

Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра «Системы обработки информации и управления»

Дисциплина «Технологии машинного обучения»

Отчёт

по лабораторной работе №1

«Разведочный анализ данных. Исследование и визуализация данных»

Вариант 3

Студент:

Белкина Е.В.

Группа ИУ5-61Б

Преподаватель:

Гапанюк Ю.Е.

Москва, 2020 г.

Цель лабораторной работы:

Изучение различных методов визуализации данных.

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

Задание:

- Выбрать набор данных (датасет).

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 работы:

1. Текстовое описание выбранного набора данных

В качестве набора данных мы будем использовать набор данных, задачей которого является попытка предсказания объема проката велосипедов в будущем.

<https://www.kaggle.com/hmavrodiiev/london-bike-sharing-dataset>

Решение этой задачи может быть актуально для городских организаций проката в сфере прогнозирования спроса на предоставление их услуг в разные сезоны, при возникновении различных природных и социальных условий.

Данные взяты из 3 источников:

- <https://cycling.data.tfl.gov.uk/> 'Contains OS data © Crown copyright and database rights 2016' and Geomni UK Map data © and database rights [2019] 'Powered by TfL Open Data'
- freemeteo.com - weather data
- <https://www.gov.uk/bank-holidays>
From 1/1/2015 to 31/12/2016

Датасет состоит из 1 файла: london_merged.csv

Каждый файл содержит следующие колонки:

"timestamp" - поле метки времени для группировки данных

"cnt" – количество проката нового велосипеда

"t1" – реальная температура воздуха в Цельсиях

"t2" – ощущаемая температура воздуха

"hum" - влажность в процентах

"windspeed" – скорость ветра в км/ч

"weather_code" – категория погодных условий

"isholiday" – булево число - 1 = праздничный день / 0 = обычный день

"isweekend" - булево число - 1 = выходной день / 0 = рабочий день

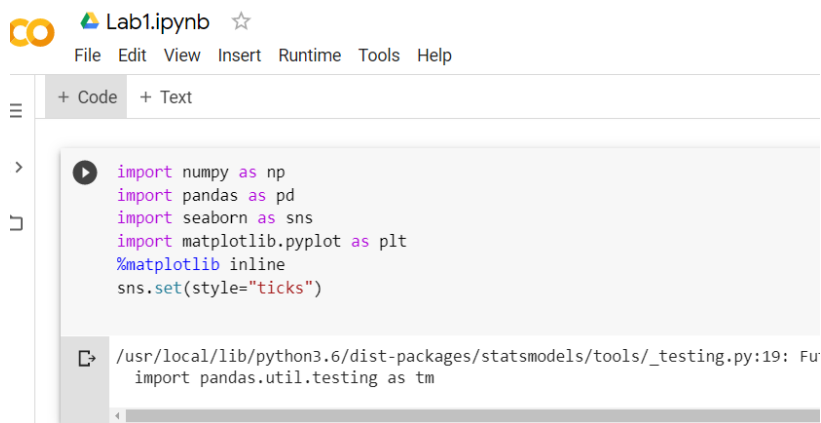
"season" – поле категории сезона: 0 - весна ; 1 - лето; 2 - осень; 3 - зима.

"weather_code" описание категории:

1 = ясно; 2 = рассеянные облака; 3 = облачность; 4 = сильная облачность; 7 = дождь/светло; 10 = дождь с грозой; 26 = снегопад; 94 = мороз

2. Основные характеристики датасета

Импорт библиотек



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

/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: Fur
import pandas.util.testing as tm

Загрузка данных

```
[10] data = pd.read_csv('/london_merged.csv')
```

Основные характеристики датасета

```
[11] data.head()
```

	timestamp	cnt	t1	t2	hum	wind_speed	weather_code	is_holiday	is_weekend	season
0	2015-01-04 00:00:00	182	3.0	2.0	93.0	6.0	3.0	0.0	1.0	3.0
1	2015-01-04 01:00:00	138	3.0	2.5	93.0	5.0	1.0	0.0	1.0	3.0
2	2015-01-04 02:00:00	134	2.5	2.5	96.5	0.0	1.0	0.0	1.0	3.0
3	2015-01-04 03:00:00	72	2.0	2.0	100.0	0.0	1.0	0.0	1.0	3.0
4	2015-01-04 04:00:00	47	2.0	0.0	93.0	6.5	1.0	0.0	1.0	3.0

```
data.shape
```

```
(17414, 10)
```

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

↳ Всего строк: 17414

```
▶ # Список колонок
data.columns
```

↳ Index(['timestamp', 'cnt', 't1', 't2', 'hum', 'wind_speed', 'weather_code', 'is_holiday', 'is_weekend', 'season'], dtype='object')

