ЛР 3. Работа с табличными данными на Python

1. Цель работы

Научиться основам работы с табличными данными для их чтения из открытых источников, статистического анализа и визуализации результатов.

Работу рекомендуется выполнять в бригаде из двух человек.

В примерах ниже рассматриваются следующие вопросы:

- основы работы с табличными данными - эти навыки нужны, чтобы делать меньше рутинных ручных операций в задачах анализа данных и поиска информации в больших объемах данных;
- вычисления с табличными данными - иногда быстрее написать несколько строк кода, чем считать на калькуляторе или использовать программы для работы с таблицами (Excel, LibreOffice Calc);
- визуализация табличных данных - для быстрого создания наглядных отчетов;
- получение данных из открытых источников - чтобы не тратить время на копирование вручную и внесение тоже вручную мелких поправок для приведения данных к нужному виду.

2. Работа с табличными данными в Pandas

Пример доступен по ссылке https://colab.research.google.com/drive/13eeLL0qaUzZCCQdB-qCca8oeAP7G14mS.

2.1. Создание таблицы, добавление строк, столбцов

```
# 1. Подключение Pandas и NumPy
import pandas as pd
import numpy as np


# 2. Пример создания таблицы (DataFrame) с помощью кода
df_init = pd.DataFrame({
    'имя': ['ТЭЦ-4', 'ТЭЦ-5', 'ГЭС'],
    'мощность': [384, 1200, 480],
    'компания': ['СГК', 'СГК', 'РусГидро']
})

df_init.head()


# 3. Добавление строк
```

```
# Для добавления строк можно использовать функцию append. По сути здесь создается еще одна таблица тем же способом, что df_init. Затем добавляется к df_init
df_exted = df_init.append(pd.DataFrame({'имя': ['ТЭЦ-2', 'ТЭЦ-3'], 'мощность': [340, 511.5], 'компания': ['СГК', 'СГК']}), ignore_index = True)

# первые строки таблицы можно вывести на экран в более простом виде, применив функцию print.
print(df_exted)
print()

# Получение списка имен столбцов
print('имена столбцов:', df_exted.columns.values)


# 4. Добавление столбца.
# Например, добавим каждой станции условное обозначение - идентификатор
df_exted.insert(0, 'id', ['t4', 't5', 'g', 't2', 't3'])

print(df_exted)
print()

# Установка id в качестве индекса строки
df_exted = df_exted.set_index('id')

print(df_exted)
```

В результате должна получиться таблица, показанная на Рисунке 1. Все данные в ней прописаны с помощью кода выше.

id	имя	мощность	компания
t4	ТЭЦ-4	384.0	СГК
t5	ТЭЦ-5	1200.0	СГК
g	ГЭС	480.0	РусГидро
t2	ТЭЦ-2	340.0	СГК
t3	ТЭЦ-3	511.5	СГК

Рисунок 1. Вид таблицы исходных данных по электростанциями

2.2. Обработка данных: сортировка, вычисления

Выполнение пятой ячейки дает две таблицы. Первая отсортирована по именам электростанций, вторая - по мощностям.

Выполнение шестой ячейки дает таблицу с новыми столбцами “выработка_за_сутки”, “тип” и “проверка_расчетов”, данные в которых получены путем вычислений над исходными данными.

```
# 5. Сортировка
df_exted = df_exted.sort_values(['имя'])
print(df_exted)
print()

# ascending задает сортировку по убыванию или возрастанию
# по умолчанию True - сортировка по убыванию, если поставить False, то будет по возрастанию
df_exted = df_exted.sort_values(['мощность'], ascending = False)
print(df_exted)


# 6. Создание столбцов на основе вычислений.
# Чтобы случайно не испортить df_exted, будем работать с копией
df_exted_2 = df_exted.copy()

# Добавление столбца на основе формулы делается легко
df_exted_2['выработка_за_сутки'] = df_exted_2['мощность'] * 24

# Сложности могут возникнуть когда нужно реализовать условия и работать с текстовыми данными.
# Например, нужно по названию определить тип электростанции.
# Условия можно задавать через NumPy функцию where, а поиск подстроки в строке через find.
# Так можно одной строкой задать целый алгоритм:
# если в имени есть ГЭС:
#   тип - ГЭС;
# иначе
#   если в имени есть ТЭЦ:
#     тип - ТЭЦ
#   иначе
#     тип - не определен

df_exted_2['тип'] = np.where(df_exted_2['имя'].str.find('ГЭС') >= 0, 'ГЭС', np.where(df_exted_2['имя'].str.find('ТЭЦ') >= 0, 'ТЭЦ', '?'))

print(df_exted_2)
print()

# Легко применить расчеты, используя несколько столбцов сразу
df_exted_2['проверка_расчетов'] =  df_exted_2['выработка_за_сутки'] / df_exted_2['мощность'] == 24
print(df_exted_2)
```

