eda_visualization

February 20, 2019

1 Цель лабораторной работы

Изучить различные методы визуализации данных [1].

2 Задание

Требуется выполнить следующие действия [1]:

- Выбрать набор данных (датасет).
- Создать ноутбук, который содержит следующие разделы:
  1. Текстовое описание выбранного 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 Ход выполнения работы

3.1 Текстовое описание набора данных

В качестве набора данных используются метрологические данные с метеостанции HI-SEAS (Hawaii Space Exploration Analog and Simulation) за четыре месяца (с сентября по декабрь 2016 года) [2].

Данный набор данных состоит из одного файла `SolarPrediction.csv`, содержащего все данные этого датасета. Данный файл содержит следующие колонки:

- `UNIXTime` — временная метка измерения в формате UNIX;
- `Data` — дата измерения;
- `Time` — время измерения (в местной временной зоне);
- `Radiation` — солнечное излучение (Вт/м^2);
- `Temperature` — температура (°F);
- `Pressure` — атмосферное давление (дюймов ртутного столба);
- `Humidity` — относительная влажность (%);
- `WindDirection(Degrees)` — направление ветра (°);
- `Speed` — скорость ветра (миль/ч);
- `TimeSunRise` — время восхода (в местной временной зоне);
- `TimeSunSet` — время заката (в местной временной зоне).

3.2 Основные характеристики набора данных

Подключим все необходимые библиотеки:

```
In [1]: from datetime import datetime
        import matplotlib.pyplot as plt
        import numpy as np
        import pandas as pd
        import seaborn as sns
```

Настроим отображение графиков [3, 4]:

```
In [2]: # Enable inline plots
        %matplotlib inline

        # Set plot style
        sns.set(style="ticks")

        # Set plots formats to save high resolution PNG
        from IPython.display import set_matplotlib_formats
        set_matplotlib_formats("retina")
```

Зададим ширину текстового представления данных, чтобы в дальнейшем текст в отчёте влезал на A4:

```
In [3]: pd.set_option("display.width", 70)
```

Загрузим непосредственно данные:

```
In [4]: data = pd.read_csv("./SolarPrediction.csv")
```

Преобразуем временные колонки в соответствующий временной формат:

```
In [5]: data["UNIXTime"] = pd.to_datetime(data["UNIXTime"], unit="s", utc=True).dt.
        data["Data"] = data["UNIXTime"].dt.date
        data["Time"] = data["UNIXTime"].dt.time
        data["TimeSunRise"] = pd.to_datetime(data["TimeSunRise"], infer_datetime_fo
        data["TimeSunSet"] = pd.to_datetime(data["TimeSunSet"], infer_datetime_form
        data = data.rename({"Data": "Date", "WindDirection(Degrees)": "WindDirectio
```

Проверим полученные типы:

```
In [6]: data.dtypes
```

```
Out[6]: UNIXTime          datetime64[ns, Pacific/Honolulu]
        Date                                       object
        Time                                       object
        Radiation                                 float64
        Temperature                                 int64
        Pressure                                  float64
        Humidity                                    int64
        WindDirection                             float64
        Speed                                     float64
        TimeSunRise                                object
        TimeSunSet                                 object
        dtype: object
```

Посмотрим на данные в данном наборе данных:

```
In [7]: data.head()
```

```
Out[7]:                    UNIXTime        Date      Time  Radiation  \
        0 2016-09-29 23:55:26-10:00  2016-09-29  23:55:26       1.21
        1 2016-09-29 23:50:23-10:00  2016-09-29  23:50:23       1.21
        2 2016-09-29 23:45:26-10:00  2016-09-29  23:45:26       1.23
        3 2016-09-29 23:40:21-10:00  2016-09-29  23:40:21       1.21
        4 2016-09-29 23:35:24-10:00  2016-09-29  23:35:24       1.17

           Temperature  Pressure  Humidity  WindDirection  Speed  \
        0           48     30.46        59         177.39   5.62
        1           48     30.46        58         176.78   3.37
        2           48     30.46        57         158.75   3.37
        3           48     30.46        60         137.71   3.37
        4           48     30.46        62         104.95   5.62

          TimeSunRise TimeSunSet
        0    06:13:00   18:13:00
        1    06:13:00   18:13:00
        2    06:13:00   18:13:00
        3    06:13:00   18:13:00
        4    06:13:00   18:13:00
```

