Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Методы машинного обучения»

Отчет по лабораторной работе №1
«Создание "истории о данных" (Data Storytelling)»

Выполнила:
студент группы ИУ5-22М
Павловская А.А.
14.02.2023

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.Е.

Москва, 2023 г.

Цель работы: изучение различных методов визуализация данных и создание истории на основе данных.

Задание:

- Выбрать набор данных (датасет).

Для лабораторных работ не рекомендуется выбирать датасеты очень большого размера.

- Создать "историю о данных" в виде юпитер-ноутбука, с учетом следующих требований:

1. История должна содержать не менее 5 шагов (где 5 - рекомендуемое количество шагов). Каждый шаг содержит график и его текстовую интерпретацию.
2. На каждом шаге наряду с удачным итоговым графиком рекомендуется в юпитер-ноутбуке оставлять результаты предварительных "неудачных" графиков.
3. Не рекомендуется повторять виды графиков, желательно создать 5 графиков различных видов.
4. Выбор графиков должен быть обоснован использованием методологии data-to-viz. Рекомендуется учитывать типичные ошибки построения выбранного вида графика по методологии data-to-viz. Если методология Вами отвергается, то просьба обосновать Ваше решение по выбору графика.
5. История должна содержать итоговые выводы. В реальных "историях о данных" именно эти выводы представляют собой основную ценность для предприятия.

- Сформировать отчет и разместить его в своем репозитории на github.

Ход работы

Для выполнения лабораторной работы использовался Jupiter notebook.

Ссылка на репозиторий github: https://github.com/PavlAA79/MMO_IU5-22M.git

ММО Лаб1 Создание истории о данных (Data Storytelling)

Павловская А.А. ИУ5-22М

Ввод [39]:
```
import seaborn as sns
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
import matplotlib.pyplot as plt
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import MinMaxScaler, StandardScaler, Normalizer
from sklearn.model_selection import train_test_split
```

Датасет London Weather Data https://www.kaggle.com/datasets/emmanuelfwerr/london-weather-data (https://www.kaggle.com/datasets/emmanuelfwerr/london-weather-data)

1. date - записанная дата измерения - (int)
2. cloud_cover - измерение облачного покрова в октантах - (float)
3. sunshine - измерение солнечного света в часах (hrs) - (float)
4. global_radiation - измерение интенсивности излучения в ваттах на квадратный метр (W/m2) - (float)
5. max_temp - максимальная зарегистрированная температура в градусах Цельсия (°C) - (float)
6. mean_temp - средняя температура в градусах Цельсия (°C) - (float)
7. min_temp – минимальная зарегистрированная температура в градусах Цельсия (°C) - (float)
8. precipitation - измерение осадков в миллиметрах (mm) - (float)
9. pressure - измерение давления в паскалях (Pa) - (float)
10. snow_depth - измерение глубины снега в сантиметрах (cm) - (float)

Ввод [40]:
```
#Загрузка данных
data = pd.read_csv(r"D:\Py\MAD\london_weather.csv")
```

Ввод [41]:
```
data.head()
```

Out[41]:

	date	cloud_cover	sunshine	global_radiation	max_temp	mean_temp	min_temp	precipitation	pressure	snow_depth
0	19790101	2.0	7.0	52.0	2.3	-4.1	-7.5	0.4	101900.0	9.0
1	19790102	6.0	1.7	27.0	1.6	-2.6	-7.5	0.0	102530.0	8.0
2	19790103	5.0	0.0	13.0	1.3	-2.8	-7.2	0.0	102050.0	4.0
3	19790104	8.0	0.0	13.0	-0.3	-2.6	-6.5	0.0	100840.0	2.0
4	19790105	6.0	2.0	29.0	5.6	-0.8	-1.4	0.0	102250.0	1.0

Ввод [42]:
```
data.columns
```

Out[42]:
```
Index(['date', 'cloud_cover', 'sunshine', 'global_radiation', 'max_temp',
       'mean_temp', 'min_temp', 'precipitation', 'pressure', 'snow_depth'],
      dtype='object')
```