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

↳

timestamp	object
cnt	int64
t1	float64
t2	float64
hum	float64
wind_speed	float64
weather_code	float64
is_holiday	float64
is_weekend	float64
season	float64
dtype:	object

```
▶ # 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))
```

↳ timestamp - 0
cnt - 0
t1 - 0
t2 - 0
hum - 0
wind_speed - 0
weather_code - 0
is_holiday - 0
is_weekend - 0
season - 0

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

	cnt	t1	t2	hum	wind_speed	weather_code	is_holiday	is_weekend	season
count	17414.000000	17414.000000	17414.000000	17414.000000	17414.000000	17414.000000	17414.000000	17414.000000	17414.000000
mean	1143.101642	12.468091	11.520836	72.324954	15.913063	2.722752	0.022051	0.285403	1.492075
std	1085.108068	5.571818	6.615145	14.313186	7.894570	2.341163	0.146854	0.451619	1.118911
min	0.000000	-1.500000	-6.000000	20.500000	0.000000	1.000000	0.000000	0.000000	0.000000
25%	257.000000	8.000000	6.000000	63.000000	10.000000	1.000000	0.000000	0.000000	0.000000
50%	844.000000	12.500000	12.500000	74.500000	15.000000	2.000000	0.000000	0.000000	1.000000
75%	1671.750000	16.000000	16.000000	83.000000	20.500000	3.000000	0.000000	1.000000	2.000000
max	7860.000000	34.000000	34.000000	100.000000	56.500000	26.000000	1.000000	1.000000	3.000000

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

```
array([1., 0.])
```

3. Визуальное исследование датасета

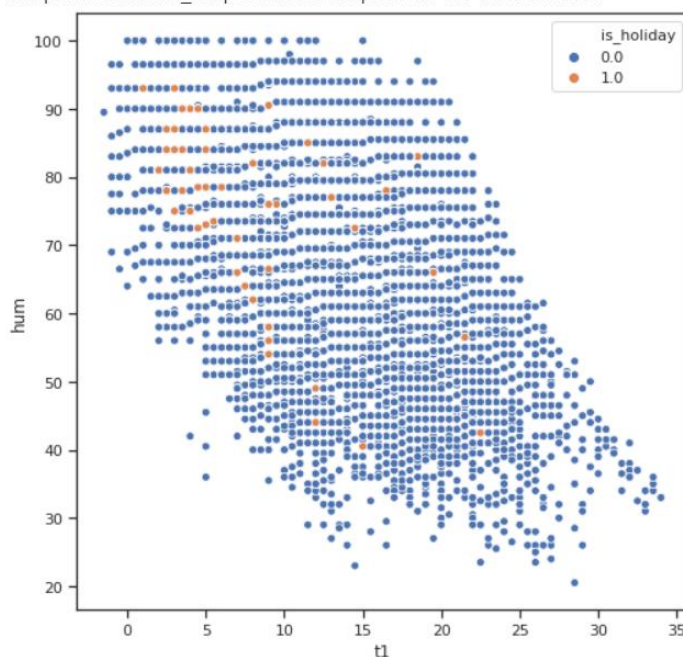
Для визуального исследования могут быть использованы различные виды диаграмм, мы построим только некоторые варианты диаграмм, которые используются достаточно часто.