Очевидно, что все эти временные характеристики в таком виде нам не особо интересны. Преобразуем все нечисловые столбцы в числовые. В целом колонка `UNIXTime` нам не интересна, дата скорее интереснее в виде дня в году. Время измерения может быть интересно в двух видах: просто секунды с полуночи, и время, нормализованное относительно рассвета и заката.

```
In [8]: df = data.copy()

        df["Day"] = df["UNIXTime"].dt.dayofyear

        # Using method from [5]
        df["TimeInSeconds"] = df["Time"].map(lambda t: (datetime.combine(datetime.m

        sunrise = df["TimeSunRise"].map(lambda t: (datetime.combine(datetime.min, t
        sunset = df["TimeSunSet"].map(lambda t: (datetime.combine(datetime.min, t)
        df["DayPart"] = (df["TimeInSeconds"] - sunrise) / (sunset - sunrise)

        df = df.drop(["UNIXTime", "Date", "Time", "TimeSunRise", "TimeSunSet"], axi

        df.head()
```

```
Out[8]:    Radiation  Temperature  Pressure  Humidity  WindDirection  Speed  \
        0       1.21           48     30.46        59         177.39   5.62
        1       1.21           48     30.46        58         176.78   3.37
        2       1.23           48     30.46        57         158.75   3.37
        3       1.21           48     30.46        60         137.71   3.37
```

```
4       1.17           48     30.46        62         104.95   5.62

   Day  TimeInSeconds   DayPart
0  273        86126.0  1.475602
1  273        85823.0  1.468588
2  273        85526.0  1.461713
3  273        85221.0  1.454653
4  273        84924.0  1.447778
```

```
In [9]: df.dtypes
```

```
Out[9]: Radiation        float64
        Temperature        int64
        Pressure         float64
        Humidity           int64
        WindDirection    float64
        Speed            float64
        Day                int64
        TimeInSeconds    float64
        DayPart          float64
        dtype: object
```

С такими данными уже можно работать. Проверим размер набора данных:

```
In [10]: df.shape
```

```
Out[10]: (32686, 9)
```

Проверим основные статистические характеристики набора данных:

```
In [11]: df.describe()
```

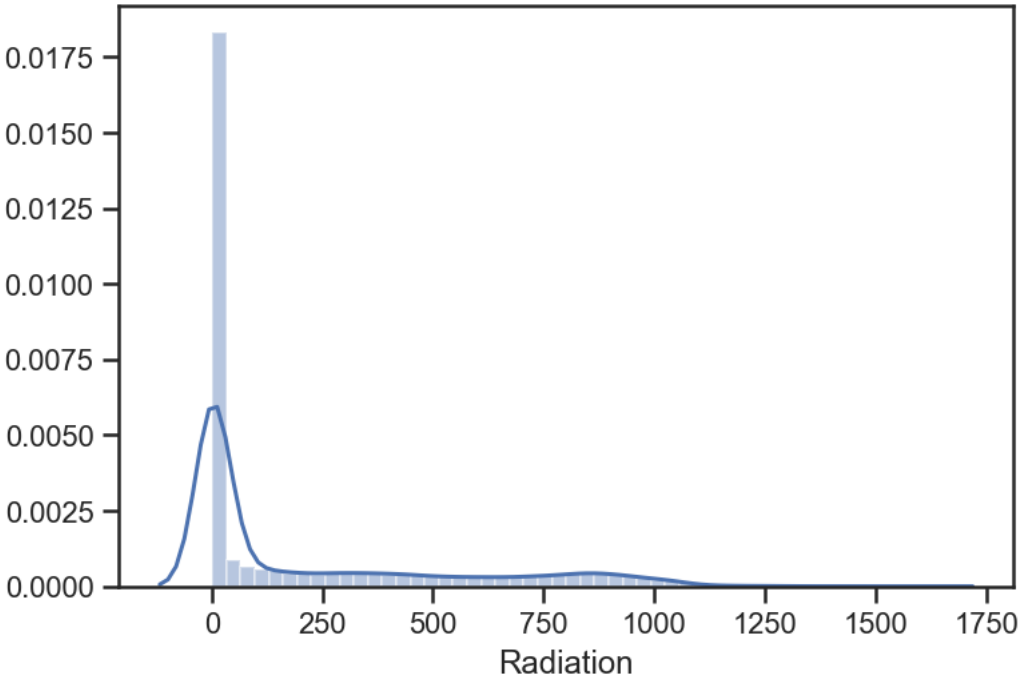
```
Out[11]:           Radiation   Temperature      Pressure      Humidity  \
         count  32686.000000  32686.000000  32686.000000  32686.000000
         mean     207.124697     51.103255     30.422879     75.016307
         std      315.916387      6.201157      0.054673     25.990219
         min        1.110000     34.000000     30.190000      8.000000
         25%        1.230000     46.000000     30.400000     56.000000
         50%        2.660000     50.000000     30.430000     85.000000
         75%      354.235000     55.000000     30.460000     97.000000
         max     1601.260000     71.000000     30.560000    103.000000

                WindDirection         Speed           Day  TimeInSeconds  \
         count   32686.000000  32686.000000  32686.000000   32686.000000
         mean      143.489821      6.243869    306.110965   43277.574068
         std        83.167500      3.490474     34.781367   24900.749819
         min         0.090000      0.000000    245.000000       1.000000
         25%        82.227500      3.370000    277.000000   21617.000000
         50%       147.700000      5.620000    306.000000   43230.000000
         75%       179.310000      7.870000    334.000000   64849.000000
         max       359.950000     40.500000    366.000000   86185.000000
```

```
              DayPart
count  32686.000000
mean       0.482959
std        0.602432
min       -0.634602
25%       -0.040139
50%        0.484332
75%        1.006038
max        1.566061
```

