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, ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ И СИСТЕМЫ УПРАВЛЕНИЯ

КАФЕДРА _____СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ (ИУ5)_____

ОТЧЕТ

Лабораторная работа №1

«Создание истории о данных»

по курсу «Методы машинного обучения»

ИСПОЛНИТЕЛЬ: Савченко Г. А.
ФИО

группа ИУ5-21М _____
подпись

"__"_____2023 г.

ПРЕПОДАВАТЕЛЬ: Гапанюк Ю.Е.
ФИО

_____
подпись

"__"_____2023 г.

Москва - 2023

Лабораторная работа №1

Разведочный анализ данных. Исследование и визуализация данных.

Цель лабораторной работы: изучение различных методов визуализация данных.

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

1) Текстовое описание набора данных

В качестве набора данных мы будем использовать набор данных Campus Recruitment - https://www.kaggle.com/datasets/benroshan/factors-affecting-campus-placement/versions/1?resource=download

Этот набор данных состоит из данных о размещении студентов в кампусе XYZ. Он включает среднюю и высшую среднюю школу и специализацию. Он также включает специализацию по степени, тип и опыт работы, а также предложения по заработной плате для размещенных студентов.

Файл Placement_Data_Full_Class.csv содержит следующие колонки:

- sl_no - Serial Number
- gender - Gender Male='M', Female='F'
- ssc_p - Secondary Education percentage - 10th Grade
- ssc_b - Board of Education - Central/Others
- hsc_p - Higher Secondary Education percentage - 12th Grade
- hsc_b - Board of Education - Central/Others
- hsc_s - Specialization in Higher Secondary Education
- degree_p - Degree Percentage
- degree_t - Under Graduation(Degree type) - Field of degree education
- workex - Work Experience
- etest_p - Employability test percentage (conducted by college)
- specialisation - Post Graduation(MBA) - Specialization
- mba_p - MBA percentage
- status - Status of placement - Placed/Not placed
- salary - Salary offered by corporate to candidates

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 [1]:
```
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 в помощью библиотеки Pandas.

Файлы имеют расширение CSV (https://ru.wikipedia.org/wiki/CSV). Часто в файлах такого формата в качестве разделителей используются символы ",", ";" или табуляция. Поэтому вызывая метод read_csv всегда стоит явно указывать разделитель данных с помощью параметра sep. Чтобы узнать какой разделитель используется в файле его рекомендуется предварительно посмотреть в любом текстовом редакторе.

In [4]:
```
# Будем анализировать данные файла Placement_Data_Full_Class.csv
data = pd.read_csv('data/Placement_Data_Full_Class.csv', sep=",")
```

2) Основные характеристики датасета

In [5]:
```
# Первые 5 строк датасета
data.head()
```

Out[5]:

	sl_no	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	degree_t	workex	etest_p
0	1	M	67.00	Others	91.00	Others	Commerce	58.00	Sci&Tech	No	55.(
1	2	M	79.33	Central	78.33	Others	Science	77.48	Sci&Tech	Yes	86.5
2	3	M	65.00	Central	68.00	Central	Arts	64.00	Comm&Mgmt	No	75.(
3	4	M	56.00	Central	52.00	Central	Science	52.00	Sci&Tech	No	66.(
4	5	M	85.80	Central	73.60	Central	Commerce	73.30	Comm&Mgmt	No	96.8

In [7]:
```
# Размер датасета - 215 строк, 15 колонок
data.shape
```

Out[7]:
```
(215, 15)
```

In [10]:
```
total_count = data.shape[0]
print('Всего строк: {}'.format(total_count))
```

```
Всего строк: 215
```

In [11]:
```
# Список колонок
data.columns
```

Out[11]:
```
Index(['sl_no', 'gender', 'ssc_p', 'ssc_b', 'hsc_p', 'hsc_b', 'hsc_s',
       'degree_p', 'degree_t', 'workex', 'etest_p', 'specialisation', 'mba_p',
       'status', 'salary'],
      dtype='object')
```

In [12]:
```
# Список колонок с типами данных
data.dtypes
```

Out[12]:
```
sl_no              int64
gender            object
ssc_p            float64
ssc_b             object
hsc_p            float64
hsc_b             object
hsc_s             object
degree_p         float64
degree_t          object
workex            object
etest_p          float64
specialisation    object
mba_p            float64
status            object
salary           float64
dtype: object
```

In [13]:
```
# Проверим наличие пустых значений
# 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))
```

