Московский авиационный институт

(Национальный исследовательский университет)

Лабораторная работа № 1

Тема: Данные и проблемы в них

По курсу: Машинное обучение

Студент: Подгорная Виолетта

Группа: М80-304Б-17

Дата:

Оценка:

Москва, 2020

Задание: необходимо сформировать два набора данных для приложений машинного обучения. Первый датасет должен представлять из себя табличный набор данных для задачи классификации. Второй датасет должен быть отличен от первого, и может представлять из себя набор изображений, корпус документов, другой табличный датасет или датасет из соревнования Kaggle, предназначенный для решения интересующей вас задачи машинного обучения. Необходимо провести анализ обоих наборов данных, поставить решаемую вами задачу, определить признаки необходимые для решения задачи, в случае необходимости заняться генерацией новых признаков, устранением проблем в данных, визуализировать распределение и зависимость целевого признака от выбранных признаков.

Датасеты:

1. heart.csv – для задачи классификации.

Колонки:

- age
- sex
- chest pain type (4 values)
- resting blood pressure
- serum cholestoral in mg/dl
- fasting blood sugar > 120 mg/dl
- resting electrocardiographic results (values 0,1,2)
- maximum heart rate achieved
- exercise induced angina
- oldpeak = ST depression induced by exercise relative to rest
- the slope of the peak exercise ST segment
- number of major vessels (0-3) colored by flourosopy
- thal: 3 = normal; 6 = fixed defect; 7 = reversable defect

Задача: необходимо определить, кто из людей болен.

2. CarPrice.csv — для задачи регрессии.

Колонки:

- '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'

Задача: необходимо определить стоимость машины

Задача классификации

Анализ и подготовка данных

Для начала посмотрим общую информацию по датасету:

```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
age         303 non-null int64
sex         303 non-null int64
cp          303 non-null int64
trestbps    303 non-null int64
chol        303 non-null int64
fbs         303 non-null int64
restecg     303 non-null int64
thalach     303 non-null int64
exang       303 non-null int64
oldpeak     303 non-null float64
slope       303 non-null int64
ca          303 non-null int64
thal        303 non-null int64
target      303 non-null int64
dtypes: float64(1), int64(13)
memory usage: 33.2 KB
```

Замечаем, что все данные датасета имеют численные значения, поэтому степень значимости признаков будем определять по графикам зависимости и значениям корреляции.

Теперь проверим, есть ли у нас незаполненные поля в датасете:

```
1 df.isnull().sum()
```

```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

Пустых полей в датасете нет.

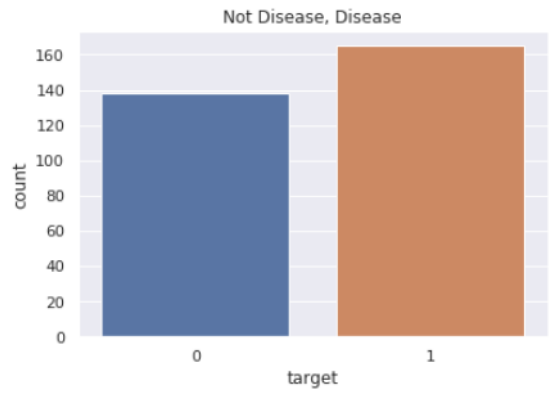
Далее посмотрим, насколько сбалансирован датасет:

```
1 df["target"].value_counts()
```

```
1    165
0    138
Name: target, dtype: int64
```

```
1 sns.countplot(df.target)
2 plt.title("Not Disease, Disease")
```

```
Text(0.5, 1.0, 'Not Disease, Disease')
```



Можно сделать вывод что датасет достаточно сбалансирован.

Обратим внимание на численную информацию по датасету:

```
df.describe()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	30
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.326733	1.039604	1.399340	0.729373	
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.161075	0.616226	1.022606	
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.000000	1.000000	0.000000	
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	0.800000	1.000000	0.000000	
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.000000	1.600000	2.000000	1.000000	
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000	

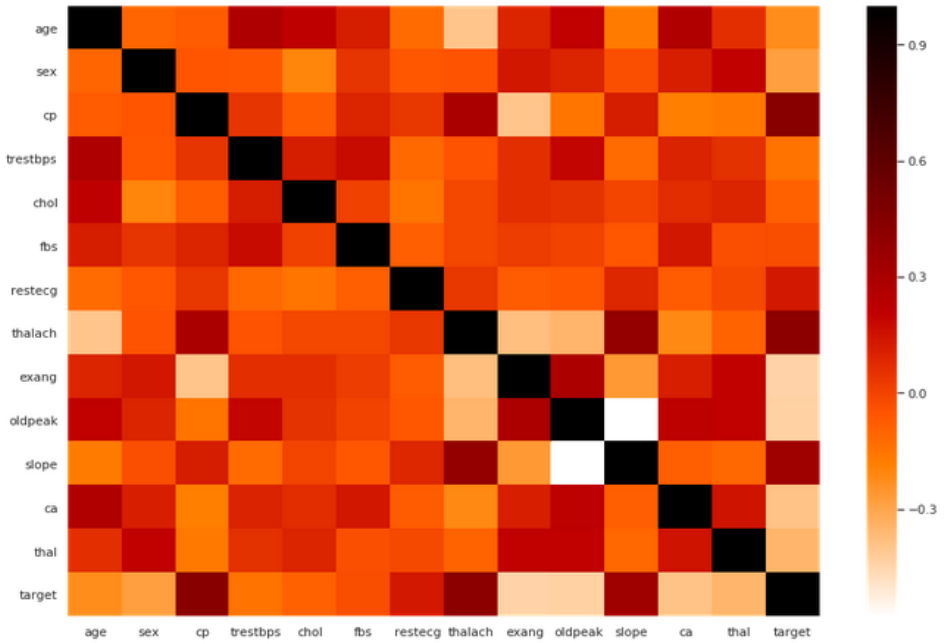
Заметим, что данные датасета практически не нуждаются в нормализации за исключением некоторых признаков.