Ввод [43]:
```
#Удаление колонки date
data = data.drop(columns=['date'])
```

Ввод [44]:
```
data.shape
```

Out[44]:
```
(15341, 9)
```

Ввод [45]:
```
#Удаление дубликатов
data = data.drop_duplicates()
```

Ввод [46]:
```
data.shape
```

Out[46]:
```
(15341, 9)
```

Ввод [47]:
```
# Проверка наличия 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))
```

```
cloud_cover - 19
sunshine - 0
global_radiation - 19
max_temp - 6
mean_temp - 36
min_temp - 2
precipitation - 6
pressure - 4
snow_depth - 1441
```

Ввод [48]:
```
#Заполнение пропусков нулями в колонках snow_depth,precipitation, cloud_cover
data.snow_depth.fillna(0, inplace=True)
```

Ввод [49]:
```
data.precipitation.fillna(0, inplace=True)
```

Ввод [50]:
```
data.cloud_cover.fillna(0, inplace=True)
```

Ввод [51]:
```
# Проверка наличия 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))
```

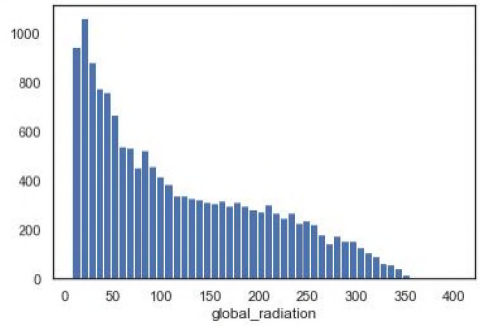
```
cloud_cover - 0
sunshine - 0
global_radiation - 19
max_temp - 6
mean_temp - 36
min_temp - 2
precipitation - 0
pressure - 4
snow_depth - 0
```

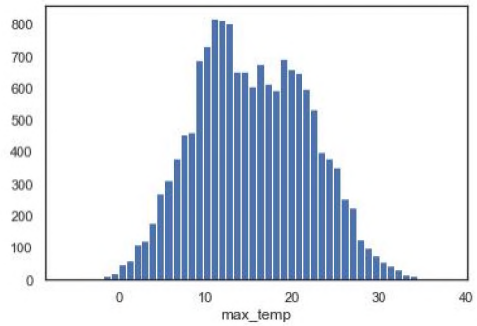
Ввод [52]:
```
# Выбор числовых колонок с пропущенными значениями
# Цикл по колонкам датасета
num_cols = []
for col in data.columns:
    # Количество пустых значений
    temp_null_count = data[data[col].isnull()].shape[0]
    dt = str(data[col].dtype)
    if temp_null_count>0 and (dt=='float64' or dt=='int64'):
        num_cols.append(col)
        temp_perc = round((temp_null_count / data.shape[0]) * 100.0, 2)
        print('Колонка {}. Тип данных {}. Количество пустых значений {}, {}%.'.format(col, dt, temp_null_count, temp_perc))
```

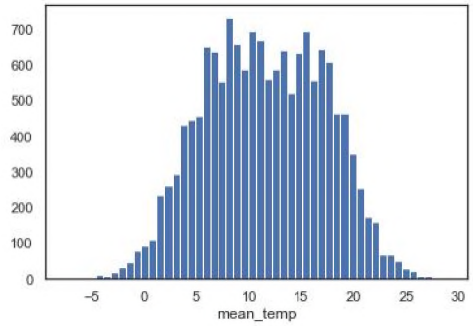
```
Колонка global_radiation. Тип данных float64. Количество пустых значений 19, 0.12%.
Колонка max_temp. Тип данных float64. Количество пустых значений 6, 0.04%.
Колонка mean_temp. Тип данных float64. Количество пустых значений 36, 0.23%.
Колонка min_temp. Тип данных float64. Количество пустых значений 2, 0.01%.
Колонка pressure. Тип данных float64. Количество пустых значений 4, 0.03%.
```

