*Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования*

***«Московский государственный технический университет имени Н.Э. Баумана»***

***(МГТУ им. Н.Э. Баумана)***

---

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Системы обработки информации и управления (ИУ5)

# О т ч е т

## по лабораторной работе №1

## Разведочный анализ данных. Исследование и визуализация данных

## Дисциплина: Технологии машинного обучения

Студент гр. ИУ5-63Б _____ Терентьев В.О.

(Подпись, дата) (Фамилия И.О.)

Преподаватель _____ Гапанюк Ю.Е.

(Подпись, дата) (Фамилия И.О.)

Москва, 2021

# 1. Цель работы

Цель лабораторной работы: изучение различных методов визуализация данных.

# 2. Описание задания

- Выбрать набор данных (датасет). Вы можете найти список свободно распространяемых датасетов здесь.
- Для первой лабораторной работы рекомендуется использовать датасет без пропусков в данных, например из Scikit-learn.
- Пример преобразования датасетов Scikit-learn в Pandas Dataframe можно посмотреть здесь.

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.

# 3. Основная часть

# Лабораторная работа №1

## Разведочный анализ данных. Исследование и визуализация данных

## 1) Текстовое описание набора данных

В качестве набора данных использован набор данных по предсказанию уровня солнечной радиации - https://www.kaggle.com/dronio/SolarEnergy (https://www.kaggle.com/dronio/SolarEnergy)

Датасет состоит из одного файла: **SolarPrediction.csv**

Файл содержит следующие колонки:

- **UNIXTime** - количество секунд с 1 января 1970 г.;
- **Data** - дата-время в формате месяц/день/год часы:минуты:секунды;
- **Time** - время в 24-х часовом формате, часы:минуты:секунды;
- **Radiation** - солнечная радиация, ватт/м^2;
- **Temperature** - температура, °F;
- **Pressure** - атмосферное давление, дюйм рт. ст.;
- **Humidity** - относительная влажность, %;
- **WindDirection(Degrees)** - направление ветра, градусы;
- **Speed** - скорость ветра, мили/ч;
- **TimeSunRise** - время восхода солнца, часы:минуты:секунды;
- **TimeSunSet** - время захода солнца, часы:минуты:секунды.

## Импорт библиотек

In [1]:

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

## Загрузка данных

Загрузка файла датасета с помощью библиотеки Pandas:

In [2]:

```
data = pd.read_csv('SolarPrediction.csv', sep=",")
# Преобразование данных датасета в нужный формат
data['Data'] = pd.to_datetime(data['UNIXTime'], unit='s') + pd.DateOffset(hours=-10)
data.rename(columns={'Data': 'DateTime'}, inplace = True)
data['TimeSunRise'] = pd.to_datetime(data['TimeSunRise'], format='%H:%M:%S').dt.time
data['TimeSunSet'] = pd.to_datetime(data['TimeSunSet'], format='%H:%M:%S').dt.time
# Удаление ненужных столбцов
data.drop(['UNIXTime', 'Time'], inplace=True, axis=1)
```

## 2) Основные характеристики датасета

Первые 5 строк датасета:

In [3]:

```
data.head()
```

Out[3]:

| | DateTime | Radiation | Temperature | Pressure | Humidity | WindDirection(Degrees) | Speed | Time |
|---|---|---|---|---|---|---|---|---|
| **0** | 2016-09-29 23:55:26 | 1.21 | 48 | 30.46 | 59 | 177.39 | 5.62 | |
| **1** | 2016-09-29 23:50:23 | 1.21 | 48 | 30.46 | 58 | 176.78 | 3.37 | |
| **2** | 2016-09-29 23:45:26 | 1.23 | 48 | 30.46 | 57 | 158.75 | 3.37 | |
| **3** | 2016-09-29 23:40:21 | 1.21 | 48 | 30.46 | 60 | 137.71 | 3.37 | |
| **4** | 2016-09-29 23:35:24 | 1.17 | 48 | 30.46 | 62 | 104.95 | 5.62 | |

Размер датасета - 32686 строк, 9 колонок.

In [4]:

```
data.shape
```

Out[4]:

```
(32686, 9)
```

Список колонок:

In [5]:

```
data.columns
```

Out[5]:

```
Index(['DateTime', 'Radiation', 'Temperature', 'Pressure', 'Humidity',
       'WindDirection(Degrees)', 'Speed', 'TimeSunRise', 'TimeSunSet'],
      dtype='object')
```

Список колонок с типами данных:

In [6]:

```
data.dtypes
```

Out[6]:

```
DateTime                 datetime64[ns]
Radiation                       float64
Temperature                       int64
Pressure                        float64
Humidity                          int64
WindDirection(Degrees)          float64
Speed                           float64
TimeSunRise                      object
TimeSunSet                       object
dtype: object
```

Количество пустых значений - все значения заполнены:

In [7]:

```
for col in data.columns:
    temp_null_count = data[data[col].isnull()].shape[0]
    print('{} - {}'.format(col, temp_null_count))
```

```
DateTime - 0
Radiation - 0
Temperature - 0
Pressure - 0
Humidity - 0
WindDirection(Degrees) - 0
Speed - 0
TimeSunRise - 0
TimeSunSet - 0
```

Основные статистические характеристки набора данных:

In [8]:

```
data.describe()
```

Out[8]:

| | Radiation | Temperature | Pressure | Humidity | WindDirection(Degrees) | |
|---|---|---|---|---|---|---|
| **count** | 32686.000000 | 32686.000000 | 32686.000000 | 32686.000000 | 32686.000000 | 32686.0 |
| **mean** | 207.124697 | 51.103255 | 30.422879 | 75.016307 | 143.489821 | 6.2 |
| **std** | 315.916387 | 6.201157 | 0.054673 | 25.990219 | 83.167500 | 3.4 |
| **min** | 1.110000 | 34.000000 | 30.190000 | 8.000000 | 0.090000 | 0.0 |
| **25%** | 1.230000 | 46.000000 | 30.400000 | 56.000000 | 82.227500 | 3.3 |
| **50%** | 2.660000 | 50.000000 | 30.430000 | 85.000000 | 147.700000 | 5.6 |
| **75%** | 354.235000 | 55.000000 | 30.460000 | 97.000000 | 179.310000 | 7.8 |
| **max** | 1601.260000 | 71.000000 | 30.560000 | 103.000000 | 359.950000 | 40.5 |

Целевой признак **Radiation** является действительным числом и содержит значения от 1.11 ватт/м^2 до 1601.26 ватт/м^2.