Обработка данных

Сравним корреляцию признаков с таргетом:

```
fig = plt.figure(figsize=(15,10))
sns.heatmap(df.corr(), cmap="gist_heat_r")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fee8018f898>
```



Можно заметить, что наименьшая корреляция с признаками 'chol', 'fbs'. Поэтому удаляем их:

```
not_need = ['chol', 'fbs']

x = df.drop(not_need, axis=1)
x.head()
```

	age	sex	cp	trestbps	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	0	150	0	2.3	0	0	1	1
1	37	1	2	130	1	187	0	3.5	0	0	2	1
2	41	0	1	130	0	172	0	1.4	2	0	2	1
3	56	1	1	120	1	178	0	0.8	2	0	2	1
4	57	0	0	120	1	163	1	0.6	2	0	2	1

Далее проверим некоторые зависимости признаков между собой, чтобы графически определить, что нам может пригодиться.

1. Зависимость максимальной доли здоровья от возраста и диагноза:

```
plt.scatter(x=x.age[x.target==1], y=x.thalach[(x.target==1)], c="re
plt.scatter(x=x.age[x.target==0], y=x.thalach[(x.target==0)])
plt.legend(["Disease", "Not Disease"])
plt.xlabel("Age")
plt.ylabel("Maximum Heart Rate")
plt.show()
```

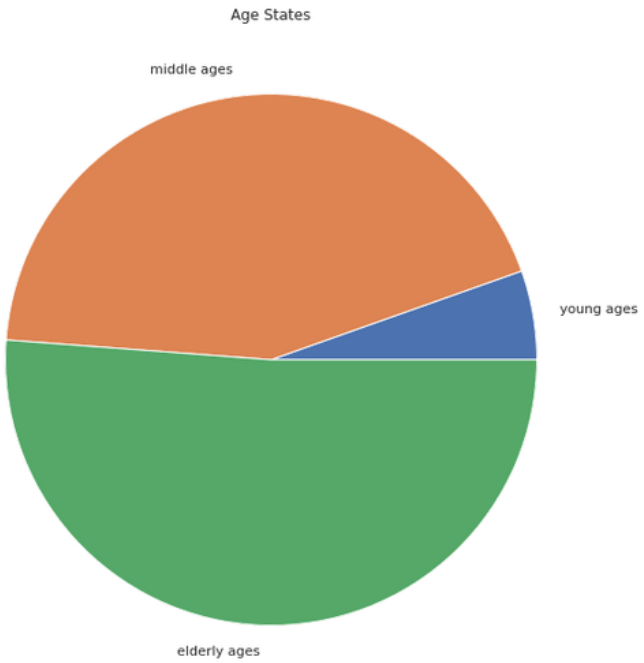
Видно, что некая зависимость прослеживается.

2. Распределение по возрасту:

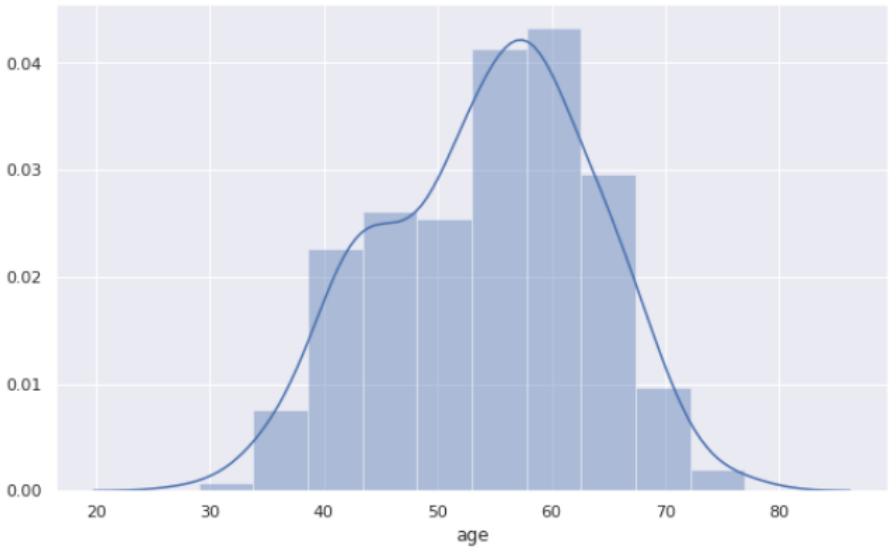
```
young_ages = x[(x.age>=29) & (x.age<40)]
middle_ages = x[(x.age>=40) & (x.age<55)]
elderly_ages = x[(x.age>55)]
print('Young Ages :',len(young_ages))
print('Middle Ages :',len(middle_ages))
print('Elderly Ages :',len(elderly_ages))
```

```
Young Ages : 16
Middle Ages : 128
Elderly Ages : 151
```