Диаграмма рассеяния

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

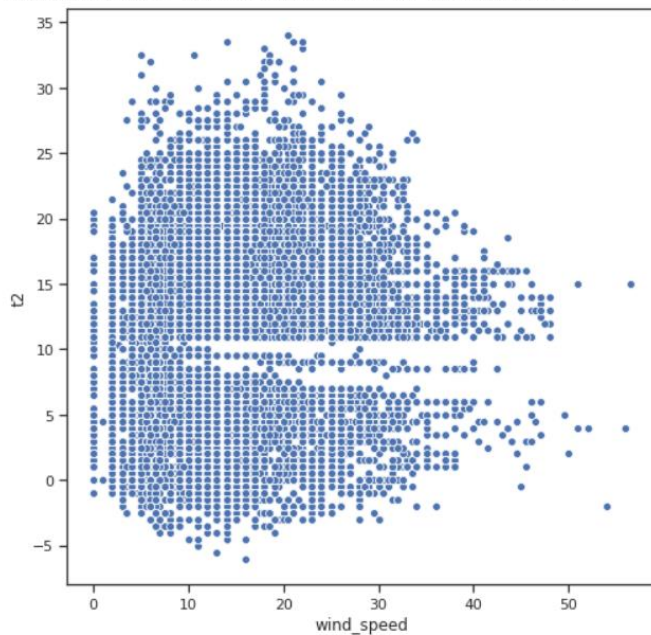
```
fig, ax = plt.subplots(figsize=(8,8))
sns.scatterplot(ax=ax, x='t1', y='hum', data=data, hue='is_holiday')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0aa19b72e8>
```



```
fig, ax = plt.subplots(figsize=(8,8))
sns.scatterplot(ax=ax, x='wind_speed', y='t2', data=data)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f0aa1de17f0>

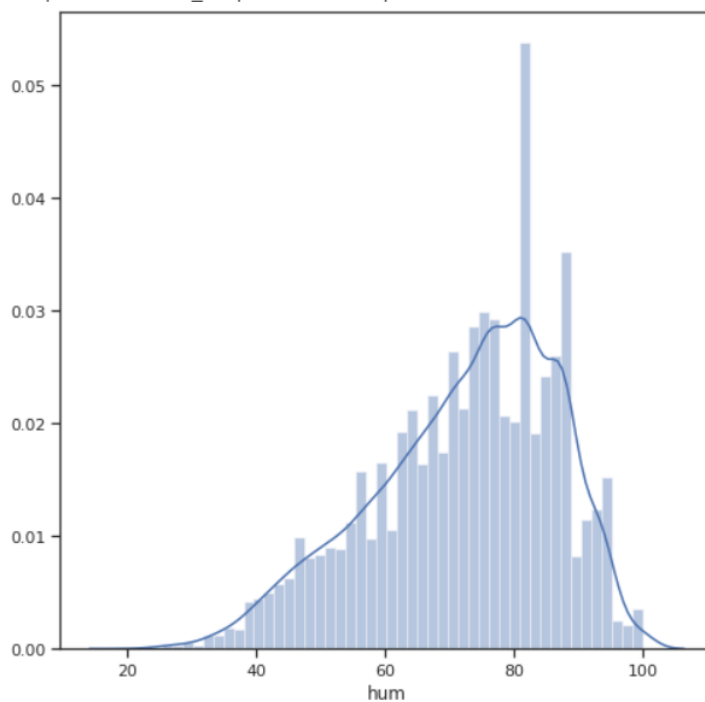


Гистограмма

