Московский государственный технический
университет им. Н. Э. Баумана

Факультет «Радиотехнический»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Технологии машинного обучения»

Отчет по лабораторной работе №1
«Разведочный анализ данных. Исследование и визуализация данных»

Выполнил:
студент группы РТ5-61Б

Алиев Тимур
Подпись и дата:

Проверил:
доцент каф. ИУ5

Гапанюк Ю.Е.
Подпись и дата:

Москва, 2022 г.

Описание задания

- Выбрать набор данных (датасет).
- Для первой лабораторной работы рекомендуется использовать датасет без пропусков в данных, например из Scikit-learn.

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.

Jupyter notebook

Разведочный анализ данных. Исследование и визуализация данных.

1) Текстовое описание набора данных

В качестве набора данных мы будем использовать набор данных для распознавания вин - https://scikit-learn.org/stable/datasets/toy_dataset.html

Эта задача является очень актуальной для продавцов вина, которым необходимо найти нужное вино.

Количество экземпляров: 178

Сорта:

- class_0 (59)
- class_1 (71)
- class_2 (48)

Информация об атрибутах:

- Alcohol - алкоголь
- Malic acid - яблочная кислота
- Ash - пепел
- Alcalinity of ash - щелочность пепла
- Magnesium - магний
- Total phenols - всего фенолов
- Flavanoids - флавоноиды
- Nonflavanoid phenols - нефлаваноидные фенолы
- Proanthocyanins - проантоцианы
- Color intensity - интенсивность цвета
- Hue - оттенок
- OD280/OD315 of diluted wines - OD280/OD315 разбавленных вин
- Proline - пролин

Импорт библиотек

Импортируем библиотеки с помощью команды import.

In [1]:
```
!pip install numpy
import numpy as np
!pip install pandas
import pandas as pd
!pip install sklearn
from sklearn.datasets import *
!pip install seaborn
import seaborn as sns
!pip install matplotlib
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")
```

```
Requirement already satisfied: numpy in c:\users\truma\appdata\local\programs\python\python310\lib\site-packages (1.22.3)
Requirement already satisfied: pandas in c:\users\truma\appdata\local\programs\python\python310\lib\site-packages (1.4.1)
Requirement already satisfied: numpy>=1.21.0 in c:\users\truma\appdata\local\programs\python\python310\lib\site-packages (from pandas) (1.22.3)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\truma\appdata\local\programs\python\python310\lib\site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\truma\appdata\local\programs\python\python310\lib\site-packages (from pandas) (2021.3)
Requirement already satisfied: six>=1.5 in c:\users\truma\appdata\local\programs\python\python310\lib\site-packages (from python-dateutil>=2.8.1->panda
s) (1.16.0)
Requirement already satisfied: sklearn in c:\users\truma\appdata\local\programs\python\python310\lib\site-packages (0.0)
Requirement already satisfied: scikit-learn in c:\users\truma\appdata\local\programs\python\python310\lib\site-packages (from sklearn) (1.0.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\truma\appdata\local\programs\python\python310\lib\site-packages (from scikit-learn->skle
arn) (3.1.0)
Requirement already satisfied: scipy>=1.1.0 in c:\users\truma\appdata\local\programs\python\python310\lib\site-packages (from scikit-learn->sklearn) (1.
8.0)
Requirement already satisfied: numpy>=1.14.6 in c:\users\truma\appdata\local\programs\python\python310\lib\site-packages (from scikit-learn->sklearn)
(1.22.3)
Requirement already satisfied: joblib>=0.11 in c:\users\truma\appdata\local\programs\python\python310\lib\site-packages (from scikit-learn->sklearn) (1.
1.0)
Requirement already satisfied: seaborn in c:\users\truma\appdata\local\programs\python\python310\lib\site-packages (0.11.2)
Requirement already satisfied: numpy>=1.15 in c:\users\truma\appdata\local\programs\python\python310\lib\site-packages (from seaborn) (1.22.3)
Requirement already satisfied: scipy>=1.0 in c:\users\truma\appdata\local\programs\python\python310\lib\site-packages (from seaborn) (1.8.0)
Requirement already satisfied: pandas>=0.23 in c:\users\truma\appdata\local\programs\python\python310\lib\site-packages (from seaborn) (1.4.1)
Requirement already satisfied: matplotlib>=2.2 in c:\users\truma\appdata\local\programs\python\python310\lib\site-packages (from seaborn) (3.5.1)
Requirement already satisfied: packaging>=20.0 in c:\users\truma\appdata\local\programs\python\python310\lib\site-packages (from matplotlib>=2.2->seabor
n) (21.3)
Requirement already satisfied: pillow>=6.2.0 in c:\users\truma\appdata\local\programs\python\python310\lib\site-packages (from matplotlib>=2.2->seaborn)
(9.0.1)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\truma\appdata\local\programs\python\python310\lib\site-packages (from matplotlib>=2.2->seabo
rn) (3.0.7)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\truma\appdata\local\programs\python\python310\lib\site-packages (from matplotlib>=2.2->seab
orn) (4.30.0)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\truma\appdata\local\programs\python\python310\lib\site-packages (from matplotlib>=2.2->s
eaborn) (2.8.2)
Requirement already satisfied: cycler>=0.10 in c:\users\truma\appdata\local\programs\python\python310\lib\site-packages (from matplotlib>=2.2->seaborn)
(0.11.0)

Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\truma\appdata\local\programs\python\python310\lib\site-packages (from matplotlib>=2.2->seab
orn) (1.3.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\truma\appdata\local\programs\python\python310\lib\site-packages (from pandas>=0.23->seaborn) (20
21.3)
Requirement already satisfied: six>=1.5 in c:\users\truma\appdata\local\programs\python\python310\lib\site-packages (from python-dateutil>=2.7->matplotl
ib>=2.2->seaborn) (1.16.0)
Requirement already satisfied: matplotlib in c:\users\truma\appdata\local\programs\python\python310\lib\site-packages (3.5.1)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\truma\appdata\local\programs\python\python310\lib\site-packages (from matplotlib) (1.3.2)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\truma\appdata\local\programs\python\python310\lib\site-packages (from matplotlib) (4.30.0)
Requirement already satisfied: cycler>=0.10 in c:\users\truma\appdata\local\programs\python\python310\lib\site-packages (from matplotlib) (0.11.0)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\truma\appdata\local\programs\python\python310\lib\site-packages (from matplotlib) (2.8.
2)
Requirement already satisfied: numpy>=1.17 in c:\users\truma\appdata\local\programs\python\python310\lib\site-packages (from matplotlib) (1.22.3)
Requirement already satisfied: pillow>=6.2.0 in c:\users\truma\appdata\local\programs\python\python310\lib\site-packages (from matplotlib) (9.0.1)
Requirement already satisfied: packaging>=20.0 in c:\users\truma\appdata\local\programs\python\python310\lib\site-packages (from matplotlib) (21.3)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\truma\appdata\local\programs\python\python310\lib\site-packages (from matplotlib) (3.0.7)
Requirement already satisfied: six>=1.5 in c:\users\truma\appdata\local\programs\python\python310\lib\site-packages (from python-dateutil>=2.7->matplotl
ib) (1.16.0)
```