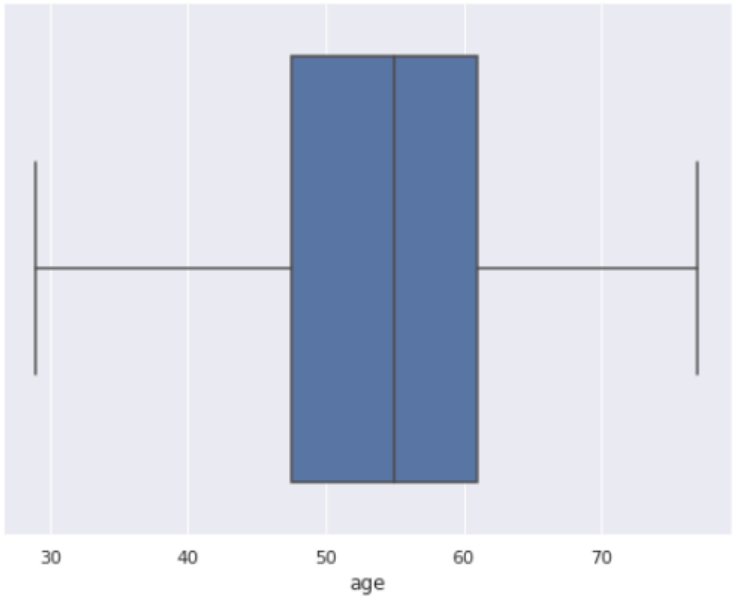
```
  #we can see in pie.

plt.figure(figsize = (10,10))
plt.pie([len(young_ages),len(middle_ages),len(elderly_ages)],labels=['young ages','middle ages','elderly ages']
plt.title('Age States')
plt.show()
```




```
f, ax = plt.subplots(figsize=(10,6))
ax = sns.distplot(x['age'], bins=10)
plt.show()
```



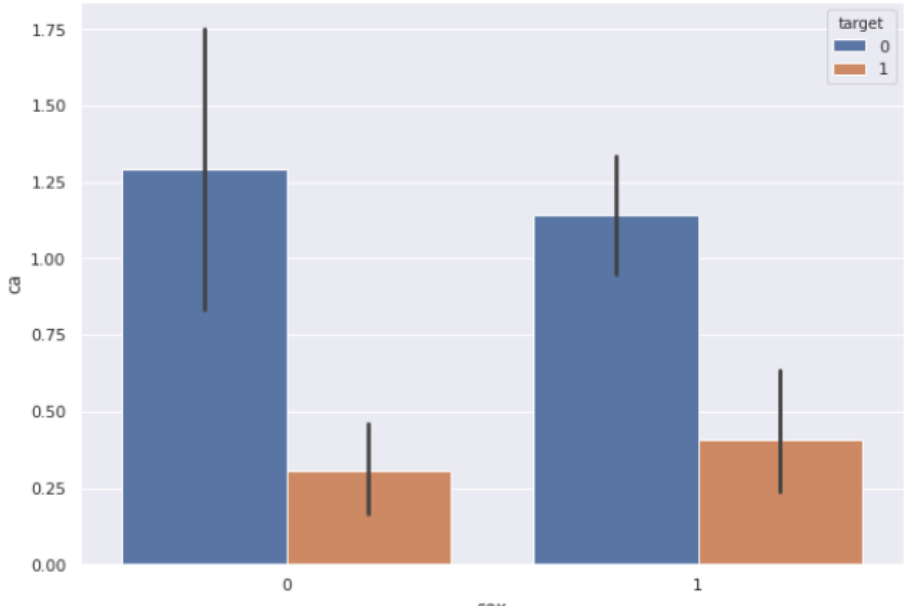
```
f, ax = plt.subplots(figsize=(8, 6))
sns.boxplot(x=x["age"])
plt.show()
```



3. Некоторые показатели по здоровью:

