

Московский государственный технический университет им. Н.Э. Баумана
Факультет «Информатика и системы управления»
Кафедра «Автоматизированные системы обработки информации и управления»



Отчет
Лабораторная работа № 1
По курсу «Технологии машинного обучения»

ИСПОЛНИТЕЛЬ:

Группа ИУ5-65Б

Уристимбек Г.

" " _____ 2021 г.

ПРЕПОДАВАТЕЛЬ:

Гапанюк Ю.Е.

" " _____ 2021 г.

Москва 2021

Текстовое описание

Эти данные являются результатами химического анализа вин, выращенных в одном и том же регионе Италии тремя различными культиваторами. Существует тринадцать различных измерений, проведенных для различных компонентов, содержащихся в трех типах вина.

- Алкоголь
- Яблочная кислота
- Шлак
- Щелочность шлака
- Магний
- Всего фенолов
- Флаваноиды
- Нефлаваноидные фенолы
- Проантоцианы
- Интенсивность цвета:
- Оттенок
- OD280/OD315 разбавленных вин
- Пролин

Импорт библиотек

```
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")
from sklearn.datasets import *
```

Загрузка данных

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

Основные характеристики датасета

```
In [3]: type(wine)
```

```
Out[3]: sklearn.utils.Bunch
```

```
In [4]: wine.target[[10, 50, 85]]
```

```
Out[4]: array([0, 0, 1])
```

```
In [5]:
```

```
wine['target_names']
```

```
Out[5]: array(['class_0', 'class_1', 'class_2'], dtype='<U7')
```

```
In [6]: wine['feature_names']
```

```
Out[6]: ['alcohol',
        'malic_acid',
        'ash',
        'alcalinity_of_ash',
        'magnesium',
        'total_phenols',
        'flavanoids',
        'nonflavanoid_phenols',
        'proanthocyanins',
        'color_intensity',
        'hue',
        'od280/od315_of_diluted_wines', 'proline']
```

```
In [7]: #Преобразование Scikit-learn в Pandas DataFrame.
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
```

```
In [8]: temp_df = make_dataframe(load_wine)
temp_df.head() #Выводятся первые 5 строк датасета
```

```
Out[8]:
```

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoi
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	



```
In [9]: wine['data'].shape
#Размерность датасета - 178 записей, 13 атрибутов
```

```
Out[9]: (178, 13)
```

```
In [10]: wine['target'].shape
#Размерность целевого признака
```

```
Out[10]: (178,)
```

```
In [11]: temp_df.columns
#Список атрибутов
```

```
Out[11]: Index(['alcohol', 'malic_acid', 'ash', 'alcalinity_of_ash', 'magnesium',
```

```

        'total_phenols', 'flavanoids', 'nonflavanoid_phenols','proanthocyanins',
        'color_intensity', 'hue', 'od280/od315_of_diluted_wines', 'proline',
        'target'],
        dtype='object')

```

```

In [12]: temp_df.dtypes
         #Типы данных атрибутов

```

```

Out[12]: 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 [13]: # Проверка наличия пустых значений в датасете
for col in temp_df.columns:
    temp_null_count = temp_df[temp_df[col].isnull()].shape[0]
    print('{} - {}'.format(col, temp_null_count))
    #Пустых значений нет

```

```

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 [14]: # Основные статистические характеристики набора данных
temp_df.describe()

```

```

Out[14]:

```

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoi
count	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.0000
mean	13.000618	2.336348	2.366517	19.494944	99.741573	2.295112	2.0292
std	0.811827	1.117146	0.274344	3.339564	14.282484	0.625851	0.9988
min	11.030000	0.740000	1.360000	10.600000	70.000000	0.980000	0.3400
25%	12.362500	1.602500	2.210000	17.200000	88.000000	1.742500	1.2050
50%	13.050000	1.865000	2.360000	19.500000	98.000000	2.355000	2.1350
75%	13.677500	3.082500	2.557500	21.500000	107.000000	2.800000	2.8750
max	14.830000	5.800000	3.230000	30.000000	162.000000	3.880000	5.0800