Загрузка данных

Загрузим файлы датасета в помощью библиотеки Pandas и Sklearn.

Нужный нам датасет можно загрузить с помощью команды:

In [2]:
```
wine = load_wine()
type(wine)
```

Out[2]:
```
sklearn.utils.Bunch
```

Преобразование датасета Scikit-learn в Pandas Dataframe:

In [3]:
```
data1 = pd.DataFrame(data= np.c_[wine['data'], wine['target']],
                     columns= wine['feature_names'] + ['target'])
```

2) Основные характеристики датасета

Датасет возвращается в виде словаря с ключами, которые можно получить, выполнив следующую команду:

In [4]:
```
for x in wine:
    print(x)
```

```
data
target
frame
target_names
DESCR
feature_names
```

In [5]:
```
# Названиe сортов:
print(wine['target_names'])
```

```
['class_0' 'class_1' 'class_2']
```

In [6]:
```
# Список колонок с типами данных
data1.dtypes
```

Out[6]:
```
alcohol                         float64
malic_acid                      float64
ash                             float64
alcalinity_of_ash               float64
magnesium                       float64
total_phenols                   float64
flavanoids                      float64
nonflavanoid_phenols            float64
proanthocyanins                 float64
color_intensity                 float64
hue                             float64
od280/od315_of_diluted_wines    float64
proline                         float64
target                          float64
dtype: object
```

In [7]:
```
# Размерность данных
print(wine['data'].shape)
```

```
(178, 13)
```

In [8]:
```
# Размерность целевого признака
print(wine['target'].shape)
```

```
(178,)
```

In [9]:
```
# Проверим наличие пустых значений
# Цикл по колонкам датасета
for col in data1.columns:
    # Количество пустых значений - все значения заполнены
    temp_null_count = data1[data1[col].isnull()].shape[0]
    print('{} - {}'.format(col, temp_null_count))
```

```
alcohol - 0
malic_acid - 0
ash - 0
alcalinity_of_ash - 0
magnesium - 0
total_phenols - 0
flavanoids - 0
nonflavanoid_phenols - 0
proanthocyanins - 0
color_intensity - 0
hue - 0
od280/od315_of_diluted_wines - 0
proline - 0
target - 0
```

In [10]:
```
# Основные статистические характеристки набора данных
data1.describe()
```

Out[10]:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	hue	od280/od31
count	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	
mean	13.000618	2.336348	2.366517	19.494944	99.741573	2.295112	2.029270	0.361854	1.590899	5.058090	0.957449	
std	0.811827	1.117146	0.274344	3.339564	14.282484	0.625851	0.998859	0.124453	0.572359	2.318286	0.228572	
min	11.030000	0.740000	1.360000	10.600000	70.000000	0.980000	0.340000	0.130000	0.410000	1.280000	0.480000	
25%	12.362500	1.602500	2.210000	17.200000	88.000000	1.742500	1.205000	0.270000	1.250000	3.220000	0.782500	
50%	13.050000	1.865000	2.360000	19.500000	98.000000	2.355000	2.135000	0.340000	1.555000	4.690000	0.965000	
75%	13.677500	3.082500	2.557500	21.500000	107.000000	2.800000	2.875000	0.437500	1.950000	6.200000	1.120000	
max	14.830000	5.800000	3.230000	30.000000	162.000000	3.880000	5.080000	0.660000	3.580000	13.000000	1.710000	