# 3) Визуальное исследование датасета

## Диаграмма рассеяния:

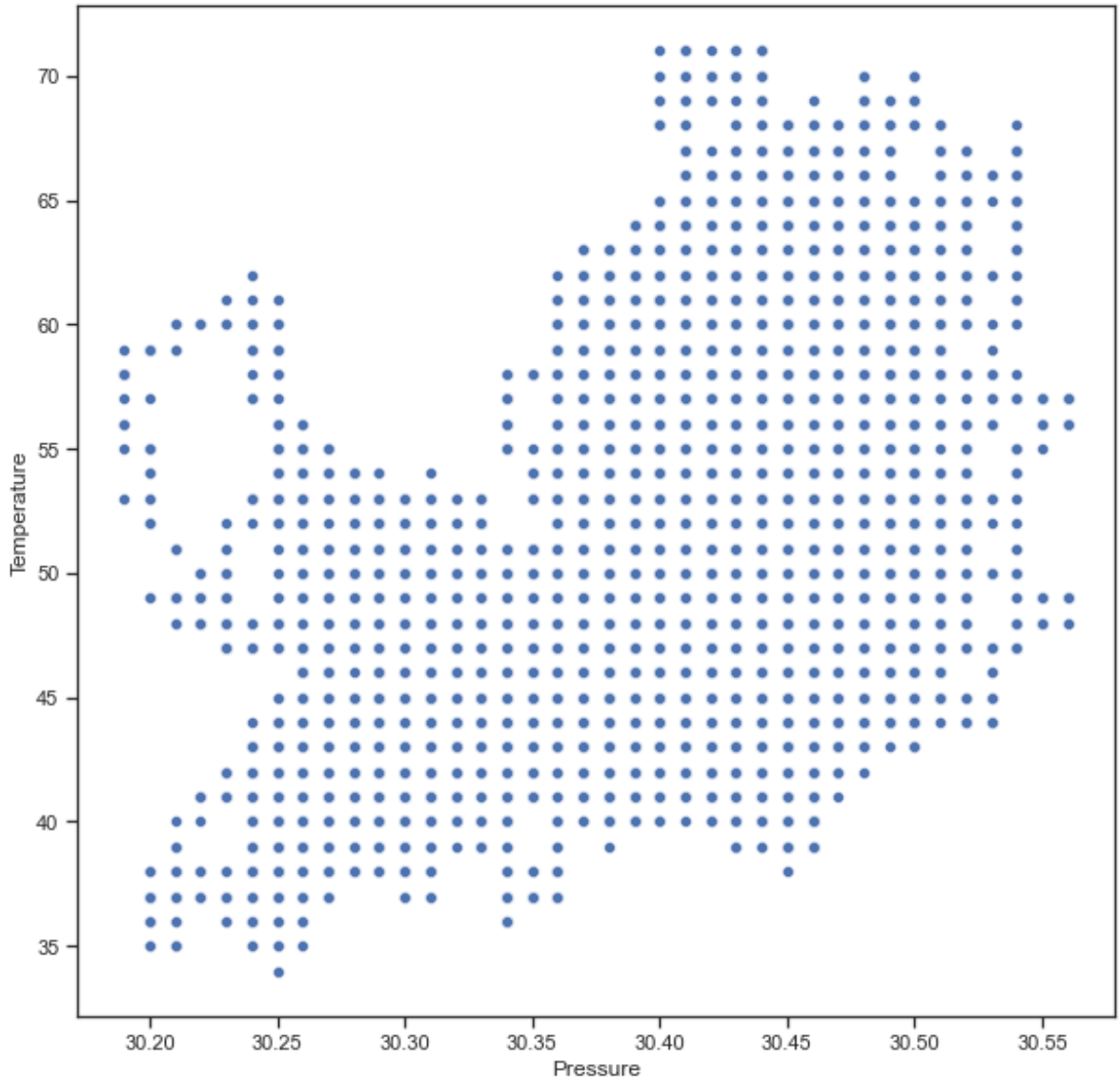
Позволяет построить распределение двух колонок данных и визуально обнаружить наличие зависимости.

In [9]:

```
fig, ax = plt.subplots(figsize=(10,10))
sns.scatterplot(ax=ax, x='Pressure', y='Temperature', data=data)
```

Out[9]:

```
<AxesSubplot:xlabel='Pressure', ylabel='Temperature'>
```



Можно увидеть, что между полями **Pressure** и **Temperature** почти отсутвует корреляция.

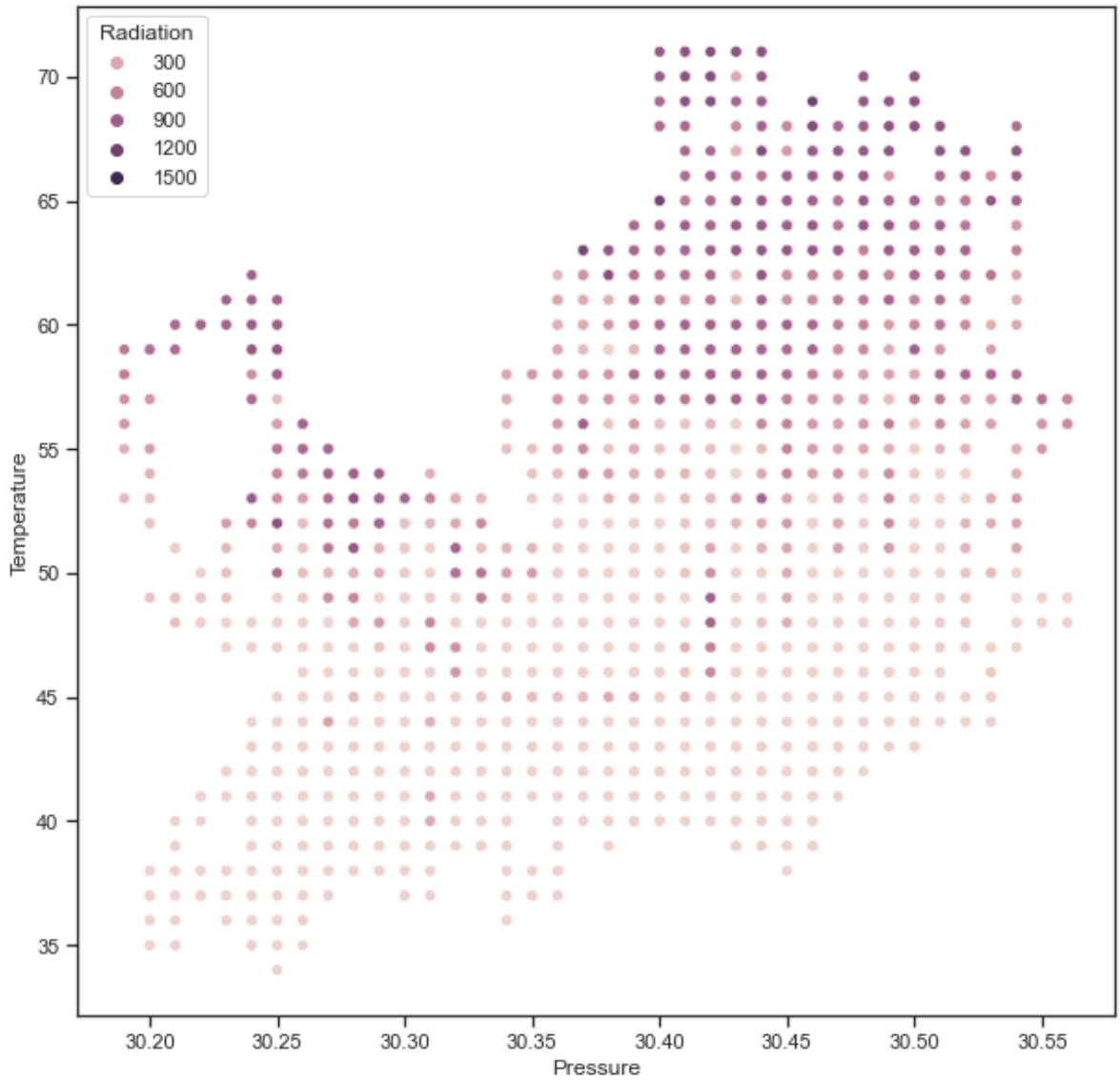
Посмотрим, наколько на это влияет целевой признак:

In [10]:

```
fig, ax = plt.subplots(figsize=(10,10))
sns.scatterplot(ax=ax, x='Pressure', y='Temperature', data=data, hue='Radiation')
```

Out[10]:

```
<AxesSubplot:xlabel='Pressure', ylabel='Temperature'>
```




Можно заметить, что значение целевого признака немного зависит от значения признака **Temperature**.