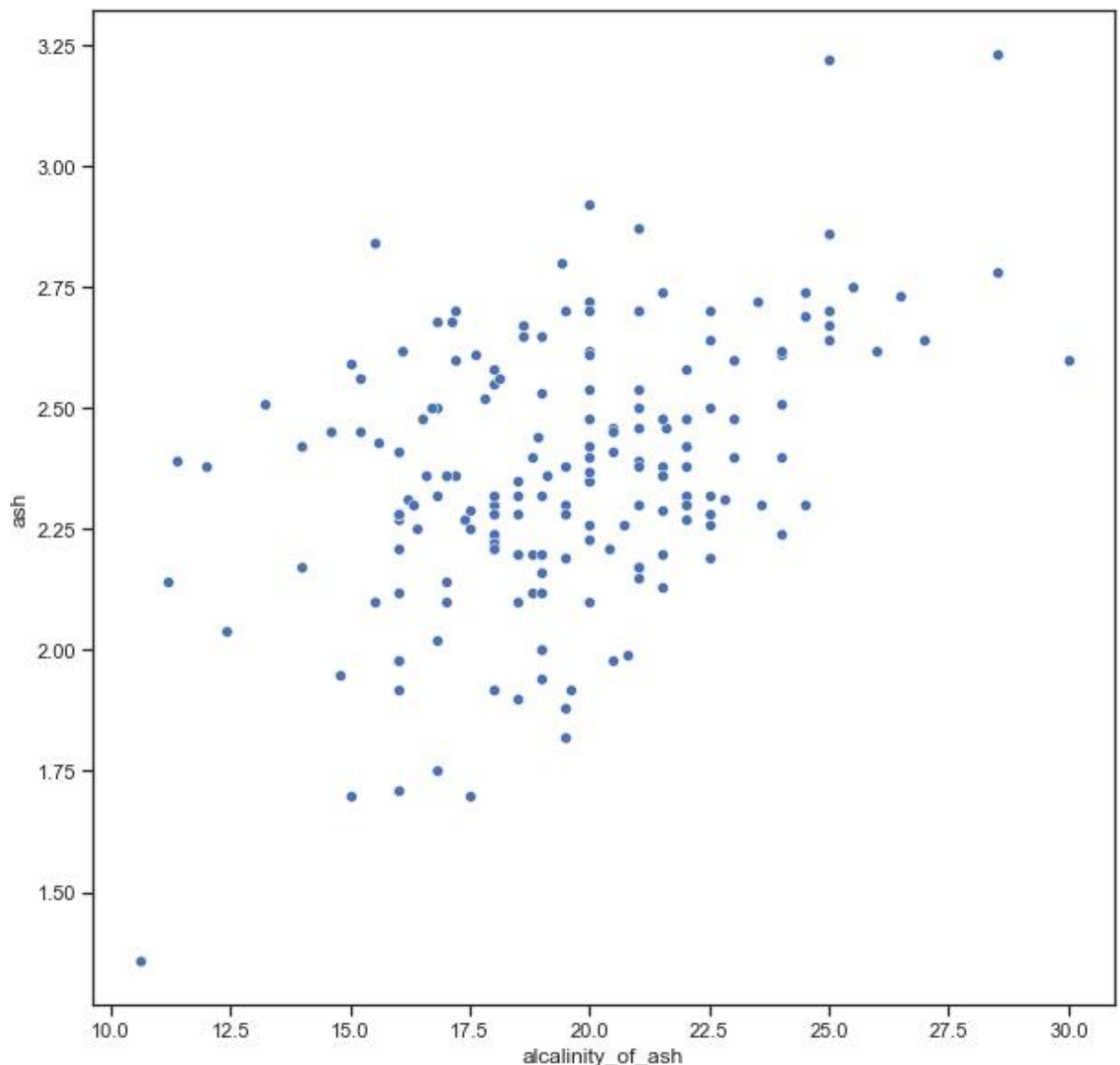
Визуальное исследование датасета

Диаграмма рассеяния

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

```
In [15]: fig, ax = plt.subplots(figsize=(10,10))
sns.scatterplot(ax=ax, x='alkalinity_of_ash', y='ash', data=temp_df)
```

```
Out[15]: <AxesSubplot:xlabel='alkalinity_of_ash', ylabel='ash'>
```