3) Визуальное исследование датасета

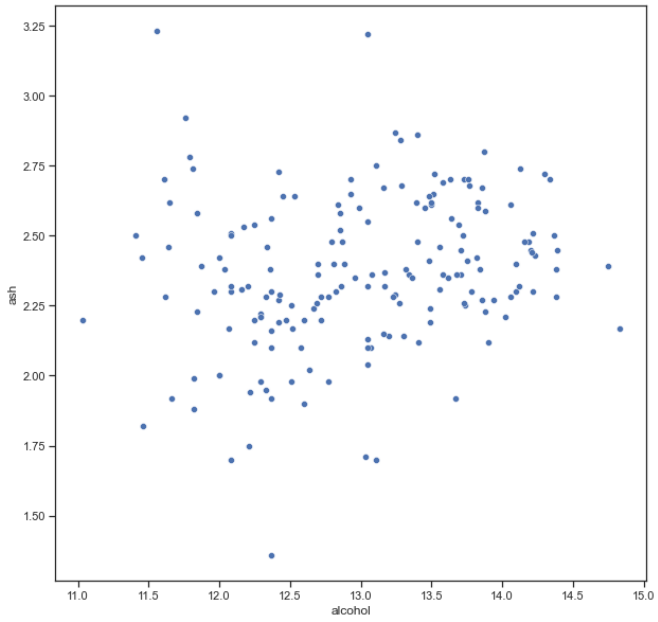
Для визуального исследования могут быть использованы различные виды диаграмм, мы построим только некоторые варианты диаграмм, которые используются достаточно часто.

Диаграмма рассеяния

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

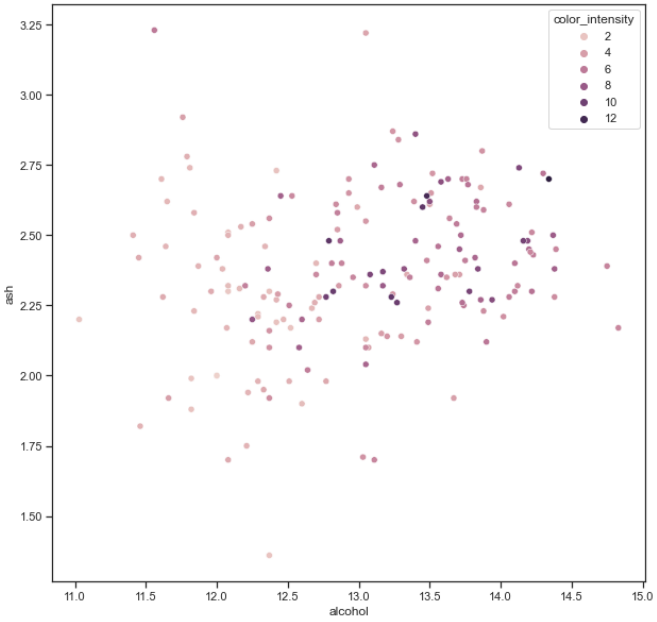
In [11]:
```
fig, ax = plt.subplots(figsize=(10,10))
sns.scatterplot(ax=ax, x='alcohol', y='ash', data=data1)
```

Out[11]:
```
<AxesSubplot:xlabel='alcohol', ylabel='ash'>
```



In [12]:
```
fig, ax = plt.subplots(figsize=(10,10))
sns.scatterplot(ax=ax, x='alcohol', y='ash', data=data1, hue='color_intensity')
```

Out[12]:
```
<AxesSubplot:xlabel='alcohol', ylabel='ash'>
```

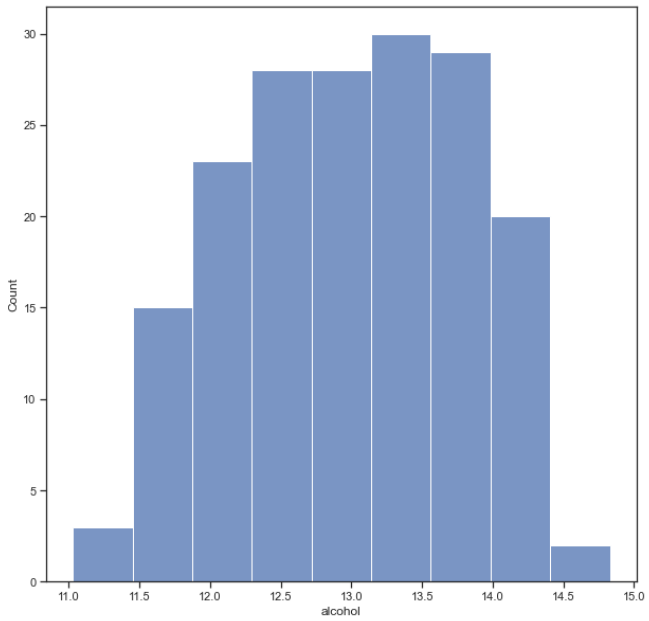


Гистограмма

Позволяет оценить плотность вероятности распределения данных.

In [13]:
```
fig, ax = plt.subplots(figsize=(10,10))
sns.histplot(data1['alcohol'])
```

Out[13]:
```
<AxesSubplot:xlabel='alcohol', ylabel='Count'>
```

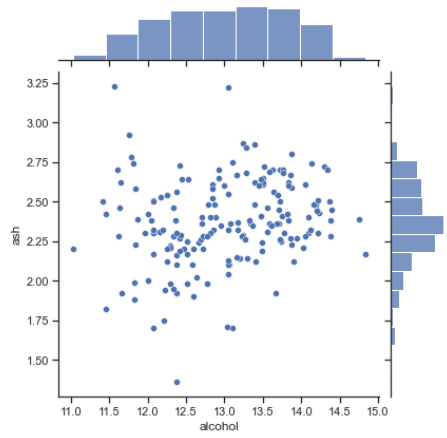


Jointplot

Комбинация гистограмм и диаграмм рассеивания.