```
sl_no - 0
gender - 0
ssc_p - 0
ssc_b - 0
hsc_p - 0
hsc_b - 0
hsc_s - 0
degree_p - 0
degree_t - 0
workex - 0
etest_p - 0
specialisation - 0
mba_p - 0
status - 0
salary - 67
```

In [14]:
```
# Основные статистические характеристки набора данных
data.describe()
```

Out[14]:

	sl_no	ssc_p	hsc_p	degree_p	etest_p	mba_p	salary
count	215.000000	215.000000	215.000000	215.000000	215.000000	215.000000	148.000000
mean	108.000000	67.303395	66.333163	66.370186	72.100558	62.278186	288655.405405
std	62.209324	10.827205	10.897509	7.358743	13.275956	5.833385	93457.452420
min	1.000000	40.890000	37.000000	50.000000	50.000000	51.210000	200000.000000
25%	54.500000	60.600000	60.900000	61.000000	60.000000	57.945000	240000.000000
50%	108.000000	67.000000	65.000000	66.000000	71.000000	62.000000	265000.000000
75%	161.500000	75.700000	73.000000	72.000000	83.500000	66.255000	300000.000000
max	215.000000	89.400000	97.700000	91.000000	98.000000	77.890000	940000.000000

In [15]:
```
# Определим уникальные значения
data['hsc_s'].unique()
```

Out[15]:
```
array(['Commerce', 'Science', 'Arts'], dtype=object)
```

hsc_s (Специализация в высшем среднем образовании) содержит значения 'Commerce', 'Science', 'Arts'.

3) Визуальное исследование датасета

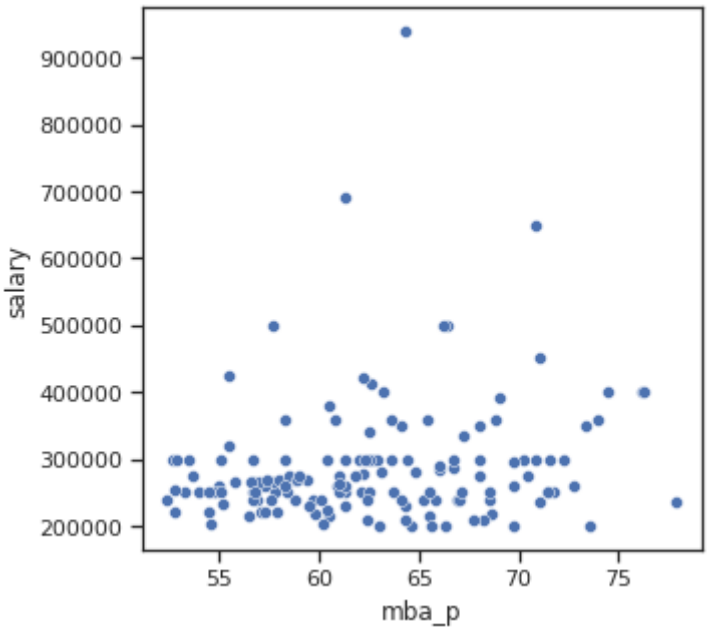
Для визуального исследования могут быть использованы различные виды диаграмм, мы построим только некоторые варианты диаграмм, которые используются достаточно часто.

Диаграмма рассеяния

Позволяет построить распределение двух колонок данных и визуально обнаружить наличие зависимости. Не предполагается, что значения упорядочены (например, по времени).

In [18]:
```
fig, ax = plt.subplots(figsize=(5,5))
sns.scatterplot(ax=ax, x='mba_p', y='salary', data=data)
```

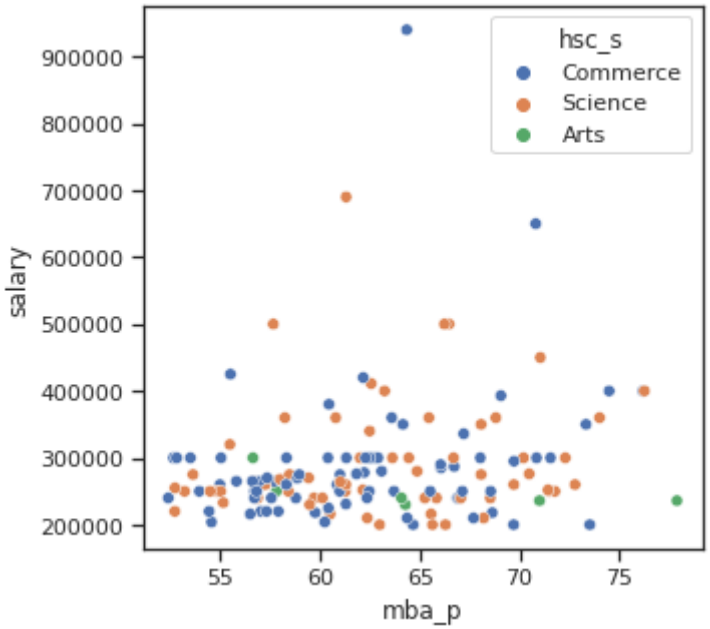
Out[18]:
```
<AxesSubplot:xlabel='mba_p', ylabel='salary'>
```



Посмотрим насколько на эту зависимость влияет hsc_s.

In [41]:
```
fig, ax = plt.subplots(figsize=(5,5))
sns.scatterplot(ax=ax, x='mba_p', y='salary', data=data, hue='hsc_s')
```

Out[41]:
```
<AxesSubplot:xlabel='mba_p', ylabel='salary'>
```



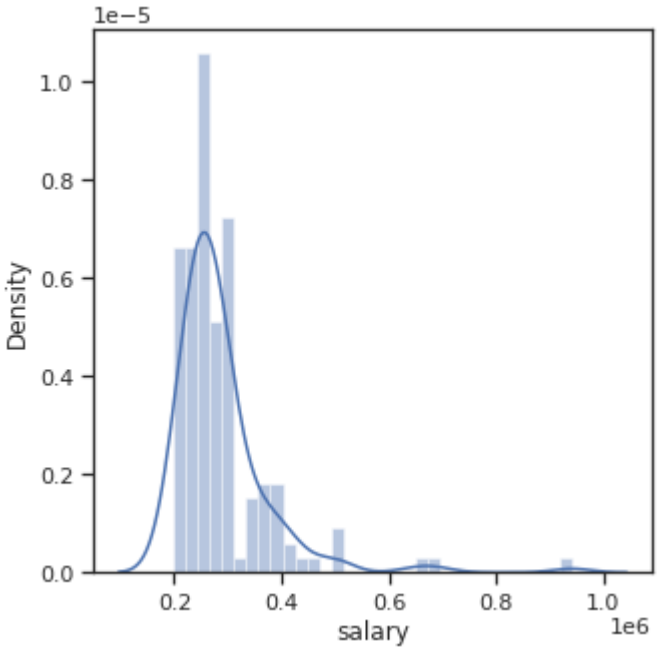
Гистограмма

Позволяет оценить плотность вероятности распределения данных.

In [23]:
```
fig, ax = plt.subplots(figsize=(5,5))
sns.distplot(data['salary'])
```

```
/root/miniconda3/lib/python3.9/site-packages/seaborn/distributions.py:2619: FutureWa
rning: `distplot` is a deprecated function and will be removed in a future version.
Please adapt your code to use either `displot` (a figure-level function with similar
flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[23]:
```
<AxesSubplot:xlabel='salary', ylabel='Density'>
```

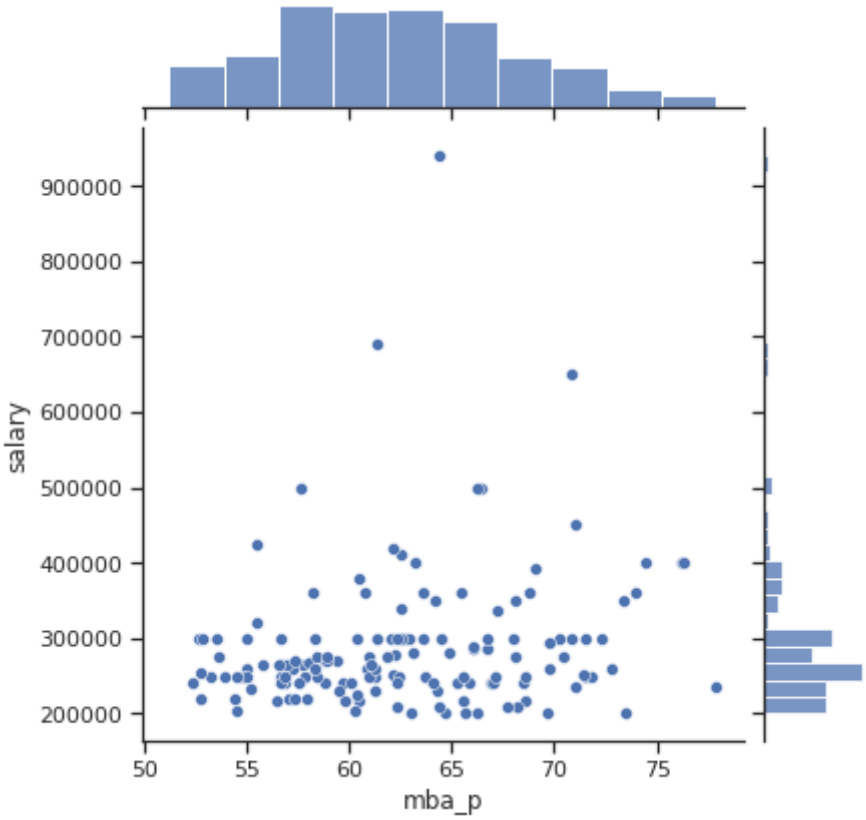


Jointplot

Комбинация гистограмм и диаграмм рассеивания.