Позволяет оценить плотность вероятности распределения данных.

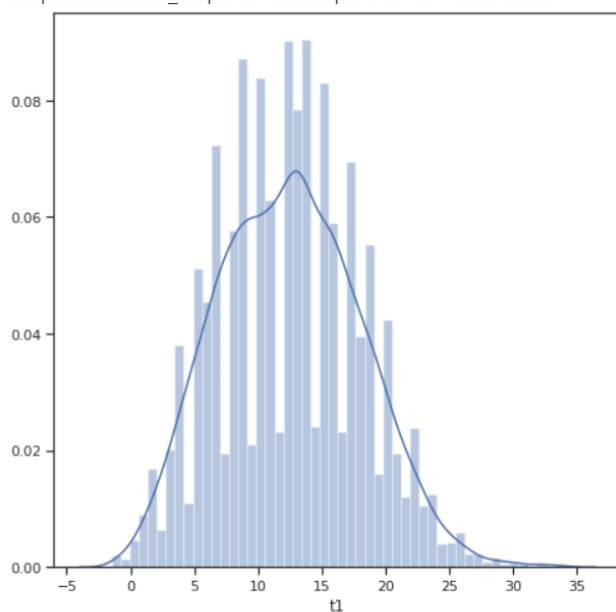
```
fig, ax = plt.subplots(figsize=(8,8))
sns.distplot(data['hum'])
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f0aa1868400>



```
fig, ax = plt.subplots(figsize=(8,8))
sns.distplot(data['t1'])
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f0aa17ad320>

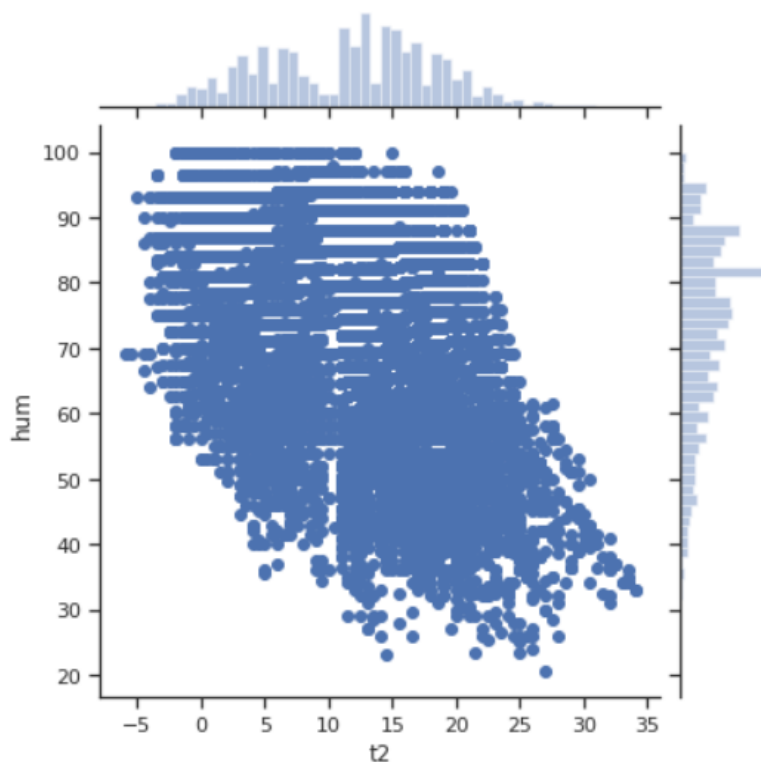


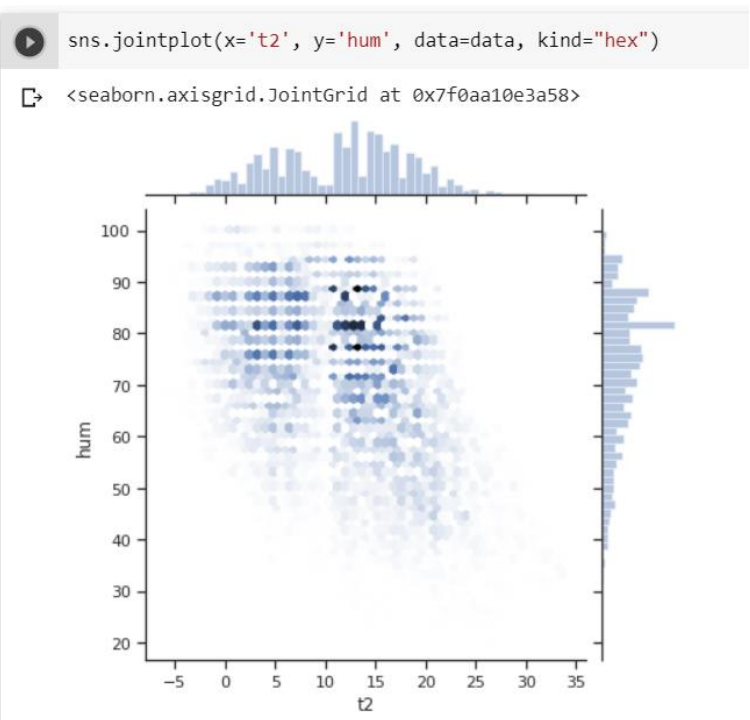
Jointplot

Комбинация гистограмм и диаграмм рассеивания.