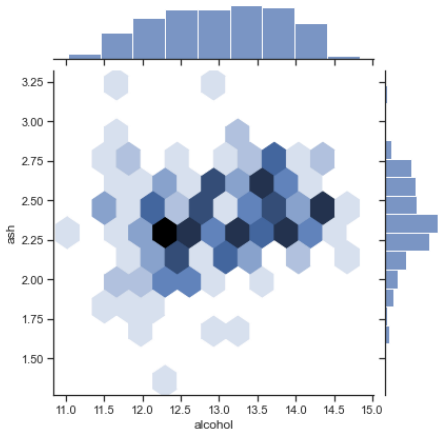
In [14]:
```
sns.jointplot(x='alcohol', y='ash', data=data1)
```

Out[14]:
```
<seaborn.axisgrid.JointGrid at 0x1f6c7e76fe0>
```



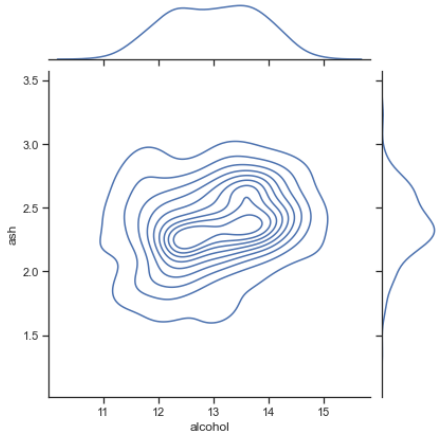
In [15]:
```
sns.jointplot(x='alcohol', y='ash', data=data1, kind="hex")
```

Out[15]:
```
<seaborn.axisgrid.JointGrid at 0x1f6c7fcf5b0>
```



In [16]:
```
sns.jointplot(x='alcohol', y='ash', data=data1, kind="kde")
```

Out[16]:
```
<seaborn.axisgrid.JointGrid at 0x1f6c7e77bb0>
```



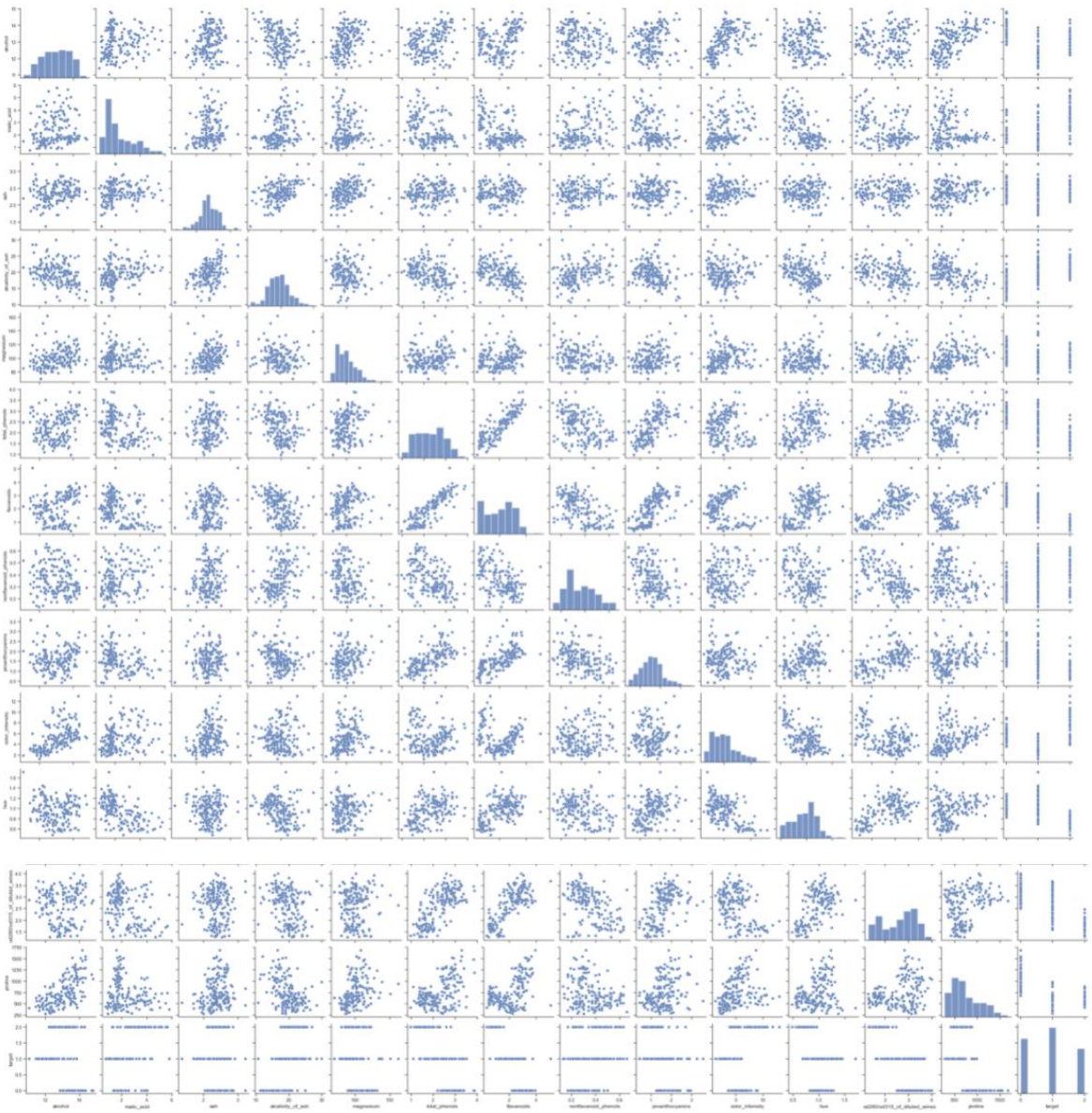
"Парные диаграммы"