Гистограмма

Позволяет оценить плотность вероятности распределения данных

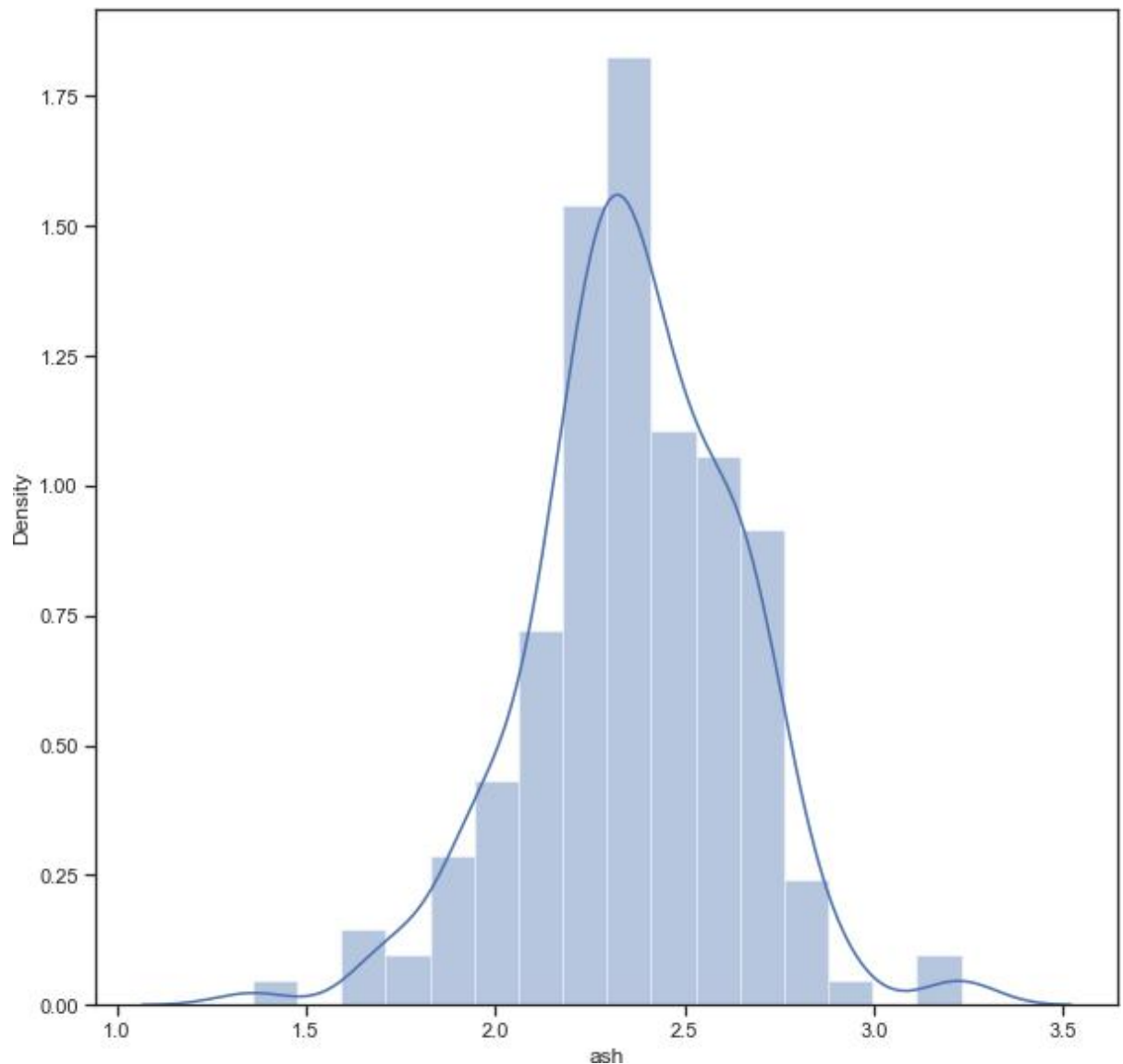
```
In [16]: fig, ax = plt.subplots(figsize=(10,10))
sns.distplot(temp_df['ash'])
```

/home/zeus/anaconda3/envs/tml_env/lib/python3.9/site-packages/seaborn/distrib

```
utions.py:2557: FutureWarning: `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[16]: <AxesSubplot:xlabel='ash', ylabel='Density'>
```

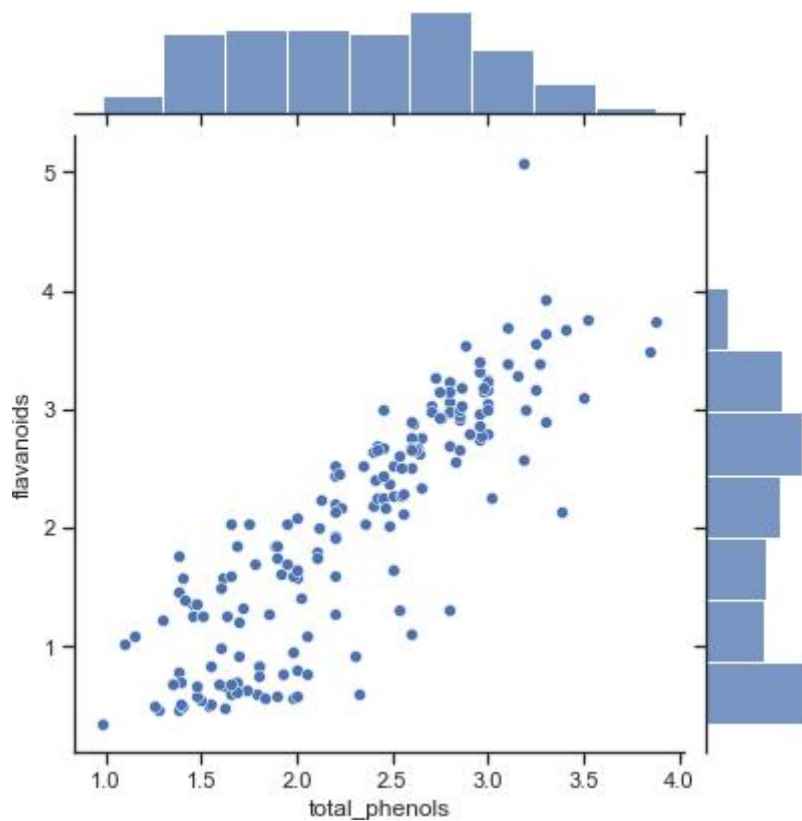


Jointplot

Комбинация гистограмм и диаграмм рассеивания.