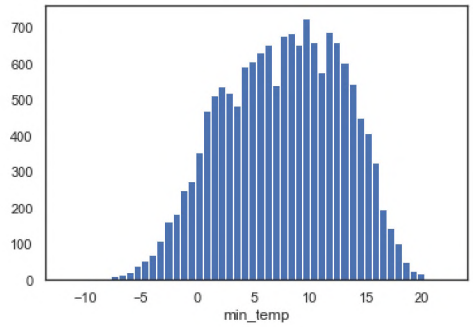
Ввод [53]:
```
# Фильтр по колонкам с пропущенными значениями
data_num = data[num_cols]
```

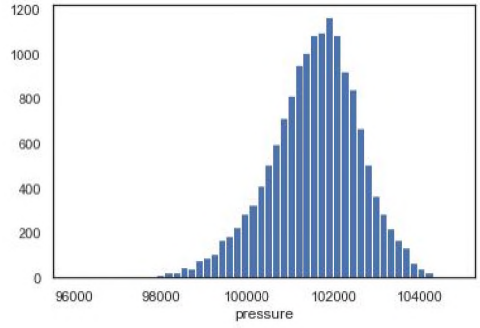
Ввод [54]:
```
# Гистограмма по признакам
for col in data_num:
    plt.hist(data[col], 50)
    plt.xlabel(col)
    plt.show()
```











Ввод [55]:
```
# Обработка пропусков
imp_num = SimpleImputer(strategy='median')
imp_num2 = SimpleImputer(strategy='most_frequent')
data[['pressure']] = imp_num2.fit_transform(data[['pressure']])
data[['min_temp']] = imp_num2.fit_transform(data[['min_temp']])
data[['mean_temp']] = imp_num2.fit_transform(data[['mean_temp']])
data[['max_temp']] = imp_num2.fit_transform(data[['max_temp']])
data[['global_radiation']] = imp_num.fit_transform(data[['global_radiation']])
```

Ввод [56]:
```
# Проверка наличия 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))
```

```
cloud_cover - 0
sunshine - 0
global_radiation - 0
max_temp - 0
mean_temp - 0
min_temp - 0
precipitation - 0
pressure - 0
snow_depth - 0
```

Ввод [57]:
```
# Числовые колонки для масштабирования
scale_cols = ['cloud_cover','sunshine','global_radiation','max_temp','mean_temp','min_temp','precipitation',
'pressure','snow_depth']
```

Ввод [58]:
```
# Масштабирование данных
sc1 = MinMaxScaler()
sc1_data = sc1.fit_transform(data[scale_cols])
data_scaled = pd.DataFrame()
```

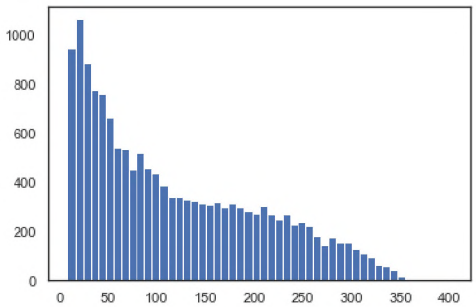
Ввод [59]:
```
# Добавим масштабированные данные в набор данных
for i in range(len(scale_cols)):
    col = scale_cols[i]
    new_col_name = col
    data_scaled[new_col_name] = sc1_data[:,i]
```

Ввод [60]:
```
data_scaled.head()
```

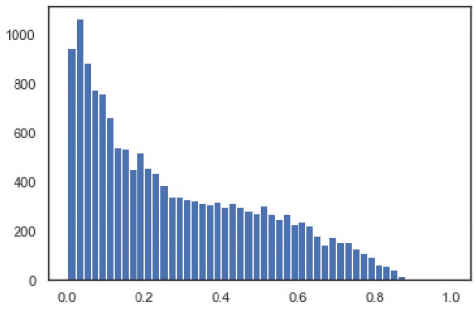
Out[60]:

	cloud_cover	sunshine	global_radiation	max_temp	mean_temp	min_temp	precipitation	pressure	snow_depth
0	0.222222	0.43750	0.111675	0.192744	0.095628	0.126100	0.006472	0.670429	0.409091
1	0.666667	0.10625	0.048223	0.176871	0.136612	0.126100	0.000000	0.741535	0.363636
2	0.555556	0.00000	0.012690	0.170068	0.131148	0.134897	0.000000	0.687359	0.181818
3	0.888889	0.00000	0.012690	0.133787	0.136612	0.155425	0.000000	0.550790	0.090909
4	0.666667	0.12500	0.053299	0.267574	0.185792	0.304985	0.000000	0.709932	0.045455

