**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_____

Отчёт
по лабораторной работе № 1
**«Разведочный анализ данных. Исследование и визуализация данных.»**
по курсу «Технологии машинного обучения»

Выполнил:
Студент группы ИУ5-63 _____ **Волков А.С.**
(Подпись, дата) (Фамилия И.О.)

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

Москва – 2020

## Цель лабораторной работы:

Изучение различных методов визуализация данных.

**Краткое описание.** Построение основных графиков, входящих в этап разведочного анализа данных.

## Задание:

- Выбрать набор данных (датасет).
- Для первой лабораторной работы рекомендуется использовать датасет без пропусков в данных.
- Выполнить преобразования датасетов Scikit-learn в Pandas Dataframe, если потребуется.

Для лабораторных работ не рекомендуется выбирать датасеты большого размера.

- Создать ноутбук, который содержит следующие разделы:

1. Текстовое описание выбранного Вами набора данных.
2. Основные характеристики датасета.
3. Визуальное исследование датасета.
4. Информация о корреляции признаков.

- Сформировать отчет и разместить его в своем репозитории на github.

## Выполнение лабораторной работы:

# 1) Текстовое описание набора данных

В качестве набора данных мы будем использовать набор данных о ценах домов в Бостоне.
https://github.com/selva86/datasets/blob/master/BostonHousing.csv
(https://github.com/selva86/datasets/blob/master/BostonHousing.csv)

Анализ подобного набора данных позволить выявить прежде всего факторы, которые больше всего влияют на цену собственности, а также позволяют отметить некоторые закономерности между определенными характеристиками собственности.

Датасет состоит из одного файла BostonHousing.csv .

Файл содержит следующие колонки:

CRIM - уровень преступности на душу населения
ZN - доля жилых земель, предназначенных для участков площадью более 25 000 кв. футов.
INDUS - доля нерозничных предприятий в акрах на город
CHAS - искусственная переменная близости к р. Чарльз (1 если здание у реки; 0 иначе)
NOX - концентрация оксидов азота (частей на 10 миллионов)
RM - среднее количество комнат на одно жилище
AGE - количество единиц жилья, занимаемых владельцами, построенных до 1940 года
DIS - взвешенные расстояния до пяти Бостонских центров занятости
RAD - индекс доступности радиальных магистралей
TAX - налог на недвижимость полной стоимости за $10 000
PTRATIO - соотношение числа учащихся и учителей
B - доля чернокожих (Bk): 1000(Bk - 0.63)^2
LSTAT - процент более низкого статуса населения
MEDV - средняя стоимость домов, занятых владельцами, в тысячах долларов США

# Импорт библиотек

Импортируем библиотеки с помощью команды import. Как правило, все команды import размещают в первой ячейке ноутбука, но мы в этом примере будем подключать все библиотеки последовательно, по мере их использования.

In [144]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")
```

# Загрузка данных

Загрузим файлы датасета с помощью библиотеки Pandas.

In [145]:

```
data = pd.read_csv('data/BostonHousing.csv', sep=",")
```

# 2) Основные характеристики датасета

In [146]:

```
# Первые 5 строк датасета
data.head()
```

Out[146]:

| | crim | zn | indus | chas | nox | rm | age | dis | rad | tax | ptratio | b | lstat | m |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.00632 | 18.0 | 2.31 | 0 | 0.538 | 6.575 | 65.2 | 4.0900 | 1 | 296 | 15.3 | 396.90 | 4.98 | |
| 1 | 0.02731 | 0.0 | 7.07 | 0 | 0.469 | 6.421 | 78.9 | 4.9671 | 2 | 242 | 17.8 | 396.90 | 9.14 | |
| 2 | 0.02729 | 0.0 | 7.07 | 0 | 0.469 | 7.185 | 61.1 | 4.9671 | 2 | 242 | 17.8 | 392.83 | 4.03 | |
| 3 | 0.03237 | 0.0 | 2.18 | 0 | 0.458 | 6.998 | 45.8 | 6.0622 | 3 | 222 | 18.7 | 394.63 | 2.94 | |
| 4 | 0.06905 | 0.0 | 2.18 | 0 | 0.458 | 7.147 | 54.2 | 6.0622 | 3 | 222 | 18.7 | 396.90 | 5.33 | |

In [147]:

```
# Размер датасета - 100 строк, 6 колонок
data.shape
```

Out[147]:

```
(506, 14)
```

In [148]:

```
total_count = data.shape[0]
print('Всего строк: {}'.format(total_count))
```

```
Всего строк: 506
```

In [149]:

```
# Список колонок
data.columns
```

Out[149]:

```
Index(['crim', 'zn', 'indus', 'chas', 'nox', 'rm', 'age', 'dis', 'rad', 't
ax',
       'ptratio', 'b', 'lstat', 'medv'],
      dtype='object')
```

In [150]:

```
# Список колонок с типами данных
data.dtypes
```

Out[150]:

```
crim       float64
zn         float64
indus      float64
chas         int64
nox        float64
rm         float64
age        float64
dis        float64
rad          int64
tax          int64
ptratio    float64
b          float64
lstat      float64
medv       float64
dtype: object
```

In [151]:

```
# Цикл по колонкам датасета
for col in data.columns:
    # Количество пустых значений - все значения заполнены
    temp_null_count = data[data[col].isnull()].shape[0]
    print('{} - {}'.format(col, temp_null_count))
```

```
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
medv - 0
```

In [152]:

```
# Основные статистические характеристки набора данных
data.describe()
```

Out[152]:

| | crim | zn | indus | chas | nox | rm | age |
|---|---|---|---|---|---|---|---|
| count | 506.000000 | 506.000000 | 506.000000 | 506.000000 | 506.000000 | 506.000000 | 506.000000 |
| mean | 3.613524 | 11.363636 | 11.136779 | 0.069170 | 0.554695 | 6.284634 | 68.574901 |
| std | 8.601545 | 23.322453 | 6.860353 | 0.253994 | 0.115878 | 0.702617 | 28.148861 |
| min | 0.006320 | 0.000000 | 0.460000 | 0.000000 | 0.385000 | 3.561000 | 2.900000 |
| 25% | 0.082045 | 0.000000 | 5.190000 | 0.000000 | 0.449000 | 5.885500 | 45.025000 |
| 50% | 0.256510 | 0.000000 | 9.690000 | 0.000000 | 0.538000 | 6.208500 | 77.500000 |
| 75% | 3.677082 | 12.500000 | 18.100000 | 0.000000 | 0.624000 | 6.623500 | 94.075000 |
| max | 88.976200 | 100.000000 | 27.740000 | 1.000000 | 0.871000 | 8.780000 | 100.000000 |

In [153]:

```
# Определим уникальные значения для целевого признака
data['medv'].unique()
```

Out[153]:

```
array([24. , 21.6, 34.7, 33.4, 36.2, 28.7, 22.9, 27.1, 16.5, 18.9, 15. ,
       21.7, 20.4, 18.2, 19.9, 23.1, 17.5, 20.2, 13.6, 19.6, 15.2, 14.5,
       15.6, 13.9, 16.6, 14.8, 18.4, 21. , 12.7, 13.2, 13.1, 13.5, 20. ,
       24.7, 30.8, 34.9, 26.6, 25.3, 21.2, 19.3, 14.4, 19.4, 19.7, 20.5,
       25. , 23.4, 35.4, 31.6, 23.3, 18.7, 16. , 22.2, 33. , 23.5, 22. ,
       17.4, 20.9, 24.2, 22.8, 24.1, 21.4, 20.8, 20.3, 28. , 23.9, 24.8,
       22.5, 23.6, 22.6, 20.6, 28.4, 38.7, 43.8, 33.2, 27.5, 26.5, 18.6,
       20.1, 19.5, 19.8, 18.8, 18.5, 18.3, 19.2, 17.3, 15.7, 16.2, 18. ,
       14.3, 23. , 18.1, 17.1, 13.3, 17.8, 14. , 13.4, 11.8, 13.8, 14.6,
       15.4, 21.5, 15.3, 17. , 41.3, 24.3, 27. , 50. , 22.7, 23.8, 22.3,
       19.1, 29.4, 23.2, 24.6, 29.9, 37.2, 39.8, 37.9, 32.5, 26.4, 29.6,
       32. , 29.8, 37. , 30.5, 36.4, 31.1, 29.1, 33.3, 30.3, 34.6, 32.9,
       42.3, 48.5, 24.4, 22.4, 28.1, 23.7, 26.7, 30.1, 44.8, 37.6, 46.7,
       31.5, 31.7, 41.7, 48.3, 29. , 25.1, 17.6, 24.5, 26.2, 42.8, 21.9,
       44. , 36. , 33.8, 43.1, 48.8, 31. , 36.5, 30.7, 43.5, 20.7, 21.1,
       25.2, 35.2, 32.4, 33.1, 35.1, 45.4, 46. , 32.2, 28.5, 37.3, 27.9,
       28.6, 36.1, 28.2, 16.1, 22.1, 19. , 32.7, 31.2, 17.2, 16.8, 10.2,
       10.4, 10.9, 11.3, 12.3,  8.8,  7.2, 10.5,  7.4, 11.5, 15.1,  9.7,
       12.5,  8.5,  5. ,  6.3,  5.6, 12.1,  8.3, 11.9, 17.9, 16.3,  7. ,
        7.5,  8.4, 16.7, 14.2, 11.7, 11. ,  9.5, 14.1,  9.6,  8.7, 12.8,
       10.8, 14.9, 12.6, 13. , 16.4, 17.7, 12. , 21.8,  8.1])
```

Целевой признак содержит множество уникальных значений.

# 3) Визуальное исследование датасета

# Диаграмма рассеяния

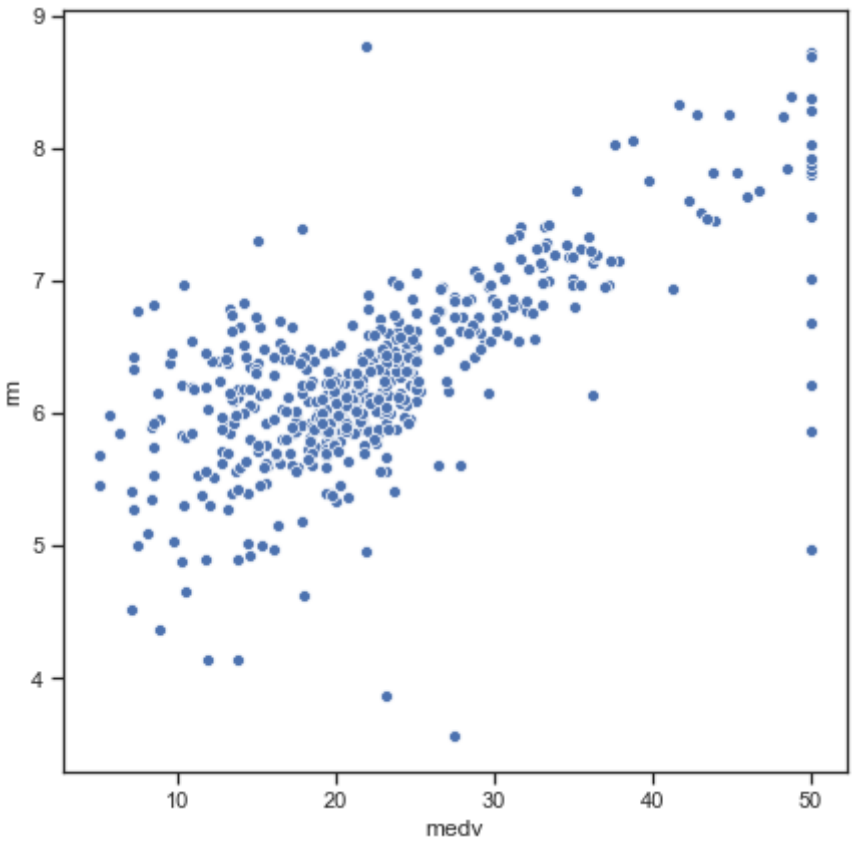
В данном случае рассмотрим **отношение средней стоимости дома и количества комнат в нем.**

In [154]:

```
fig, ax = plt.subplots(figsize=(7,7))
sns.scatterplot(ax=ax, x='medv', y='rm', data=data)
```

Out[154]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1d0d3a93688>
```




Больше комнат - стоимость, очевидно, больше. Однако большинство зданий имеют до 7 комнат.

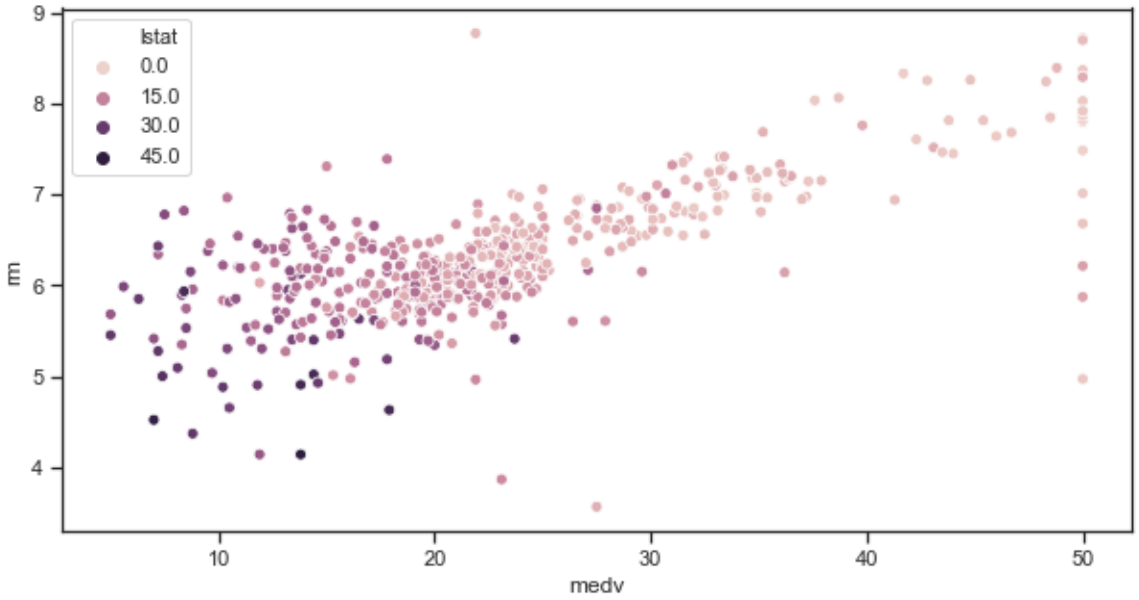
А теперь **выделим цветом процент населения более низкого статуса.**

In [155]:

```
plt.figure(figsize = (10,5))
sns.scatterplot(x='medv', y='rm', hue = 'lstat', data=data)
```