```
sns.jointplot(x='t2', y='hum', data=data)
```

<seaborn.axisgrid.JointGrid at 0x7f0aa390e390>

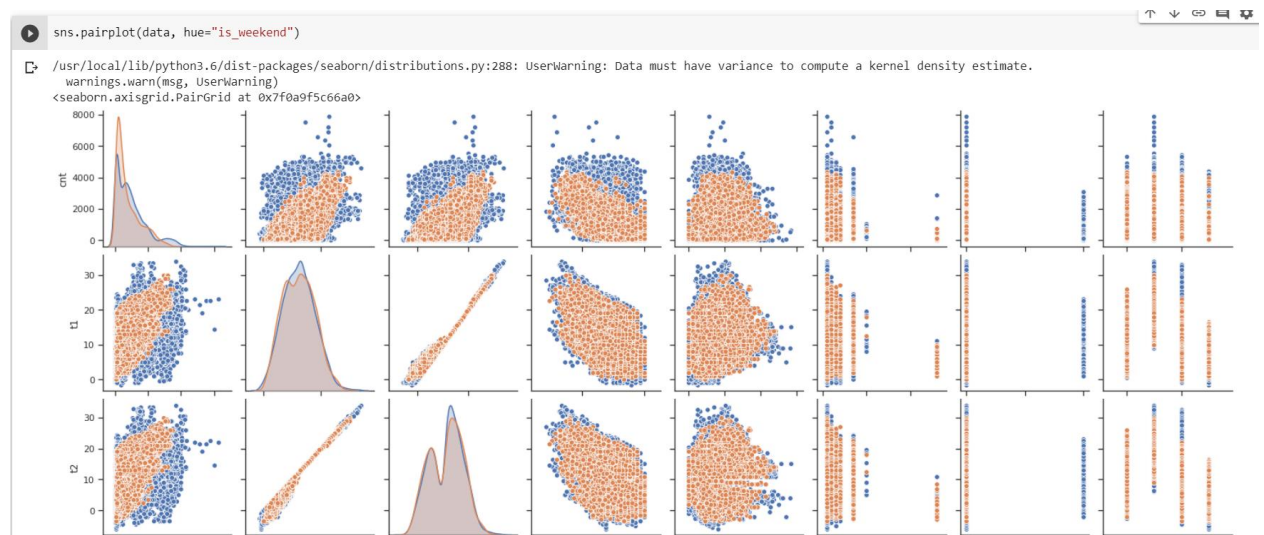


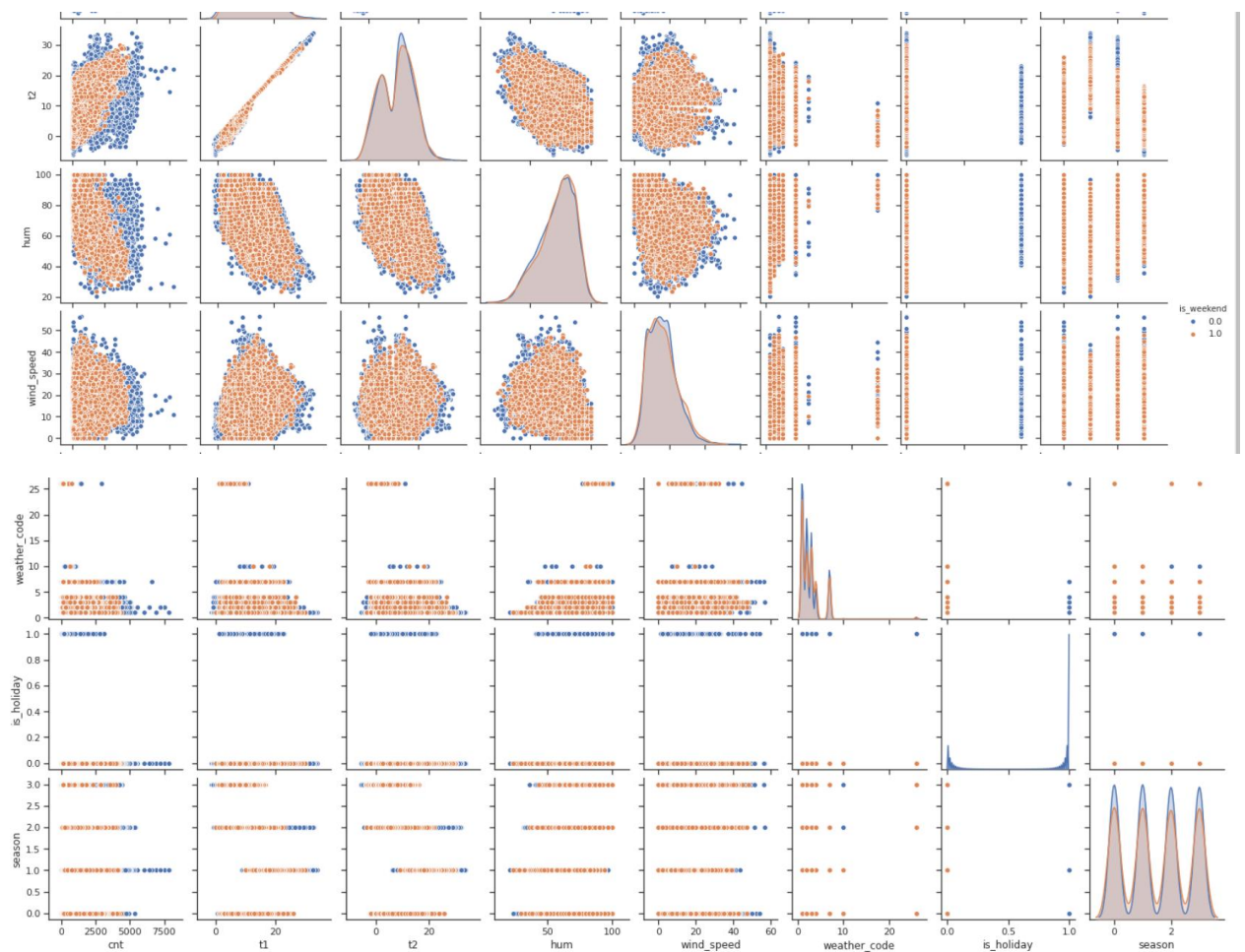


"Парные диаграммы"

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.



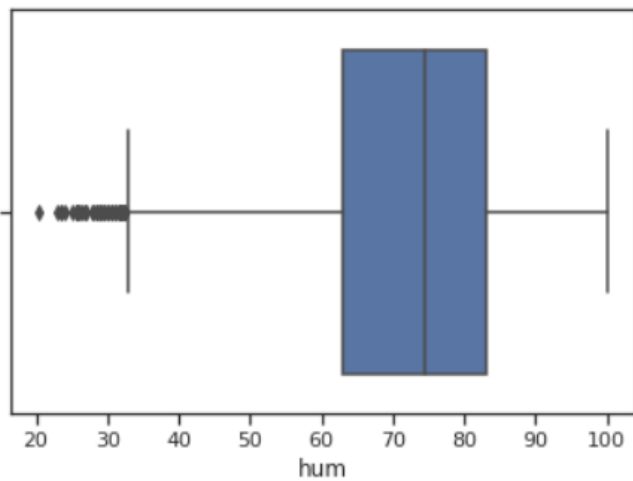


Ящик с усами

Отображает одномерное распределение вероятности.

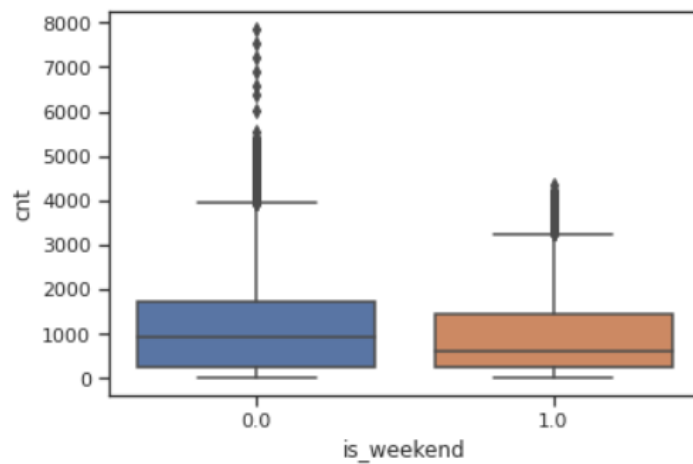
```
sns.boxplot(x=data['hum'])
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f0a9d959128>



```
sns.boxplot(x='is_weekend', y='cnt', data=data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0a9cfb23c8>
```