## Гистограмма:

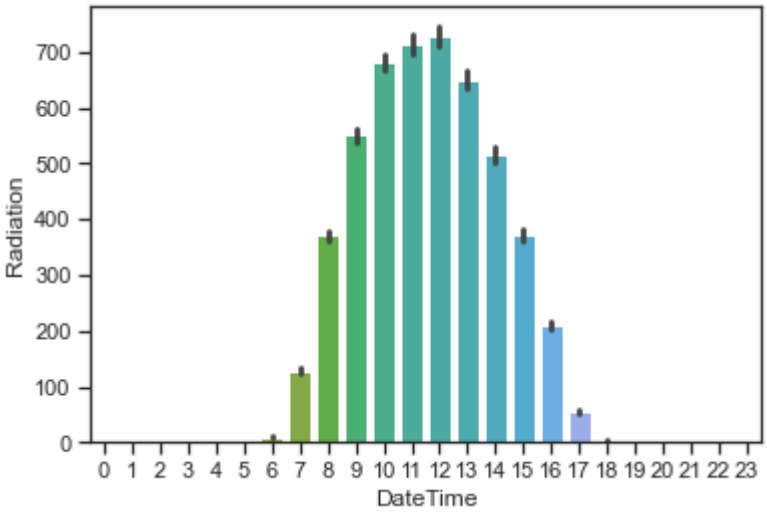
Отображает точечные оценки и доверительные интервалы в виде прямоугольных столбцов:

In [11]:

```
sns.barplot(x=data['DateTime'].dt.hour, y='Radiation', data=data)
```

Out[11]:

```
<AxesSubplot:xlabel='DateTime', ylabel='Radiation'>
```




Можно увидеть, что значение целевого признака **Radiation** зависит от времени.

Позволяет оценить плотность вероятности распределения данных:

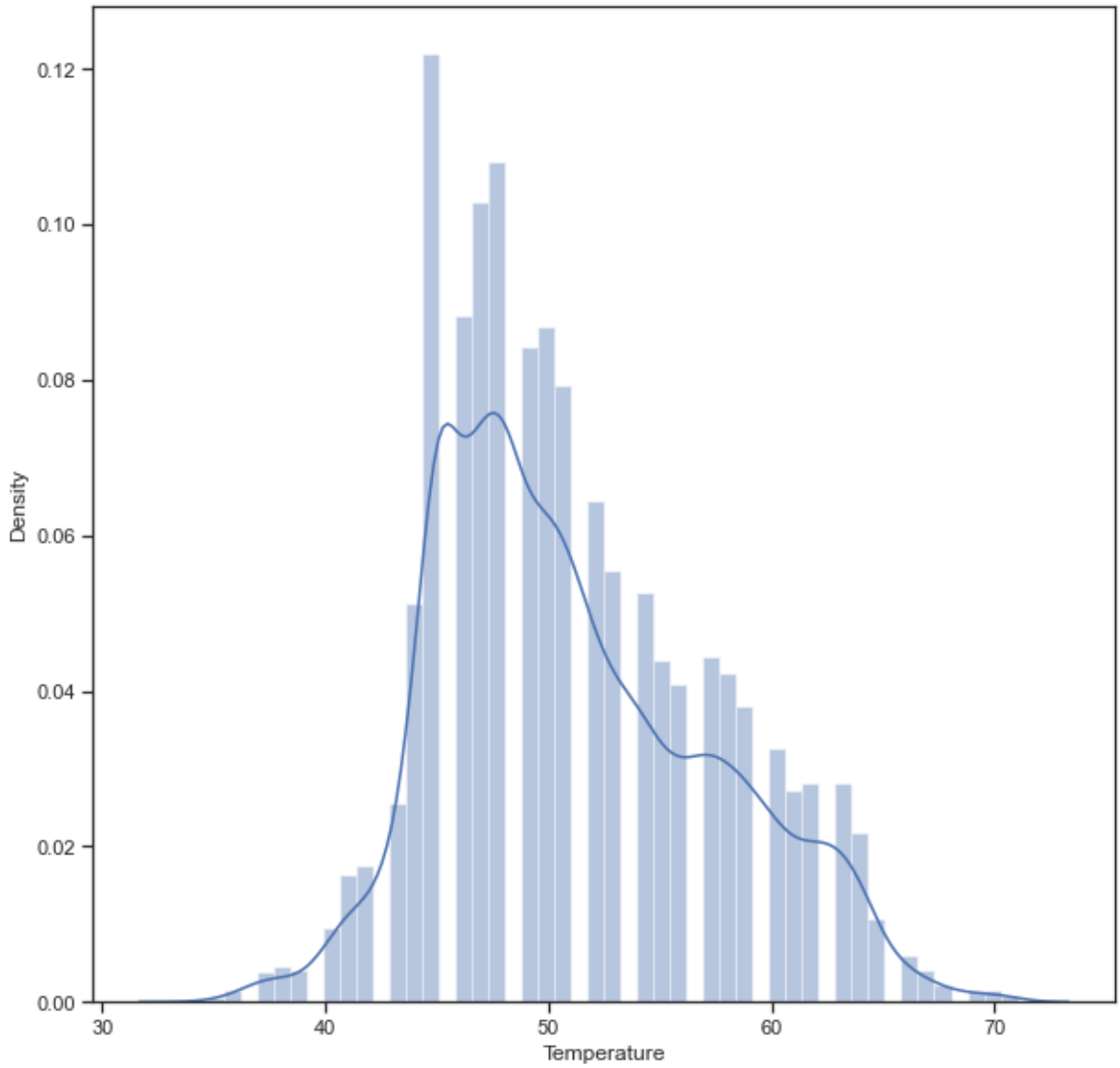
In [12]:

```
fig, ax = plt.subplots(figsize=(10,10))
sns.distplot(data['Temperature'])
```

```
D:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2551: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture 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[12]:

```
<AxesSubplot:xlabel='Temperature', ylabel='Density'>
```



## Jointplot

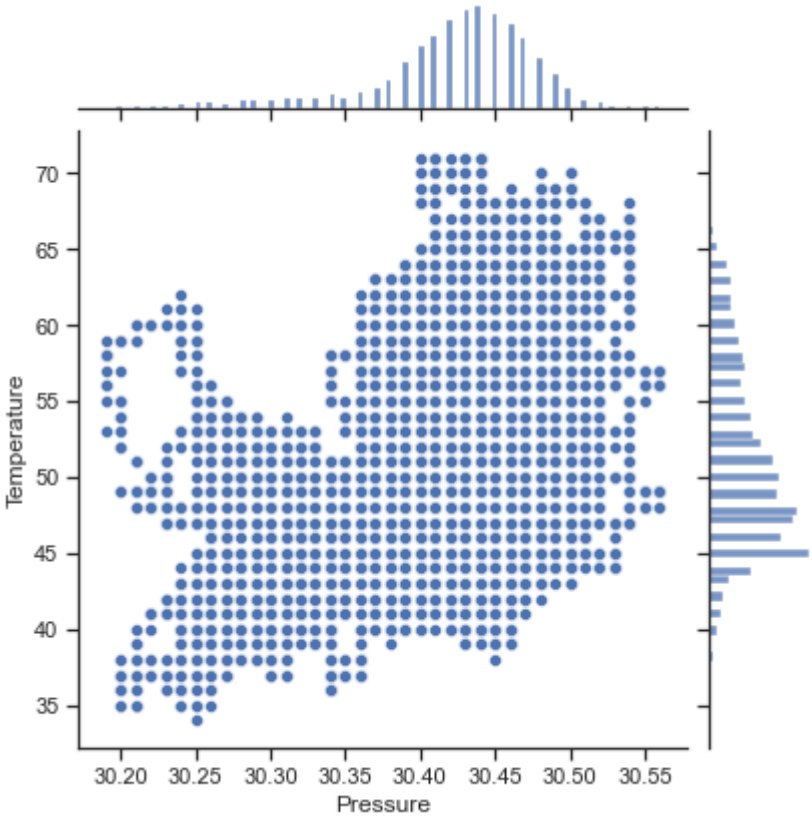
Комбинация гистограмм и диаграмм рассеивания.

In [13]:

```
sns.jointplot(x='Pressure', y='Temperature', data=data)
```

Out[13]:

```
<seaborn.axisgrid.JointGrid at 0x11aaaac39d0>
```

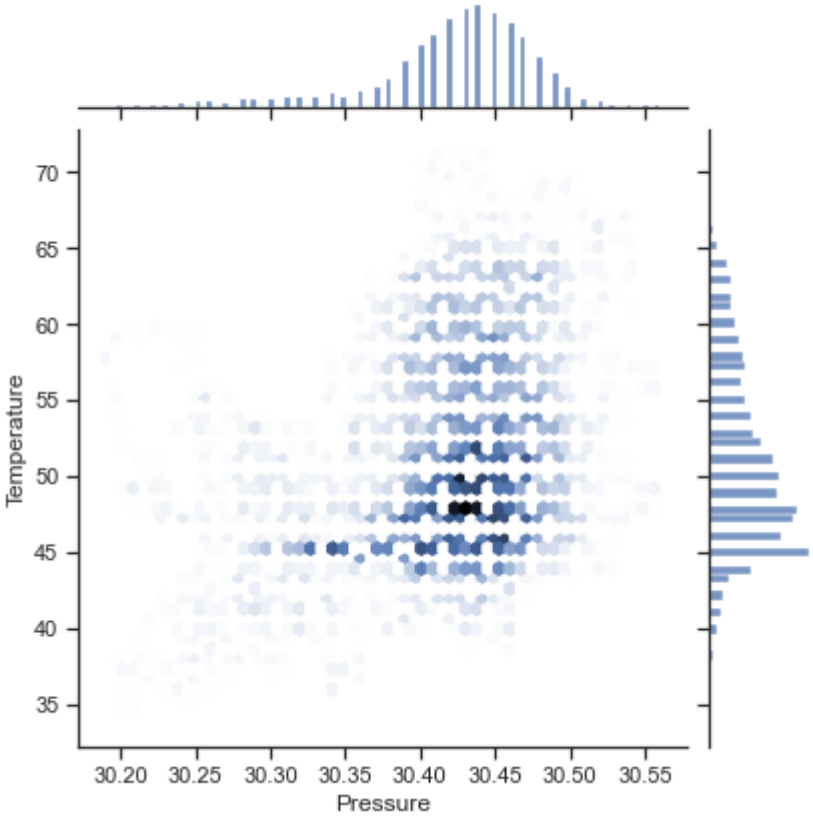


In [14]:

```
sns.jointplot(x='Pressure', y='Temperature', data=data, kind="hex")
```

Out[14]:

```
<seaborn.axisgrid.JointGrid at 0x11aab33a370>
```

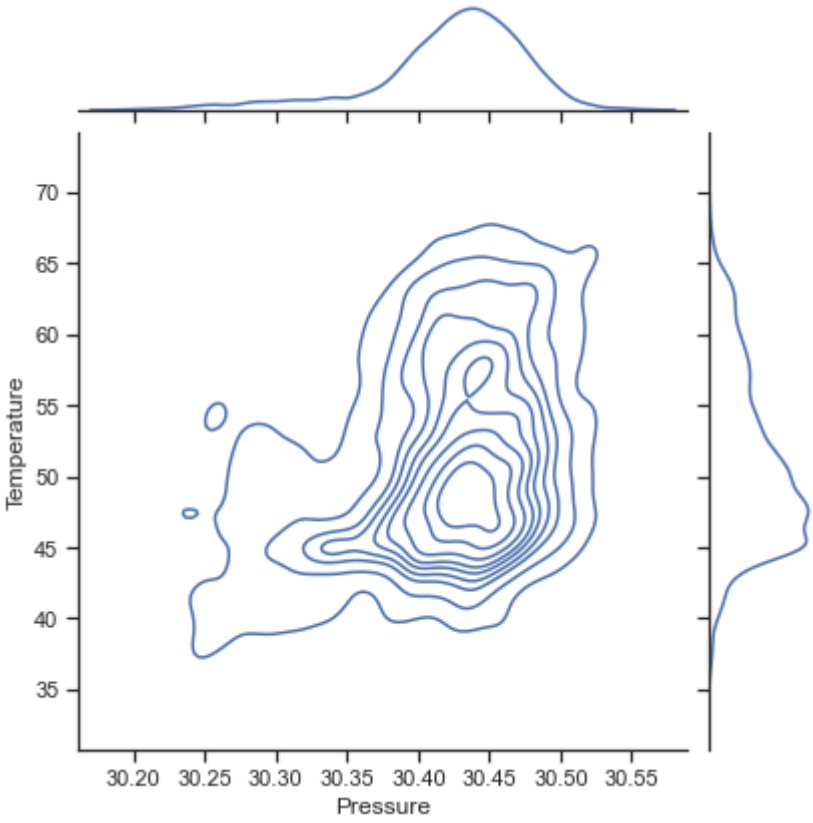


In [15]:

```
sns.jointplot(x='Pressure', y='Temperature', data=data, kind="kde")
```

Out[15]:

```
<seaborn.axisgrid.JointGrid at 0x11aaade2a60>
```




## "Парные диаграммы"

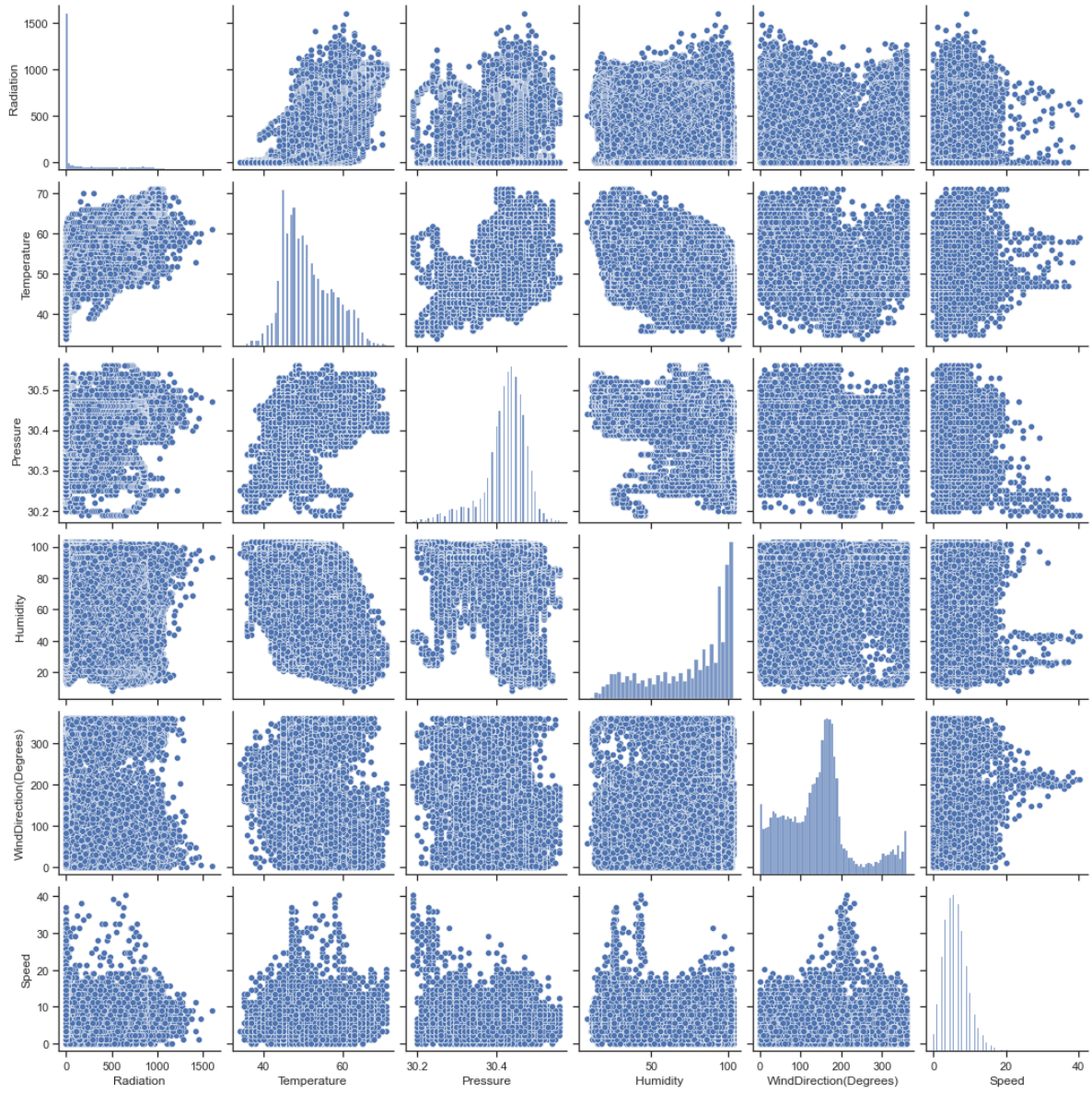
Комбинация гистограмм и диаграмм рассеивания для всего набора данных.

In [16]:

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

Out[16]:

```
<seaborn.axisgrid.PairGrid at 0x11aac941a90>
```



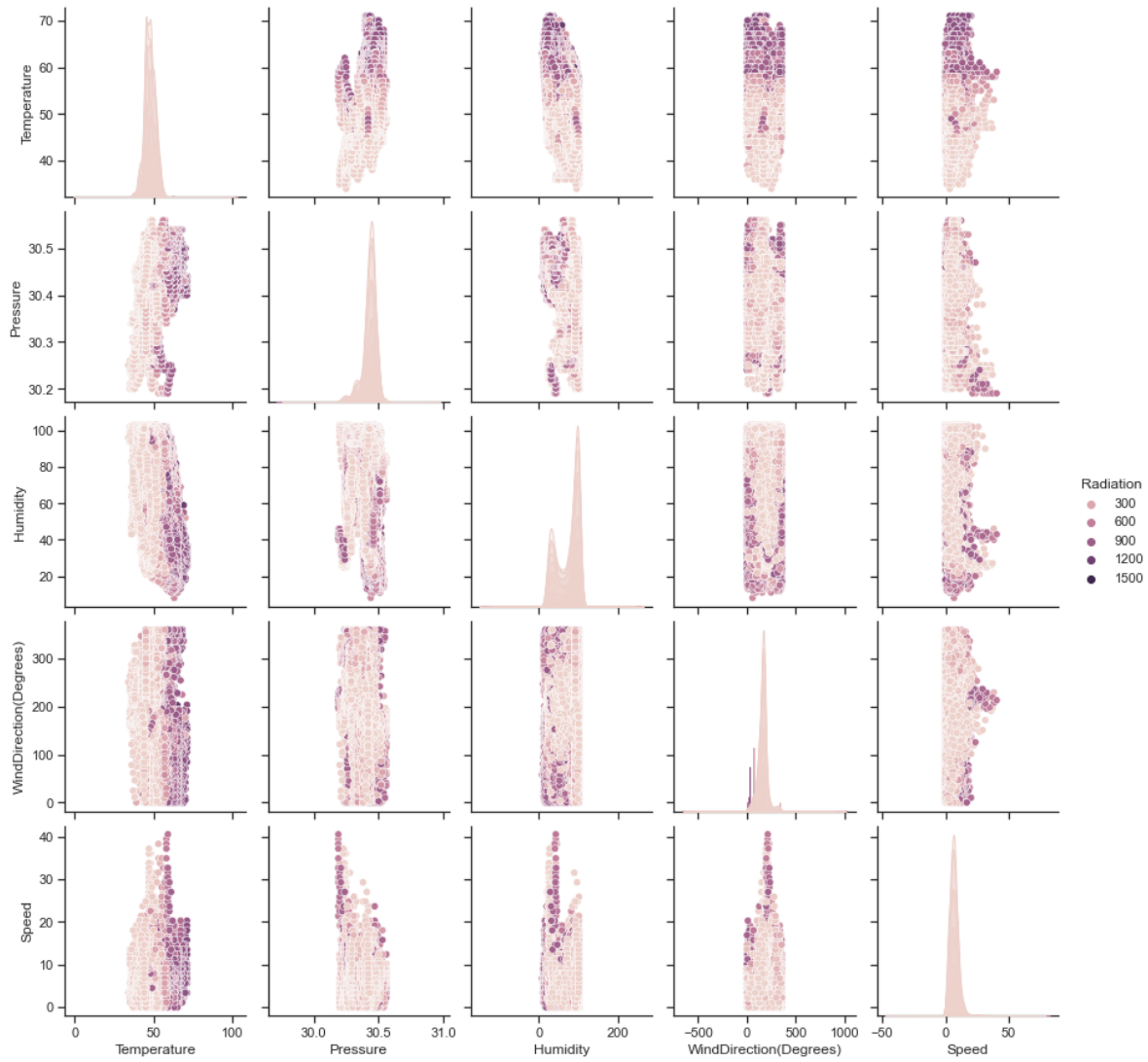

С помощью параметра "hue" возможна группировка по значениям какого-либо признака:

In [17]:

```
# Фильтр предупреждений
warnings.simplefilter('ignore', category=UserWarning)
sns.pairplot(data, hue='Radiation')
```

Out[17]:

```
<seaborn.axisgrid.PairGrid at 0x11aae4d0820>
```




## Ящик с усами

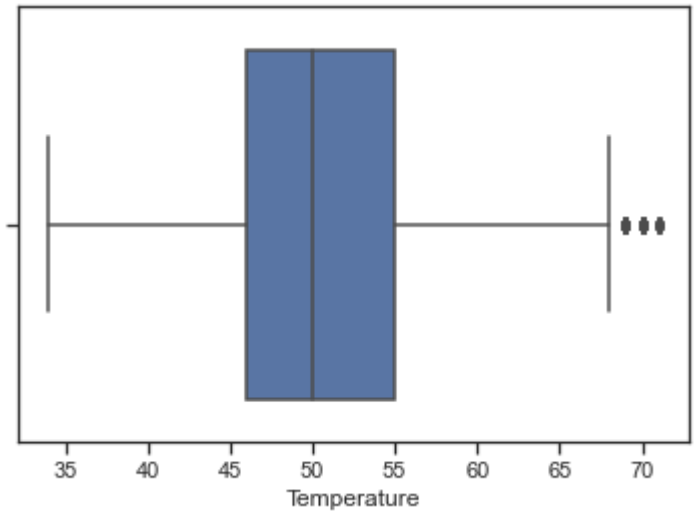
Отображает одномерное распределение вероятности.

In [18]:

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

Out[18]:

```
<AxesSubplot:xlabel='Temperature'>
```



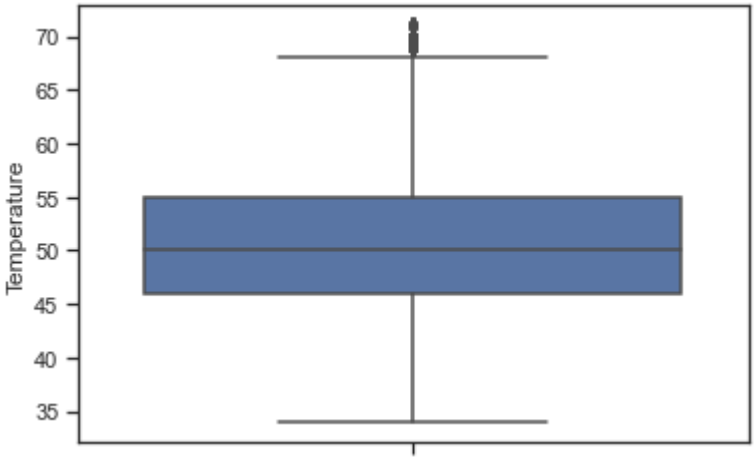

По вертикали:

In [19]:

```
sns.boxplot(y=data['Temperature'])
```

Out[19]:

```
<AxesSubplot:ylabel='Temperature'>
```



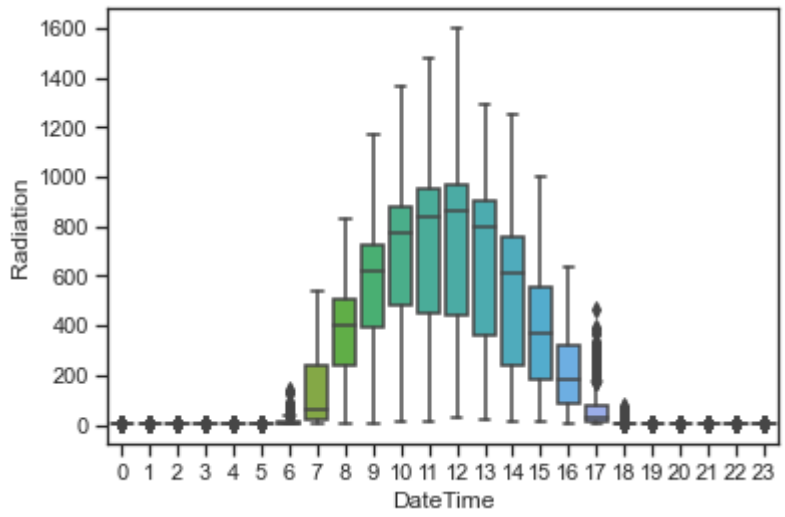

Распределение целевого признака **Radiation** сгруппированное по времени:

In [20]:

```
sns.boxplot(x=data['DateTime'].dt.hour, y='Radiation', data=data)
```

Out[20]:

```
<AxesSubplot:xlabel='DateTime', ylabel='Radiation'>
```




## Violin plot

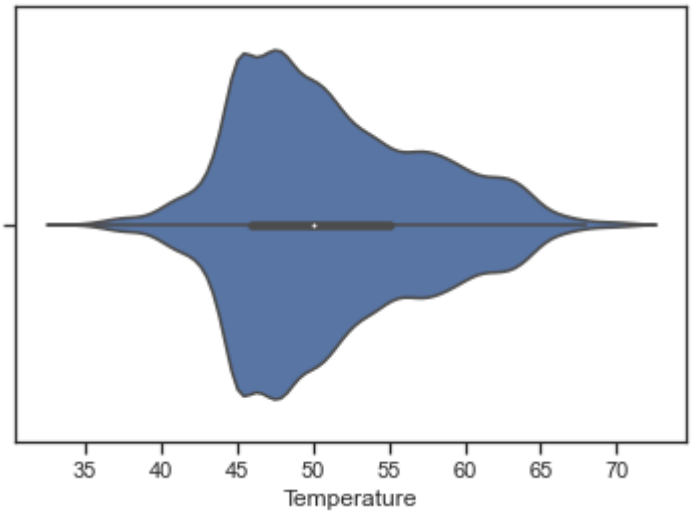
Похоже на предыдущую диаграмму, но по краям отображаются распределения плотности:

In [21]:

```
sns.violinplot(x=data['Temperature'])
```

Out[21]:

```
<AxesSubplot:xlabel='Temperature'>
```



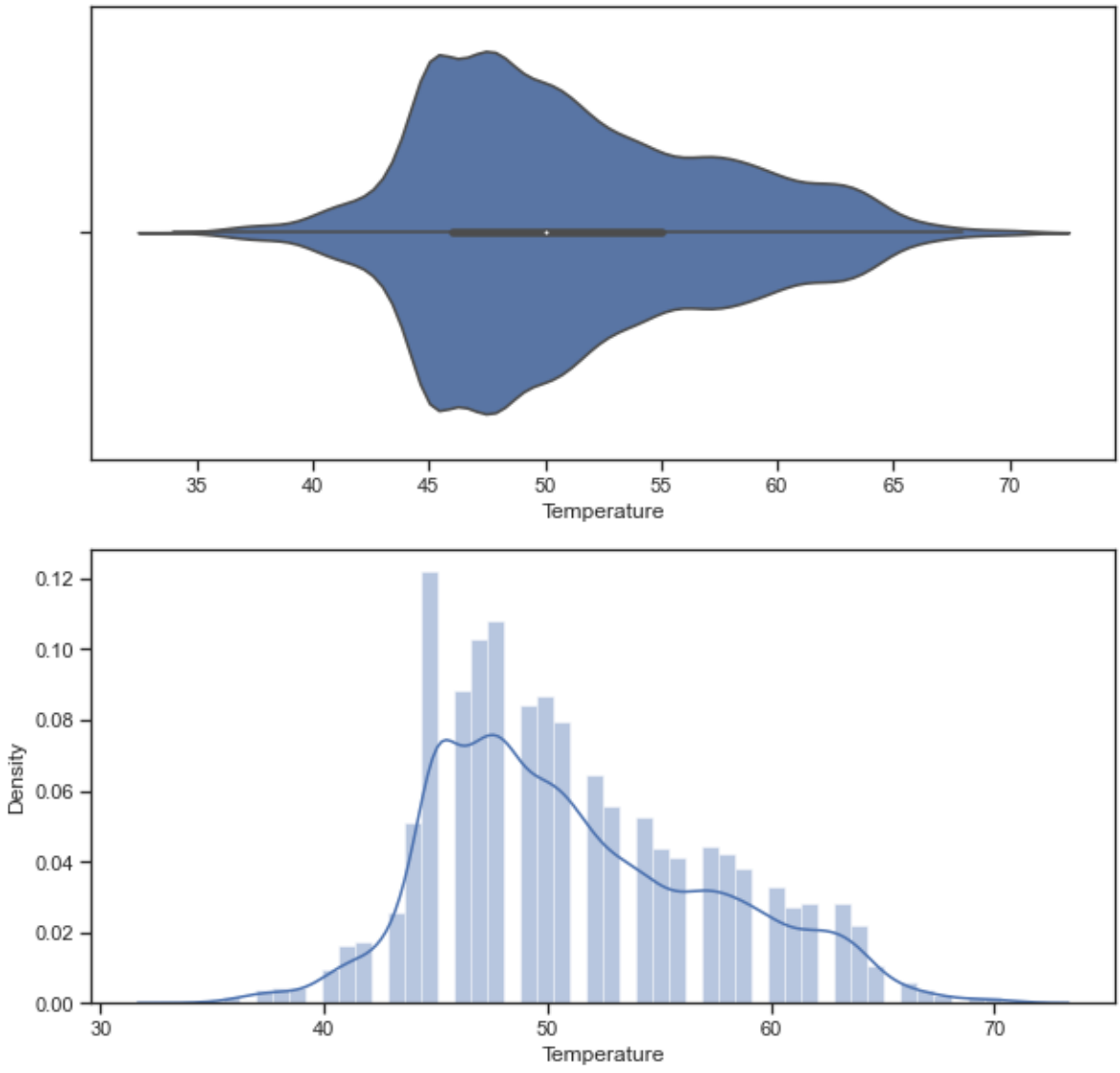
In [22]:

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

```
D:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2551: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture 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[22]:

```
<AxesSubplot:xlabel='Temperature', ylabel='Density'>
```