Ввод [61]:
```
plt.hist(data['global_radiation'], 50)
plt.show()
```



Ввод [62]:
```
plt.hist(data_scaled['global_radiation'], 50)
plt.show()
```

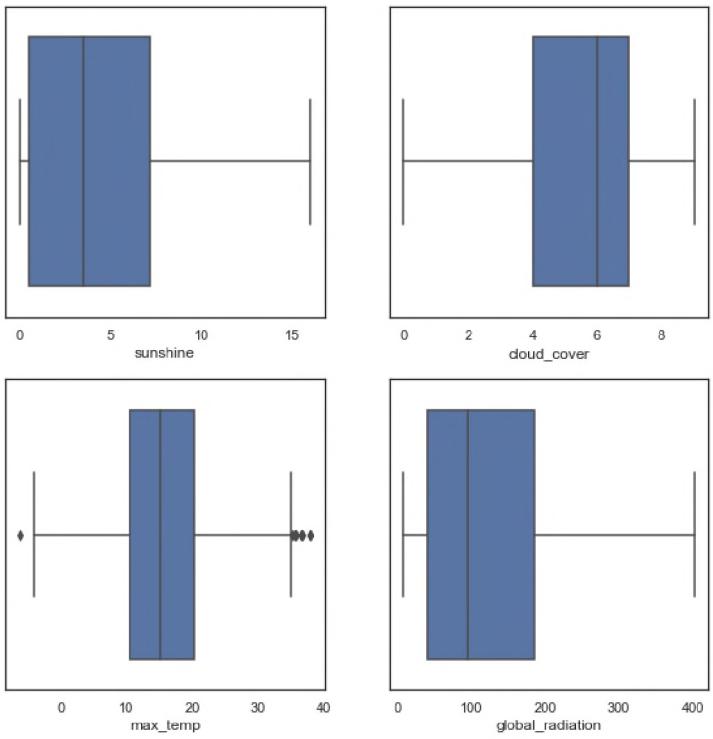


Ящик с усами

Отображает одномерное распределение вероятности

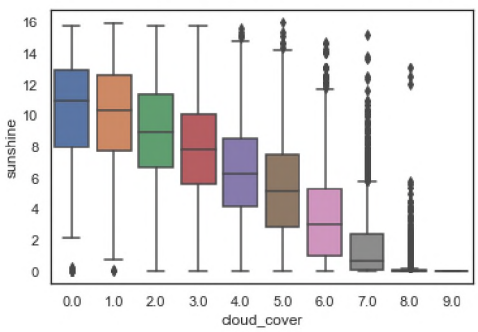
Ввод [63]:
```
fig, ax = plt.subplots(2, 2, figsize=(10,10))
sns.boxplot(ax = ax[0,0],x=data['sunshine'])
sns.boxplot(ax = ax[0,1],x=data['cloud_cover'])
sns.boxplot(ax = ax[1,0],x=data['max_temp'])
sns.boxplot(ax = ax[1,1],x=data['global_radiation'])
```

Out[63]: <AxesSubplot:xlabel='global_radiation'>



Ввод [64]:
```
# Распределения параметра sunshine сгруппированные по cloud_cover
sns.boxplot(x='cloud_cover', y='sunshine', data=data)
```