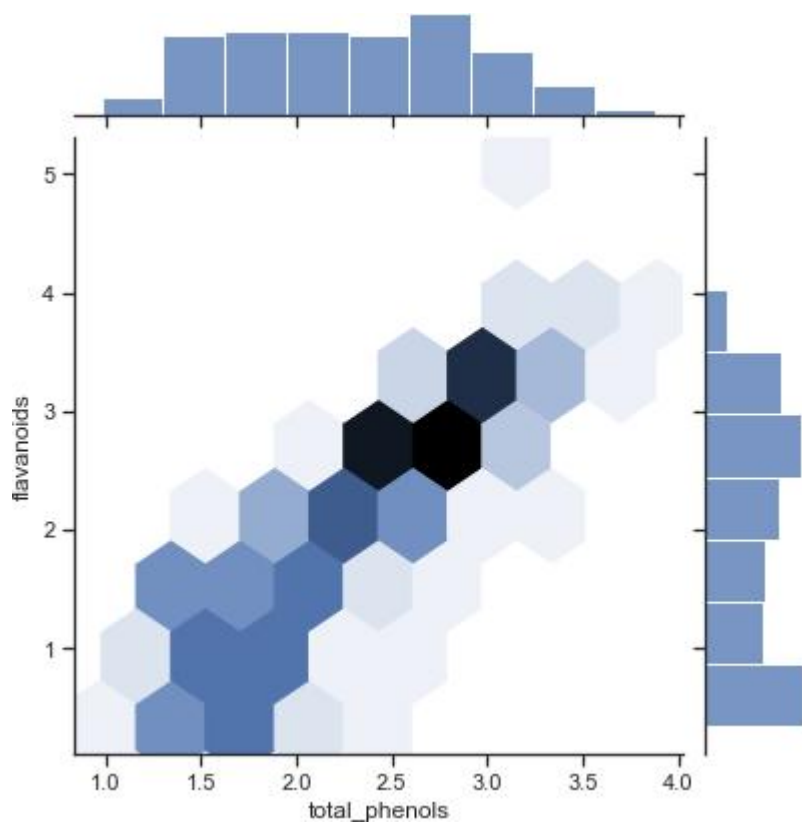
```
In [17]: sns.jointplot(x='total_phenols', y='flavanoids', data=temp_df)
```

```
Out[17]: <seaborn.axisgrid.JointGrid at 0x7feb5b619580>
```



```
In [18]: sns.jointplot(x='total_phenols', y='flavanoids', data=temp_df, kind = 'hex')
```

```
Out[18]: <seaborn.axisgrid.JointGrid at 0x7feb5b9bae80>
```