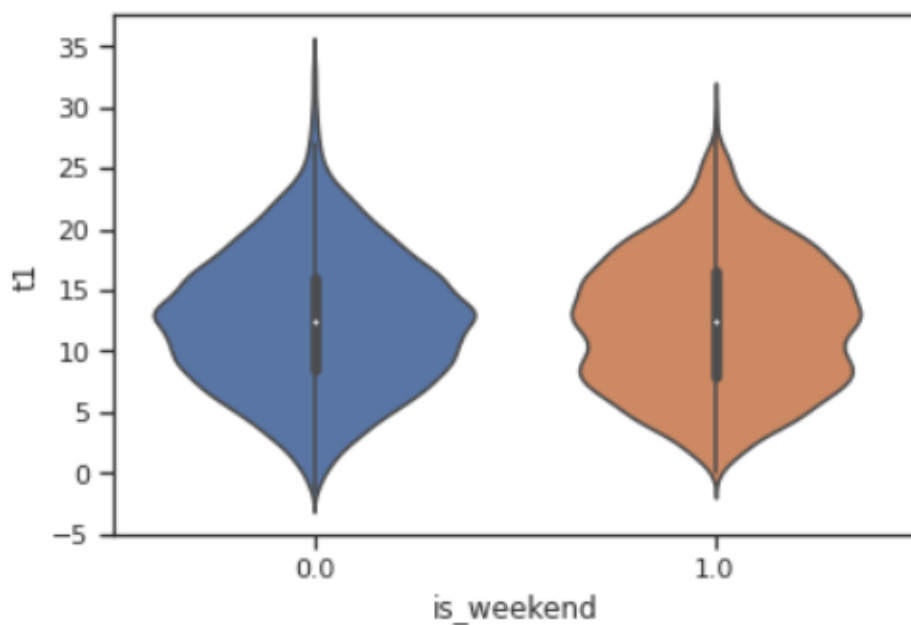


Violin plot

Похоже на предыдущую диаграмму, но по краям отображаются распределения плотности.

```
sns.violinplot(x='is_weekend', y='t1', data=data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0a9cf0cb38>
```



4. Информация о корреляции признаков

Проверка корреляции признаков позволяет решить две задачи:

1. Понять какие признаки (колонки датасета) наиболее сильно коррелируют с целевым признаком (в нашем примере нет целевого признака) 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.
2. 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.