Out[64]: <AxesSubplot:xlabel='cloud_cover', ylabel='sunshine'>

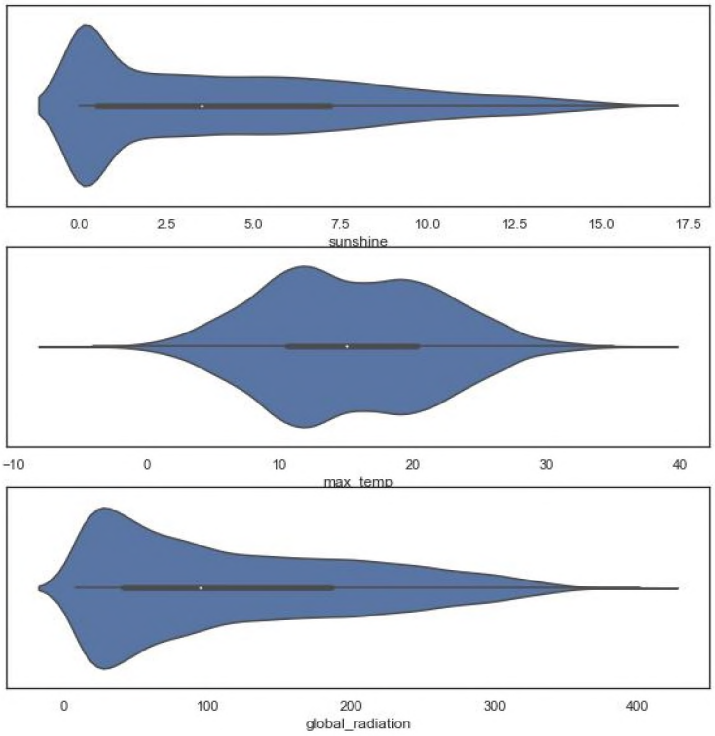


Violin plot

Распределение вероятности и распределение плотности

Ввод [65]:
```
fig, ax = plt.subplots(3, 1, figsize=(10,10))
sns.violinplot(ax = ax[0],x=data['sunshine'])
sns.violinplot(ax = ax[1],x=data['max_temp'])
sns.violinplot(ax = ax[2],x=data['global_radiation'])
```