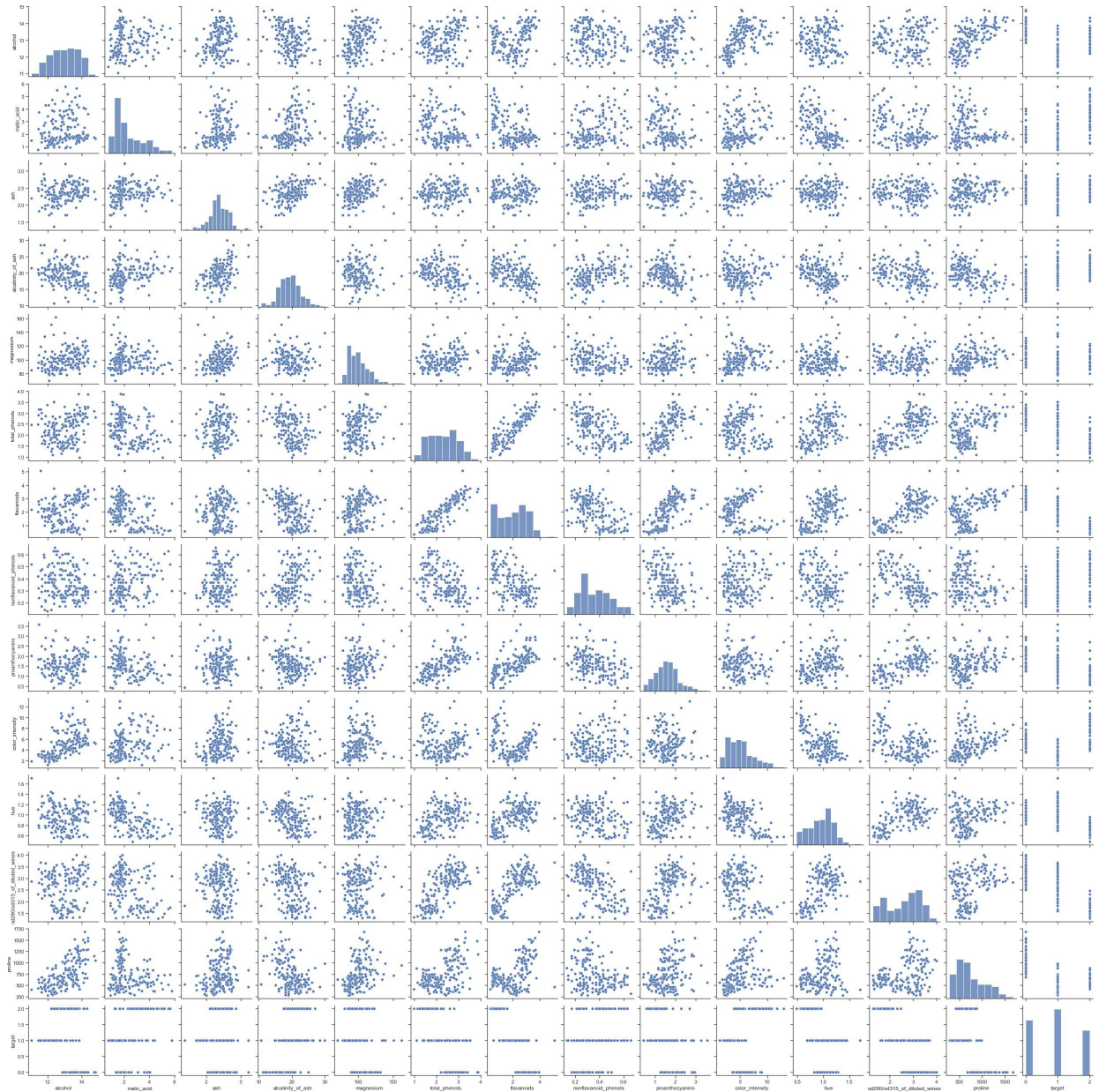
Парные диаграммы

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 [19]: sns.pairplot(temp_df)
```

```
Out[19]: <seaborn.axisgrid.PairGrid at 0x7feb5b7b4730>
```

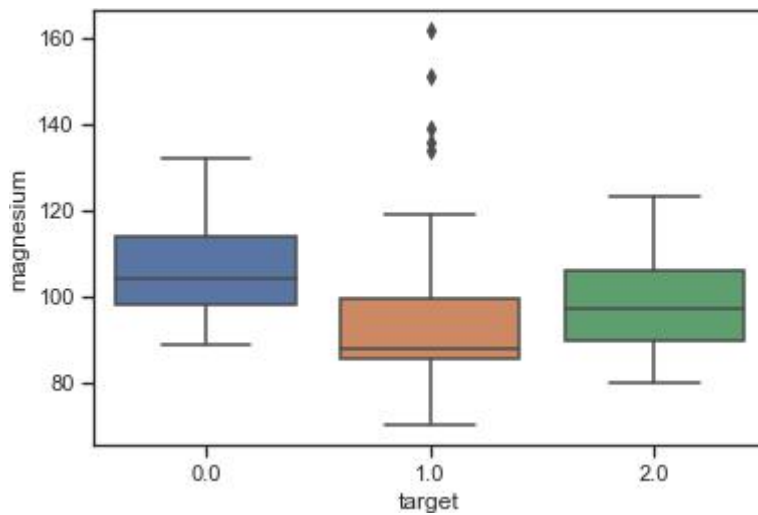


Ящик с усами