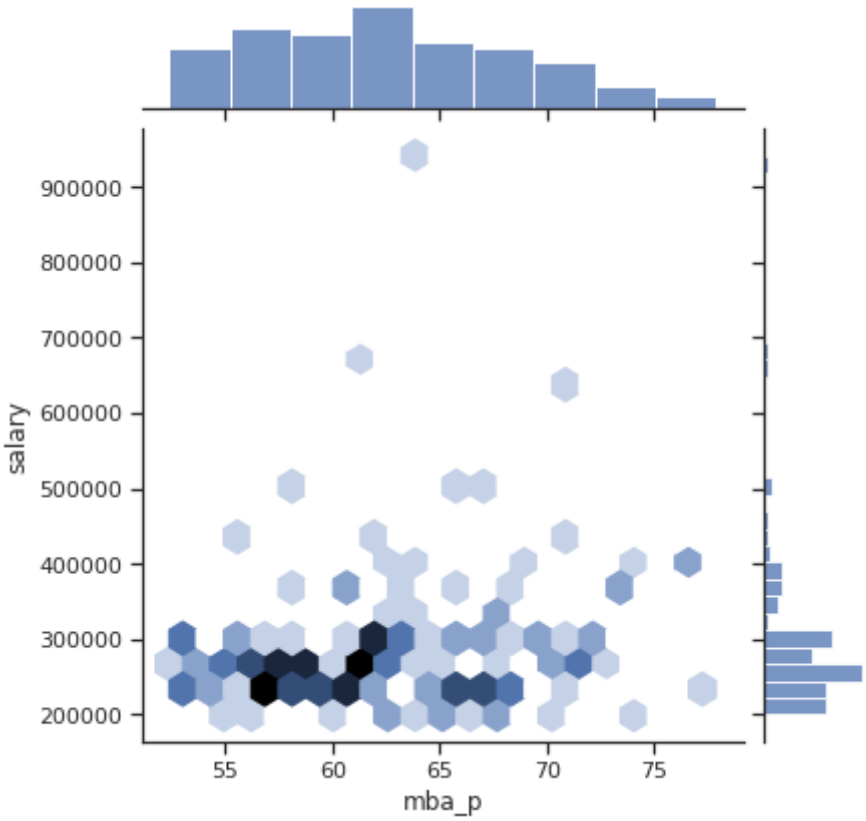
In [26]:
```
sns.jointplot(x='mba_p', y='salary', data=data)
```

Out[26]: <seaborn.axisgrid.JointGrid at 0x7f4d9f4fa160>



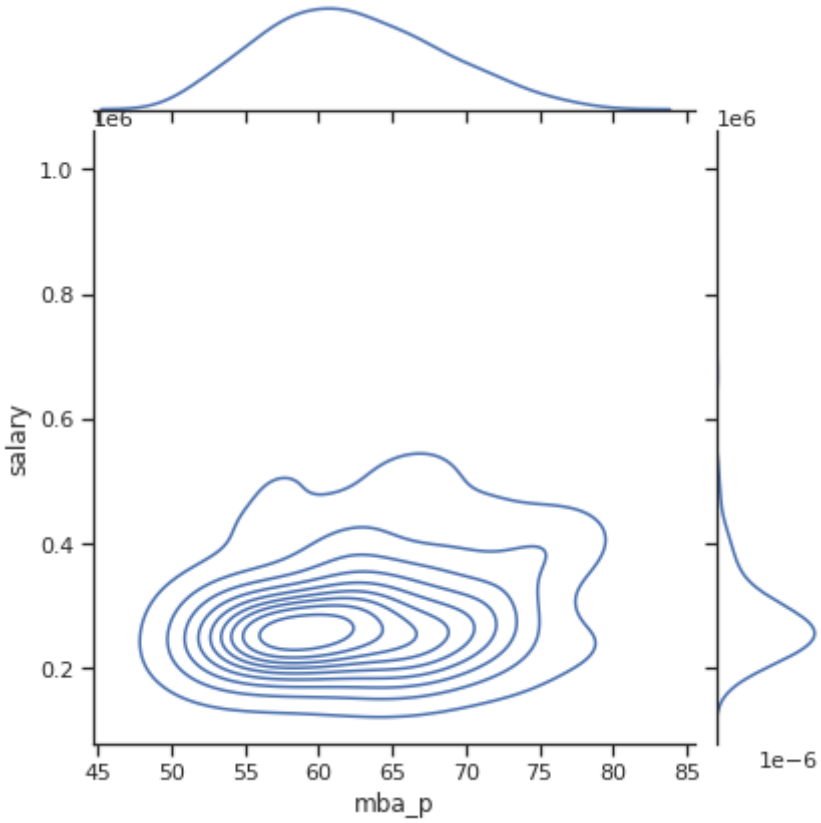
In [27]:
```
sns.jointplot(x='mba_p', y='salary', data=data, kind="hex")
```