2.3. Выборка данных по индексам и условиям

Приведенный здесь код показывает различными примеры отбора данных. Рекомендуется поэкспериментировать с данными кодом, чтобы лучше понять, как он работает.

```
# 7. Выборка нужных фрагментов. Есть три основных способа выделения фрагментов.

# 7.1. Через имя столбца или столбцов
print('все данные из столбцов \'имя\' и \'мощность\'')
print(df_exted_2[['имя', 'мощность']])
print()
print('все данные из столбца \'имя\'')
print(df_exted_2.имя)
print()

# 7.2. Через loc по именам строк и столбцов
print('данные из строк \'g\' и \'t5\' и столбцов \'имя\' и \'мощность\'')
print(df_exted_2.loc[['g', 't5'], ['имя', 'мощность']])
print()
print('все данные из строки \'t2\'')
print(df_exted_2.loc[['t2']])
print()

# 7.3. Через iloc по номерам строк и столбцов, номера идут с нуля
print('данные из первой строки и второго столбца')
print(df_exted_2.iloc[1 , 2])
print()
print('данные из всех строк кроме двух последних и из столбцов со второго по
четвертый включительно')
print(df_exted_2.iloc[: -2, 2 : 5])
print()
print('данные из нечетных строк и третьего столбца')
print(df_exted_2.iloc[ : : 2, 3])
print()


# 8. Удаление столбцов.
df_exted = df_exted_2.drop(['проверка_расчетов'], axis='columns')

print(df_exted)


# 9. Выборка данных по условию.
# Выбрать электростанции с мощностью выше 400 МВт
print(df_exted[df_exted['мощность'] > 400.])
print()
```

```
# Можно писать короче, если имя столбца не содержит пробела
print(df_exted[df_exted.мощность > 400.])
print()

# Выбрать электростанции с мощностью выше 400 МВт и не от СГК
print(df_exted[(df_exted.мощность > 400.) & (df_exted.компания != 'СГК')])
print()

# Выбрать электростанции с мощностью выше 1000 МВт или относящиеся к ГЭС
print(df_exted[(df_exted.мощность > 1000.) | (df_exted.тип == 'ГЭС')])
```

2.4. Группировка данных, статистика

```
# 10. Статистические расчеты по сгруппированным данным.
# Суммарная мощность по типа электростанций.
print(df_exted.groupby(['тип'])['мощность'].sum())
print()

# Максимальная мощность по типам электростанций.
print(df_exted.groupby(['тип'])['мощность'].max())
print()

# Число электростанций по мощности ниже и выше 500 МВт и разделением по
компаниям
print(df_exted.groupby(['компания', df_exted['мощность'] >
500])['мощность'].count())
print()


# 11. Быстрая визуализация данных
# В предыдущей работе для визуализации данных применялись отдельные библиотеки.
# Но для быстрого получения графиков и гистограмм можно использовать очень
простой код.
df_exted.plot(kind = 'bar')
print()


# 12. Выбор столбцов для визуализации
df_exted['мощность'].plot(kind = 'bar')
print()


# 13. Гистограмма после группировки
df_exted.groupby(['тип'])['мощность'].sum().plot(kind = 'barh')
print()
```

```
# 14. Удаление строк по условию
df_del = df_exted.drop(df_exted[df_exted['мощность'] < 400].index)
print(df_del)
```

В результате применения группировки должны получиться следующие результаты. Суммарная мощность по типам электростанций в МВт:

```
тип
ГЭС     480.0
ТЭЦ    2435.5
```

Максимальная мощность по типам электростанций:

```
тип
ГЭС     480.0
ТЭЦ    1200.0
```

Определение числа электростанций по компаниям, мощность которых выше 500 МВт:

```
компания  мощность
РусГидро  False       1
СГК       False       2
          True        2
```

Результат показывает, что у РусГидро одна электростанция мощностью ниже 500 МВт (False 1), а у СГК в выборке две электростанции ниже 500 МВт (False 2) и две более мощные (True1).