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 [17]:
```
sns.pairplot(data1)
```

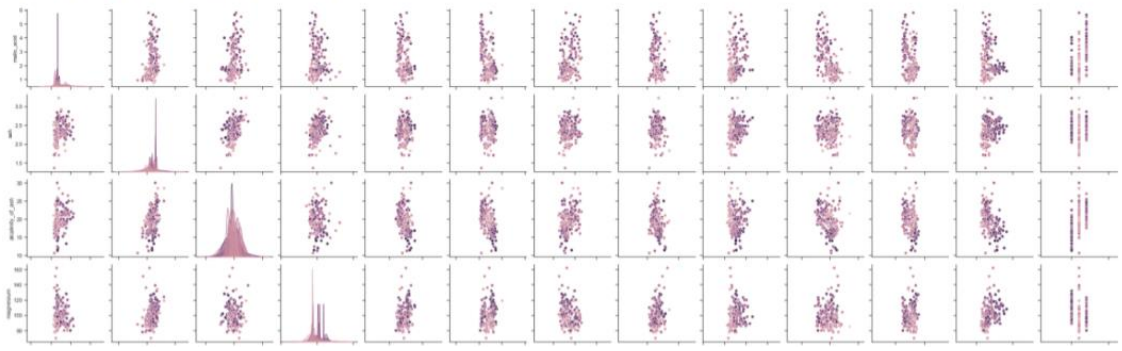
Out[17]: <seaborn.axisgrid.PairGrid at 0x1f6c87d9510>



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

In [18]:
```
sns.pairplot(data1, hue="alcohol")
```

Out[18]: <seaborn.axisgrid.PairGrid at 0x1f6c87c3310>

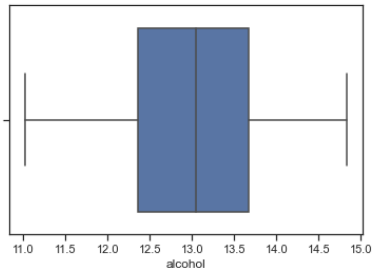


Ящик с усами

Отображает одномерное распределение вероятности.

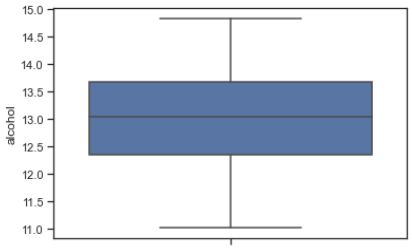
In [19]:
```
sns.boxplot(x=data1['alcohol'])
```

Out[19]:
```
<AxesSubplot:xlabel='alcohol'>
```



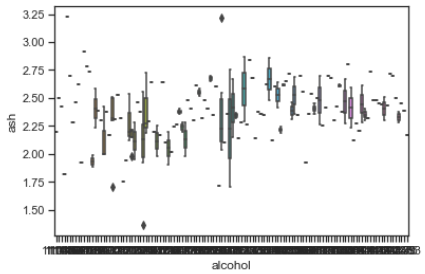
In [20]:
```
# По вертикали
sns.boxplot(y=data1['alcohol'])
```

Out[20]:
```
<AxesSubplot:ylabel='alcohol'>
```



In [21]:
```
# Распределение параметра alcohol сгруппированные по ash.
sns.boxplot(x='alcohol', y='ash', data=data1)
```

Out[21]:
```
<AxesSubplot:xlabel='alcohol', ylabel='ash'>
```

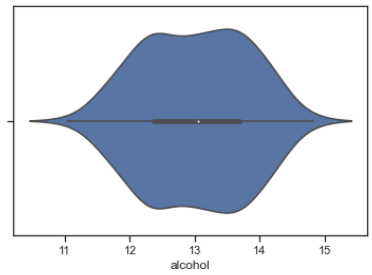


Violin plot

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

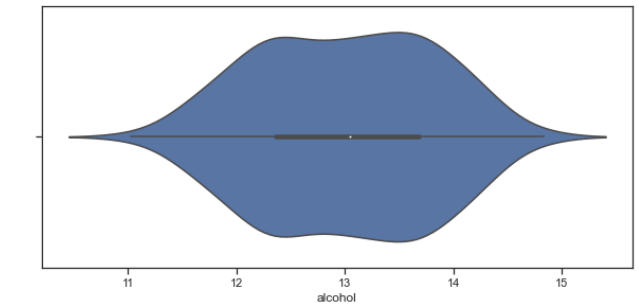
In [22]:
```
sns.violinplot(x=data1['alcohol'])
```

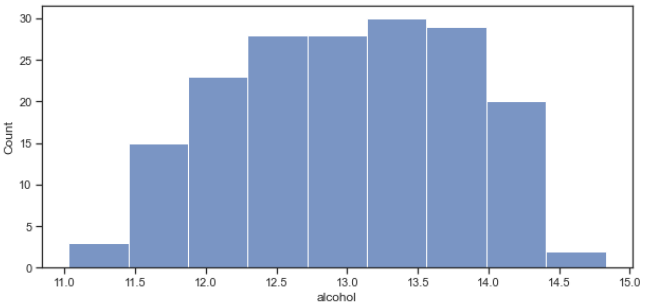
Out[22]:
```
<AxesSubplot:xlabel='alcohol'>
```