Out[27]: <seaborn.axisgrid.JointGrid at 0x7f4d9f4cd370>



In [28]:
```
sns.jointplot(x='mba_p', y='salary', data=data, kind="kde")
```

Out[28]: <seaborn.axisgrid.JointGrid at 0x7f4d9f331af0>



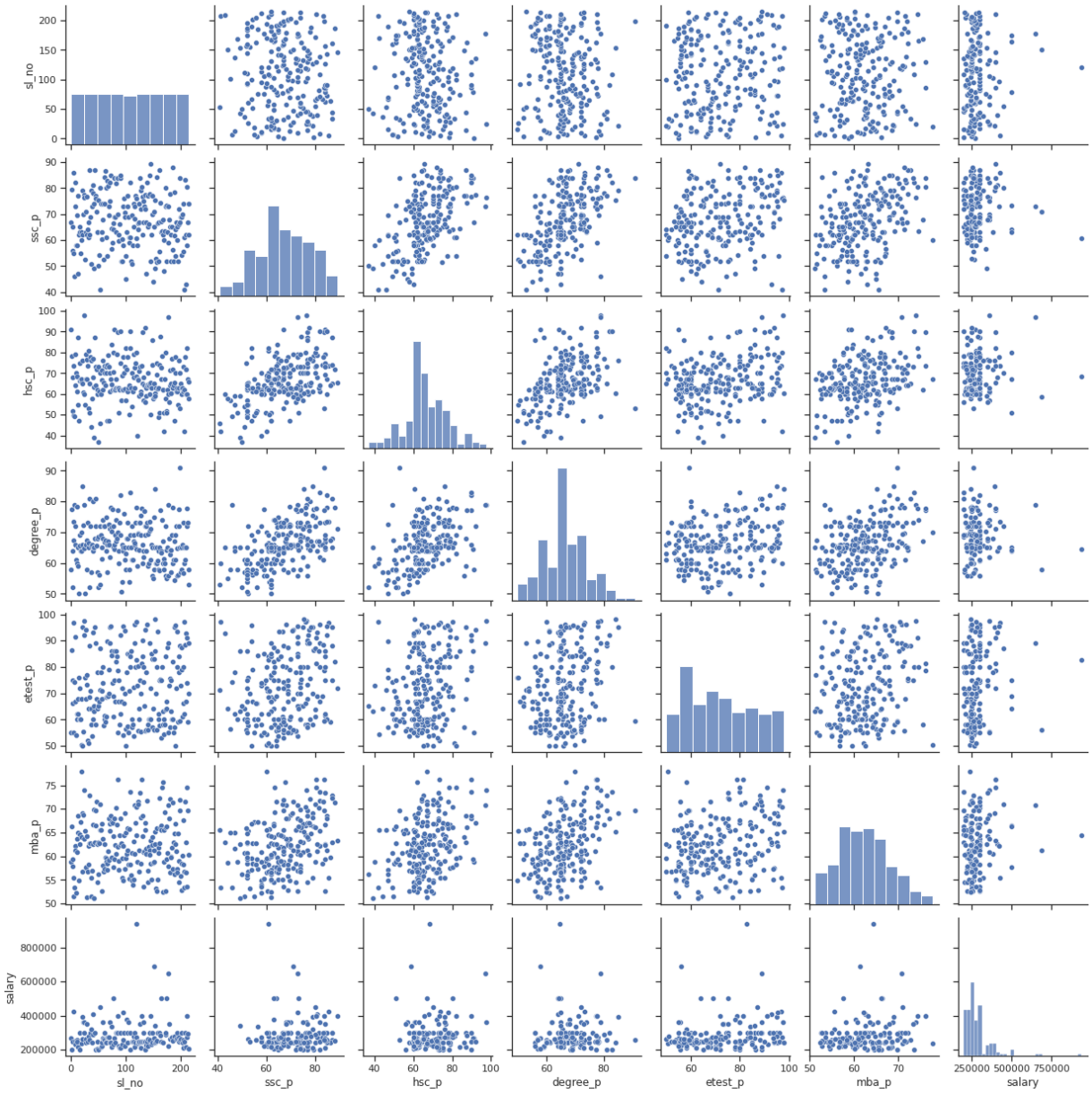
"Парные диаграммы"

Комбинация гистограмм и диаграмм рассеивания для всего набора данных.

Выводится матрица графиков. На пересечении строки и столбца, которые соответствуют двум показателям, строится диаграмма рассеивания. В главной диагонали матрицы строятся гистограммы распределения соответствующих показателей.

In [29]:
```
sns.pairplot(data)
```

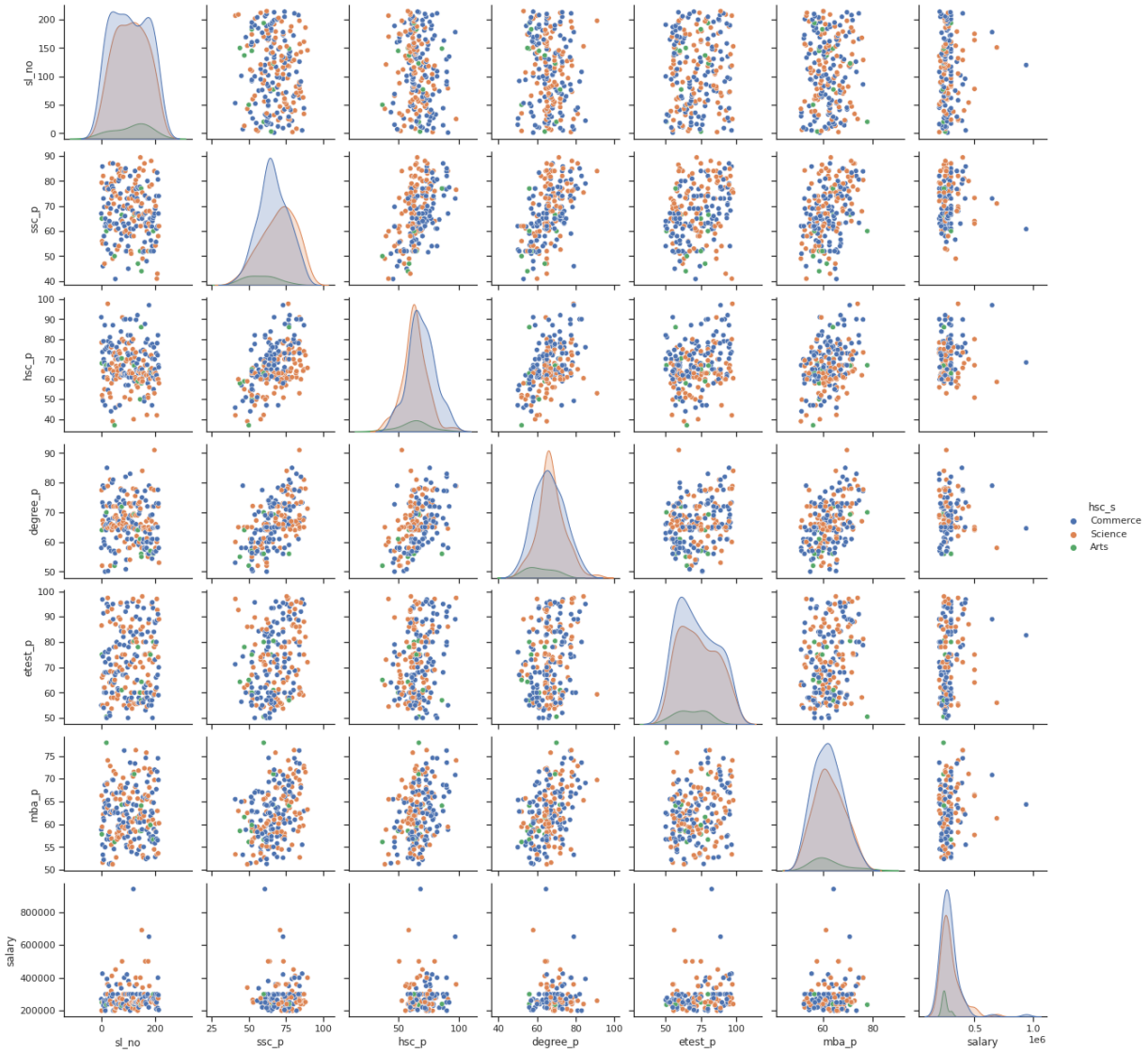
Out[29]: <seaborn.axisgrid.PairGrid at 0x7f4d9f137190>



С помощью параметра "hue" возможна группировка по значениям какого-либо признака.

