Московский государственный технический университет им. Н.Э. Баумана
Кафедра «Системы обработки информации и управления»

Рубежный контроль №1
по дисциплине
«Методы машинного обучения»
Вариант №1

Выполнил:
студент группы ИУ5-23М
Богомолов Д. Н.

Москва — 2020 г.

Богомолов Дмитрий ИУ5-23М
РК 1 по дисциплине Методы машинного обучения
Вариант 1 (задача 1, boxplot, датасет boston)

[0]:
```
import numpy as np
import pandas as pd
import seaborn as  sns
import matplotlib.pyplot as plt
```

[32]:
```
from sklearn.datasets import load_boston
X, y = load_boston(return_X_y=True)
print(X.shape)
```
```
(506, 13)
```

```
(506,  13)
```

[32]:
```
(506, 13)
```

Создание Pandas Dataframe

[0]:
```
def make_dataframe(ds_function):
    ds = ds_function()
    df = pd.DataFrame(data= np.c_[ds["data"], ds["target"]],
                      columns= list(ds["feature_names"]) + ["target"])
    return  df
```

[34]:
```
data  =  make_dataframe(load_boston)    #Создание  датафрейма
data.head()                             #Вывод первых 5  строк
```

[34]:
```
      CRIM    ZN  INDUS  CHAS    NOX  ...    TAX  PTRATIO       B  LSTAT
↪target
0  0.00632  18.0   2.31   0.0  0.538  ...  296.0     15.3  396.90   4.98
↪   24.0
1  0.02731   0.0   7.07   0.0  0.469  ...  242.0     17.8  396.90   9.14
↪   21.6
2  0.02729   0.0   7.07   0.0  0.469  ...  242.0     17.8  392.83   4.03
↪   34.7
3  0.03237   0.0   2.18   0.0  0.458  ...  222.0     18.7  394.63   2.94
↪   33.4
4  0.06905   0.0   2.18   0.0  0.458  ...  222.0     18.7  396.90   5.33
↪   36.2

[5 rows x 14 columns]
```

Поиск пустых значений в колонках

[35]:
```
for col in data.columns:
    # Количество пустых значений
    temp_null_count = data[data[col].isnull()].shape[0]
    print("{} - {}".format(col,  temp_null_count))
    #Пустых значений не обнаружено
```

```
CRIM - 0
ZN - 0
INDUS - 0
CHAS - 0
NOX - 0
RM - 0
AGE - 0
DIS - 0
RAD - 0
TAX - 0
PTRATIO - 0
B - 0
LSTAT - 0
target - 0
```

[36]:
```
data.describe() #Описательные статистики
```

[36]:

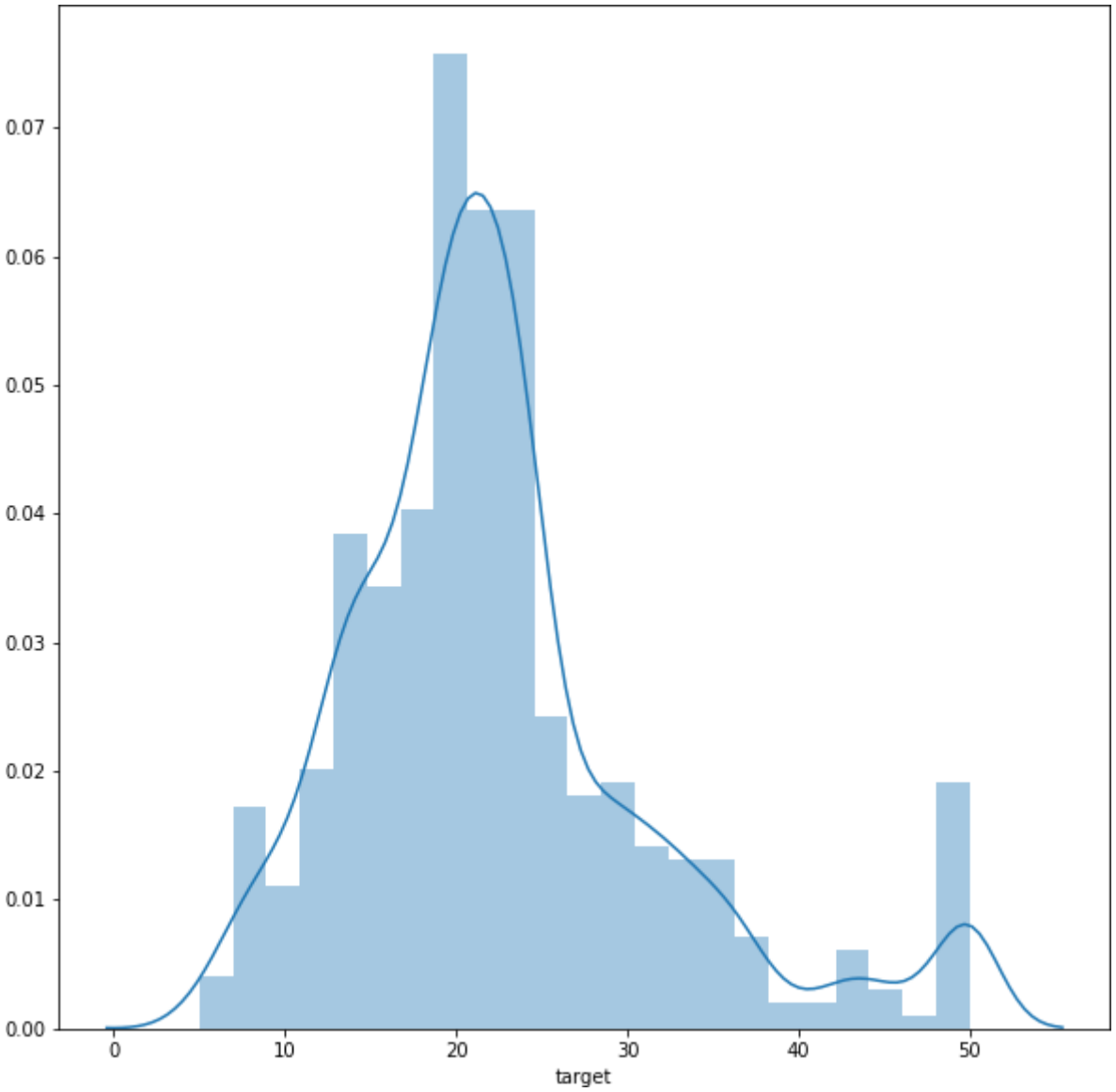
	CRIM	ZN	INDUS	…	B	LSTAT	target
count	506.000000	506.000000	506.000000	…	506.000000	506.000000	506.000000
mean	3.613524	11.363636	11.136779	…	356.674032	12.653063	22.532806
std	8.601545	23.322453	6.860353	…	91.294864	7.141062	9.197104
min	0.006320	0.000000	0.460000	…	0.320000	1.730000	5.000000
25%	0.082045	0.000000	5.190000	…	375.377500	6.950000	17.025000
50%	0.256510	0.000000	9.690000	…	391.440000	11.360000	21.200000
75%	3.677083	12.500000	18.100000	…	396.225000	16.955000	25.000000
max	88.976200	100.000000	27.740000	…	396.900000	37.970000	50.000000

[8 rows x 14 columns]

Распределениие значений целевого признака

[37]:
```
fig, ax = plt.subplots(figsize=(10,10))
sns.distplot(data["target"])
```