3.3 Визуальное исследование датасета

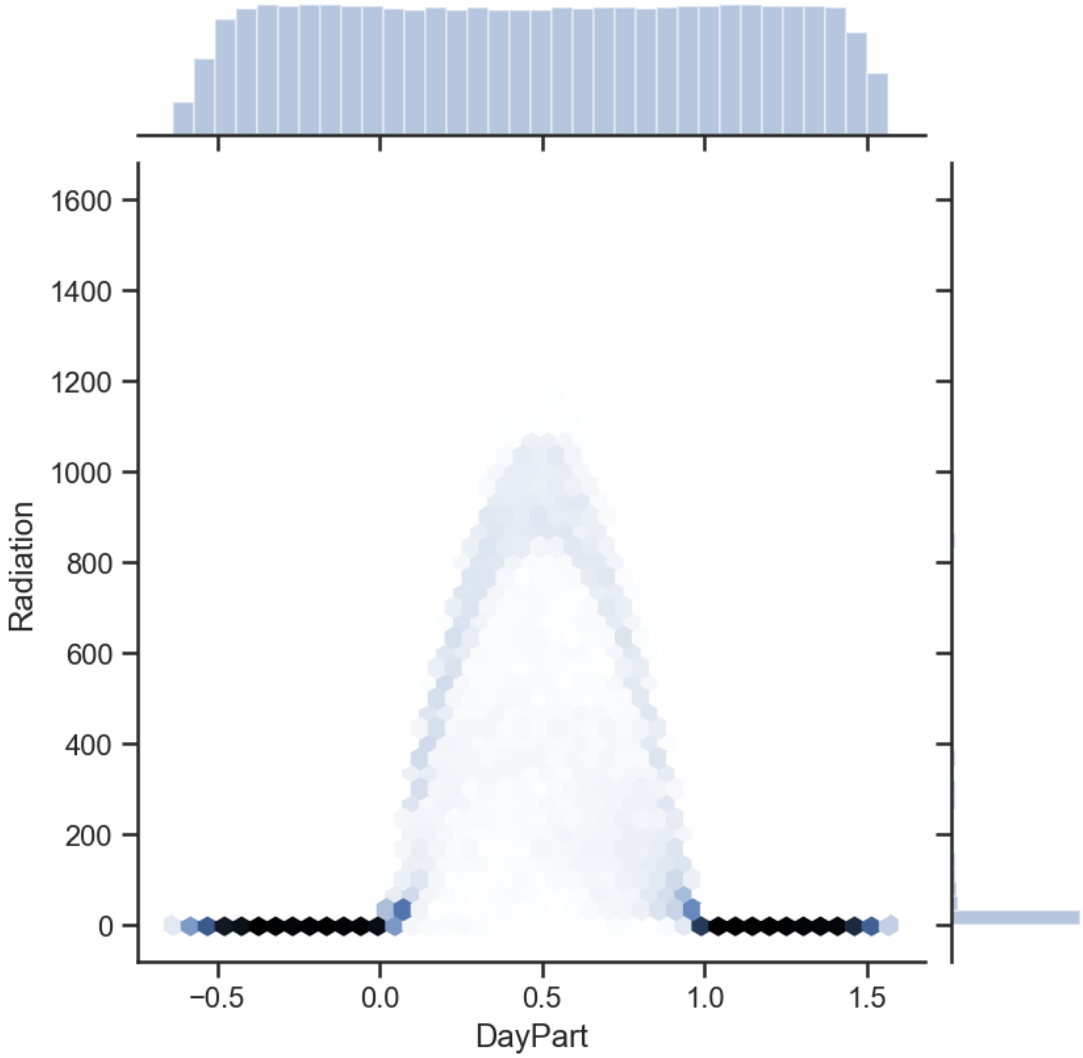
Оценим распределение целевого признака — мощности солнечного излучения:

```
In [12]: sns.distplot(df["Radiation"]);
```