In [23]:
```
fig, ax = plt.subplots(2, 1, figsize=(10,10))
sns.violinplot(ax=ax[0], x=data1['alcohol'])
sns.histplot(data1['alcohol'], ax=ax[1])
```

Out[23]:
```
<AxesSubplot:xlabel='alcohol', ylabel='Count'>
```

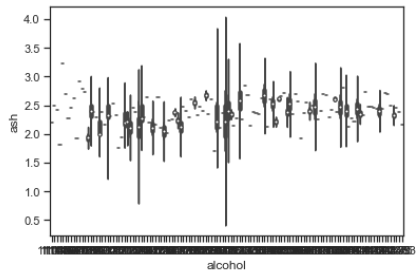




Из приведенных графиков видно, что violinplot действительно показывает распределение плотности.

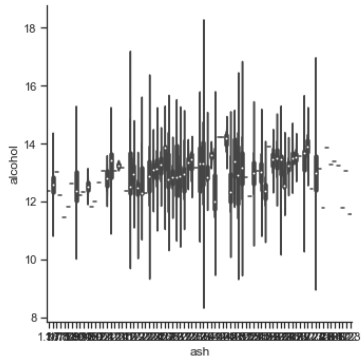
In [24]:
```
# Распределение параметра alcohol сгруппированные по ash.
sns.violinplot(x='alcohol', y='ash', data=data1)
```

Out[24]:
```
<AxesSubplot:xlabel='alcohol', ylabel='ash'>
```



In [25]:
```
sns.catplot(y='alcohol', x='ash', data=data1, kind="violin", split=True)
```

Out[25]:
```
<seaborn.axisgrid.FacetGrid at 0x1f6df9192a0>
```



4) Информация о корреляции признаков

Проверка корреляции признаков позволяет решить две задачи:

Понять какие признаки (колонки датасета) наиболее сильно коррелируют с целевым признаком (в нашем примере это колонка "alcohol"). Именно эти признаки будут наиболее информативными для моделей машинного обучения. Признаки, которые слабо коррелируют с целевым признаком, можно попробовать исключить из построения модели, иногда это повышает качество модели. Нужно отметить, что некоторые алгоритмы машинного обучения автоматически определяют ценность того или иного признака для построения модели. Понять какие нецелевые признаки линейно зависимы между собой. Линейно зависимые признаки, как правило, очень плохо влияют на качество моделей. Поэтому если несколько признаков линейно зависимы, то для построения модели из них выбирают какой-то один признак.

In [26]:
```
data1.corr()
```

Out[26]:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	
alcohol	1.000000	0.094397	0.211545	-0.310235	0.270798	0.289101	0.236815	-0.155929	0.136698	0.546364	-0.07
malic_acid	0.094397	1.000000	0.164045	0.288500	-0.054575	-0.335167	-0.411007	0.292977	-0.220746	0.248985	-0.56
ash	0.211545	0.164045	1.000000	0.443367	0.286587	0.128980	0.115077	0.186230	0.009652	0.258887	-0.07
alcalinity_of_ash	-0.310235	0.288500	0.443367	1.000000	-0.083333	-0.321113	-0.351370	0.361922	-0.197327	0.018732	-0.27
magnesium	0.270798	-0.054575	0.286587	-0.083333	1.000000	0.214401	0.195784	-0.256294	0.236441	0.199950	0.05
total_phenols	0.289101	-0.335167	0.128980	-0.321113	0.214401	1.000000	0.864564	-0.449935	0.612413	-0.055136	0.43
flavanoids	0.236815	-0.411007	0.115077	-0.351370	0.195784	0.864564	1.000000	-0.537900	0.652692	-0.172379	0.54
nonflavanoid_phenols	-0.155929	0.292977	0.186230	0.361922	-0.256294	-0.449935	-0.537900	1.000000	-0.365845	0.139057	-0.26
proanthocyanins	0.136698	-0.220746	0.009652	-0.197327	0.236441	0.612413	0.652692	-0.365845	1.000000	-0.025250	0.29
color_intensity	0.546364	0.248985	0.258887	0.018732	0.199950	-0.055136	-0.172379	0.139057	-0.025250	1.000000	-0.52
hue	-0.071747	-0.561296	-0.074667	-0.273955	0.055398	0.433681	0.543479	-0.262640	0.295544	-0.521813	1.00
od280/od315_of_diluted_wines	0.072343	-0.368710	0.003911	-0.276769	0.066004	0.699949	0.787194	-0.503270	0.519067	-0.428815	0.56
proline	0.643720	-0.192011	0.223626	-0.440597	0.393351	0.498115	0.494193	-0.311385	0.330417	0.316100	0.23
target	-0.328222	0.437776	-0.049643	0.517859	-0.209179	-0.719163	-0.847498	0.489109	-0.499130	0.265668	-0.61

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 [27]:
```
data1.corr(method='pearson')
```