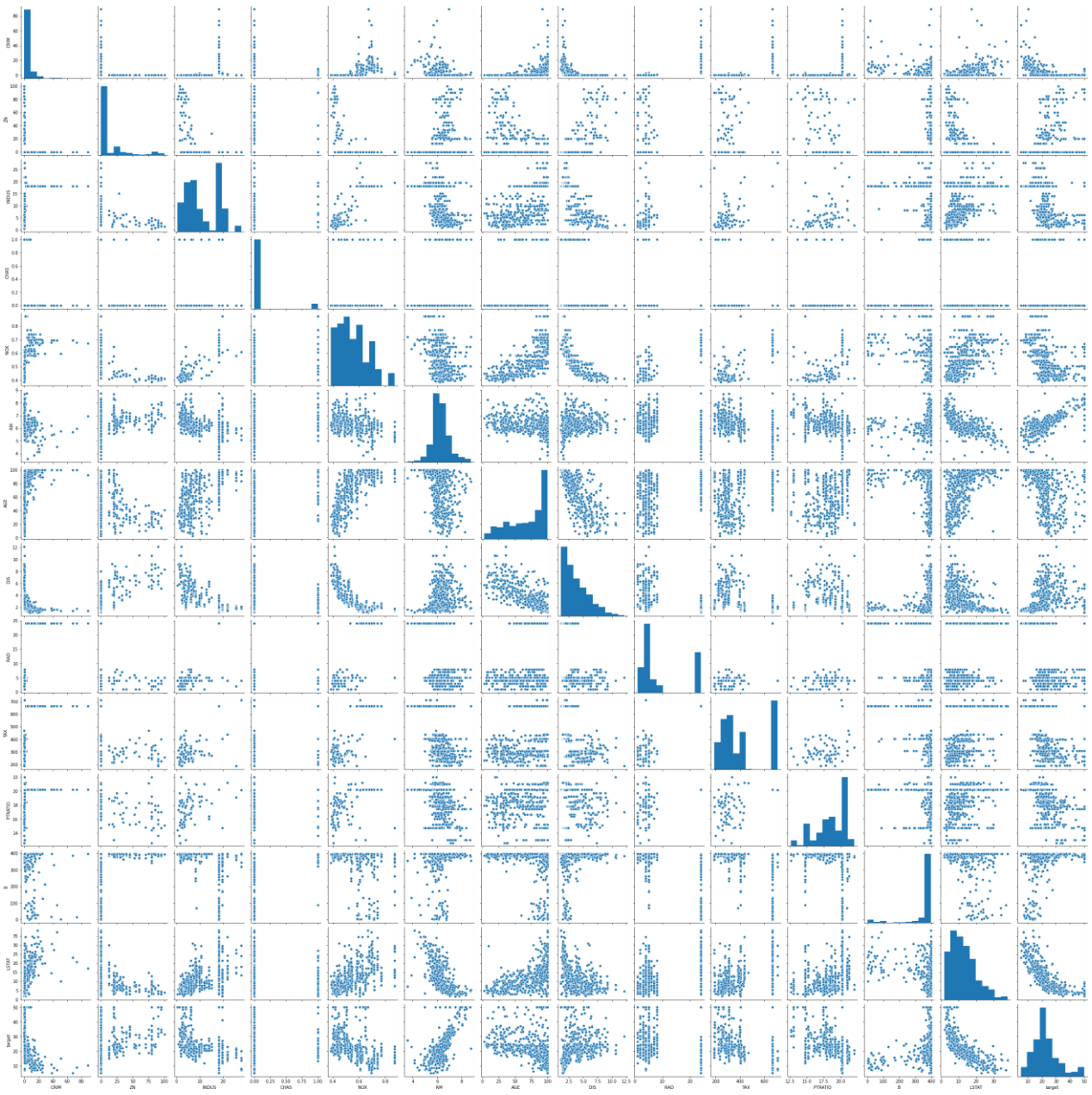
[37] : <matplotlib.axes._subplots.AxesSubplot at 0x7fcff7006390>