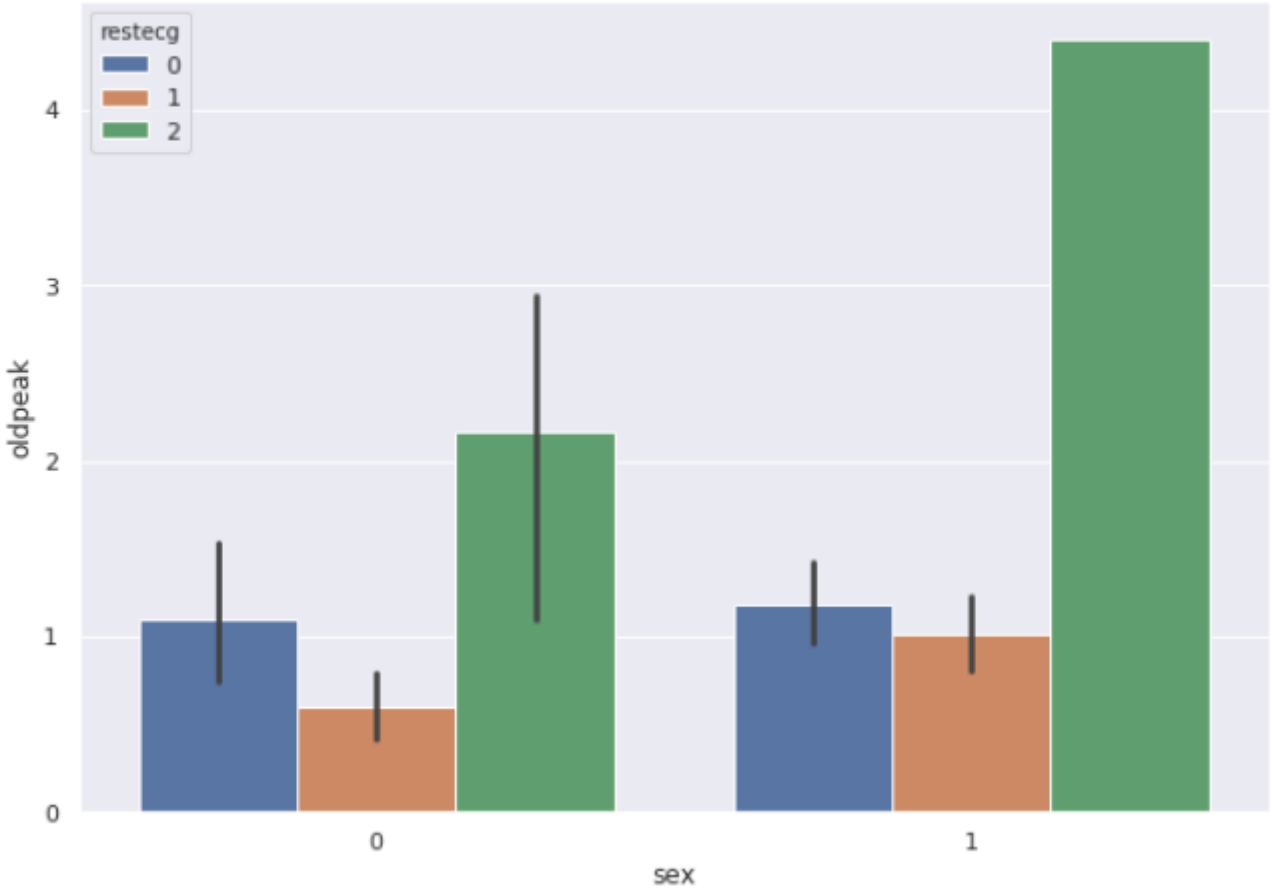
```
plt.figure(figsize=(10,7))
sns.barplot(x="sex",y = 'ca',hue = 'target',data=x)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fee7bee5d30>
```



```
plt.figure(figsize=(10,7))
sns.barplot(x="sex",y = 'oldpeak',hue = 'restecg',data=x)
```

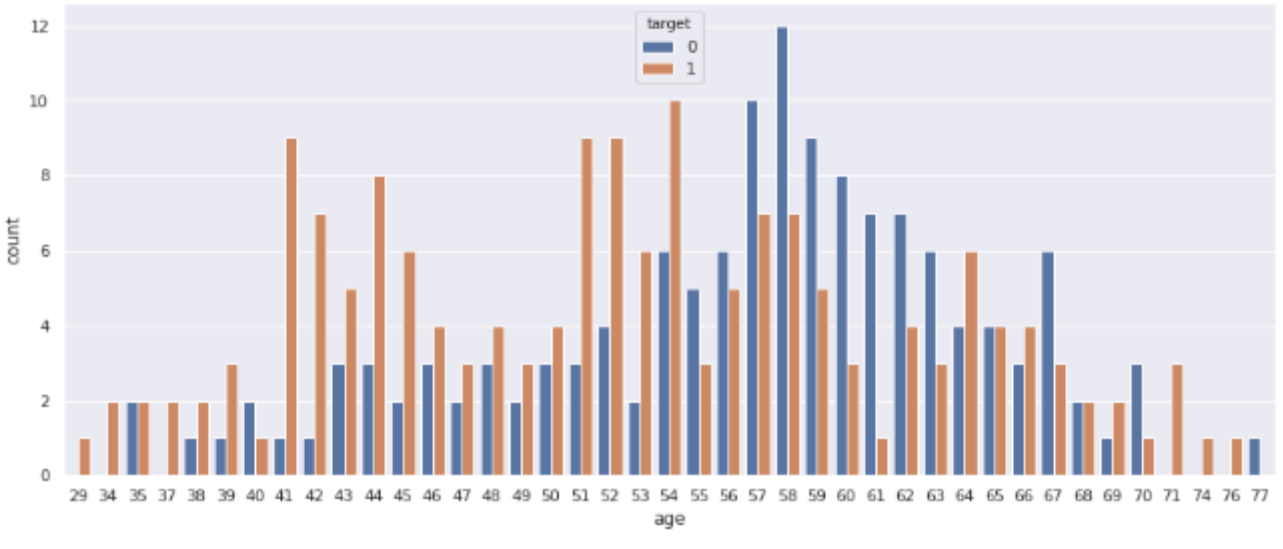
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fee7bead080>
```



4. Зависимость пола и диагноза:

```
sns.countplot(x.target,hue=x.sex)
plt.xlabel('Target')
plt.ylabel('Count')
plt.title('Target & Sex Counter')
plt.show()
```