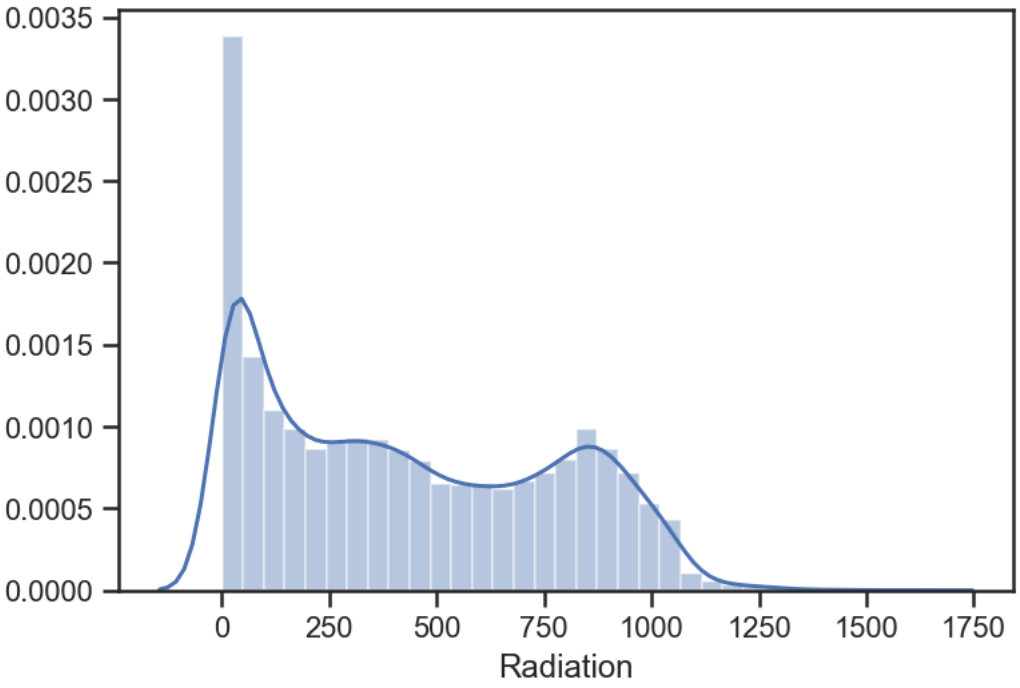
Видно, что имеется большой перевес в пользу практически нулевого излучения. Оценим, насколько мощность солнечного излучения зависит от наличия солнца на небе:

```
In [13]: sns.jointplot(x="DayPart", y="Radiation", data=df, kind="hex");
```