Распределение похоже на нормальное
Парные диаграммы для понимания общей картины

[38]:
```
sns.pairplot(data)
```

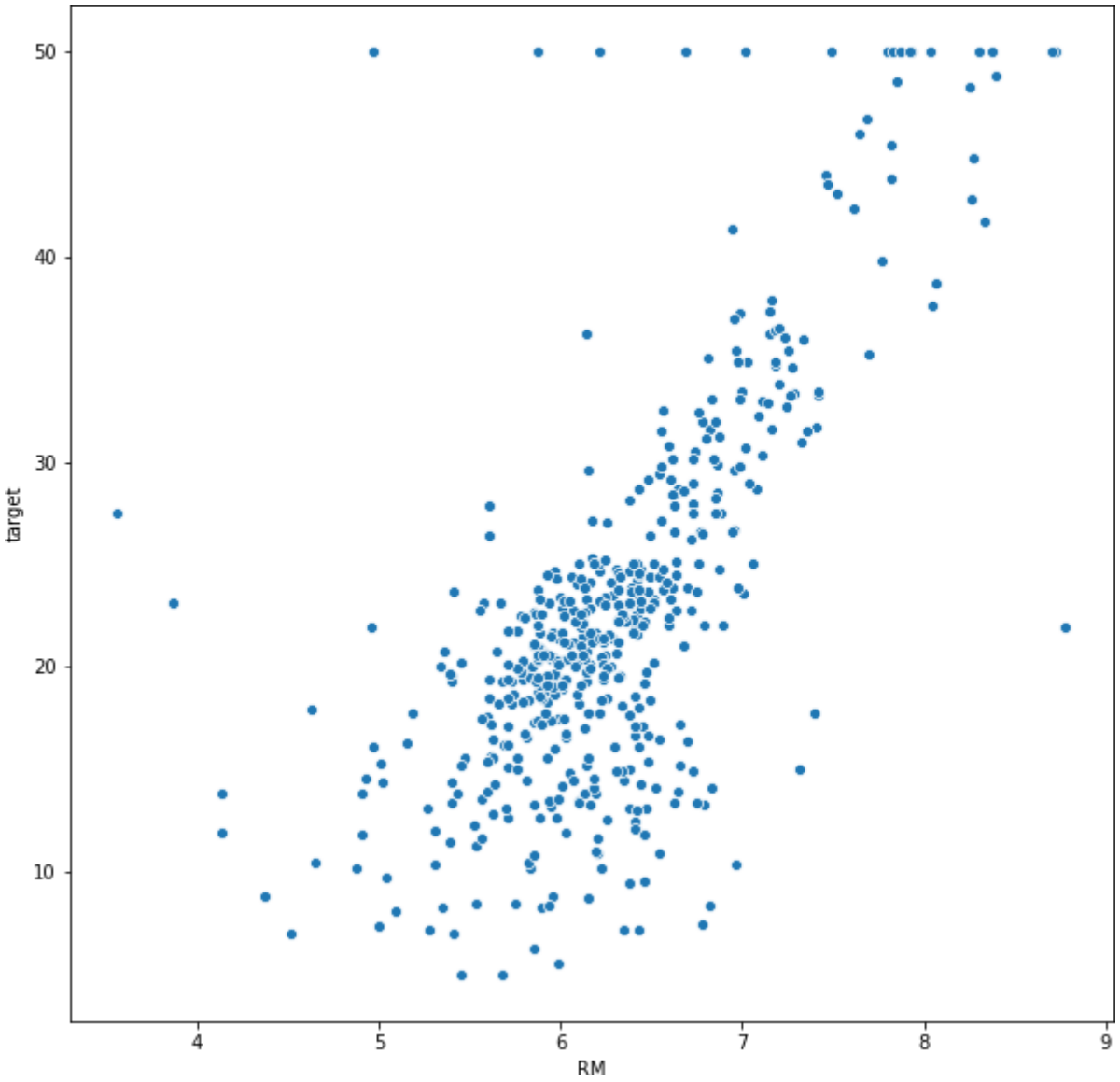
[38] : <seaborn.axisgrid.PairGrid at 0x7fcff7849908>



Находим почти линейную зависимость между значениями двух колонок с содержанием “выбросов”

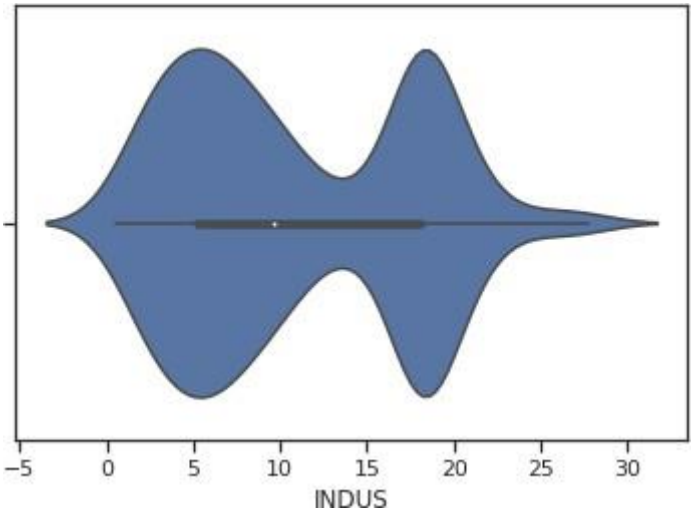
[39]:
```
fig, ax = plt.subplots(figsize=(10,10))
sns.scatterplot(ax=ax, x="RM", y="target", data=data)
```