```
plt.figure(figsize=(15,6))
sns.countplot(x='age',data = x, hue = 'target')
plt.show()
```

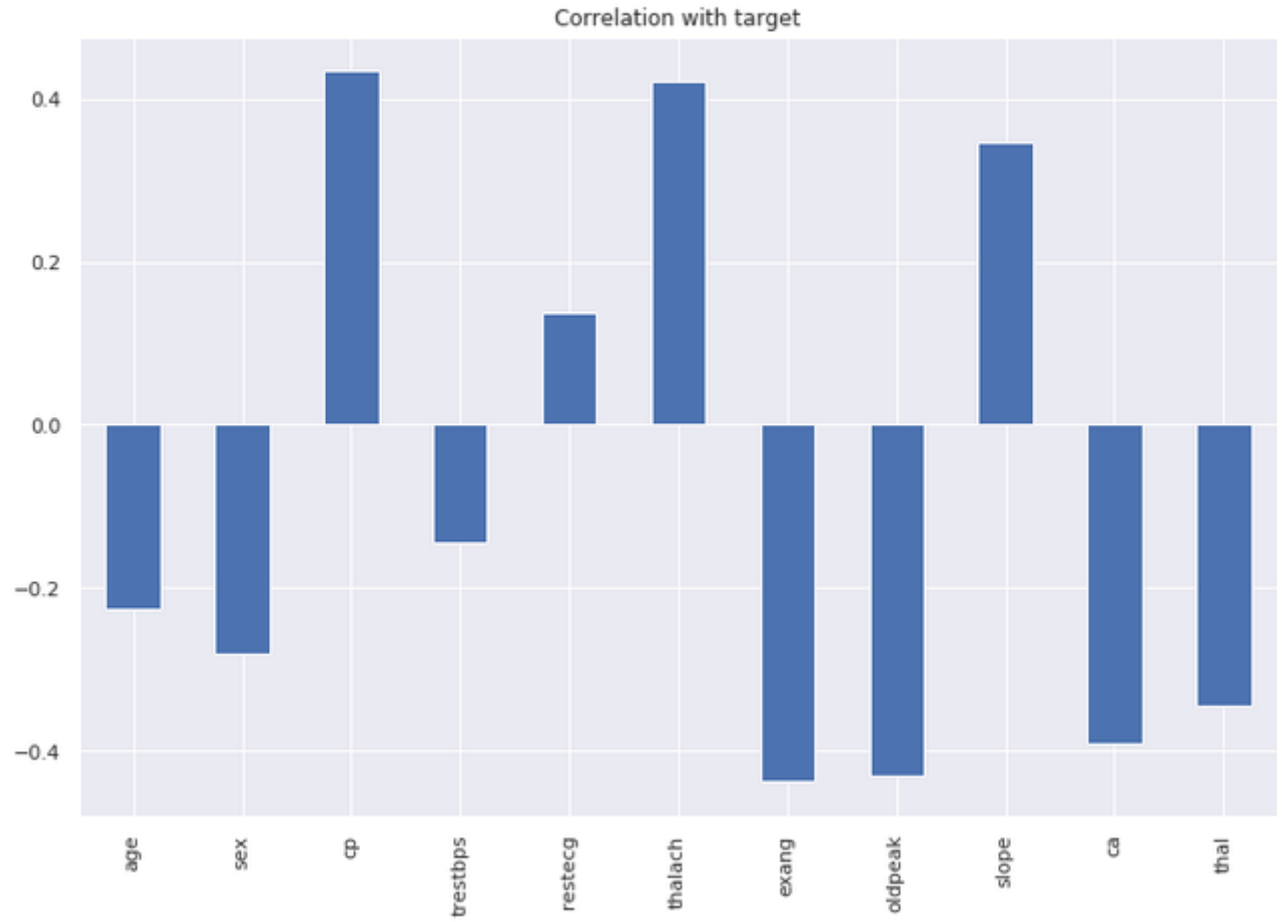


5. Зависимости между всеми признаками определим с помощью sns.pairplot().

6. Далее еще раз взглянем на корреляцию и сопоставим с таблицей зависимостей:

```
1 x.drop('target', axis=1).corrwith(df.target).plot(kind='bar', grid=
2                                                   title="Correlati
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fee77ec6eb8>
```



Выводы:

- Нет почти ни одного признака со строгой корреляцией с таргетом.
- Нет корреляции с 'trestbps', 'restecg'.
- Рассмотренные показатели по здоровью находятся в неплохой зависимости с таргетом.
- У людей с не подтвержденным диагнозом более высокий уровень здоровья.

По итогу проведенного анализа избавимся еще от двух признаков по причине низкой корреляции и слабо выраженной зависимости:

```
x = x.drop(['trestbps', 'restecg'], axis=1)
```

```
x.head()
```

	age	sex	cp	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	150	0	2.3	0	0	1	1
1	37	1	2	187	0	3.5	0	0	2	1
2	41	0	1	172	0	1.4	2	0	2	1
3	56	1	1	178	0	0.8	2	0	2	1
4	57	0	0	163	1	0.6	2	0	2	1

Также не забудем нормировать оставшийся «большой» признак:

```
x['thalach'] = (x['thalach'] - x['thalach'].mean()) / x['thalach'].std()
x.head()
```

	age	sex	cp	thalach	exang	oldpeak	slope	ca	thal
0	63	1	3	0.015417	0	2.3	0	0	1
1	37	1	2	1.630774	0	3.5	0	0	2
2	41	0	1	0.975900	0	1.4	2	0	2
3	56	1	1	1.237849	0	0.8	2	0	2
4	57	0	0	0.582975	1	0.6	2	0	2

Задача регрессии

Анализ и подготовка данных

Посмотрим общую информацию по датасету (количество незаполненных полей и различную статистику):