Отображает одномерное распределение вероятности.

```
In [20]: sns.boxplot(x='target', y='magnesium', data=temp_df)
```

```
Out[20]: <AxesSubplot:xlabel='target', ylabel='magnesium'>
```

Violin Plot

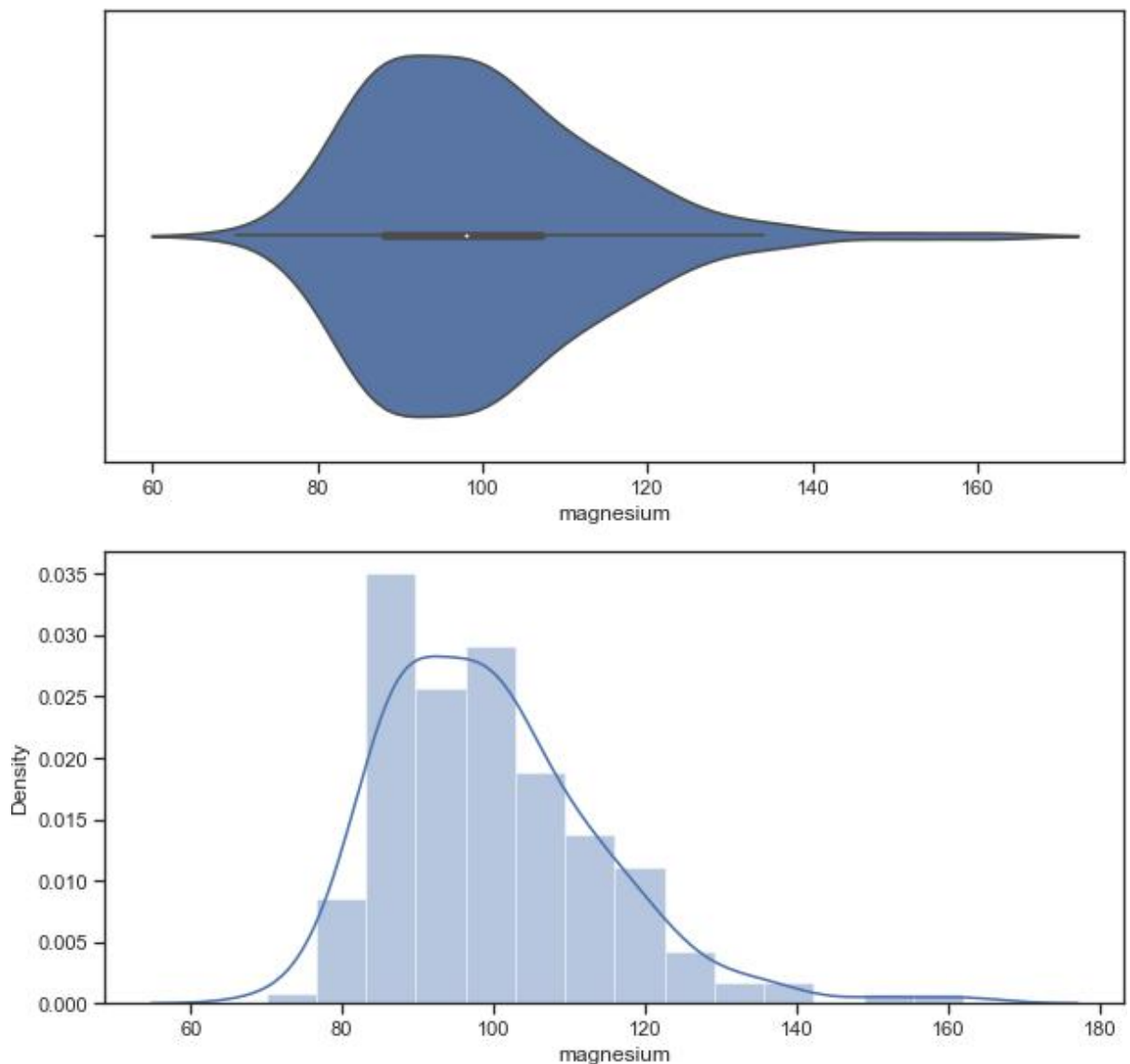
Распределение плотности