Визуализация должна дать результат, показанный на Рисунке 2. В верхней части приведена гистограмма заданных в таблице численных значений по идентификаторам электростанций. Сразу видно, что выработка за сутки t5 (ТЭЦ-5) намного выше всех остальных представленных станций.

В середине Рисунка 2 та же по сути гистограмма, но взяты только данные по мощности (без данных по выработке за сутки).

В нижней части Рисунка 3 показана гистограмма, отражающая суммарную мощность по типам электростанций. Хорошо видно, насколько для города важны ТЭЦ, но при этом вклад ГЭС в электроснабжение Новосибирска тоже существенный.

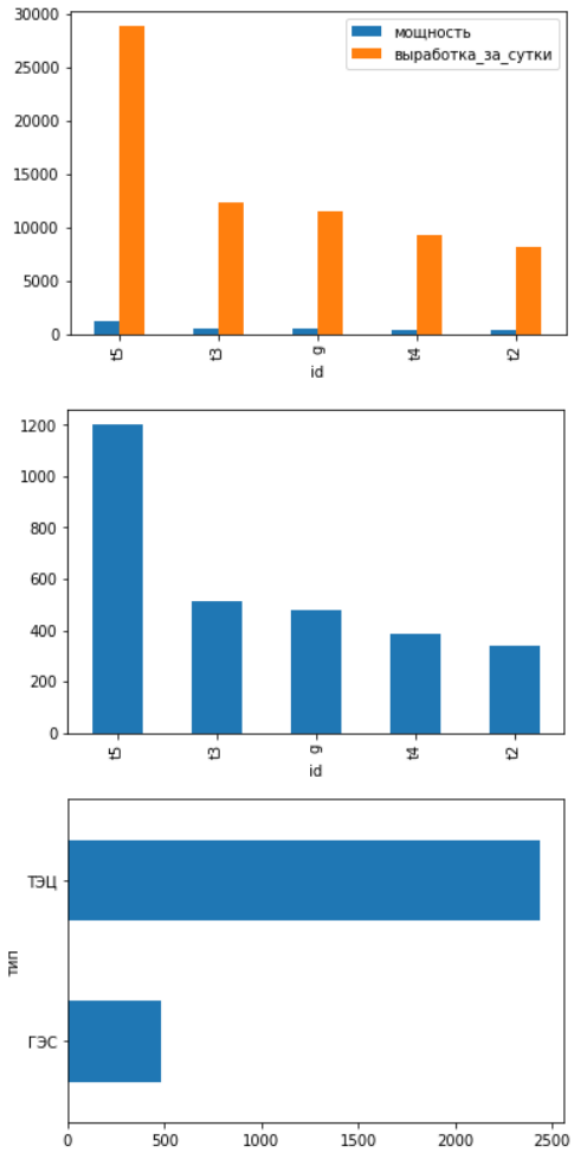

Рисунок 2. Простая визуализация в Pandas

3. Чтение табличных данных с Web страниц, обработка и визуализация результатов

В предыдущей работе данные читались с файлов из облачных хранилищ. Но далеко не всегда можно получить готовый к работе файл. Часто нужные данные можно найти на страницах различных сайтов. В этом случае требуется выполнить отделение таблицы от прочего содержимого Web-страницы. Можно сделать это вручную, но это не всегда удобно и часто слишком медленно.

Автоматическое чтение данных с Web-страниц называется веб-скрапингом или веб-скрейпингом (Web-scraping). Это непростая задача, но используемая библиотека Pandas содержит готовые функции, решающие ее.

Примеры ниже доступны по ссылке https://colab.research.google.com/drive/14AA-HGmpdhBEMhdhrjgyM-HP-aIACV68.

Цифровой мониторинг становится все более доступным и прозрачным. На сайте Системного оператора ЕЭС можно найти большое количество данных (https://so-ups.ru/). Но вручную их было бы не очень удобно скачивать, так как пришлось бы скачивать отдельный файл для каждых суток.

Чтобы определить адреса таблиц, на сайте можно открыть, например, генерацию и потребление ОЭС Сибири (Рисунок 1). В адресной строке браузера можно увидеть адрес https://so-ups.ru/index.php?id=972&tx_ms1cdu_pi1%5Bkpo%5D=610000&tx_ms1cdu_pi1%5Bdt%5D=12.02.2020. Легко понять, что окончание - это дата.