Out[65]: <AxesSubplot:xlabel='global_radiation'>



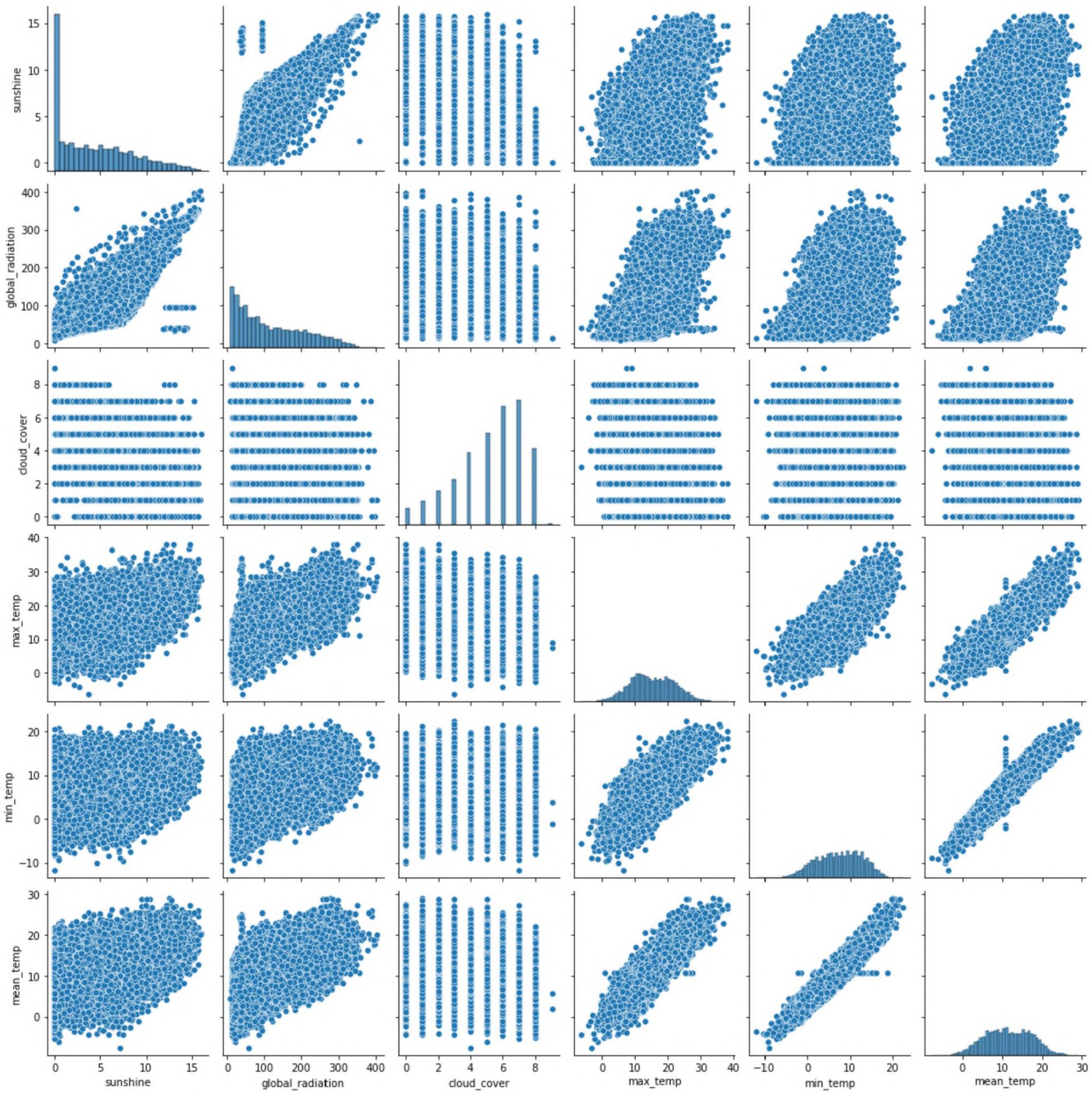
Correlogram

Комбинация гистограмм и диаграмм рассеивания для всего набора данных

Ввод [66]:
```
d = data[['sunshine','global_radiation','cloud_cover','max_temp','min_temp','mean_temp']]
```