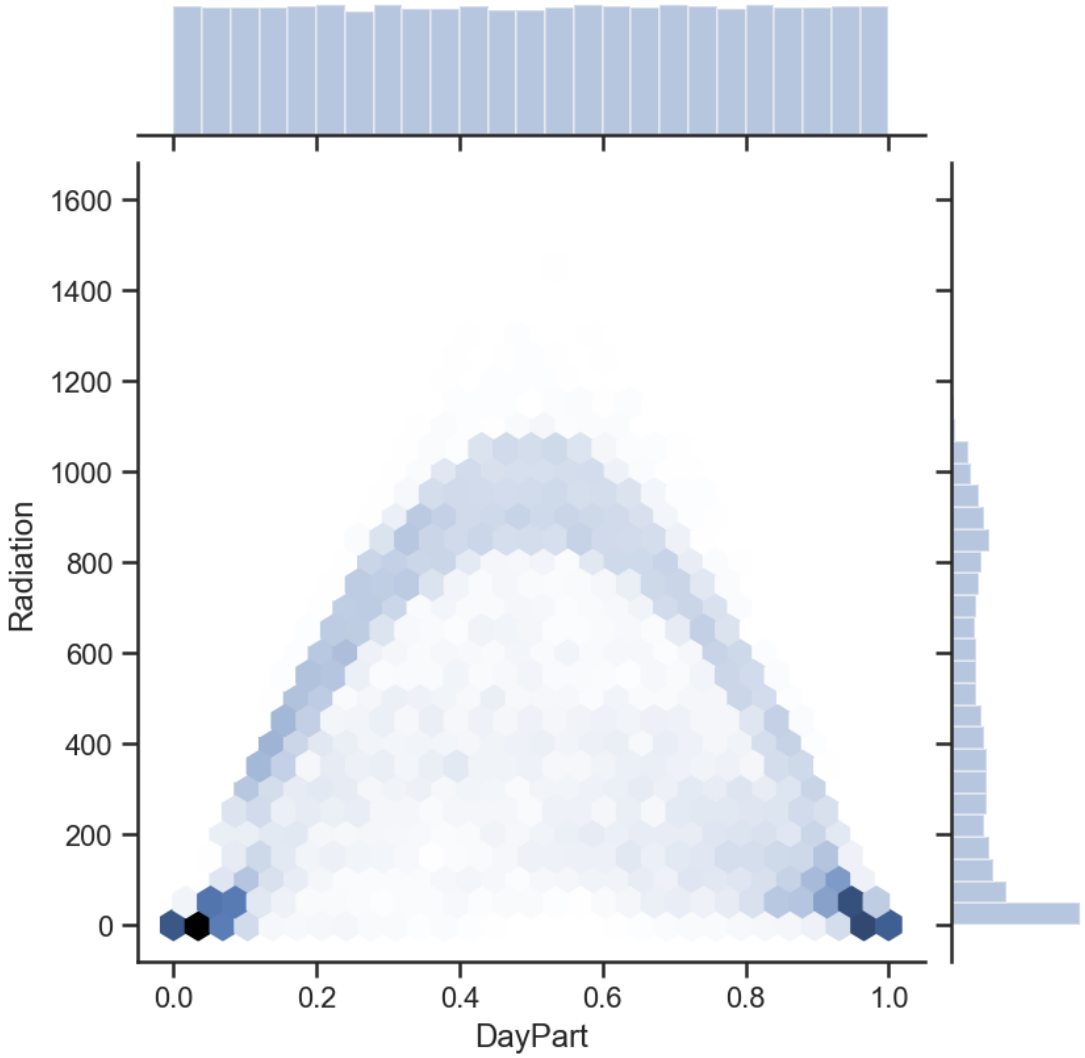
Видно, что если солнца нет на небе, то мощность солнечного излучения стремится к нулю. Посмотрим на распределение мощности излучения в течение дня:

```
In [14]: dfd = df[(df["DayPart"] >= 0) & (df["DayPart"] <= 1)]
         sns.distplot(dfd["Radiation"]);
```