Рисунок 1. Страница сайта СО ЕЭС

Сделаем чтение разных данных за три дня. Фрагмент полученной таблицы показан на Рисунке 2.

```
# 2. Получение данных с сайта Системного оператора ЕЭС
# Нужно сначала зайти на сайт, чтобы посмотреть, какие данные он предоставляет, потом использовать полученный адрес и подобрать номер таблицы.
url = "https://so-ups.ru/index.php?id=972&tx_ms1cdu_pi1%5Bkpo%5D=610000&tx_ms1cdu_pi1%5Bdt%5D=12.02.2020"
id_table = 0
df_power = pd.read_html(url, header = 0, index_col = 0)[id_table]
```

```
df_power

# 3. Сайт выдает почему-то данные не за 24 часа, а за 25, то есть сутки и еще
один час.
# Поэтому последнюю строку нужно удалять.
# Легко взять данные из несколько дней и "склеить" их в одну таблицу
df_power = (pd.read_html("https://so-
ups.ru/index.php?id=972&tx_ms1cdu_pi1%5Bkpo%5D=610000&tx_ms1cdu_pi1%5Bdt%5D=12.
02.2020", header = 0, index_col = 0)[id_table]).iloc[: - 1]
df_2 = (pd.read_html("https://so-
ups.ru/index.php?id=972&tx_ms1cdu_pi1%5Bkpo%5D=610000&tx_ms1cdu_pi1%5Bdt%5D=13.
02.2020", header = 0, index_col = 0)[id_table]).iloc[: - 1]
df_3 = (pd.read_html("https://so-
ups.ru/index.php?id=972&tx_ms1cdu_pi1%5Bkpo%5D=610000&tx_ms1cdu_pi1%5Bdt%5D=14.
02.2020", header = 0, index_col = 0)[id_table]).iloc[: - 1]

df_power = df_power.append(df_2)
df_power = df_power.append(df_3)

# Определим разность генерации и потребления
df_power['Разность (МВт)'] = df_power.iloc[: , 0] - df_power.iloc[:, 1]
df_power

# 4. Аналогично добавим к данным частоту
id_table = 0
df_freq = (pd.read_html("https://so-
ups.ru/index.php?id=971&tx_ms1cdu_pi1%5Bkpo%5D=610000&tx_ms1cdu_pi1%5Bdt%5D=12.
02.2020", header = 0, index_col = 0)[id_table]).iloc[:-1]
df_freq = (df_freq.append(pd.read_html("https://so-
ups.ru/index.php?id=971&tx_ms1cdu_pi1%5Bkpo%5D=610000&tx_ms1cdu_pi1%5Bdt%5D=13.
02.2020", header = 0, index_col = 0)[id_table])).iloc[:-1]
df_freq = (df_freq.append(pd.read_html("https://so-
ups.ru/index.php?id=971&tx_ms1cdu_pi1%5Bkpo%5D=610000&tx_ms1cdu_pi1%5Bdt%5D=14.
02.2020", header = 0, index_col = 0)[id_table])).iloc[:-1]

df_freq

# 5. Осталось совместить мощность и частоту
df_power_freq = df_power.join(df_freq)
df_power_freq
```

Время Мск	Мощность генерации (МВт)	Мощность потребления (МВт)	Разность (МВт)
12-02-2020 00:00	26345	25628	717
12-02-2020 01:00	26103	25795	308
12-02-2020 02:00	26973	26456	517
12-02-2020 03:00	27278	27334	-56
12-02-2020 04:00	27977	28065	-88

Рисунок 2. Фрагмент таблицы с генерацией, потреблением и частотой

```
# 6. Построение графиков
import matplotlib.pyplot as plt

axes = df_power_freq.plot(marker='.', figsize=(20, 10), subplots = True)
for _ in axes:
  _.grid(True)
axes = df_power_freq[df_power_freq.columns[: -2]].plot(marker='.', figsize=(20,
5), subplots = False)
axes.grid(True)

# 7. Гистограмма
axes = df_power_freq.iloc[ : 24, 1].plot(kind = 'bar', figsize=(24, 5), color =
'green')

# Добавление точных значений каждому столбцу. Решение нестандартное, найдено на
Stackoverflow по запросу "pandas bar plot with values"
for p in axes.patches:
    axes.annotate(str(p.get_height()), (p.get_x(), p.get_height() * 1.01))

axes.grid(True)
```