In [31]:
```
sns.pairplot(data, hue="hsc_s")
```

Out[31]:
```
<seaborn.axisgrid.PairGrid at 0x7f4d9f1e5580>
```

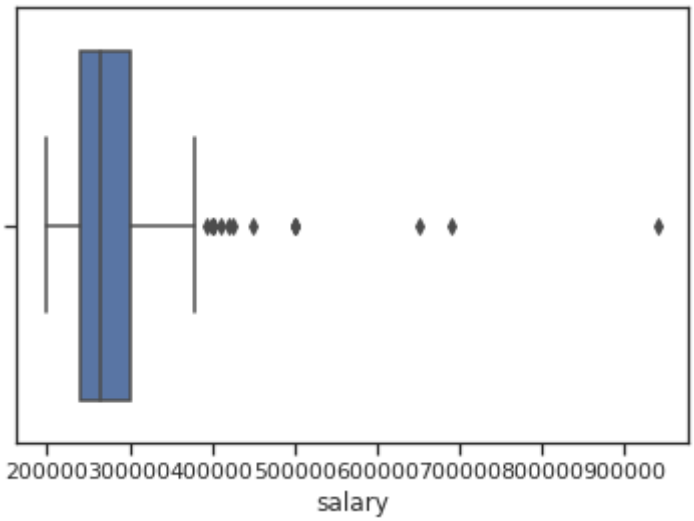


Ящик с усами

Отображает одномерное распределение вероятности.

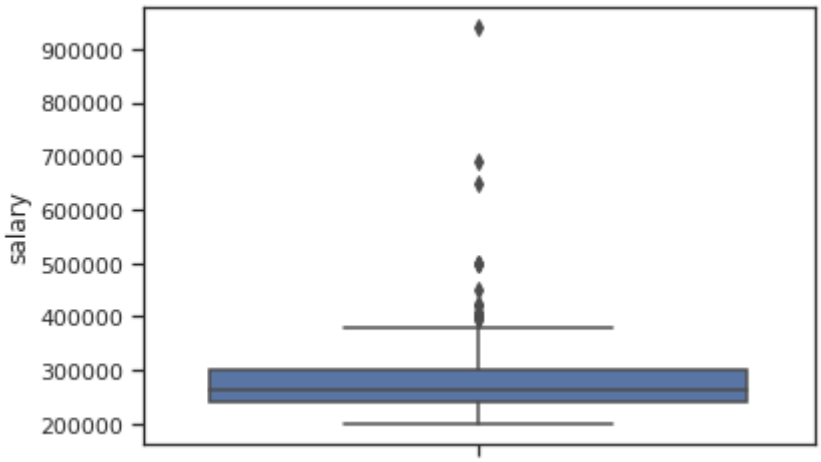
In [32]:
```
sns.boxplot(x=data['salary'])
```

Out[32]:
```
<AxesSubplot:xlabel='salary'>
```



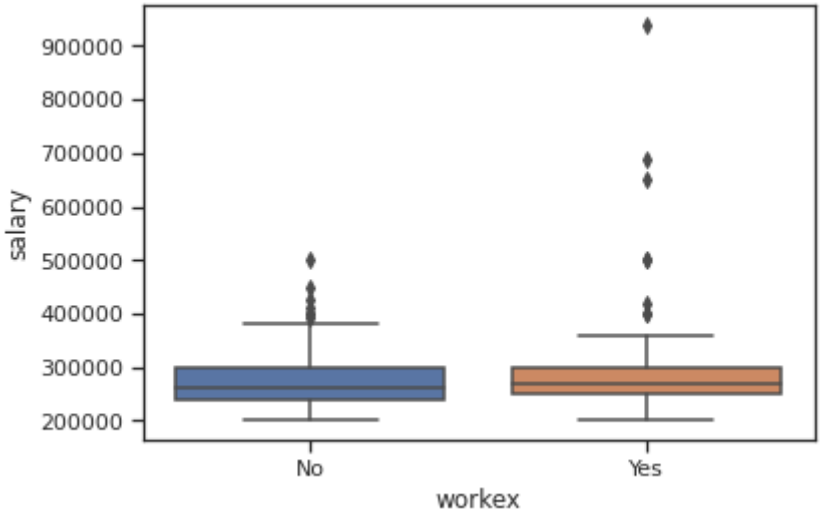
In [33]:
```
# По вертикали
sns.boxplot(y=data['salary'])
```

Out[33]:
```
<AxesSubplot:ylabel='salary'>
```



In [35]:
```
# Распределение параметра Humidity сгруппированные по Occupancy.
sns.boxplot(x='workex', y='salary', data=data)
```

Out[35]:
```
<AxesSubplot:xlabel='workex', ylabel='salary'>
```

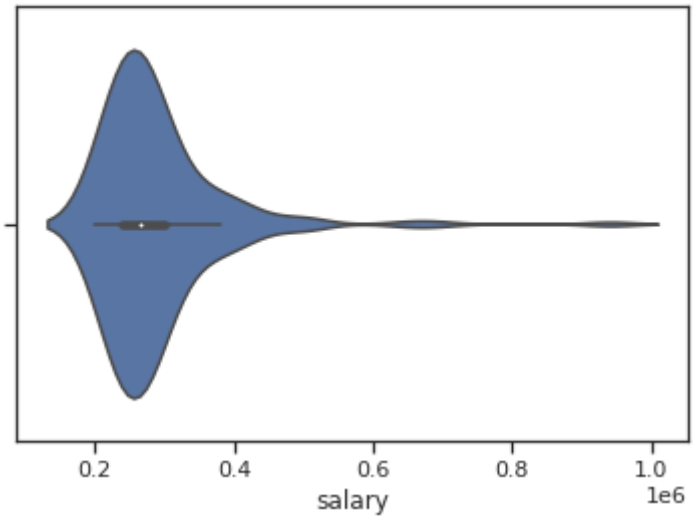


Violin plot

Похоже на предыдущую диаграмму, но по краям отображаются распределения плотности - https://en.wikipedia.org/wiki/Kernel_density_estimation

In [36]:
```
sns.violinplot(x=data['salary'])
```

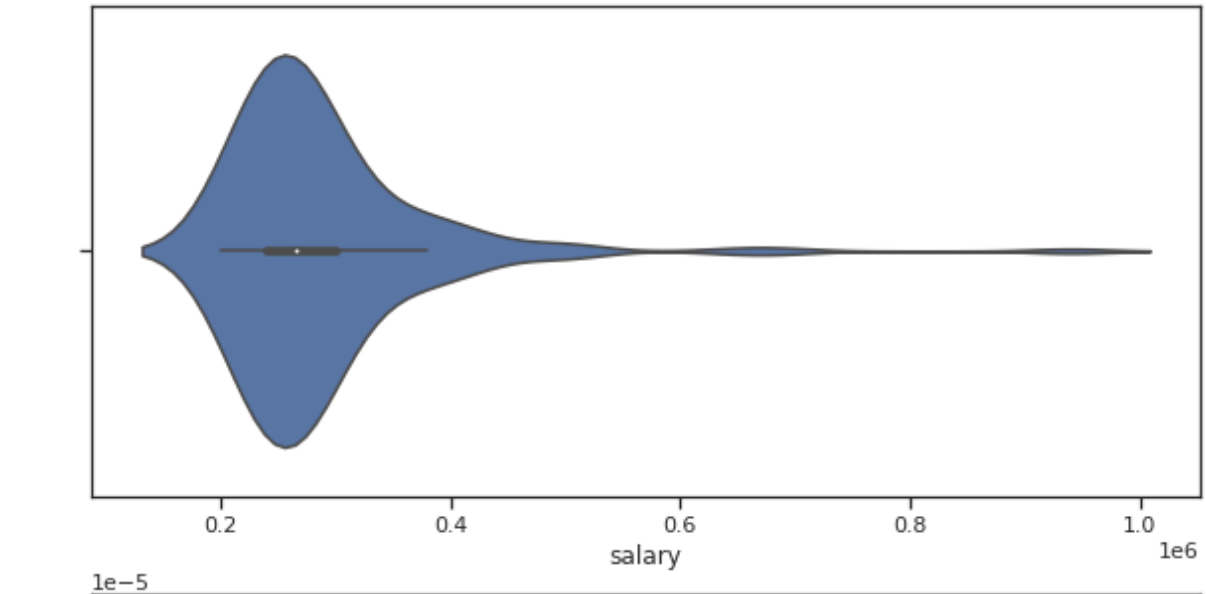
Out[36]:
```
<AxesSubplot:xlabel='salary'>
```