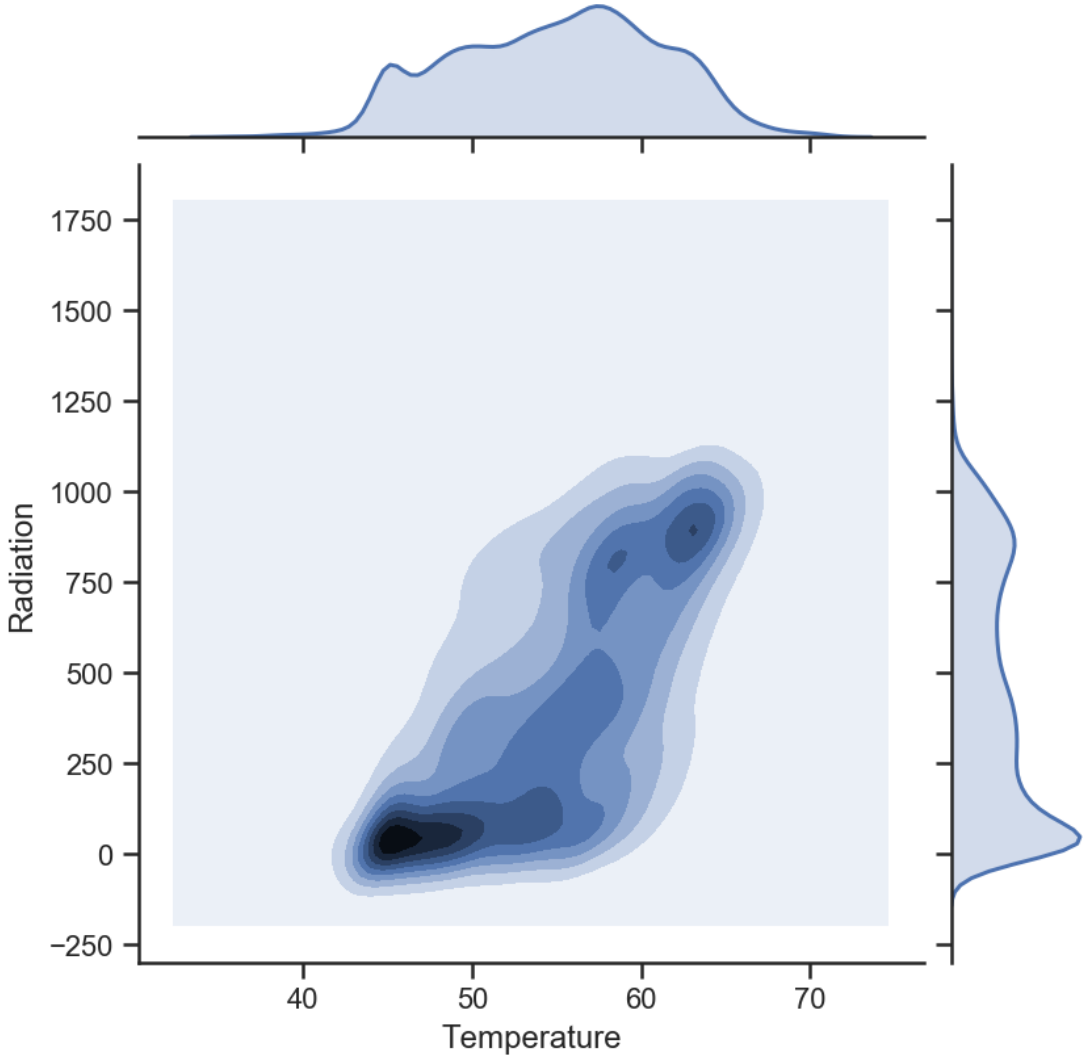
Теперь оценить влияние времени дня на мощность солнечного излучения будет заметно проще:

```
In [15]: sns.jointplot(x="DayPart", y="Radiation", data=dfd, kind="hex");