[39] : <matplotlib.axes._subplots.AxesSubplot at 0x7fcff2cea668>



[0]:
```
sns.violinplot(x=data["INDUS"])
```

[0]: <matplotlib.axes._subplots.AxesSubplot at 0x7f39e74599e8>

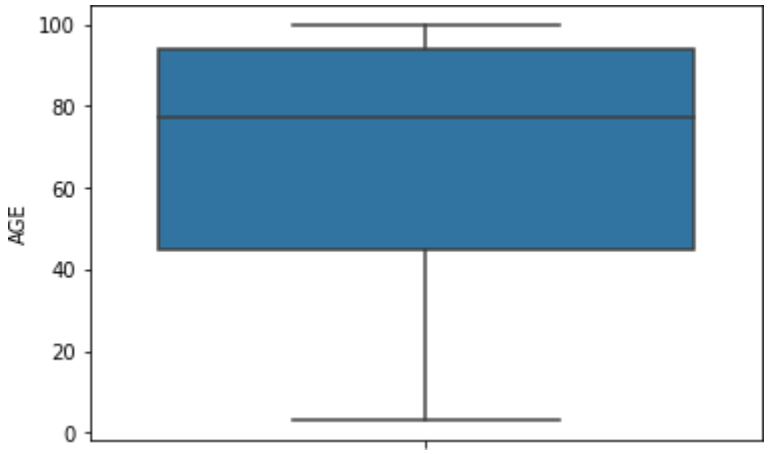


По violin plot видим, что распределение бимодальное.

Задание для ИУ5-23М (boxplot для колонки с возрастом)