Корреляционная матрица Пирсона:

```
data.corr(method='pearson')
```

	cnt	t1	t2	hum	wind_speed	weather_code	is_holiday	is_weekend	season
cnt	1.000000	0.388798	0.369035	-0.462901	0.116295	-0.166633	-0.051698	-0.096499	-0.116180
t1	0.388798	1.000000	0.988344	-0.447781	0.145471	-0.097114	-0.042233	-0.005342	-0.285851
t2	0.369035	0.988344	1.000000	-0.403495	0.088409	-0.098385	-0.040051	-0.008510	-0.285900
hum	-0.462901	-0.447781	-0.403495	1.000000	-0.287789	0.334750	0.032068	0.028098	0.290381
wind_speed	0.116295	0.145471	0.088409	-0.287789	1.000000	0.124803	-0.002606	0.011479	0.010305
weather_code	-0.166633	-0.097114	-0.098385	0.334750	0.124803	1.000000	0.012939	0.042362	0.098976
is_holiday	-0.051698	-0.042233	-0.040051	0.032068	-0.002606	0.012939	1.000000	-0.094898	-0.032488
is_weekend	-0.096499	-0.005342	-0.008510	0.028098	0.011479	0.042362	-0.094898	1.000000	0.001067
season	-0.116180	-0.285851	-0.285900	0.290381	0.010305	0.098976	-0.032488	0.001067	1.000000

Корреляционная матрица Кендалла:

```
data.corr(method='kendall')
```

	cnt	t1	t2	hum	wind_speed	weather_code	is_holiday	is_weekend	season
cnt	1.000000	0.273141	0.268159	-0.352679	0.121909	-0.092905	-0.042391	-0.055157	-0.080437
t1	0.273141	1.000000	0.971015	-0.262035	0.108975	-0.047081	-0.034276	-0.004136	-0.218541
t2	0.268159	0.971015	1.000000	-0.246630	0.072700	-0.051825	-0.034409	-0.003625	-0.219101
hum	-0.352679	-0.262035	-0.246630	1.000000	-0.220319	0.279104	0.026410	0.023390	0.204357
wind_speed	0.121909	0.108975	0.072700	-0.220319	1.000000	0.121345	-0.007240	0.003108	-0.008390
weather_code	-0.092905	-0.047081	-0.051825	0.279104	0.121345	1.000000	0.005067	0.044535	0.097486
is_holiday	-0.042391	-0.034276	-0.034409	0.026410	-0.007240	0.005067	1.000000	-0.094898	-0.030041
is_weekend	-0.055157	-0.004136	-0.003625	0.023390	0.003108	0.044535	-0.094898	1.000000	0.000967
season	-0.080437	-0.218541	-0.219101	0.204357	-0.008390	0.097486	-0.030041	0.000967	1.000000

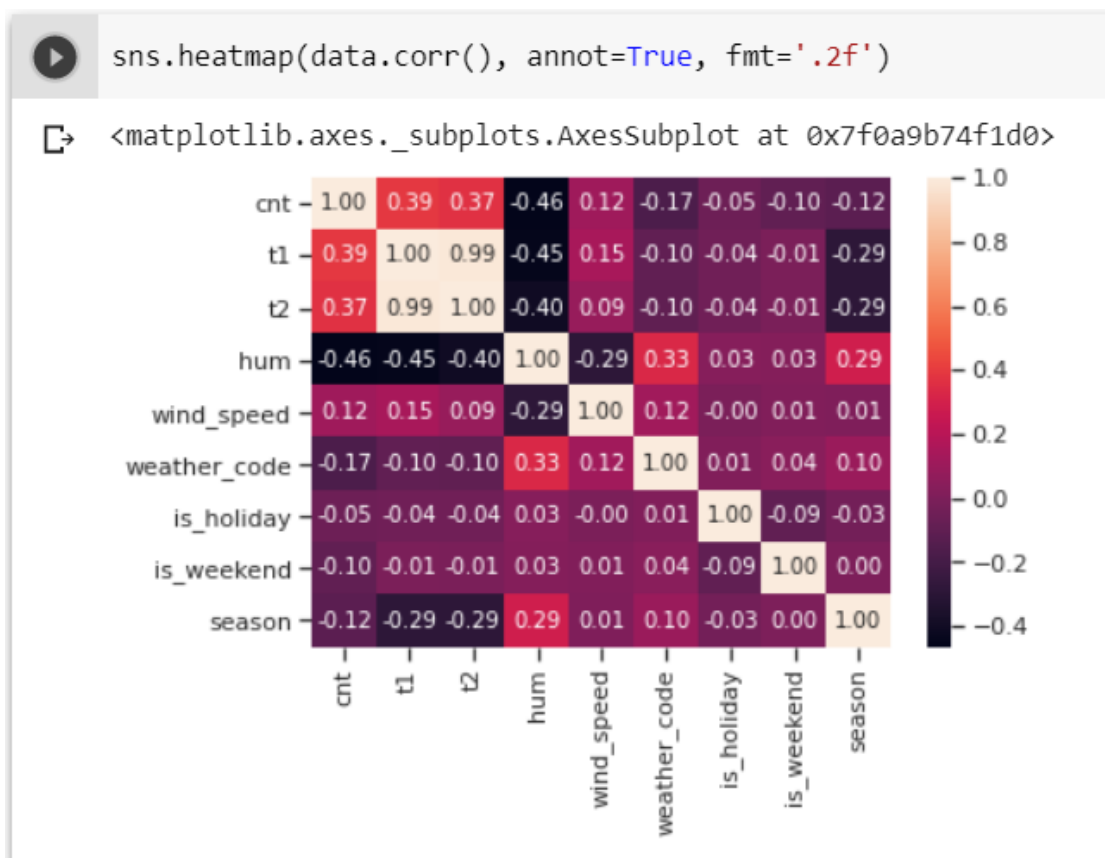
Корреляционная матрица Спирмана:

```
data.corr(method='spearman')
```