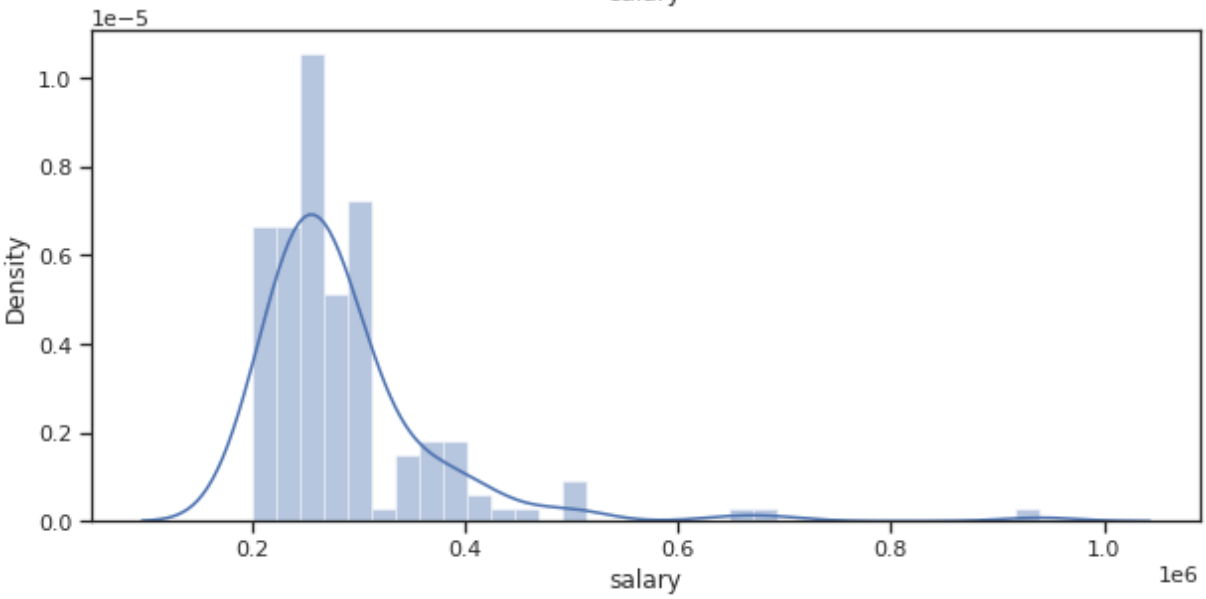


In [38]:
```
fig, ax = plt.subplots(2, 1, figsize=(10,10))
sns.violinplot(ax=ax[0], x=data['salary'])
sns.distplot(data['salary'], ax=ax[1])
```

```
/root/miniconda3/lib/python3.9/site-packages/seaborn/distributions.py:2619: FutureWa
rning: `distplot` is a deprecated function and will be removed in a future version.
Please adapt your code to use either `displot` (a figure-level function with similar
flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[38]:
```
<AxesSubplot:xlabel='salary', ylabel='Density'>
```

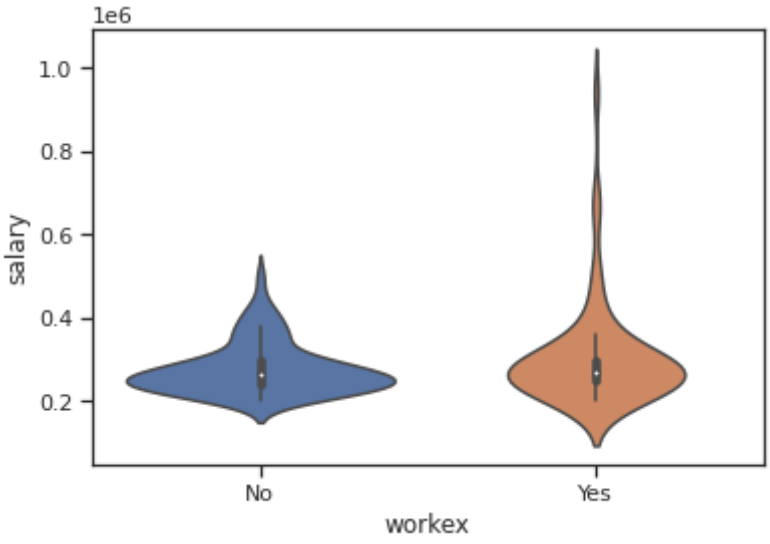




Из приведенных графиков видно, что violinplot действительно показывает распределение плотности.

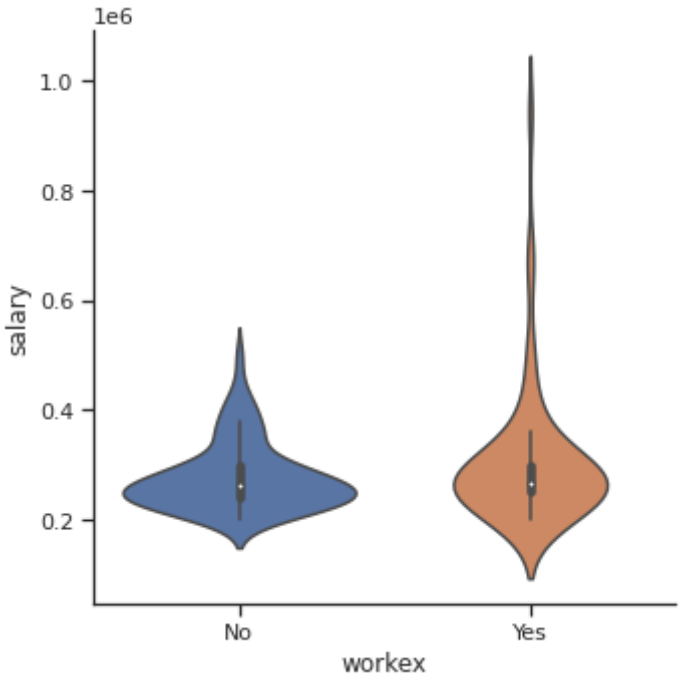
In [43]:
```
# Распределение параметра salary сгруппированные по workex.
sns.violinplot(x='workex', y='salary', data=data)
```

Out[43]:
```
<AxesSubplot:xlabel='workex', ylabel='salary'>
```



In [42]:
```
sns.catplot(y='salary', x='workex', data=data, kind="violin", split=True)
```

Out[42]:
```
<seaborn.axisgrid.FacetGrid at 0x7f4d943167c0>
```



4) Информация о корреляции признаков

In [45]:
```
data.corr()
```

Out[45]:

	sl_no	ssc_p	hsc_p	degree_p	etest_p	mba_p	salary
sl_no	1.000000	-0.078155	-0.085711	-0.088281	0.063636	0.022327	0.063764
ssc_p	-0.078155	1.000000	0.511472	0.538404	0.261993	0.388478	0.035330
hsc_p	-0.085711	0.511472	1.000000	0.434206	0.245113	0.354823	0.076819
degree_p	-0.088281	0.538404	0.434206	1.000000	0.224470	0.402364	-0.019272
etest_p	0.063636	0.261993	0.245113	0.224470	1.000000	0.218055	0.178307
mba_p	0.022327	0.388478	0.354823	0.402364	0.218055	1.000000	0.175013
salary	0.063764	0.035330	0.076819	-0.019272	0.178307	0.175013	1.000000

Корреляционная матрица содержит коэффициенты корреляции между всеми парами признаков.

Корреляционная матрица симметрична относительно главной диагонали. На главной диагонали расположены единицы (корреляция признака самого с собой).