Out[155]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1d0d379fa48>
```




Виден очевидный факт - население более низкого статуса может позволить себе только дешевые дома.
Однако этот дом необязательно будет иметь маленькое количество комнат.

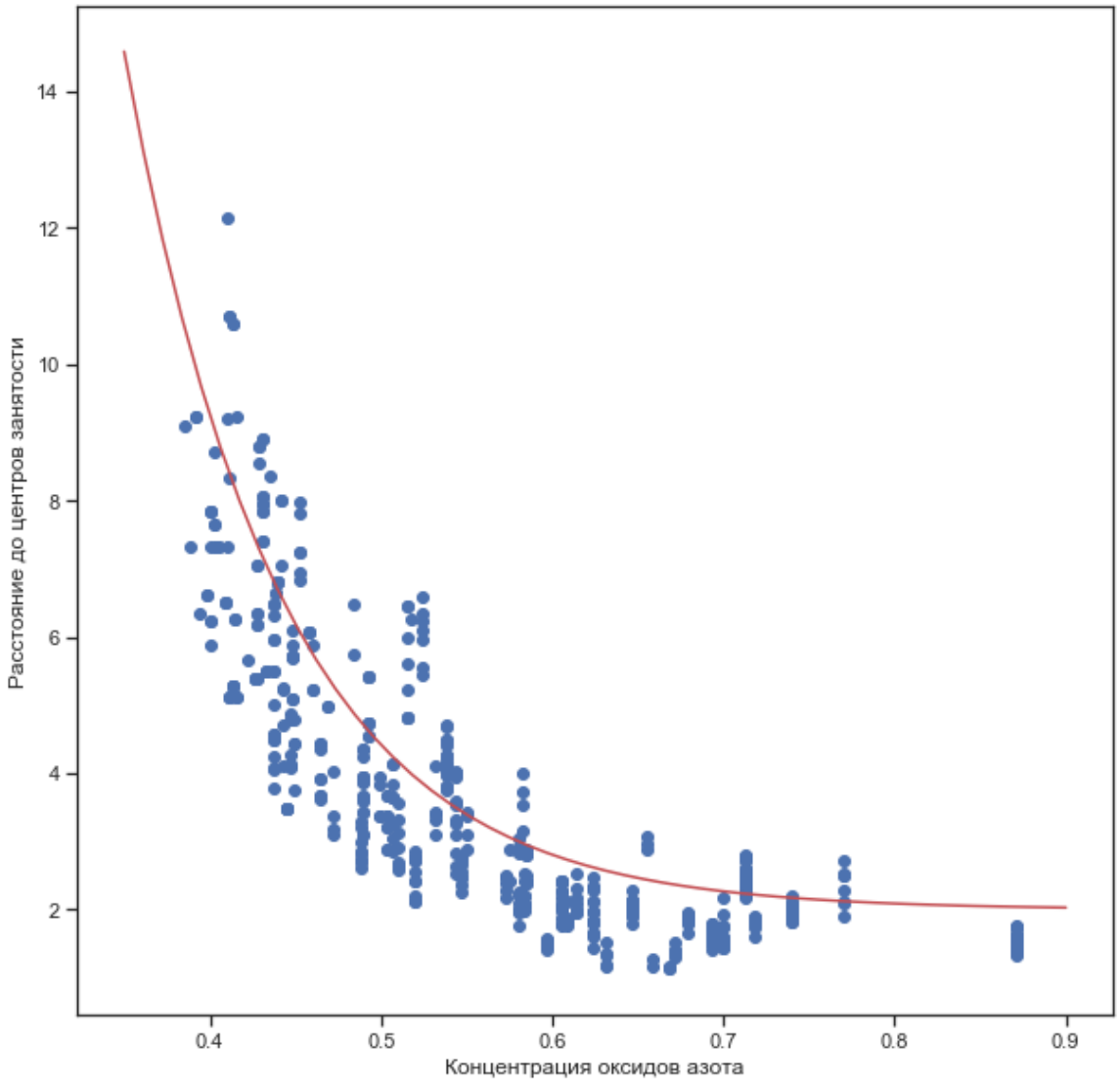
Также мною была замечена странная **связь концентрации оксидов азота и расстояния до центров занятости.**

In [156]:

```
plt.figure(figsize = (10,10))
X_plot = np.linspace(0.35, 0.9, 50)
Y_plot = np.exp(-(X_plot*11-2))*80+2
plt.scatter(data['nox'], data['dis']); plt.plot(X_plot, Y_plot, color='r')
plt.xlabel('Концентрация оксидов азота')
plt.ylabel('Расстояние до центров занятости')
```

Out[156]:

```
Text(0, 0.5, 'Расстояние до центров занятости')
```




Красным показана приблизительная "экспоненциальная зависимость".

# Гистограмма

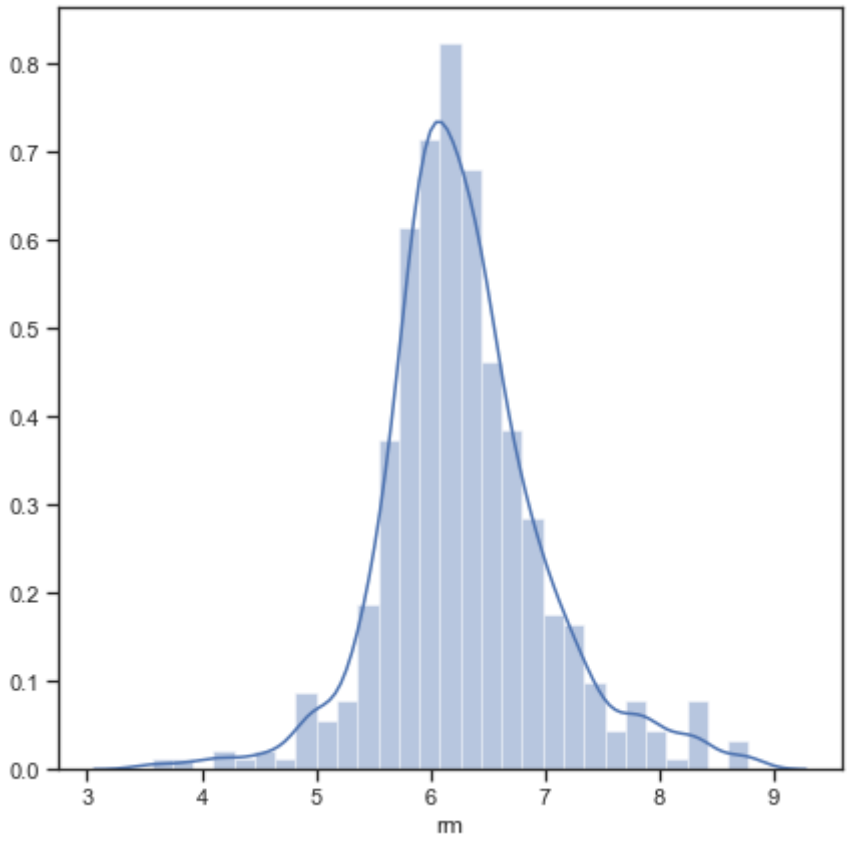
Оценим плотность вероятности **распределения среднего числа комнат.**

In [157]:

```
plt.figure(figsize = (7,7))
sns.distplot(data['rm'])
```

Out[157]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1d0d1dd5f88>
```




Построим **jointplot для отношения концентрации азота и расстояния до центров занятости.**

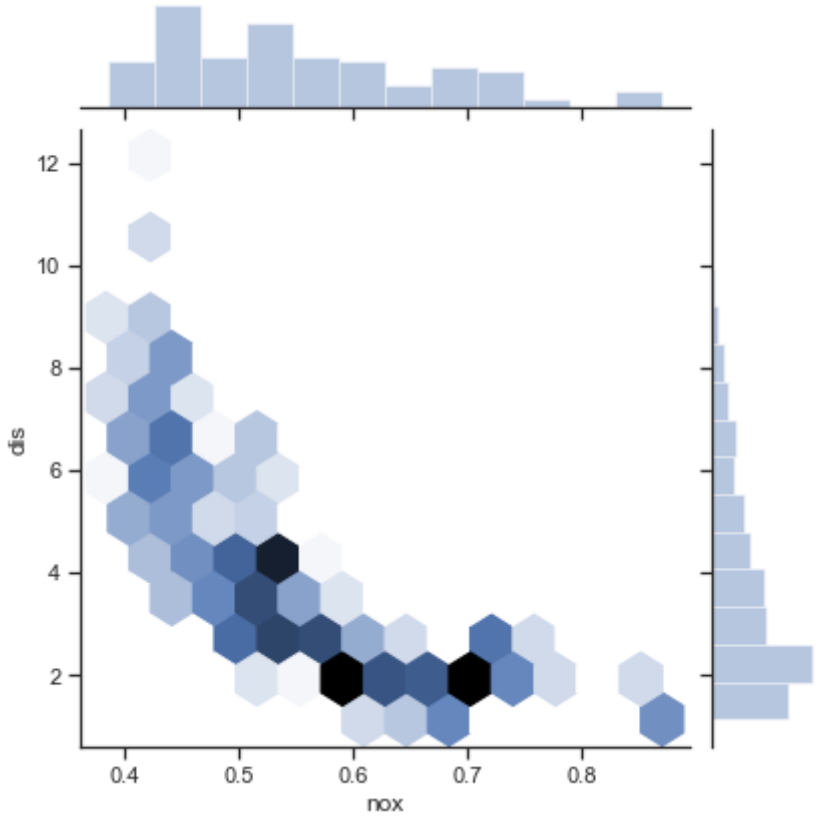
In [158]:

```
plt.figure(figsize = (7,7))
sns.jointplot(x='nox', y='dis', data=data, kind="hex")
```

Out[158]:

```
<seaborn.axisgrid.JointGrid at 0x1d0d43917c8>
```

```
<Figure size 504x504 with 0 Axes>
```