```



Посмотрим также на зависимость мощности солнечного излучения от температуры:

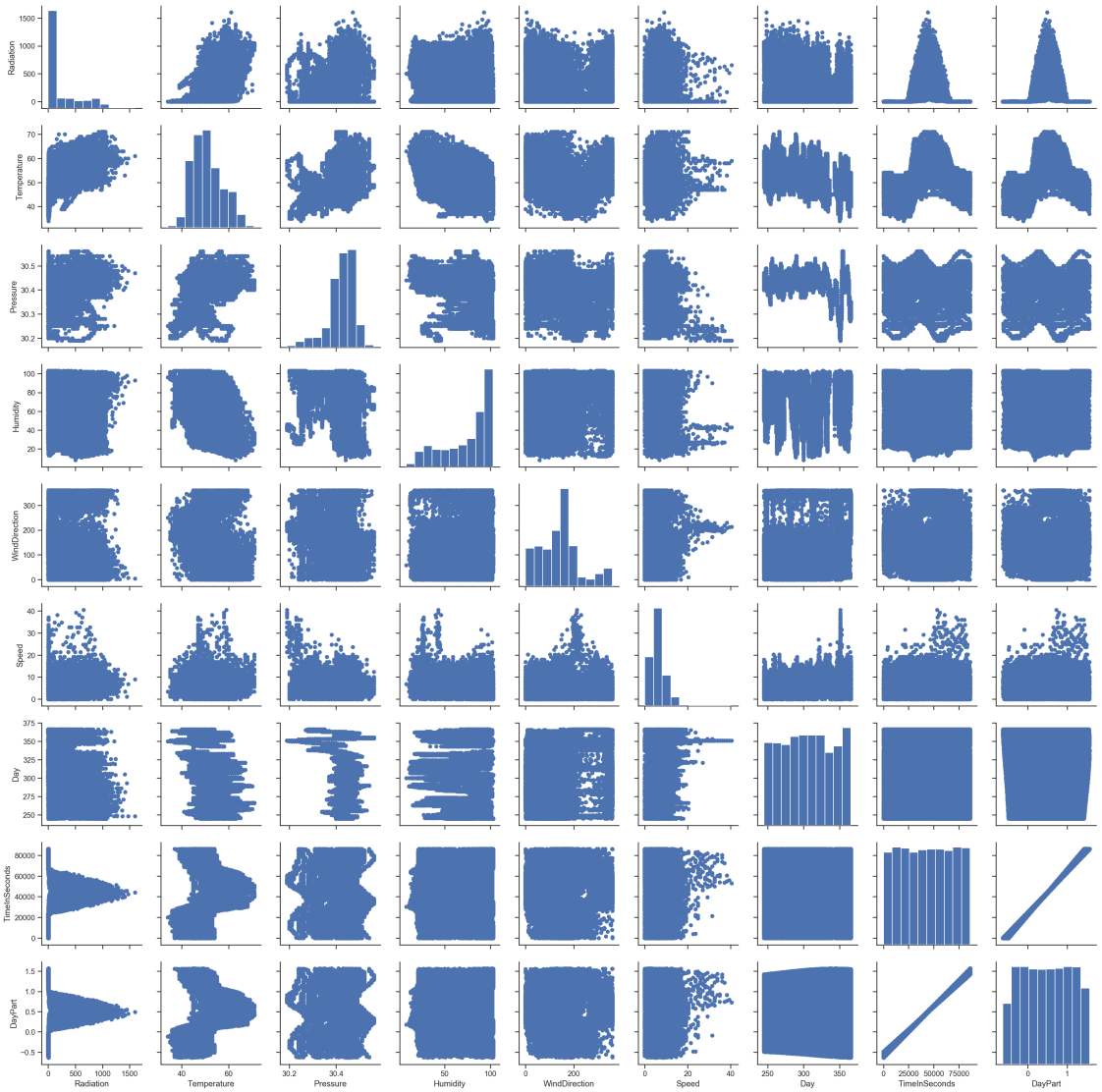
```
In [16]: sns.jointplot(x="Temperature", y="Radiation", data=dfd, kind="kde");
```