```
In [21]: fig, ax = plt.subplots(2, 1, figsize=(10,10))
sns.violinplot(ax=ax[0], x=temp_df['magnesium'])
sns.distplot(temp_df['magnesium'], ax=ax[1])
```

/home/zeus/anaconda3/envs/tml_env/lib/python3.9/site-packages/seaborn/distributions.py:2557: FutureWarning: `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[21]: <AxesSubplot:xlabel='magnesium', ylabel='Density'>
```



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 "target"). 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 [22]: temp_df.corr()
```

```
Out[22]:
```

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_sulfur
alcohol	1.000000	0.094397	0.211545	-0.310235	0.270798	0.167453

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	tota
malic_acid	0.094397	1.000000	0.164045	0.288500	-0.054575	
ash	0.211545	0.164045	1.000000	0.443367	0.286587	
alcalinity_of_ash	-0.310235	0.288500	0.443367	1.000000	-0.083333	
magnesium	0.270798	-0.054575	0.286587	-0.083333	1.000000	
total_phenols	0.289101	-0.335167	0.128980	-0.321113	0.214401	
flavanoids	0.236815	-0.411007	0.115077	-0.351370	0.195784	
nonflavanoid_phenols	-0.155929	0.292977	0.186230	0.361922	-0.256294	
proanthocyanins	0.136698	-0.220746	0.009652	-0.197327	0.236441	
color_intensity	0.546364	0.248985	0.258887	0.018732	0.199950	
hue	-0.071747	-0.561296	-0.074667	-0.273955	0.055398	
od280/od315_of_diluted_wines	0.072343	-0.368710	0.003911	-0.276769	0.066004	
proline	0.643720	-0.192011	0.223626	-0.440597	0.393351	
target	-0.328222	0.437776	-0.049643	0.517859	-0.209179	

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).

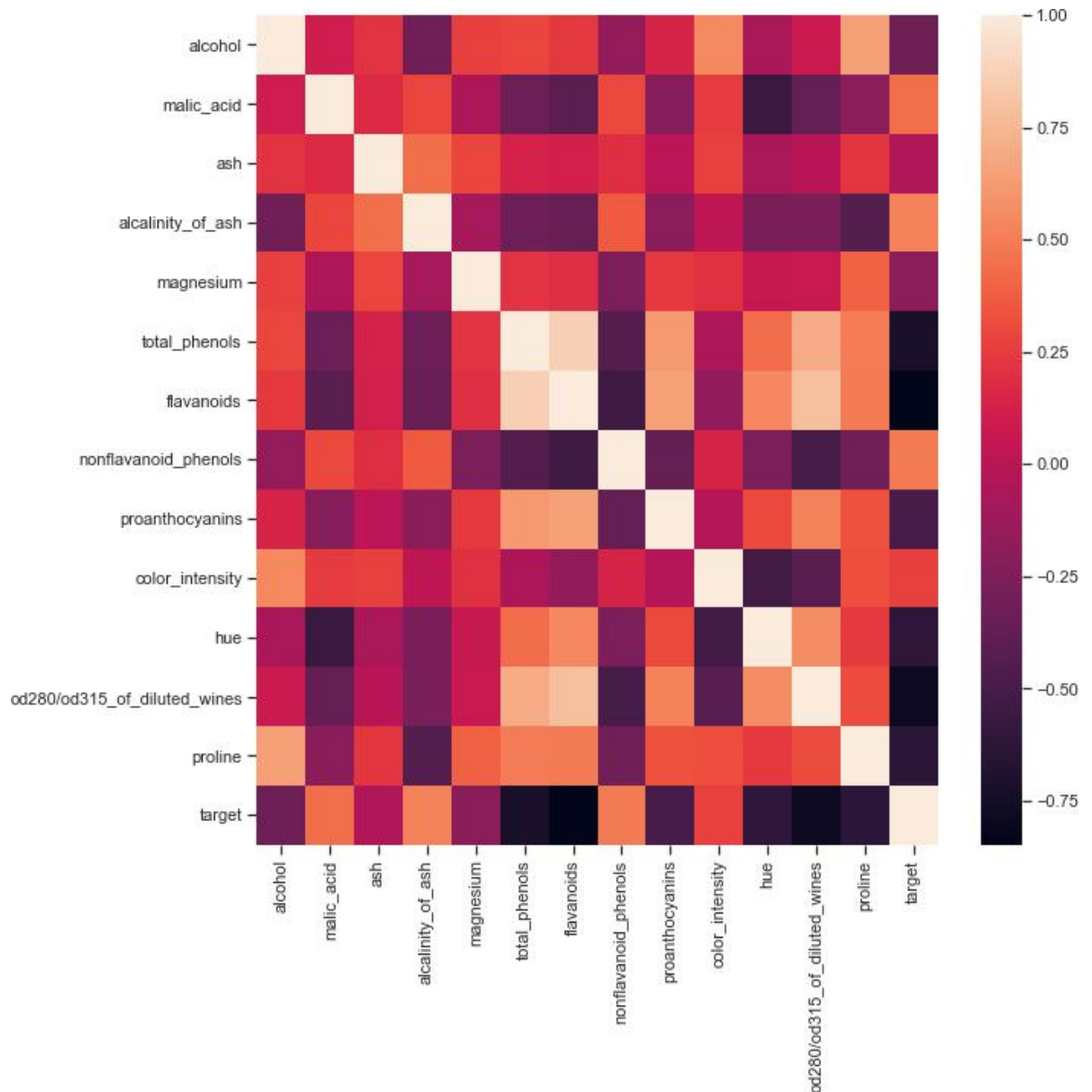
По умолчанию при построении матрицы используется коэффициент корреляции Пирсона. Возможно также построить корреляционную матрицу на основе коэффициентов корреляции Кендалла и Спирмена. На практике три метода редко дают значимые различия.

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.