Out[27]:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	
alcohol	1.000000	0.094397	0.211545	-0.310235	0.270798	0.289101	0.236815	-0.155929	0.136698	0.546364	-0.07
malic_acid	0.094397	1.000000	0.164045	0.288500	-0.054575	-0.335167	-0.411007	0.292977	-0.220746	0.248985	-0.56
ash	0.211545	0.164045	1.000000	0.443367	0.286587	0.128980	0.115077	0.186230	0.009652	0.258887	-0.07
alcalinity_of_ash	-0.310235	0.288500	0.443367	1.000000	-0.083333	-0.321113	-0.351370	0.361922	-0.197327	0.018732	-0.27
magnesium	0.270798	-0.054575	0.286587	-0.083333	1.000000	0.214401	0.195784	-0.256294	0.236441	0.199950	0.05
total_phenols	0.289101	-0.335167	0.128980	-0.321113	0.214401	1.000000	0.864564	-0.449935	0.612413	-0.055136	0.43
flavanoids	0.236815	-0.411007	0.115077	-0.351370	0.195784	0.864564	1.000000	-0.537900	0.652692	-0.172379	0.54
nonflavanoid_phenols	-0.155929	0.292977	0.186230	0.361922	-0.256294	-0.449935	-0.537900	1.000000	-0.365845	0.139057	-0.26
proanthocyanins	0.136698	-0.220746	0.009652	-0.197327	0.236441	0.612413	0.652692	-0.365845	1.000000	-0.025250	0.29
color_intensity	0.546364	0.248985	0.258887	0.018732	0.199950	-0.055136	-0.172379	0.139057	-0.025250	1.000000	-0.52
hue	-0.071747	-0.561296	-0.074667	-0.273955	0.055398	0.433681	0.543479	-0.262640	0.295544	-0.521813	1.00
od280/od315_of_diluted_wines	0.072343	-0.368710	0.003911	-0.276769	0.066004	0.699949	0.787194	-0.503270	0.519067	-0.428815	0.56
proline	0.643720	-0.192011	0.223626	-0.440597	0.393351	0.498115	0.494193	-0.311385	0.330417	0.316100	0.23
target	-0.328222	0.437776	-0.049643	0.517859	-0.209179	-0.719163	-0.847498	0.489109	-0.499130	0.265668	-0.61

In [28]:
```
data1.corr(method='kendall')
```

Out[28]:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	
alcohol	1.000000	0.093844	0.170154	-0.212978	0.250506	0.209099	0.191087	-0.109554	0.133526	0.434353	-0.02
malic_acid	0.093844	1.000000	0.158178	0.210119	0.050869	-0.174929	-0.211918	0.175129	-0.168714	0.195607	-0.38
ash	0.170154	0.158178	1.000000	0.258352	0.254246	0.089855	0.049474	0.098937	0.018240	0.187786	-0.03
alcalinity_of_ash	-0.212978	0.210119	0.258352	1.000000	-0.121005	-0.256669	-0.309865	0.278091	-0.171404	-0.057281	-0.23
magnesium	0.250506	0.050869	0.254246	-0.121005	1.000000	0.172195	0.161603	-0.158361	0.117871	0.241781	0.02
total_phenols	0.209099	-0.174929	0.089855	-0.256669	0.172195	1.000000	0.701999	-0.310443	0.466517	0.028264	0.28
flavanoids	0.191087	-0.211918	0.049474	-0.309865	0.161603	0.701999	1.000000	-0.378099	0.534615	0.028674	0.35
nonflavanoid_phenols	-0.109554	0.175129	0.098937	0.278091	-0.158361	-0.310443	-0.378099	1.000000	-0.269189	0.036065	-0.17
proanthocyanins	0.133526	-0.168714	0.018240	-0.171404	0.117871	0.466517	0.534615	-0.269189	1.000000	-0.014962	0.23
color_intensity	0.434353	0.195607	0.187786	-0.057281	0.241781	0.028264	0.028674	0.036065	-0.014962	1.000000	-0.29
hue	-0.021717	-0.388707	-0.037234	-0.239210	0.023760	0.289210	0.354372	-0.179755	0.231071	-0.291561	1.00
od280/od315_of_diluted_wines	0.061513	-0.162909	-0.006341	-0.226253	0.034307	0.478267	0.520448	-0.363787	0.369104	-0.206046	0.32
proline	0.449387	-0.044660	0.171574	-0.313218	0.343016	0.280203	0.263661	-0.174108	0.204172	0.316632	0.14
target	-0.238984	0.247494	-0.038085	0.449402	-0.184992	-0.590404	-0.725255	0.379234	-0.450225	0.065124	-0.47