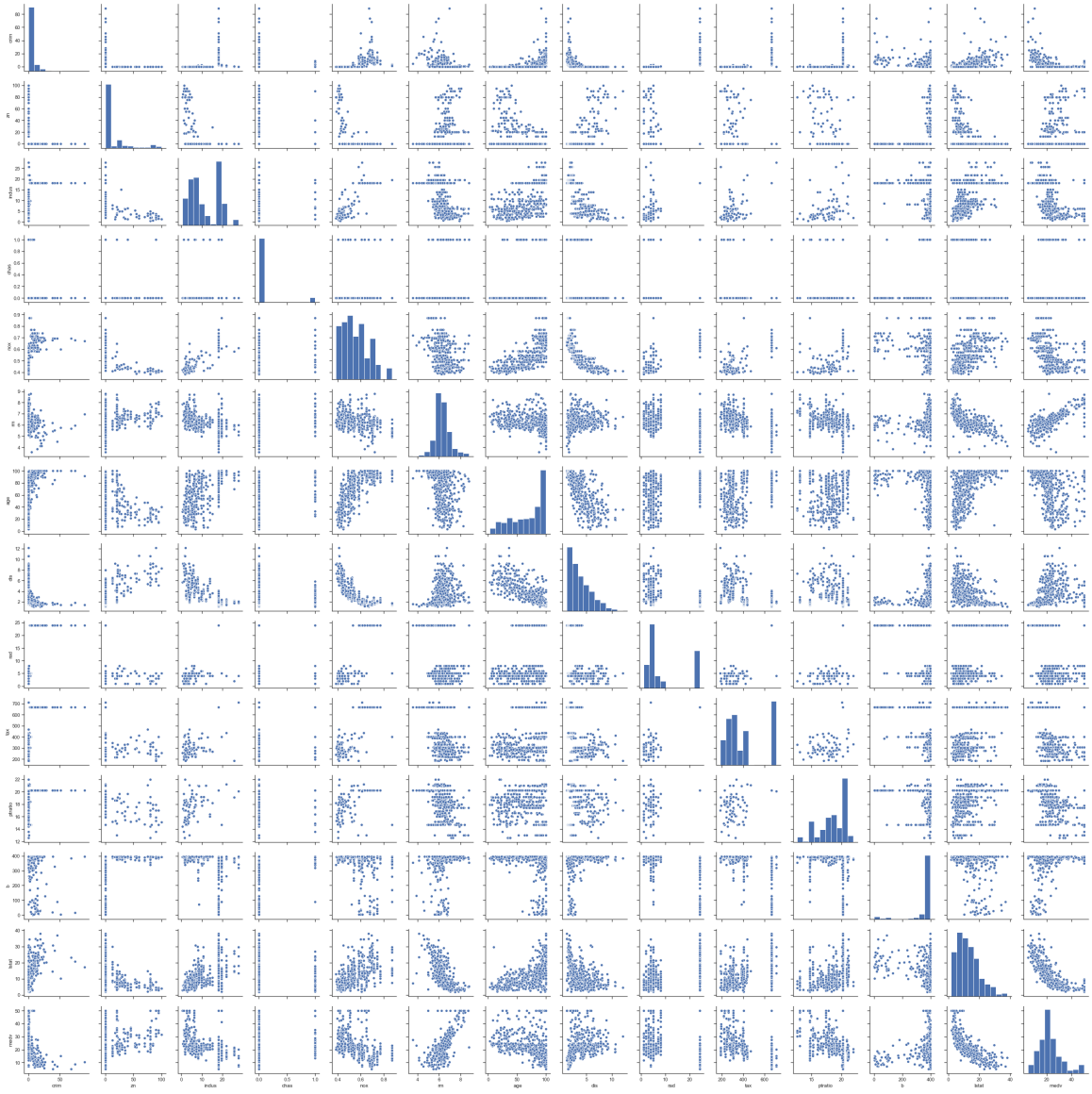

**Парная диаграмма, где цветом обозначена близость к реке.**

In [159]:

```
sns.pairplot(data)
```

Out[159]:

```
<seaborn.axisgrid.PairGrid at 0x1d0d1ebe848>
```



**Violin plot**

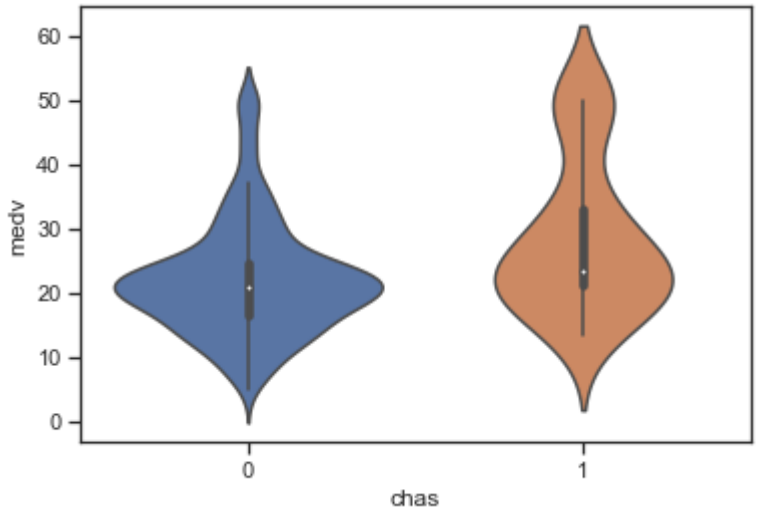
Распределение плотности для средней стоимости, сгрупированное по близости к реке.

In [160]:

```
sns.violinplot(x='chas', y='medv', data=data)
```

Out[160]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1d0d946ea48>
```




# 4)Информация о корреляции признаков

In [161]:

```
data.corr()
```

Out[161]:

| | crim | zn | indus | chas | nox | rm | age | dis |
|---|---|---|---|---|---|---|---|---|
| crim | 1.000000 | -0.200469 | 0.406583 | -0.055892 | 0.420972 | -0.219247 | 0.352734 | -0.379670 |
| zn | -0.200469 | 1.000000 | -0.533828 | -0.042697 | -0.516604 | 0.311991 | -0.569537 | 0.664408 |
| indus | 0.406583 | -0.533828 | 1.000000 | 0.062938 | 0.763651 | -0.391676 | 0.644779 | -0.708027 |
| chas | -0.055892 | -0.042697 | 0.062938 | 1.000000 | 0.091203 | 0.091251 | 0.086518 | -0.099176 |
| nox | 0.420972 | -0.516604 | 0.763651 | 0.091203 | 1.000000 | -0.302188 | 0.731470 | -0.769230 |
| rm | -0.219247 | 0.311991 | -0.391676 | 0.091251 | -0.302188 | 1.000000 | -0.240265 | 0.205246 |
| age | 0.352734 | -0.569537 | 0.644779 | 0.086518 | 0.731470 | -0.240265 | 1.000000 | -0.747881 |
| dis | -0.379670 | 0.664408 | -0.708027 | -0.099176 | -0.769230 | 0.205246 | -0.747881 | 1.000000 |
| rad | 0.625505 | -0.311948 | 0.595129 | -0.007368 | 0.611441 | -0.209847 | 0.456022 | -0.494588 |
| tax | 0.582764 | -0.314563 | 0.720760 | -0.035587 | 0.668023 | -0.292048 | 0.506456 | -0.534432 |
| ptratio | 0.289946 | -0.391679 | 0.383248 | -0.121515 | 0.188933 | -0.355501 | 0.261515 | -0.232471 |
| b | -0.385064 | 0.175520 | -0.356977 | 0.048788 | -0.380051 | 0.128069 | -0.273534 | 0.291512 |
| lstat | 0.455621 | -0.412995 | 0.603800 | -0.053929 | 0.590879 | -0.613808 | 0.602339 | -0.496996 |
| medv | -0.388305 | 0.360445 | -0.483725 | 0.175260 | -0.427321 | 0.695360 | -0.376955 | 0.249929 |

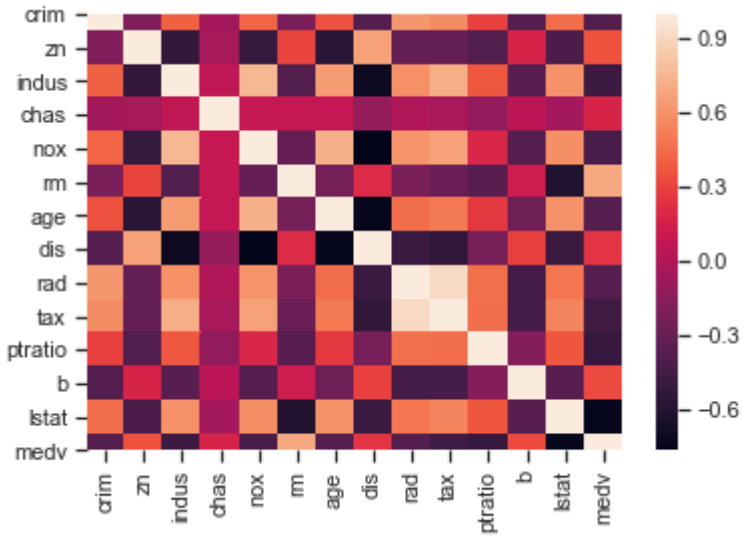
# Тепловая карта

In [162]:

```
sns.heatmap(data.corr())
```

Out[162]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1d0db026ac8>
```