	cnt	t1	t2	hum	wind_speed	weather_code	is_holiday	is_weekend	season
cnt	1.000000	0.392384	0.386731	-0.504613	0.181434	-0.119776	-0.051904	-0.067534	-0.107223
t1	0.392384	1.000000	0.995809	-0.380248	0.159299	-0.059792	-0.041370	-0.004992	-0.279427
t2	0.386731	0.995809	1.000000	-0.360955	0.107333	-0.066359	-0.041606	-0.004383	-0.280571
hum	-0.504613	-0.380248	-0.360955	1.000000	-0.319598	0.365801	0.032024	0.028362	0.272071
wind_speed	0.181434	0.159299	0.107333	-0.319598	1.000000	0.160982	-0.008737	0.003751	-0.009141
weather_code	-0.119776	-0.059792	-0.066359	0.365801	0.160982	1.000000	0.005594	0.049169	0.116525
is_holiday	-0.051904	-0.041370	-0.041606	0.032024	-0.008737	0.005594	1.000000	-0.094898	-0.032908
is_weekend	-0.067534	-0.004992	-0.004383	0.028362	0.003751	0.049169	-0.094898	1.000000	0.001060
season	-0.107223	-0.279427	-0.280571	0.272071	-0.009141	0.116525	-0.032908	0.001060	1.000000

Можем видеть, что значения во всех трёх матрицах примерно равны в соответствующих ячейках.

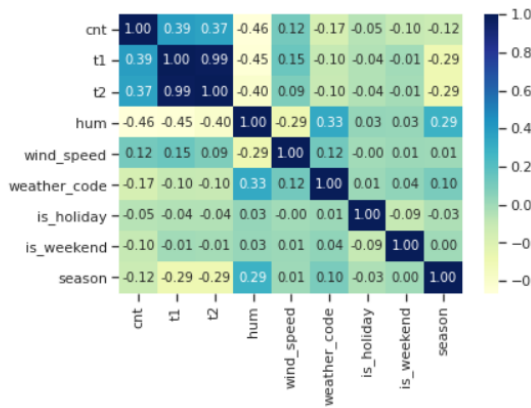
Для визуализации корреляционной матрицы будем использовать "тепловую карту" heatmap, которая показывает степень корреляции различными цветами.



Сильно коррелирующие между собой признаки: t1 и t2 (0.99) – практически линейно зависимы. При построении модели машинного обучения следует оставить только один из них, а именно тот, который больше коррелирует с целевым признаком.

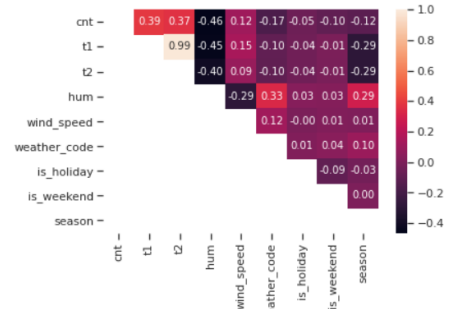
```
sns.heatmap(data.corr(), cmap='YlGnBu', annot=True, fmt='.2f')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0a9b616080>
```



```
# 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')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0a9b324160>
```



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

```
<matplotlib.figure.Figure at 0x7f0a9b324160>
```

