_test_split(data, data.price,
↪random_state=1)
```

[53]:
```
X_train.shape, y_train.shape, X_test.shape, y_test.shape
```

[53]:
```
((153, 19), (153,), (52, 19), (52,))
```

Построение моделей

Построим модели:

[54]:
```
from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsRegressor
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
```

[55]:
```
regr_models = {'LR': LinearRegression(),
               'KNN_20':KNeighborsRegressor(n_neighbors=20),
               'SVR':SVR(),
               'Tree':DecisionTreeRegressor(),
               'RF':RandomForestRegressor(),
               'GB':GradientBoostingRegressor()}
```

Рассчитаем метрики:

[56]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

[57]:
```
regrMetricLogger = MetricLogger()
```

[58]:
```
def regr_train_model(model_name, model, regrMetricLogger):
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    mae = mean_absolute_error(y_test, y_pred)
    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

    regrMetricLogger.add('MAE', model_name, mae)
    regrMetricLogger.add('MSE', model_name, mse)
    regrMetricLogger.add('R2', model_name, r2)

    print('{} \t MAE={}, MSE={}, R2={}'.format(
        model_name, round(mae, 3), round(mse, 3), round(r2, 3)))
```

[59]:
```
for model_name, model in regr_models.items():
    regr_train_model(model_name, model, regrMetricLogger)
```

```
LR       MAE=0.0, MSE=0.0, R2=1.0
KNN_20   MAE=769.6, MSE=5081904.089, R2=0.924
SVR      MAE=5494.731, MSE=70686472.641, R2=-0.053
Tree     MAE=364.109, MSE=875873.509, R2=0.987
RF       MAE=201.948, MSE=283052.672, R2=0.996
GB       MAE=140.891, MSE=126015.415, R2=0.998
```

Чем ближе значение MAE и MSE к 0 и R2 к 1 – тем лучше качество регрессии. Видно, что по трём метрикам лучшая модель регрессии – у линейной модели. Но также по метрике R2-score модели градиентного бустинга, случайного леса и ближайших соседей близки к линейной. Худшая модель по всем трём метрикам - модель опорных векторов.

Подбор оптимальной модели

Подберём оптимальную модель:

[60]:
```
from sklearn.model_selection import GridSearchCV
```

[61]:
```
n_range = np.array(range(5,100,5))
tuned_parameters = [{'n_neighbors': n_range}]
tuned_parameters
```

[61]:
```
[{'n_neighbors': array([ 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70,
75, 80, 85,
        90, 95])}]
```

```
[62]: GridSearchCV(cv=5, estimator=KNeighborsRegressor(),
                   param_grid=[{'n_neighbors': array([ 5, 10, 15, 20, 25, 30, 35, 40,
45, 50, 55, 60, 65, 70, 75, 80, 85,
       90, 95])}],
                   scoring='neg_mean_absolute_error')
```

Лучшая модель:

```
[63]: regr_gs.best_estimator_
```

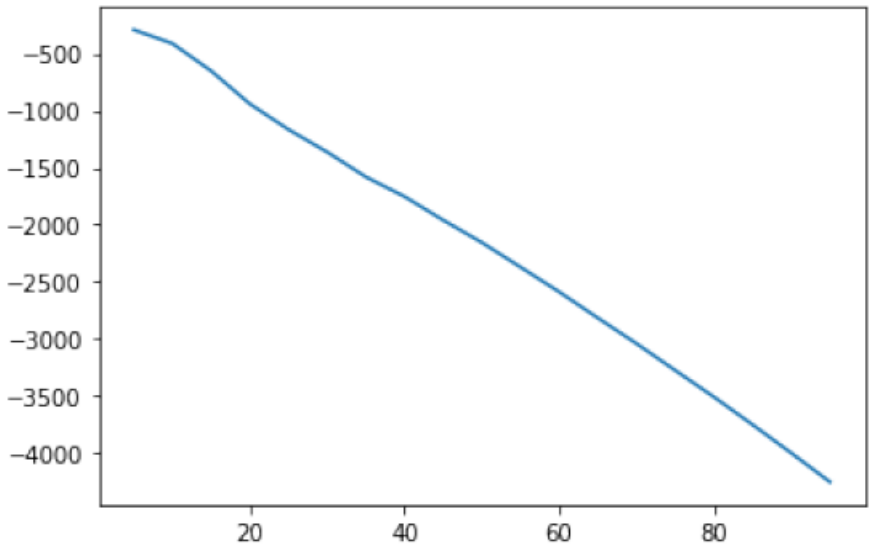
```
[63]: KNeighborsRegressor()
```

Лучшие параметры:

```
[64]: regr_gs.best_params_
```

```
[64]: {'n_neighbors': 5}
```

Зависимость качества:



Построим оптимальную модель:

```
[67]: regr_models_grid = {'KNN_20':KNeighborsRegressor(n_neighbors=20),
                          str('KNN_'+regr_gs_best_params_txt):regr_gs.best_estimator_}
```

```
[68]: for model_name, model in regr_models_grid.items():
          regr_train_model(model_name, model, regrMetricLogger)
```

```
KNN_20    MAE=769.6, MSE=5081904.089, R2=0.924
KNN_5     MAE=263.517, MSE=417700.95, R2=0.994
```

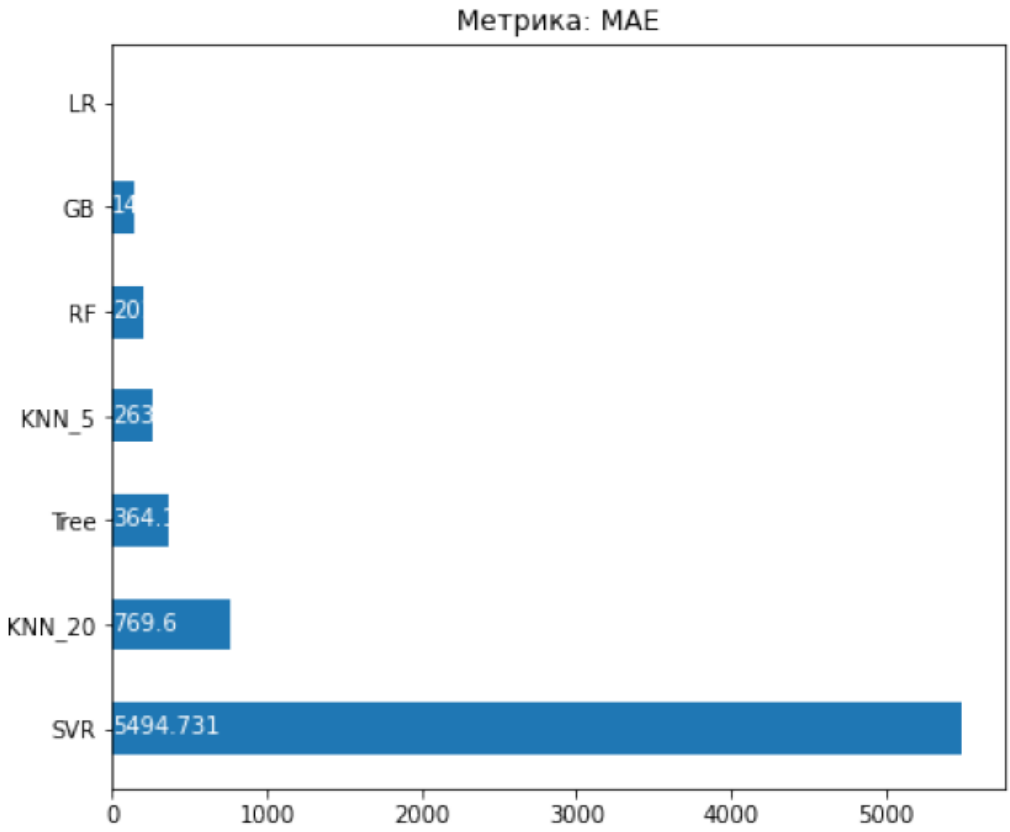
Оценка качества

Сравним все метрики:

```
[69]: regr_metrics = regrMetricLogger.df['metric'].unique()
```

Метрика Mean Absolute Error:

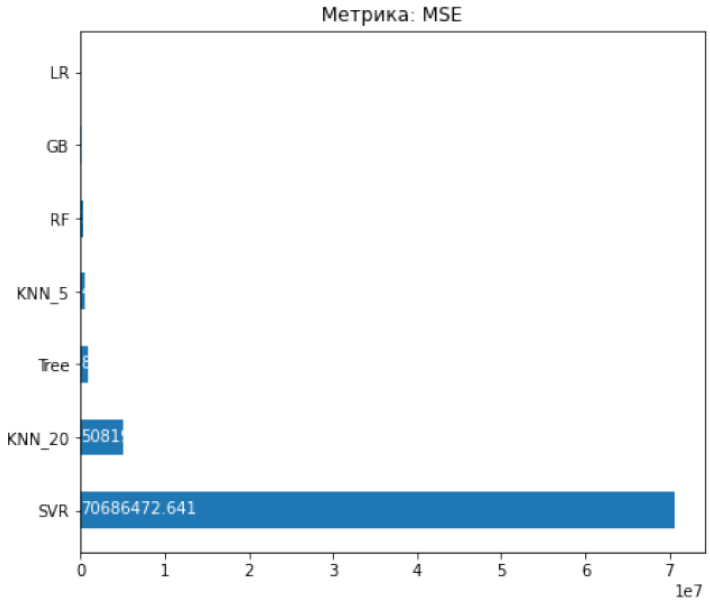
```
[70]: regrMetricLogger.plot('Метрика: ' + 'MAE', 'MAE', ascending=False, figsize=(7, 6))
```



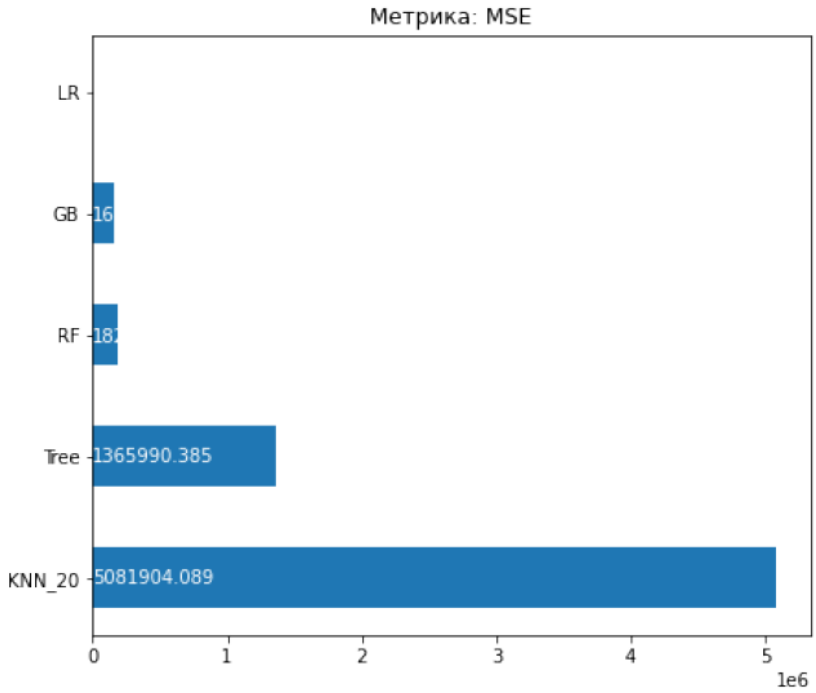
Чем ближе значение метрики к 0, тем качественнее модель. Лучший результат показывает модель линейной регрессии, худший - модель опорных векторов.

Метрика Mean Squarred Error:

```
[75]: regrMetricLogger.plot('Метрика: ' + 'MSE', 'MSE', ascending=False, figsize=(7, 6))
```



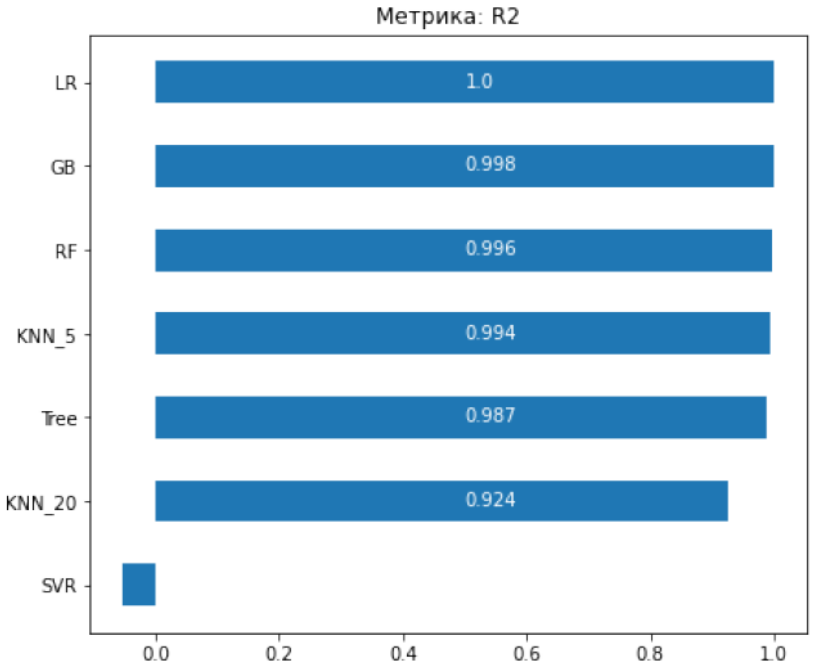
[76]:
```
regrMetricLogger_no_svr.plot('Метрика: ' + 'MSE', 'MSE', ascending=False,
↪figsize=(7, 6))
```



Чем ближе значение метрики к нулю, тем модель более качественна. Модель линейной регрессии выигрывает по качеству у остальных. Модель SVR обладает наихудшем качеством.

Метрика R2:

```
[77]: regrMetricLogger.plot('Метрика: ' + 'R2', 'R2', ascending=True, figsize=(7, 6))
```



Исходя из метрики R2-score – наихудший результат показывает модель опорных векторов. Лучшими моделями можно считать модели линейной регрессии, градиентного бустинга, случайного леса и дерева решений. Выходит, что наиболее качественной моделью регрессии можно считать модель линейной регрессии.

ЗАКЛЮЧЕНИЕ

В работе был проведён разведочный анализ данных с обработкой данных с неинформативными признаками, пропусков и модификацией структуры и самих данных. Также было проведено кодирование категориальных признаков, масштабирование данных и сравнение масштабированных данных с исходными. Был выполнен корреляционный анализ и на его основании были выбраны модели для решения задачи регрессии. Исходные данные были разделены на тестовую и обучающую выборку, на основе этих выборок были обучены выбранные модели. Также была построена наиболее оптимальная модель. Все модели подверглись сравнению для определения наилучшего качества решения задачи регрессии, для этого использовались несколько метрик регрессии.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Car Data. URL: https://www.kaggle.com/datasets/goyalshalini93/car-data (дата обращения: 22.12.2024).

2. Matplotlib – visualization via Python. URL: https://matplotlib.org/ (дата обращения: 22.12.2024).

3. sckit-learn: machine learning in Python. URL: https://scikit-learn.org/stable/ (дата обращения: 22.12.2024).