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

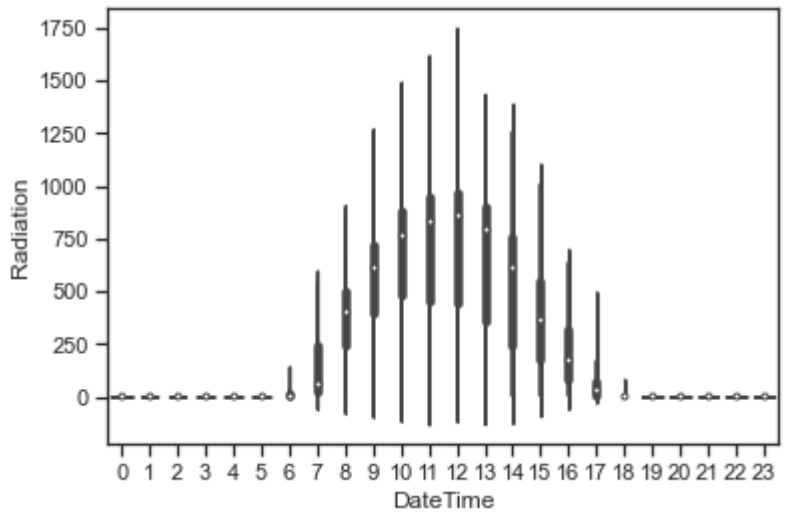
Распределение целевого признака **Radiation** сгруппированное по времени:

In [23]:

```
sns.violinplot(x=data['DateTime'].dt.hour, y='Radiation', data=data)
```

Out[23]:

```
<AxesSubplot:xlabel='DateTime', ylabel='Radiation'>
```




# 4) Информация о корреляции признаков

Проверка корреляции признаков позволяет решить две задачи:

1. Понять какие признаки (колонки датасета) наиболее сильно коррелируют с целевым признаком (колонка "Radiation"). 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.
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.

In [24]:

```
data.corr()
```

Out[24]:

| | Radiation | Temperature | Pressure | Humidity | WindDirection(Degrees) | |
|---|---|---|---|---|---|---|
| **Radiation** | 1.000000 | 0.734955 | 0.119016 | -0.226171 | -0.230324 | 0 |
| **Temperature** | 0.734955 | 1.000000 | 0.311173 | -0.285055 | -0.259421 | -0 |
| **Pressure** | 0.119016 | 0.311173 | 1.000000 | -0.223973 | -0.229010 | -0 |
| **Humidity** | -0.226171 | -0.285055 | -0.223973 | 1.000000 | -0.001833 | -0 |
| **WindDirection(Degrees)** | -0.230324 | -0.259421 | -0.229010 | -0.001833 | 1.000000 | 0 |
| **Speed** | 0.073627 | -0.031458 | -0.083639 | -0.211624 | 0.073092 | 1 |

Корреляционная матрица содержит коэффициенты корреляции между всеми парами признаков.

На основе корреляционной матрицы можно сделать следующие выводы:

- Целевой признак наиболее сильно коррелирует с **температурой** (0.73). Этот признак обязательно следует оставить в модели.
- Целевой признак слабо коррелирует с давлением (0.12), влажностью (-0.23), направлением ветра (-0.23) и скоростью ветра (-0.23). Скорее всего эти признаки стоит исключить из модели, возможно они только ухудшат качество модели.

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

In [25]:

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

Out[25]:

| | Radiation | Temperature | Pressure | Humidity | WindDirection(Degrees) | |
|---|---|---|---|---|---|---|
| **Radiation** | 1.000000 | 0.734955 | 0.119016 | -0.226171 | -0.230324 | 0 |
| **Temperature** | 0.734955 | 1.000000 | 0.311173 | -0.285055 | -0.259421 | -0 |
| **Pressure** | 0.119016 | 0.311173 | 1.000000 | -0.223973 | -0.229010 | -0 |
| **Humidity** | -0.226171 | -0.285055 | -0.223973 | 1.000000 | -0.001833 | -0 |
| **WindDirection(Degrees)** | -0.230324 | -0.259421 | -0.229010 | -0.001833 | 1.000000 | 0 |
| **Speed** | 0.073627 | -0.031458 | -0.083639 | -0.211624 | 0.073092 | 1 |

In [26]:

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

Out[26]:

| | Radiation | Temperature | Pressure | Humidity | WindDirection(Degrees) | |
|---|---|---|---|---|---|---|
| **Radiation** | 1.000000 | 0.538313 | 0.031905 | -0.084600 | -0.205663 | -( |
| **Temperature** | 0.538313 | 1.000000 | 0.199718 | -0.151589 | -0.269547 | -( |
| **Pressure** | 0.031905 | 0.199718 | 1.000000 | -0.150526 | -0.105480 | ( |
| **Humidity** | -0.084600 | -0.151589 | -0.150526 | 1.000000 | -0.041277 | -( |
| **WindDirection(Degrees)** | -0.205663 | -0.269547 | -0.105480 | -0.041277 | 1.000000 | ( |
| **Speed** | -0.015171 | -0.052562 | 0.021822 | -0.156442 | 0.060362 | 1 |

In [27]:

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

Out[27]:

| | Radiation | Temperature | Pressure | Humidity | WindDirection(Degrees) | |
|---|---|---|---|---|---|---|
| **Radiation** | 1.000000 | 0.717746 | 0.045577 | -0.117571 | -0.314843 | -( |
| **Temperature** | 0.717746 | 1.000000 | 0.279273 | -0.212946 | -0.375322 | -( |
| **Pressure** | 0.045577 | 0.279273 | 1.000000 | -0.216916 | -0.155669 | ( |
| **Humidity** | -0.117571 | -0.212946 | -0.216916 | 1.000000 | -0.062702 | -( |
| **WindDirection(Degrees)** | -0.314843 | -0.375322 | -0.155669 | -0.062702 | 1.000000 | ( |
| **Speed** | -0.021066 | -0.074065 | 0.029907 | -0.218425 | 0.086524 | 1 |

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

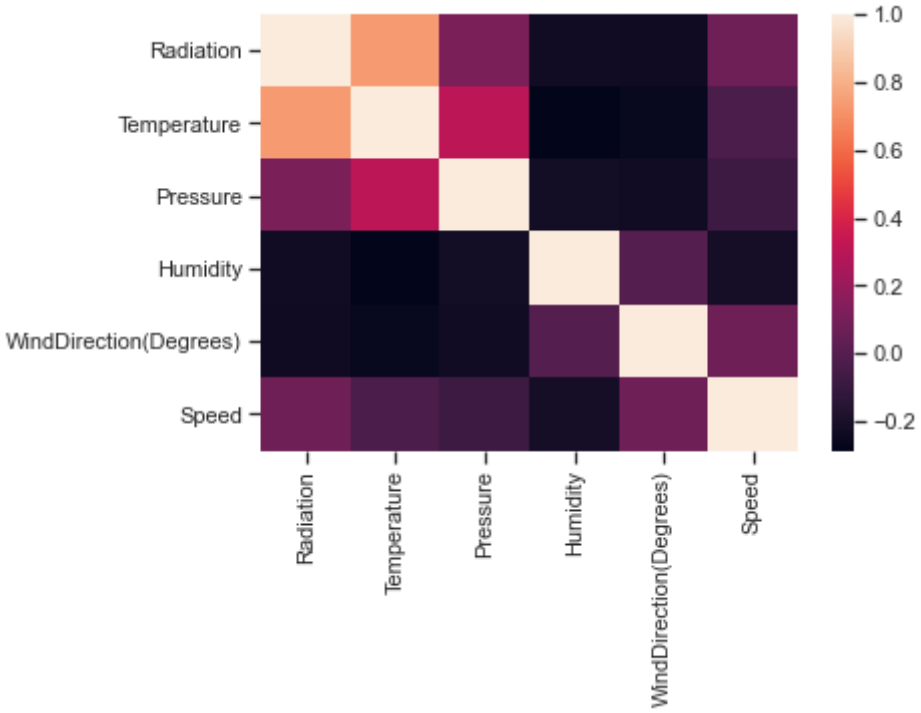
Для визуализации корреляционной матрицы будем использовать "тепловую карту" **heatmap** которая показывает степень корреляции различными цветами.