```
1 df.describe()
```

	car_ID	symboling	wheelbase	carlength	carwidth	carheight	curbweight	er
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	20
mean	103.000000	0.834146	98.756585	174.049268	65.907805	53.724878	2555.565854	12
std	59.322565	1.245307	6.021776	12.337289	2.145204	2.443522	520.680204	4
min	1.000000	-2.000000	86.600000	141.100000	60.300000	47.800000	1488.000000	6
25%	52.000000	0.000000	94.500000	166.300000	64.100000	52.000000	2145.000000	9
50%	103.000000	1.000000	97.000000	173.200000	65.500000	54.100000	2414.000000	12
75%	154.000000	2.000000	102.400000	183.100000	66.900000	55.500000	2935.000000	14
max	205.000000	3.000000	120.900000	208.100000	72.300000	59.800000	4066.000000	32

```
1 df.isnull().sum()
```

```
car_ID              0
symboling           0
CarName             0
fueltype            0
aspiration          0
doornumber          0
carbody             0
drivewheel          0
enginelocation      0
wheelbase           0
carlength           0
carwidth            0
carheight           0
curbweight          0
enginetype          0
cylindernumber      0
enginesize          0
fuelsystem          0
boreratio           0
stroke              0
compressionratio    0
horsepower          0
peakrpm             0
citympg             0
highwaympg          0
price               0
dtype: int64
```

```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
car_ID              205 non-null int64
symboling           205 non-null int64
CarName             205 non-null object
fueltype            205 non-null object
aspiration          205 non-null object
doornumber          205 non-null object
carbody             205 non-null object
drivewheel          205 non-null object
enginelocation      205 non-null object
wheelbase           205 non-null float64
carlength           205 non-null float64
carwidth            205 non-null float64
carheight           205 non-null float64
curbweight          205 non-null int64
enginetype          205 non-null object
cylindernumber      205 non-null object
enginesize          205 non-null int64
fuelsystem          205 non-null object
boreratio           205 non-null float64
stroke              205 non-null float64
compressionratio    205 non-null float64
horsepower          205 non-null int64
peakrpm             205 non-null int64
citympg             205 non-null int64
highwaympg          205 non-null int64
price               205 non-null float64
dtypes: float64(8), int64(8), object(10)
memory usage: 41.7+ KB
```

Замечаем, что нет незаполненных полей, но есть признаки типа object. Отметим, что ряд признаков в реальной жизни очень сильно влияет на стоимость автомобиля (например, марка).

Обработка данных

1. Поработаем с маркой машины. Для начала определим, какие марки представлены в датасете:

```
1 df.CarName.unique()
```

```
array(['alfa-romero giulia', 'alfa-romero stelvio',
       'alfa-romero Quadrifoglio', 'audi 100 ls', 'audi 100ls',
       'audi fox', 'audi 5000', 'audi 4000', 'audi 5000s (diesel)',
       'bmw 320i', 'bmw x1', 'bmw x3', 'bmw z4', 'bmw x4', 'bmw x5',
       'chevrolet impala', 'chevrolet monte carlo', 'chevrolet vega 2300',
       'dodge rampage', 'dodge challenger se', 'dodge d200',
       'dodge monaco (sw)', 'dodge colt hardtop', 'dodge colt (sw)',
       'dodge coronet custom', 'dodge dart custom',
       'dodge coronet custom (sw)', 'honda civic', 'honda civic cvcc',
       'honda accord cvcc', 'honda accord lx', 'honda civic 1500 gl',
       'honda accord', 'honda civic 1300', 'honda prelude',
       'honda civic (auto)', 'isuzu MU-X', 'isuzu D-Max ',
       'isuzu D-Max V-Cross', 'jaguar xj', 'jaguar xf', 'jaguar xk',
       'maxda rx3', 'maxda glc deluxe', 'mazda rx2 coupe', 'mazda rx-4',
       'mazda glc deluxe', 'mazda 626', 'mazda glc', 'mazda rx-7 gs',
       'mazda glc 4', 'mazda glc custom l', 'mazda glc custom',
       'buick electra 225 custom', 'buick century luxus (sw)',
       'buick century', 'buick skyhawk', 'buick opel isuzu deluxe',
       'buick skylark', 'buick century special',
       'buick regal sport coupe (turbo)', 'mercury cougar',
       'mitsubishi mirage', 'mitsubishi lancer', 'mitsubishi outlander',
       'mitsubishi g4', 'mitsubishi mirage g4', 'mitsubishi montero',
       'mitsubishi pajero', 'Nissan versa', 'nissan gt-r', 'nissan rogue',
       'nissan latio', 'nissan titan', 'nissan leaf', 'nissan juke',
       'nissan note', 'nissan clipper', 'nissan nv200', 'nissan dayz',
       'nissan fuga', 'nissan otti', 'nissan teana', 'nissan kicks',
       'peugeot 504', 'peugeot 304', 'peugeot 504 (sw)', 'peugeot 604sl',
       'peugeot 505s turbo diesel', 'plymouth fury iii',
       'plymouth cricket', 'plymouth satellite custom (sw)',
       'plymouth fury gran sedan', 'plymouth valiant', 'plymouth duster',
       'porsche macan', 'porcshce panamera', 'porsche cayenne',
       'porsche boxter', 'renault 12tl', 'renault 5 gtl', 'saab 99e',
       'saab 99le', 'saab 99gle', 'subaru', 'subaru dl', 'subaru brz',
       'subaru baja', 'subaru r1', 'subaru r2', 'subaru trezia',
       'subaru tribeca', 'toyota corona mark ii', 'toyota corona',
       'toyota corolla 1200', 'toyota corona hardtop',
       'toyota corolla 1600 (sw)', 'toyota carina', 'toyota mark ii',
       'toyota corolla', 'toyota corolla liftback',
       'toyota celica gt liftback', 'toyota corolla tercel',
       'toyota corona liftback', 'toyota starlet', 'toyota tercel',
       'toyota cressida', 'toyota celica gt', 'toyouta tercel',
       'vokswagen rabbit', 'volkswagen 1131 deluxe sedan',
       'volkswagen model 111', 'volkswagen type 3', 'volkswagen 411 (sw)',
       'volkswagen super beetle', 'volkswagen dasher', 'vw dasher',
       'vw rabbit', 'volkswagen rabbit', 'volkswagen rabbit custom',
       'volvo 145e (sw)', 'volvo 144ea', 'volvo 244dl', 'volvo 245',
       'volvo 264gl', 'volvo diesel', 'volvo 246'], dtype=object)
```

Можно заметить, что у нас есть ряд марок с их моделями. Немного упростим задачу: будем считать важными именно марки, а не конкретные модели машин. Приведем данный признак к численным значениям:

```
names = ['alfa', 'audi', 'bmw', 'chevrolet', 'dodge', 'honda', 'isuzu', 'jaguar', 'maxda
        'mercury', 'mitsubishi', 'Nissan', 'nissan', 'peugeot', 'plymouth', 'porsche', '
        'subaru', 'toyota', 'toyouta', 'vokswagen', 'volkswagen', 'vw', 'volvo']

un_names = df.CarName.unique()
x = df.copy()

for i, name in enumerate(names):
    for un_name in un_names:
        if name in un_name:
            x['CarName'][x['CarName'] == un_name] = i

x.CarName.unique()
```

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18,
       19, 20, 21, 22, 23, 24, 25, 26, 27], dtype=object)
```

2. У нас присутствует колонка с ID машины. Она не окажет никакого влияния на цену, поэтому удаляем:

```
x = x.drop(['car_ID'],axis=1)
x.head()
```

	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	carlength	...
0	3	0	gas	std	two	convertible	rwd	front	88.6	168.8	...
1	3	0	gas	std	two	convertible	rwd	front	88.6	168.8	...
2	1	0	gas	std	two	hatchback	rwd	front	94.5	171.2	...
3	2	1	gas	std	four	sedan	fwd	front	99.8	176.6	...
4	2	1	gas	std	four	sedan	4wd	front	99.4	176.6	...

5 rows × 25 columns

3. В реальной жизни большое влияние на ценообразование машины оказывают такие признаки, как тип топлива, расположение мотора, тип кузова. Поэтому определим уникальные значения данных признаков и заменим их численными эквивалентами:

```
x.fueltype.unique()
```

```
array(['gas', 'diesel'], dtype=object)
```

```
x['fueltype'].replace(x.fueltype.unique(), [1, 0], inplace=True)
x.head()
```

	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	carlength	...
0	3	0	1	std	two	convertible	rwd	front	88.6	168.8	...
1	3	0	1	std	two	convertible	rwd	front	88.6	168.8	...
2	1	0	1	std	two	hatchback	rwd	front	94.5	171.2	...
3	2	1	1	std	four	sedan	fwd	front	99.8	176.6	...
4	2	1	1	std	four	sedan	4wd	front	99.4	176.6	...

5 rows × 25 columns

```
x.enginelocation.unique()
```

```
array(['front', 'rear'], dtype=object)
```

```
x['enginelocation'].replace(x.enginelocation.unique(), [1, 0], inplace=True)
x.head()
```

	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	carlength	...
0	3	0	1	std	two	convertible	rwd	1	88.6	168.8	...
1	3	0	1	std	two	convertible	rwd	1	88.6	168.8	...
2	1	0	1	std	two	hatchback	rwd	1	94.5	171.2	...
3	2	1	1	std	four	sedan	fwd	1	99.8	176.6	...
4	2	1	1	std	four	sedan	4wd	1	99.4	176.6	...

5 rows × 25 columns

```
x.carbody.unique()
```

```
array(['convertible', 'hatchback', 'sedan', 'wagon', 'hardtop'],
      dtype=object)
```

```
x['carbody'].replace(x.carbody.unique(), [i for i in range(len(x.carbody.unique()))], in
x.head()
```

	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	carlength	...	en
0	3	0	1	std	two	0	rwd	1	88.6	168.8	...	
1	3	0	1	std	two	0	rwd	1	88.6	168.8	...	
2	1	0	1	std	two	1	rwd	1	94.5	171.2	...	
3	2	1	1	std	four	2	fwd	1	99.8	176.6	...	
4	2	1	1	std	four	2	4wd	1	99.4	176.6	...	