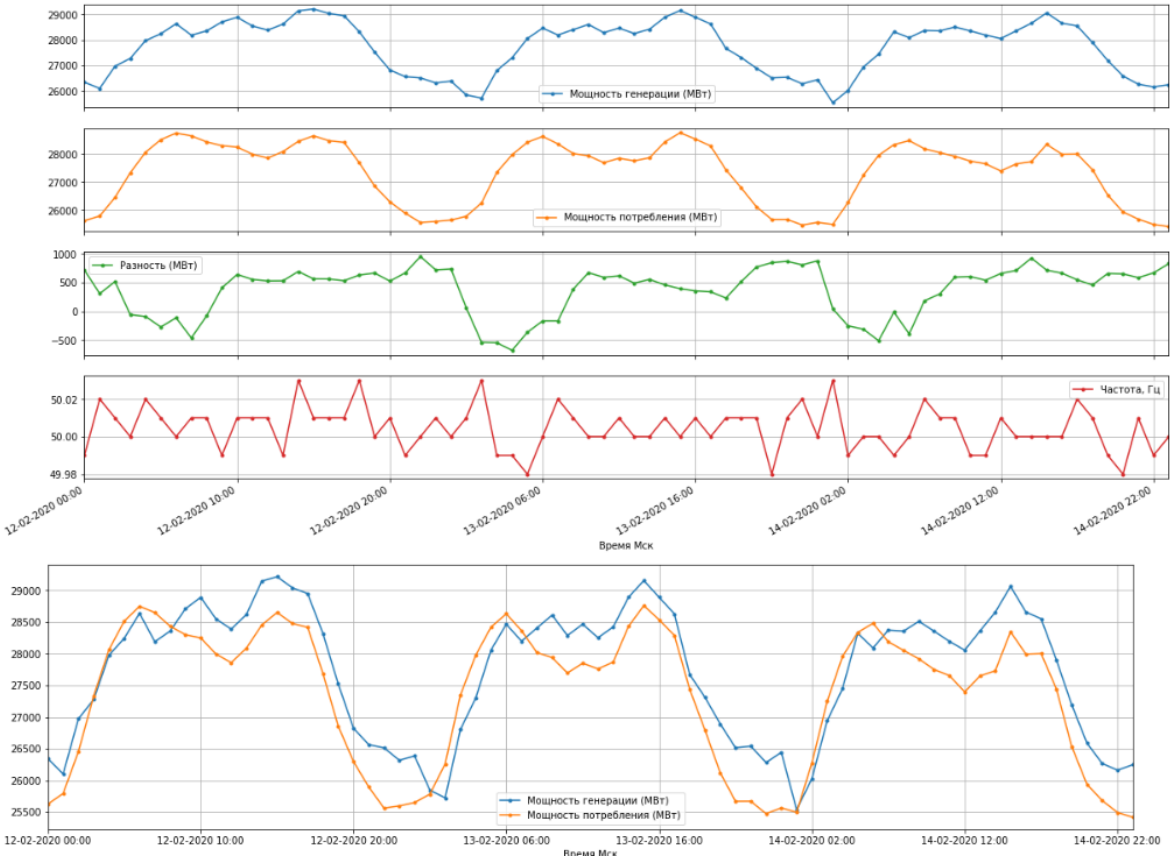

Рисунок 3. Графики потребления, генерации и частоты

Можно определить коэффициент корреляции, который показывает степень взаимозависимости величин (Рисунок 4). Коэффициент корреляции принимает значения от -1 до 1. Значение +1 говорит о связи между величинами, то есть если увеличивается одна, то линейно увеличивается и вторая. Значение -1 говорит о сильной связи, но если одна величина увеличивается, то другая пропорционально уменьшается. Значение 0 говорит, что никакой линейной связи между величинами нет. Из Рисунка 8 видно, что мощность генерации сильно связана с мощностью потребления. Другие величины друг с другом связаны или слабо, или нелинейно.

```
# 8. Одна из важных статистических метрик - корреляция
df_power_freq.corr()
```

	Мощность генерации (МВт)	Мощность потребления (МВт)	Разность (МВт)	Частота, Гц
Мощность генерации (МВт)	1.000000	0.924982	0.029980	0.059419
Мощность потребления (МВт)	0.924982	1.000000	-0.352109	0.061636
Разность (МВт)	0.029980	-0.352109	1.000000	-0.015776
Частота, Гц	0.059419	0.061636	-0.015776	1.000000