Ввод [58]:
```
sns.pairplot(d)
```

Out[58]: <seaborn.axisgrid.PairGrid at 0x28db4bfbac0>

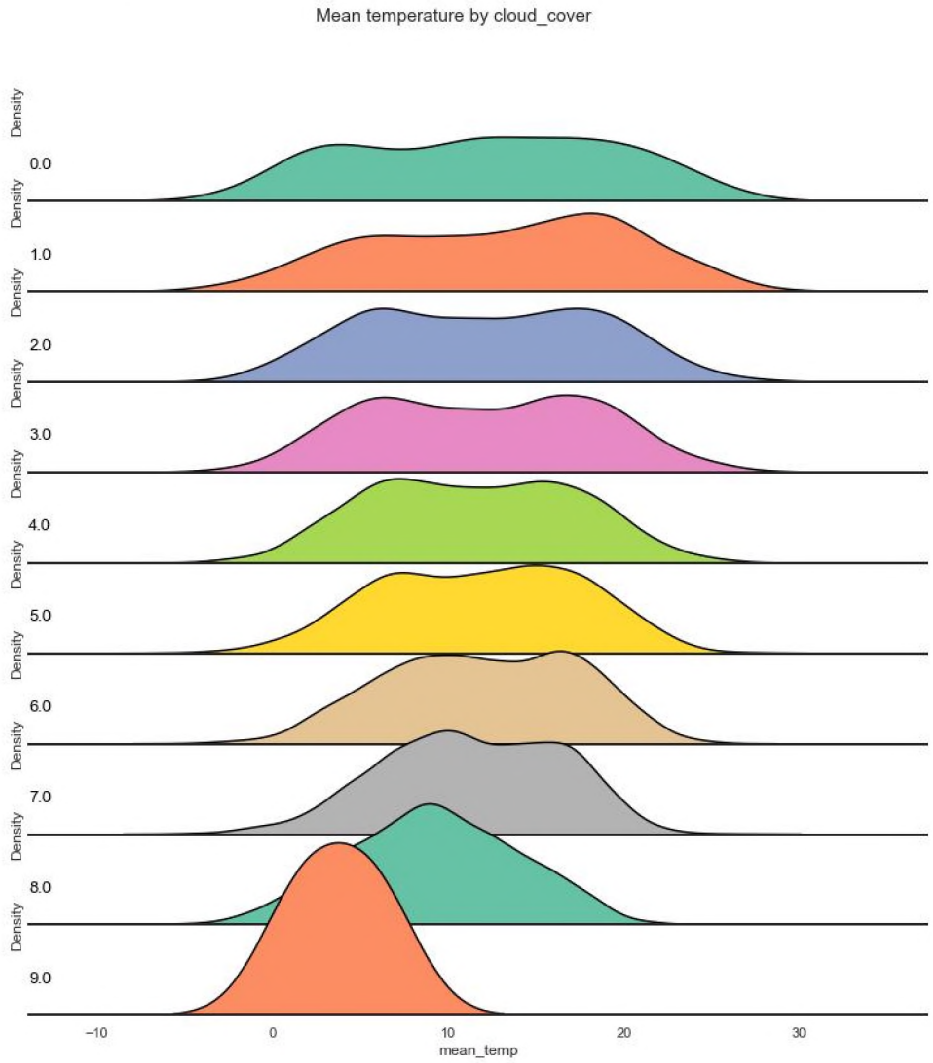


Ridgeline

Распределение количественной величины по нескольким группам

Ввод [89]:
```
# Тема
sns.set_theme(style="white", rc={"axes.facecolor": (0, 0, 0, 0), 'axes.linewidth':2})
palette = sns.color_palette("Set2", 12)


g = sns.FacetGrid(data, palette=palette, row="cloud_cover", hue="cloud_cover", aspect=9, height=1.2)


g.map_dataframe(sns.kdeplot, x="mean_temp", fill=True, alpha=1)
g.map_dataframe(sns.kdeplot, x="mean_temp", color='black')

def label(x, color, label):
    ax = plt.gca()
    ax.text(0, .2, label, color='black', fontsize=13,
            ha="left", va="center", transform=ax.transAxes)
g.map(label, "mean_temp")

g.fig.subplots_adjust(hspace=-.5)

g.set_titles("")

g.set(yticks=[], xlabel="mean_temp")

g.despine(left=True)

plt.suptitle('Mean temperature by cloud_cover', y=0.98)
```

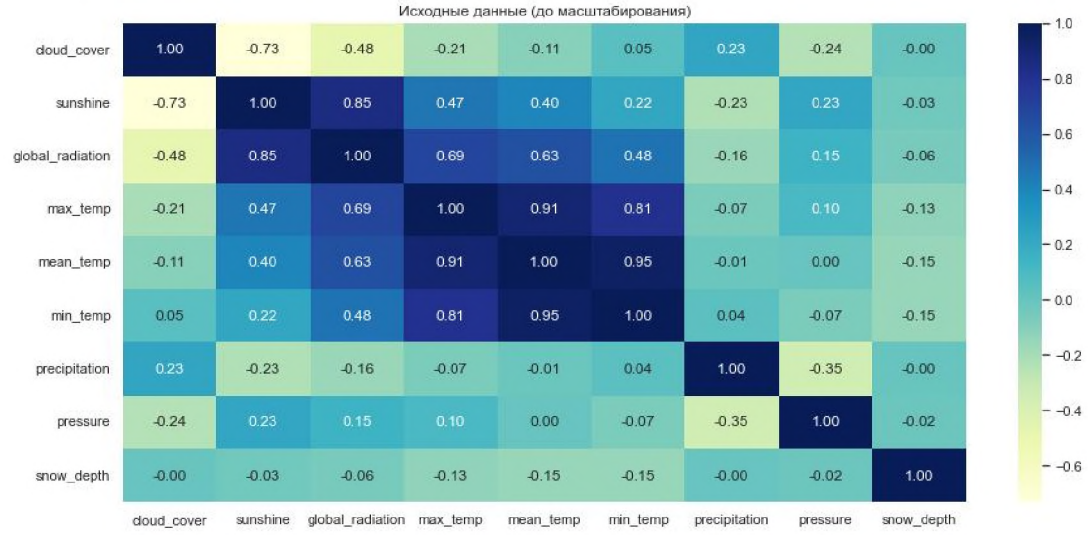
Out[89]: Text(0.5, 0.98, 'Mean temperature by cloud_cover')