Видно, что некоторая зависимость определённо есть, но не настолько большая, насколько хотелось бы. Возможно на большей выборке эта зависимость стала бы ещё менее заметной.

Построим парные диаграммы по всем показателям по исходному набору данных:

```
In [17]: sns.pairplot(df, plot_kws=dict(linewidth=0));
```



Видно, что близкая к линейной зависимость есть только между временем с начала дня и приведение этого времени к промежутку наличия солнца на небе. Кроме того, видно выброс по скорости ветра, который происходил в один день и во время которого мощность солнечного излучения была близка к нулю. Имеет смысл учесть это при дальнейшем использовании этого набора данных и либо убрать этот выброс вообще, либо следить за тем, чтобы модели не переобучались на скорость ветра.

3.4 Информация о корреляции признаков

Построим корреляционную матрицу по всему набору данных:

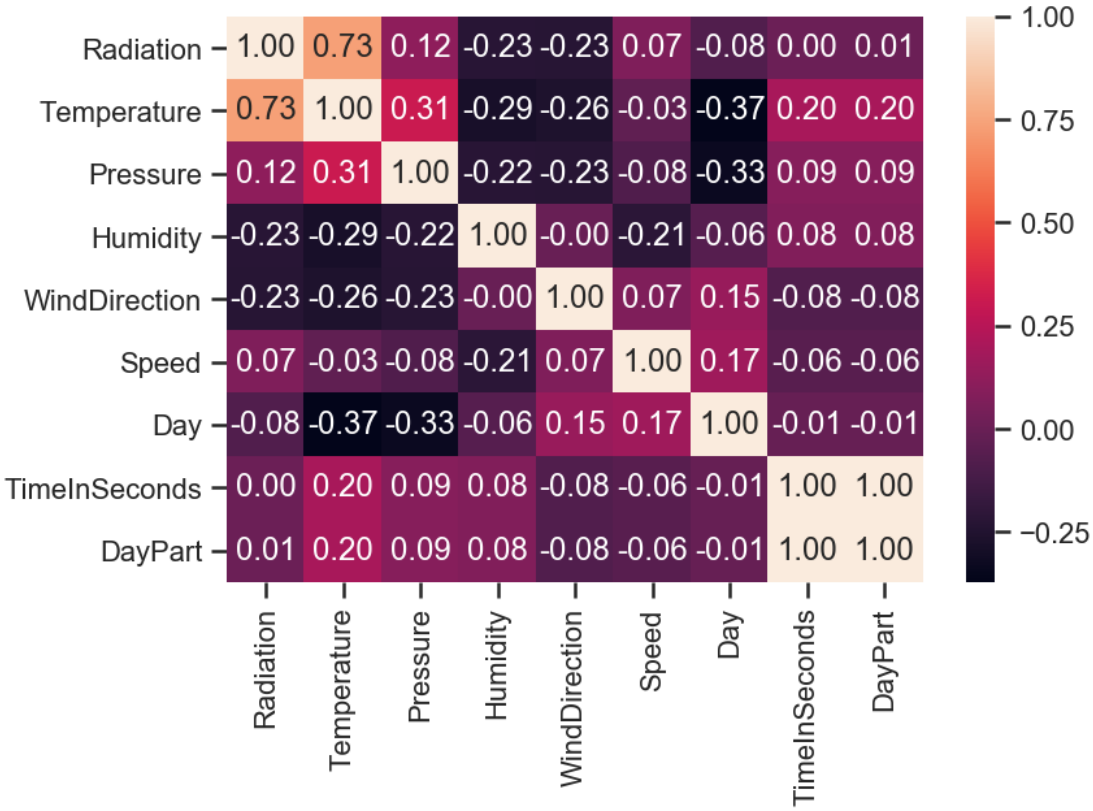
```
In [18]: df.corr()
```

```
Out[18]:                 Radiation  Temperature  Pressure  Humidity  \
         Radiation        1.000000     0.734955  0.119016 -0.226171
         Temperature      0.734955     1.000000  0.311173 -0.285055
         Pressure         0.119016     0.311173  1.000000 -0.223973
         Humidity        -0.226171    -0.285055 -0.223973  1.000000
         WindDirection   -0.230324    -0.259421 -0.229010 -0.001833
```

```
Speed           0.073627    -0.031458 -0.083639 -0.211624
Day            -0.081320    -0.370794 -0.332762 -0.063760
TimeInSeconds   0.004348     0.197227  0.091066  0.077851
DayPart         0.005980     0.198520  0.094403  0.075513

                WindDirection     Speed       Day  TimeInSeconds  \
Radiation           -0.230324  0.073627 -0.081320       0.004348
Temperature         -0.259421 -0.031458 -0.370794       0.197227
Pressure            -0.229010 -0.083639 -0.332762       0.091066
Humidity            -0.001833 -0.211624 -0.063760       0.077851
WindDirection        1.000000  0.073092  0.153255      -0.077956
Speed                0.073092  1.000000  0.174336      -0.057908
Day                  0.153255  0.174336  1.000000      -0.007094
TimeInSeconds       -0.077956 -0.057908 -0.007094       1.000000
DayPart             -0.078130 -0.056095 -0.010052       0.998980

                 DayPart
Radiation       0.005980
Temperature     0.198520
Pressure        0.094403
Humidity        0.075513
WindDirection  -0.078130
Speed          -0.056095
Day            -0.010052
TimeInSeconds   0.998980
DayPart         1.000000
```

Визуализируем корреляционную матрицу с помощью тепловой карты:

```
In [19]: sns.heatmap(df.corr(), annot=True, fmt=".2f");
```



Видно, что мощность солнечного излучения заметно коррелирует с температурой, что было показано выше с помощью парного графика. Также заметно коррелируют время дня и проекция этого времени на промежуток светового дня. При этом последняя характеристика слегка больше коррелирует с целевым признаком, так что, возможно, следует использовать именно эту характеристику.

4 Список использованной литературы

1. Гапанюк Ю.Е. Лабораторная работа «Разведочный анализ данных. Исследование и визуализация данных» [Электронный ресурс] // ugapanyuk/ml_course Wiki // GitHub. 2019. URL: https://github.com/ugapanyuk/ml_course/wiki/LAB_EDA_VISUALIZATION (дата обращения: 13.02.2019).
2. dronio. Solar Radiation Prediction [Electronic resource] // Kaggle. 2017. URL: https://www.kaggle.com/dronio/SolarEnergy (дата обращения: 18.02.2019).
3. The IPython Development Team. IPython 7.3.0 Documentation [Electronic resource] // Read the Docs. 2019. URL: https://ipython.readthedocs.io/en/stable/ (дата обращения: 20.02.2019).
4. Waskom M. seaborn 0.9.0 documentation [Electronic resource]. 2018. URL: https://seaborn.pydata.org/ (дата обращения: 20.02.2019).
5. Chrétien M. Convert datetime.time to seconds [Electronic resource] // Stack Overflow. 2017. URL: https://stackoverflow.com/a/44823381 (дата обращения: 20.02.2019).