[40]:
```
sns.boxplot(y=data["AGE"])
```

[40]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fcff2fefc18>
```



Корреляционный анализ

Построим корреляционную матрицу

[41]:
```
data.corr()
```

[41]:

	CRIM	ZN	INDUS	…	B	LSTAT	target
CRIM	1.000000	-0.200469	0.406583	…	-0.385064	0.455621	-0.388305
ZN	-0.200469	1.000000	-0.533828	…	0.175520	-0.412995	0.360445
INDUS	0.406583	-0.533828	1.000000	…	-0.356977	0.603800	-0.483725
CHAS	-0.055892	-0.042697	0.062938	…	0.048788	-0.053929	0.175260
NOX	0.420972	-0.516604	0.763651	…	-0.380051	0.590879	-0.427321
RM	-0.219247	0.311991	-0.391676	…	0.128069	-0.613808	0.695360

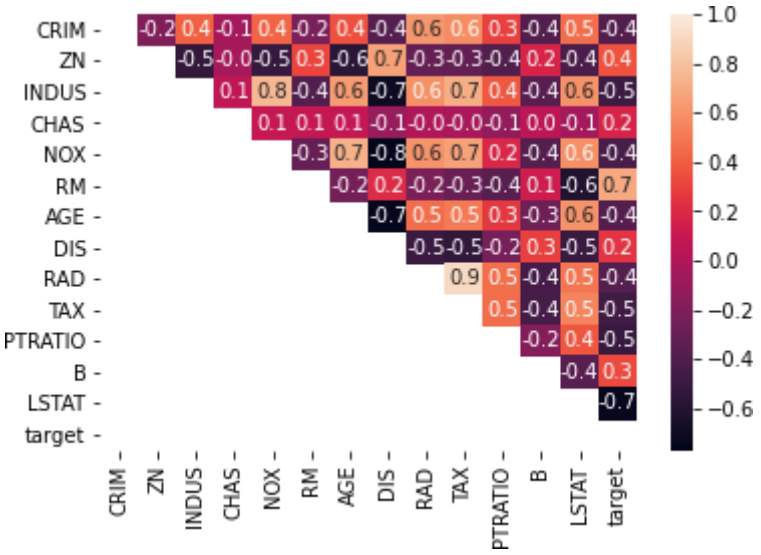
AGE	0.352734	-0.569537	0.644779	...	-0.273534	0.602339	-0.376955
DIS	-0.379670	0.664408	-0.708027	...	0.291512	-0.496996	0.249929
RAD	0.625505	-0.311948	0.595129	...	-0.444413	0.488676	-0.381626
TAX	0.582764	-0.314563	0.720760	...	-0.441808	0.543993	-0.468536
PTRATIO	0.289946	-0.391679	0.383248	...	-0.177383	0.374044	-0.507787
B	-0.385064	0.175520	-0.356977	...	1.000000	-0.366087	0.333461
LSTAT	0.455621	-0.412995	0.603800	...	-0.366087	1.000000	-0.737663
target	-0.388305	0.360445	-0.483725	...	0.333461	-0.737663	1.000000

[14 rows x 14 columns]

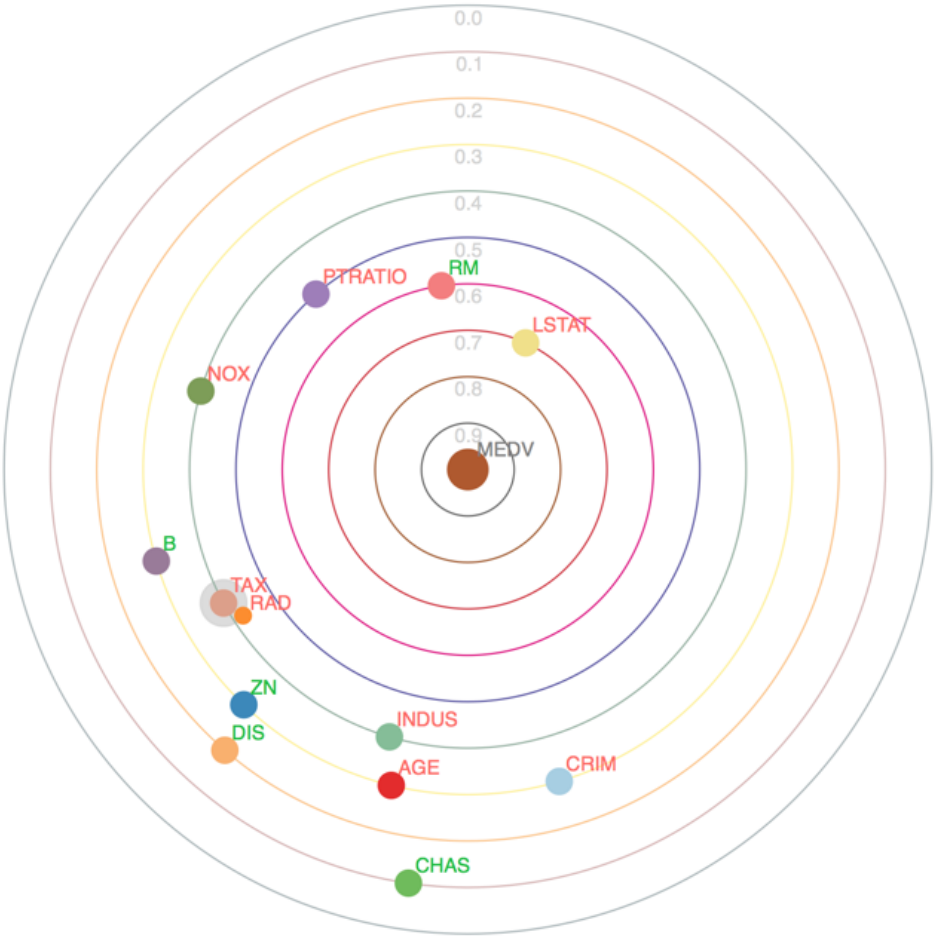
Также построим матрицу корреляций по Пирсону

[42]:
```
# Треугольный вариант матрицы Пирсона
mask = np.zeros_like(data.corr(), dtype=np.bool)
mask[np.tril_indices_from(mask)] = True
sns.heatmap(data.corr(), mask=mask, annot=True, fmt=".1f")
```

[42]: <matplotlib.axes._subplots.AxesSubplot at 0x7fcff2b41978>



Выявлена корреляция между показателями RAD и TAX
Использовав Solar correlation map, получаем ту же зависимость


0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
MEDV
PTRATIO
RM
LSTAT
NOX
B
TAX
RAD
ZN
DIS
INDUS
AGE
CRIM
CHAS