In [28]:

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

Out[28]:

<AxesSubplot:>



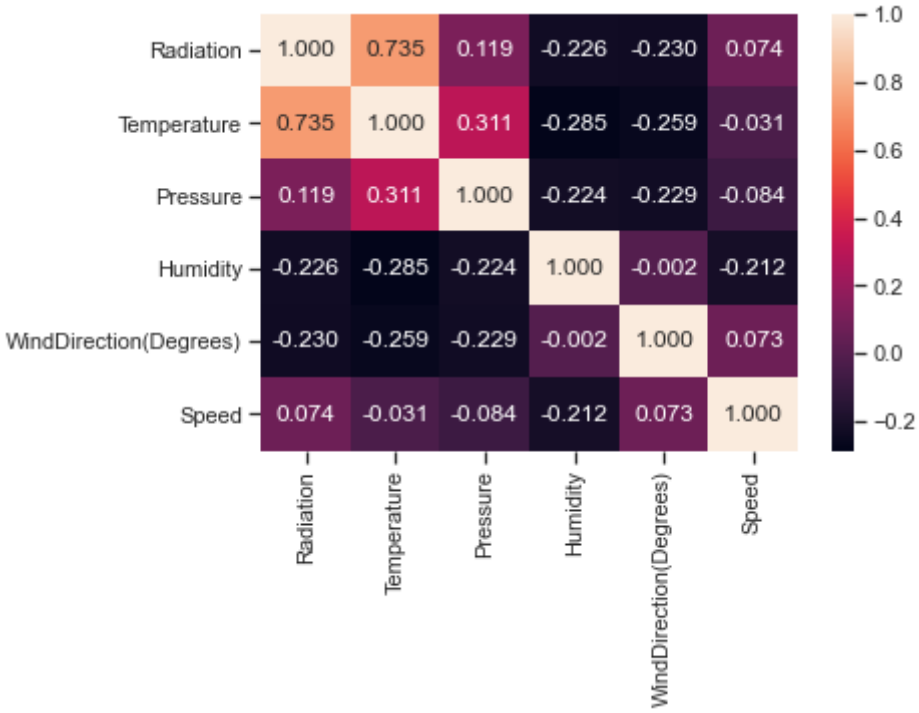

Вывод значений в ячейках:

In [29]:

```
sns.heatmap(data.corr(), annot=True, fmt='.3f')
```

Out[29]:

<AxesSubplot:>



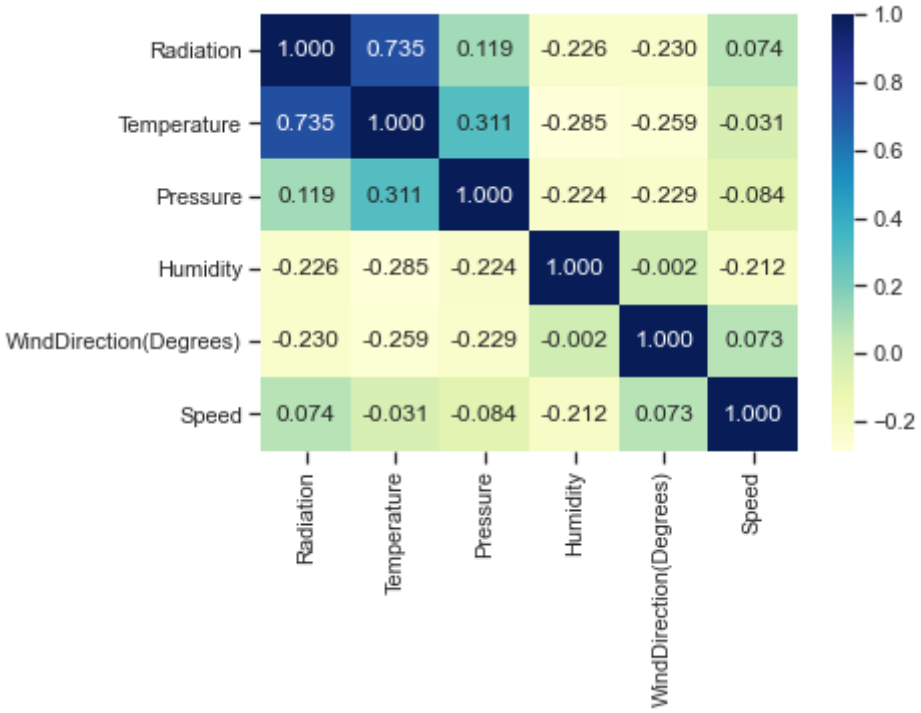

Изменение цветовой гаммы:

In [30]:

```
sns.heatmap(data.corr(), cmap='YlGnBu', annot=True, fmt='.3f')
```

Out[30]:

<AxesSubplot:>



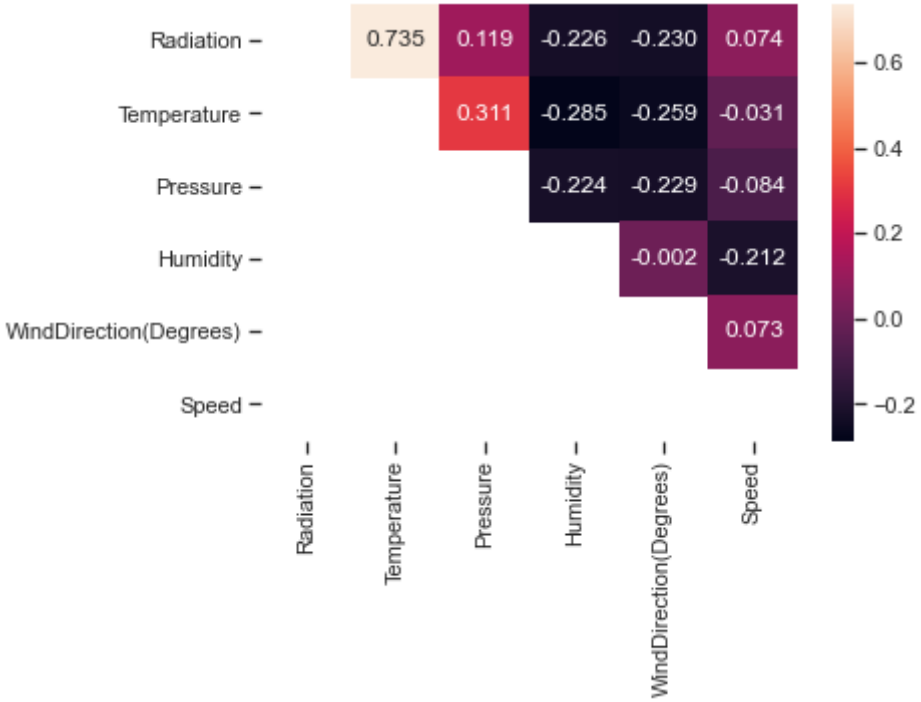

Треугольный вариант матрицы:

In [31]:

```
mask = np.zeros_like(data.corr(), dtype=np.bool)
mask[np.tril_indices_from(mask)] = True
sns.heatmap(data.corr(), mask=mask, annot=True, fmt='.3f')
```

Out[31]:

<AxesSubplot:>



In [32]:

```
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')
```