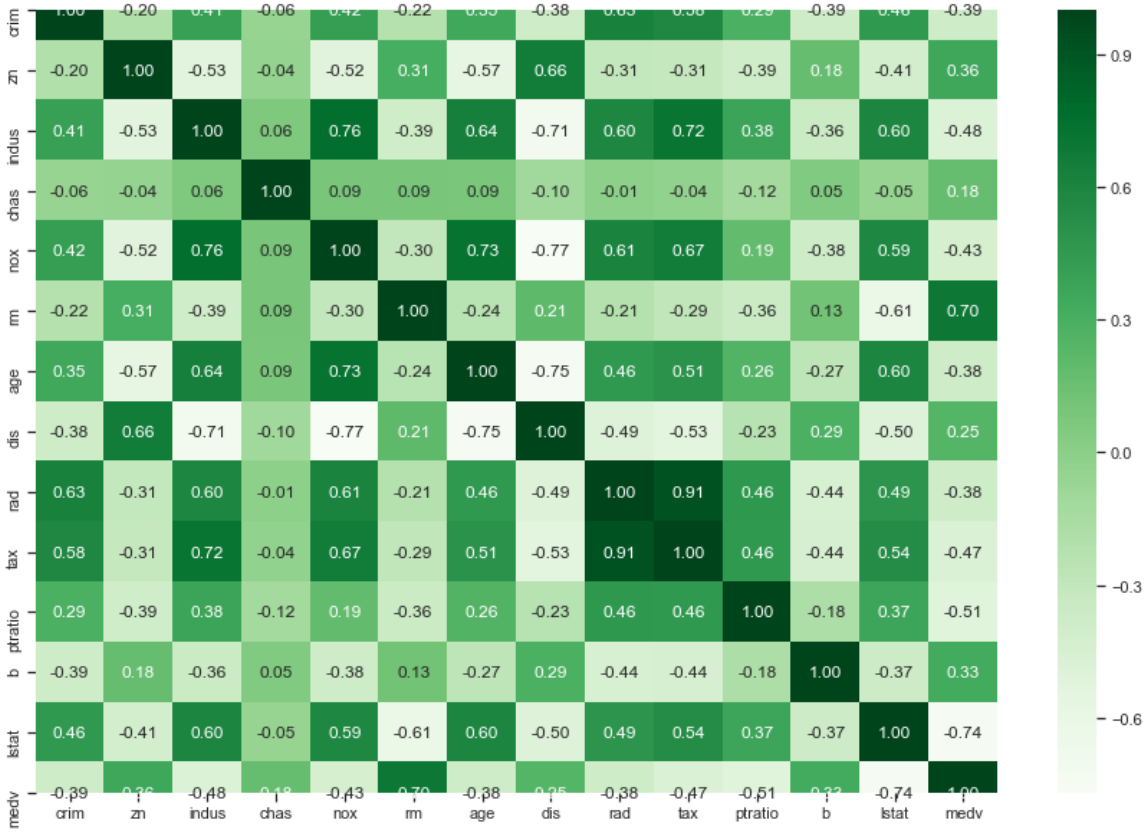

# Тепловая карта со значениями

In [163]:

```
plt.figure(figsize = (15,10))
sns.heatmap(data.corr(), cmap='Greens', annot=True, fmt='.2f')
```

Out[163]:

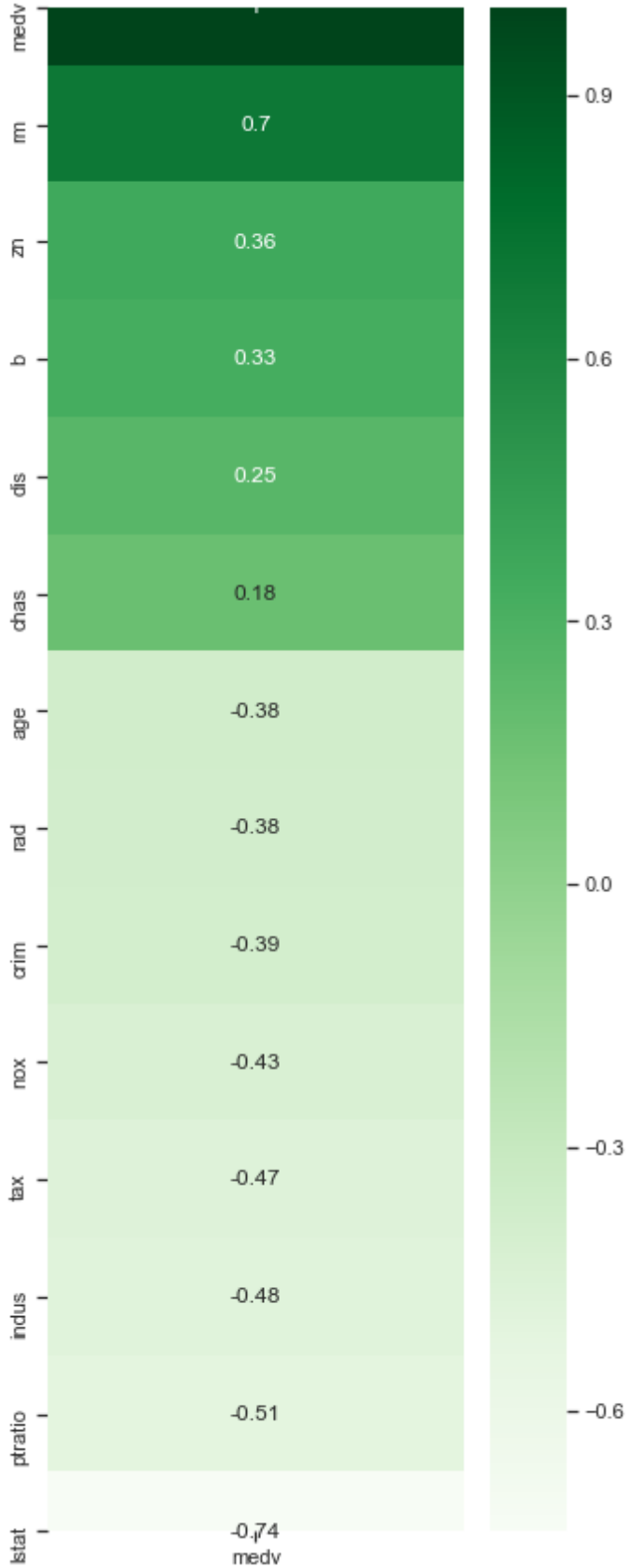
```
<matplotlib.axes._subplots.AxesSubplot at 0x1d0dcaf8cc8>
```




# Сортировка корреляций для целевого признака

In [167]:

```
plt.figure(figsize = (5,15))
sns.heatmap(data.corr()[['medv']].sort_values(by=['medv'],ascending=False),
            cmap='Greens',
            annot=True);
```



# Треугольнгая тепловая карта корреляций с обнулёнными незначительными корреляциями

In [165]:

```
new_corr = data.corr()
new_corr[np.abs(new_corr)<.4] = 0
plt.figure(figsize=(15,10))
mask = np.zeros_like(new_corr, dtype=np.bool)
mask[np.tril_indices_from(mask)] = True
sns.heatmap(new_corr, mask = mask, cmap = 'coolwarm', annot=True);
```