In [29]:
```
data1.corr(method='spearman')
```

Out[29]:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	
alcohol	1.000000	0.140430	0.243722	-0.306598	0.365503	0.310920	0.294740	-0.162207	0.192734	0.635425	-0.0
malic_acid	0.140430	1.000000	0.230674	0.304069	0.080188	-0.280225	-0.325202	0.255236	-0.244825	0.290307	-0.5
ash	0.243722	0.230674	1.000000	0.366374	0.361488	0.132193	0.078796	0.145583	0.024384	0.283047	-0.0
alcalinity_of_ash	-0.306598	0.304069	0.366374	1.000000	-0.169558	-0.376657	-0.443770	0.389390	-0.253695	-0.073776	-0.3
magnesium	0.365503	0.080188	0.361488	-0.169558	1.000000	0.246417	0.233167	-0.236786	0.173647	0.357029	0.0
total_phenols	0.310920	-0.280225	0.132193	-0.376657	0.246417	1.000000	0.879404	-0.448013	0.666689	0.011162	0.4
flavanoids	0.294740	-0.325202	0.078796	-0.443770	0.233167	0.879404	1.000000	-0.543897	0.730322	-0.042910	0.5
nonflavanoid_phenols	-0.162207	0.255236	0.145583	0.389390	-0.236786	-0.448013	-0.543897	1.000000	-0.384629	0.059639	-0.2
proanthocyanins	0.192734	-0.244825	0.024384	-0.253695	0.173647	0.666689	0.730322	-0.384629	1.000000	-0.030947	0.3
color_intensity	0.635425	0.290307	0.283047	-0.073776	0.357029	0.011162	-0.042910	0.059639	-0.030947	1.000000	-0.4
hue	-0.024203	-0.560265	-0.050183	-0.352507	0.036095	0.439457	0.535430	-0.267813	0.342795	-0.418522	1.0
od280/od315_of_diluted_wines	0.103050	-0.255185	-0.007500	-0.325890	0.056963	0.687207	0.741533	-0.494950	0.554031	-0.317516	0.4
proline	0.633580	-0.057466	0.253163	-0.456090	0.507575	0.419470	0.429904	-0.270112	0.308249	0.457096	0.2
target	-0.354167	0.346913	-0.053988	0.569792	-0.250498	-0.726544	-0.854908	0.474205	-0.570648	0.131170	-0.6

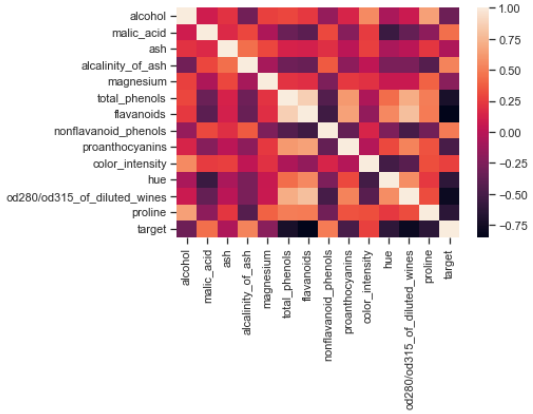
В случае большого количества признаков анализ числовой корреляционной матрицы становится неудобен.

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

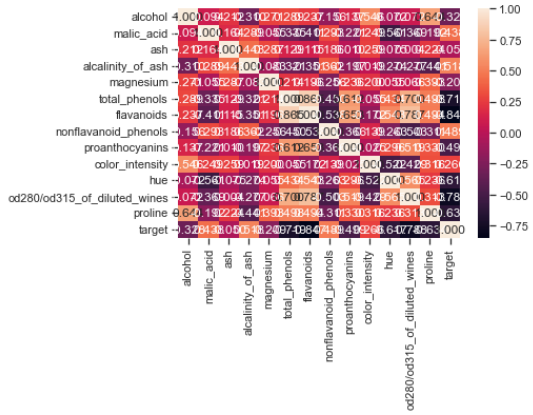
In [30]:
```
sns.heatmap(data1.corr())
```

Out[30]:
```
<AxesSubplot:>
```



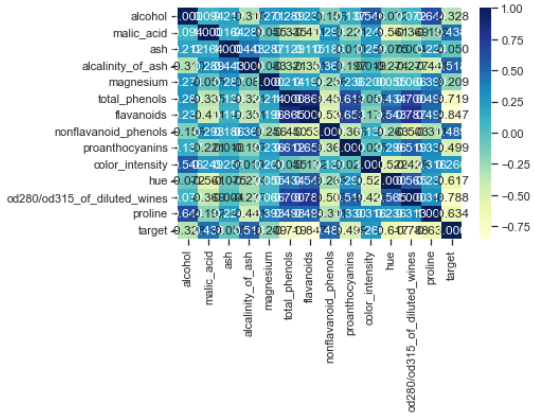
In [31]:
```
# Вывод значений в ячейках
sns.heatmap(data1.corr(), annot=True, fmt='.3f')
```

Out[31]:
```
<AxesSubplot:>
```



In [32]:
```
# Изменение цветовой гаммы
sns.heatmap(data1.corr(), cmap='YlGnBu', annot=True, fmt='.3f')
```

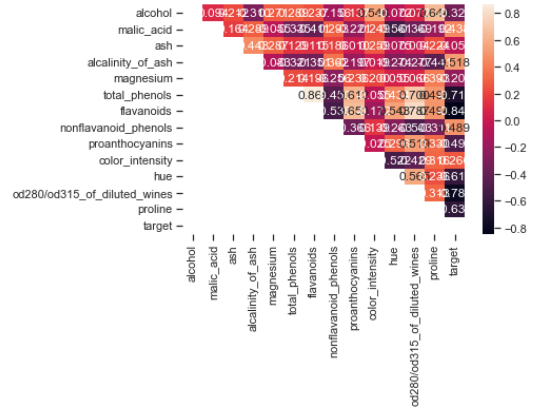
Out[32]:
```
<AxesSubplot:>
```



In [33]:

```
# Треугольный вариант матрицы
mask = np.zeros_like(data1.corr(), dtype=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(data1.corr(), mask=mask, annot=True, fmt='.3f')
```

Out[33]: <AxesSubplot:>



In [34]:

```
fig, ax = plt.subplots(1, 3, sharex='col', sharey='row', figsize=(15,5))
sns.heatmap(data1.corr(method='pearson'), ax=ax[0], annot=True, fmt='.2f')
sns.heatmap(data1.corr(method='kendall'), ax=ax[1], annot=True, fmt='.2f')
sns.heatmap(data1.corr(method='spearman'), ax=ax[2], annot=True, fmt='.2f')
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')
```