```
In [23]: fig, ax = plt.subplots(figsize=(10,10))
sns.heatmap(temp_df.corr())
```

```
Out[23]: <AxesSubplot:>
```

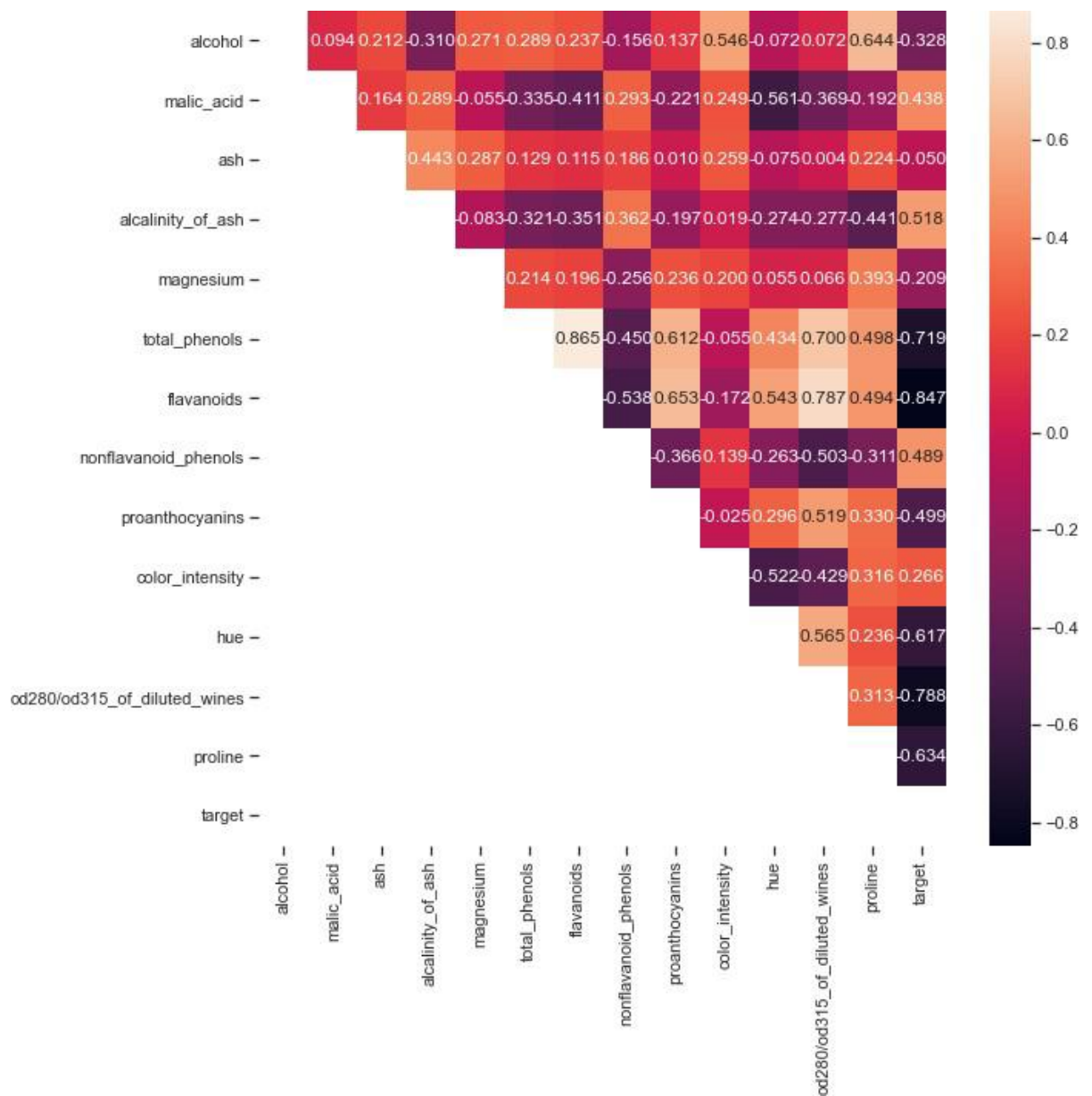


```
In [24]: # Треугольный вариант матрицы
fig, ax = plt.subplots(figsize=(10,10))
mask = np.zeros_like(temp_df.corr(), dtype=np.bool)
# чтобы оставить нижнюю часть матрицы
# mask[np.triu_indices_from(mask)] = True
# чтобы оставить верхнюю часть матрицы
mask[np.tril_indices_from(mask)] = True
sns.heatmap(temp_df.corr(), mask=mask, annot=True, fmt='.3f')
```

<ipython-input-24-4873b57a5c9d>:3: DeprecationWarning: `np.bool` is a deprecated alias for the builtin `bool`. To silence this warning, use `bool` by itself. Doing this will not modify any behavior and is safe. If you specifically wanted the numpy scalar type, use `np.bool_` here.
 Deprecated in NumPy 1.20; for more details and guidance: <https://numpy.org/doc/1.20.0-notes.html#deprecations>

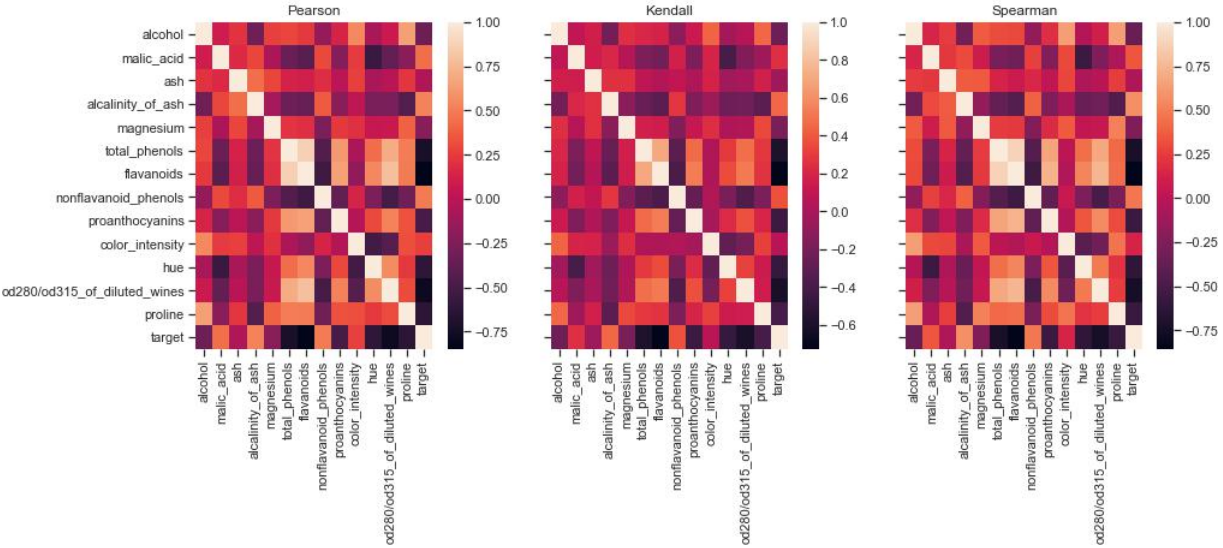
```
mask = np.zeros_like(temp_df.corr(), dtype=np.bool)
```

Out[24]: <AxesSubplot:>



```
In [25]: fig, ax = plt.subplots(1, 3, sharex='col', sharey='row', figsize=(15,5))
sns.heatmap(temp_df.corr(method='pearson'), ax=ax[0])
sns.heatmap(temp_df.corr(method='kendall'), ax=ax[1])
sns.heatmap(temp_df.corr(method='spearman'), ax=ax[2])
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')
```

Корреляционные матрицы, построенные различными методами



```
In [ ]:
```