Описание метода corr - https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.corr.html

По умолчанию при построении матрицы используется коэффициент корреляции Пирсона. Возможно также построить корреляционную матрицу на основе коэффициентов

корреляции Кендалла и Спирмена. На практике три метода редко дают значимые различия.

In [46]:
```
data.corr(method='pearson')
```

Out[46]:

	sl_no	ssc_p	hsc_p	degree_p	etest_p	mba_p	salary
sl_no	1.000000	-0.078155	-0.085711	-0.088281	0.063636	0.022327	0.063764
ssc_p	-0.078155	1.000000	0.511472	0.538404	0.261993	0.388478	0.035330
hsc_p	-0.085711	0.511472	1.000000	0.434206	0.245113	0.354823	0.076819
degree_p	-0.088281	0.538404	0.434206	1.000000	0.224470	0.402364	-0.019272
etest_p	0.063636	0.261993	0.245113	0.224470	1.000000	0.218055	0.178307
mba_p	0.022327	0.388478	0.354823	0.402364	0.218055	1.000000	0.175013
salary	0.063764	0.035330	0.076819	-0.019272	0.178307	0.175013	1.000000

In [47]:
```
data.corr(method='kendall')
```

Out[47]:

	sl_no	ssc_p	hsc_p	degree_p	etest_p	mba_p	salary
sl_no	1.000000	-0.049206	-0.083507	-0.084131	0.044573	0.004348	0.043050
ssc_p	-0.049206	1.000000	0.347797	0.394658	0.182721	0.274140	0.095484
hsc_p	-0.083507	0.347797	1.000000	0.310104	0.145742	0.211405	0.062210
degree_p	-0.084131	0.394658	0.310104	1.000000	0.134550	0.260387	-0.014889
etest_p	0.044573	0.182721	0.145742	0.134550	1.000000	0.145094	0.153130
mba_p	0.004348	0.274140	0.211405	0.260387	0.145094	1.000000	0.113096
salary	0.043050	0.095484	0.062210	-0.014889	0.153130	0.113096	1.000000

In [48]:
```
data.corr(method='spearman')
```

Out[48]:

	sl_no	ssc_p	hsc_p	degree_p	etest_p	mba_p	salary
sl_no	1.000000	-0.074884	-0.120679	-0.125622	0.058676	0.005274	0.059089
ssc_p	-0.074884	1.000000	0.490028	0.550469	0.270336	0.398587	0.142238
hsc_p	-0.120679	0.490028	1.000000	0.433140	0.214212	0.317190	0.094149
degree_p	-0.125622	0.550469	0.433140	1.000000	0.198741	0.379493	-0.026648
etest_p	0.058676	0.270336	0.214212	0.198741	1.000000	0.216701	0.225828