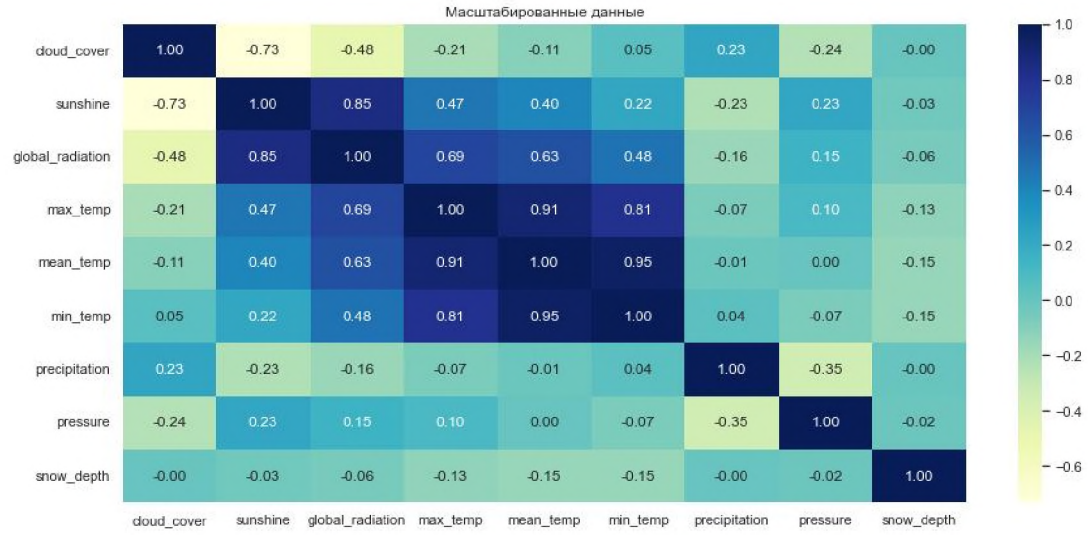
Heatmap

Матрица парных корреляций

Ввод [90]:
```
fig, ax = plt.subplots(figsize=(15,7))
sns.heatmap(data.corr(),cmap='YlGnBu', annot=True, fmt='.2f')
ax.set_title('Исходные данные (до масштабирования)')
plt.show()
```



Ввод [91]:
```
fig, ax = plt.subplots(figsize=(15,7))
sns.heatmap(data_scaled.corr(),cmap='YlGnBu', annot=True, fmt='.2f')
ax.set_title('Масштабированные данные')
plt.show()
```



• Параметр sunshine имеет довольно сильную отрицательную корреляцию с параметром cloud_cover (-0,73). Чем сильнее небо покрыто облаками, тем меньше часов прямого солнечного света в сутки.

• Параметр global_radiation сильно коррелирует с парамметром sunshine (0,85) и чуть слабее с параметром max_temp (0,69).

• Параметр max_temp сильно коррелирует с параметром mean_temp (0,91) и параметром min_temp (0,81). Это можно объяснить тем что все эти параметры содержат данные о суточной температуре, а параметр mean_temp является вычисленным средним между max_temp и min_temp.

Для предсказания целевого признака sunshine можно оставить признаки cloud_cover, global_radiation и max_temp.

Ввод [92]:
```
data_new = data_scaled.drop(columns=['mean_temp','min_temp','precipitation','pressure','snow_depth'])
```

Ввод [93]:
```
data_new.head()
```

Out[93]:

	cloud_cover	sunshine	global_radiation	max_temp
0	0.222222	0.43750	0.111675	0.192744
1	0.666667	0.10625	0.048223	0.176871
2	0.555556	0.00000	0.012690	0.170068
3	0.888889	0.00000	0.012690	0.133787
4	0.666667	0.12500	0.053299	0.267574

Вывод: в ходе выполнения лабораторной работы мы изучили различные методы визуализация данных и создали историю на основе данных.