4. Замечаем, что забыли изменить тип признака с названием машины, так же определяем ненужные строковые признаки и удаляем их:

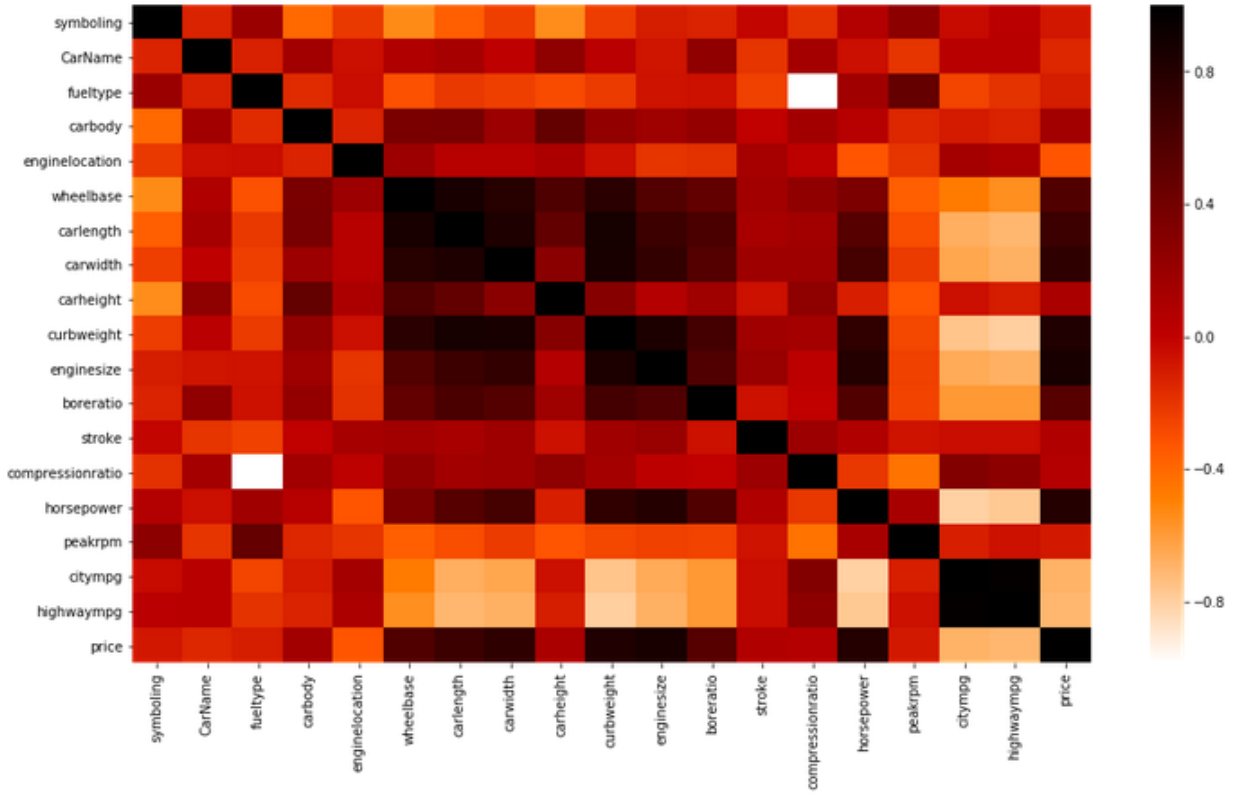
```
x['CarName'] = x['CarName'].astype('int32')

not_need = ['aspiration', 'doornumber', 'drivewheel', 'enginetype', 'cylindernumber', 'f
```

```
x = x.drop(not_need, axis=1)
x.head()
```

	symboling	CarName	fueltype	carbody	enginelocation	wheelbase	carlength	carwidth	carheight	curbweight	enginesize
0	3	0	1	0	1	88.6	168.8	64.1	48.8	2548	13(
1	3	0	1	0	1	88.6	168.8	64.1	48.8	2548	13(
2	1	0	1	1	1	94.5	171.2	65.5	52.4	2823	15
3	2	1	1	2	1	99.8	176.6	66.2	54.3	2337	10
4	2	1	1	2	1	99.4	176.6	66.4	54.3	2824	13(

5. Теперь посмотрим на корреляцию таргета с признаками:

```
plt.figure(figsize=(16,9))
ax=sns.heatmap(x.corr(),cmap='gist_heat_r')
```



Несколько признаков имеют достаточно низкую корреляцию с ценой, поэтому удаляем их:

```
# symboling, stroke, compressionratio, peakrpm
x = x.drop(['symboling', 'stroke','compressionratio','peakrpm'],axis=1)
x.head()
```

	CarName	fueltype	carbody	enginelocation	wheelbase	carlength	carwidth	carheight	curbweight	enginesize	boreratio
0	0	1	0	1	88.6	168.8	64.1	48.8	2548	130	3.47
1	0	1	0	1	88.6	168.8	64.1	48.8	2548	130	3.47
2	0	1	1	1	94.5	171.2	65.5	52.4	2823	152	2.68
3	1	1	2	1	99.8	176.6	66.2	54.3	2337	109	3.19
4	1	1	2	1	99.4	176.6	66.4	54.3	2824	136	3.19

Выводы:

- Получили относительно небольшой набор признаков.
- У ряда признаков прослеживается достаточно большая корреляция с ценой.
- Нормировку проводить не будем, так как признаки достаточно специфичны и, скорее всего, важны именно в том виде, в котором представлены.