mba_p	0.005274	0.398587	0.317190	0.379493	0.216701	1.000000	0.154517
salary	0.059089	0.142238	0.094149	-0.026648	0.225828	0.154517	1.000000

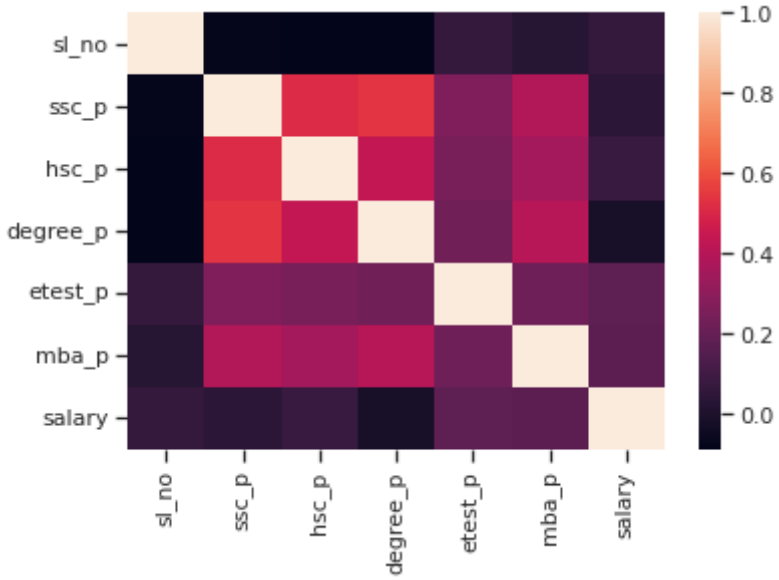
В случае большого количества признаков анализ числовой корреляционной матрицы становится неудобен.

Для визуализации корреляционной матрицы будем использовать "тепловую карту" heatmap которая показывает степень корреляции различными цветами.

Используем метод heatmap библиотеки seaborn - https://seaborn.pydata.org/generated/seaborn.heatmap.html

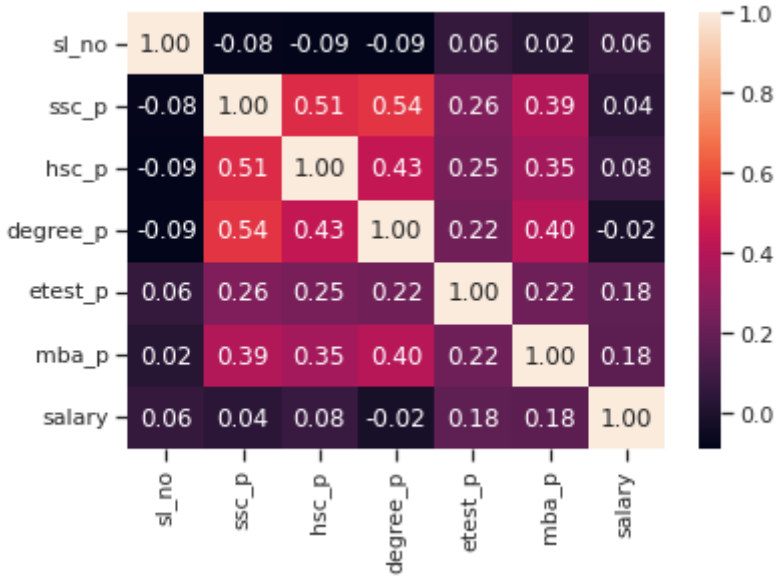
In [49]:
```
sns.heatmap(data.corr())
```

Out[49]:
```
<AxesSubplot:>
```



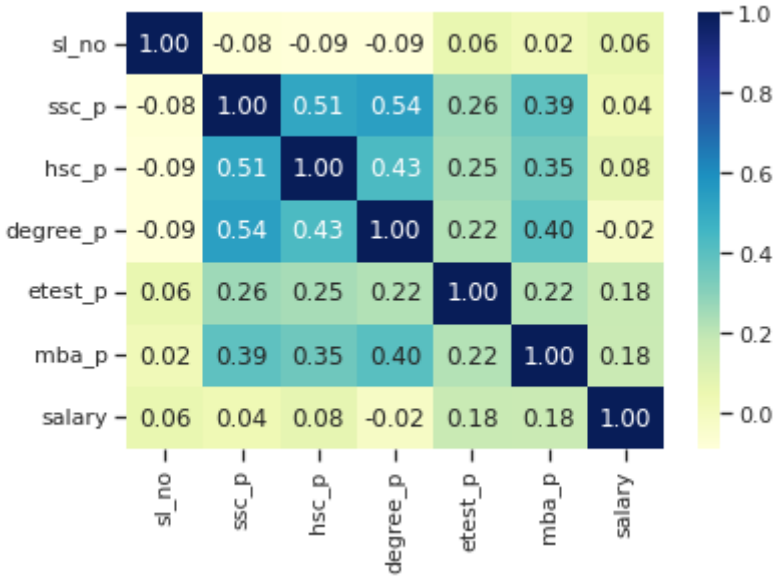
In [52]:
```
# Вывод значений в ячейках
sns.heatmap(data.corr(), annot=True, fmt='.2f')
```

Out[52]:
```
<AxesSubplot:>
```



In [53]:
```
# Изменение цветовой гаммы
sns.heatmap(data.corr(), cmap='YlGnBu', annot=True, fmt='.2f')
```

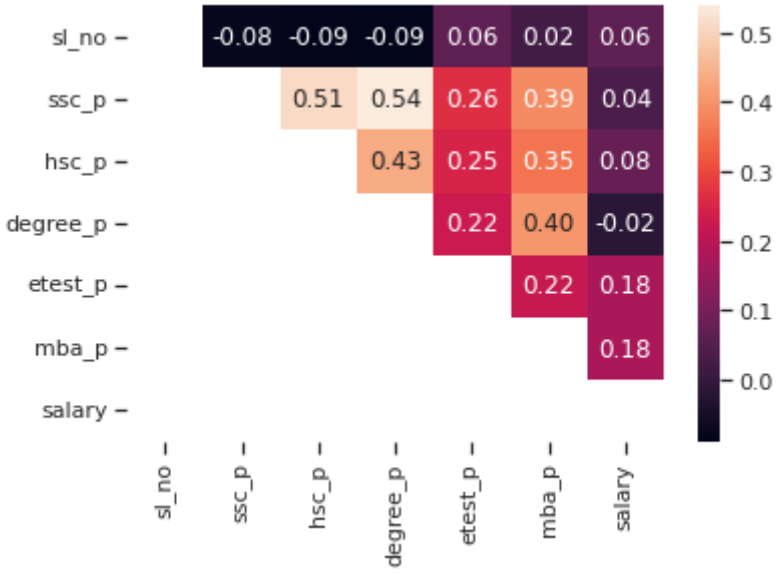
Out[53]:
```
<AxesSubplot:>
```



In [54]:

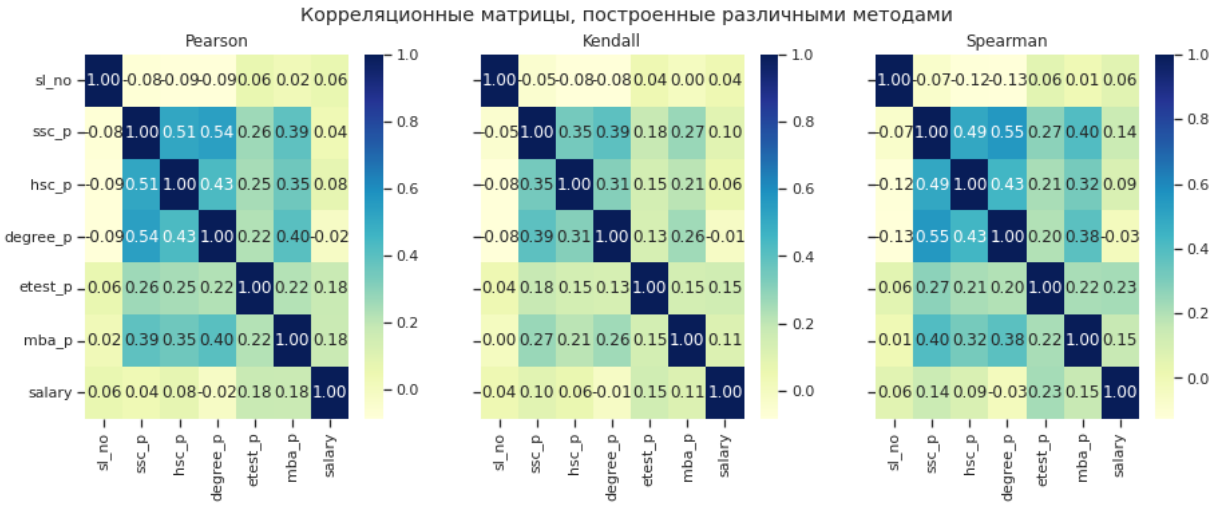
```
# Треугольный вариант матрицы
mask = np.zeros_like(data.corr(), dtype=np.bool)
# чтобы оставить нижнюю часть матрицы
# mask[np.triu_indices_from(mask)] = True
# чтобы оставить верхнюю часть матрицы
mask[np.tril_indices_from(mask)] = True
sns.heatmap(data.corr(), mask=mask, annot=True, fmt='.2f')
```

Out[54]: <AxesSubplot:>



In [56]:

```
fig, ax = plt.subplots(1, 3, sharex='col', sharey='row', figsize=(15,5))
sns.heatmap(data.corr(method='pearson'),cmap='YlGnBu', ax=ax[0], annot=True, fmt='.2
sns.heatmap(data.corr(method='kendall'),cmap='YlGnBu', ax=ax[1], annot=True, fmt='.2
sns.heatmap(data.corr(method='spearman'),cmap='YlGnBu', ax=ax[2], annot=True, fmt='.
fig.suptitle('Корреляционные матрицы, построенные различными методами')
ax[0].title.set_text('Pearson')
ax[1].title.set_text('Kendall')
ax[2].title.set_text('Spearman')
```



Необходимо отметить, что тепловая карта не очень хорошо подходит для определения корреляции нецелевых признаков между собой.