Рисунок 4. Матрица коэффициентов корреляции

Чтобы убедиться в связи или отсутствии связи, можно построить дополнительные графики (результат на Рисунке 5). Видно, что в целом при

увеличении мощности потребления увеличивается и мощность генерации. А вот изменение частоты и разности генерации и потребления за час визуально никак не связаны.

```
# 9. Точечные графики, показывающие зависимости величин
df_power_freq.plot(x = 'Мощность генерации (МВт)', y = 'Мощность потребления (МВт)', style='o')
df_power_freq.plot(x = 'Частота, Гц', y = 'Разность (МВт)', style='o')
```

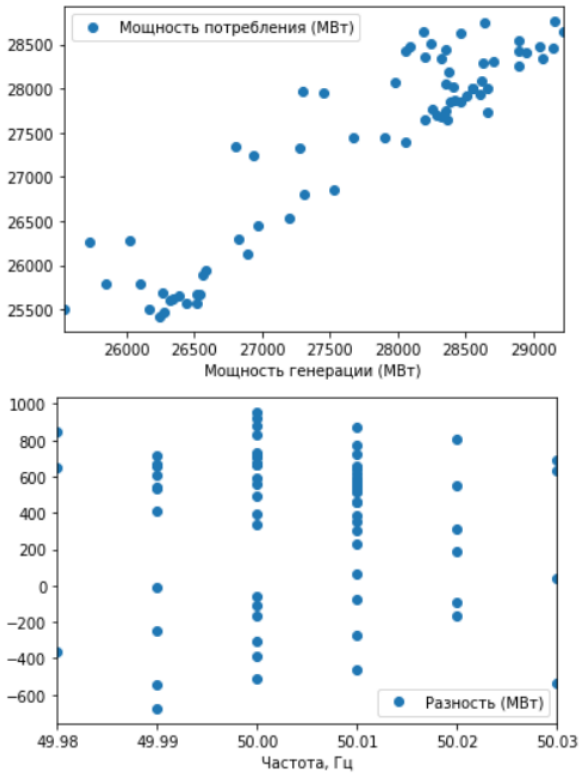

Рисунок 5. Графики зависимостей

4. Статистика по выработке электроэнергии в России

В примере использованы данные о помесячной выработке электроэнергии с сайта Министерства Энергетики РФ (https://minenergo.gov.ru/activity/statistic). Результат показан на Рисунке 6.

```
# 10. Чтение данных из Excel-файла
df_init =
pd.read_excel('https://drive.google.com/uc?export=download&id=1IFULWv72kGpZRDRpqRskm5NIM6AOCFPR', header = None)
df_init

# 11. Вывод графика
axes = df_init.plot(marker='.', figsize=(20, 5), subplots = False)
```

```
axes.grid(True)
```

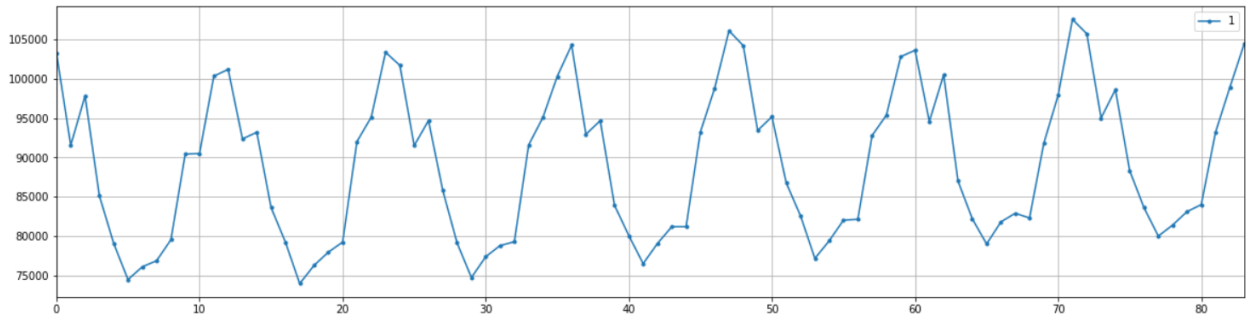

Рисунок 6. График помесячной выработки электроэнергии

Далее необходимо выполнить предобработку данных и после этого можно получить статистику по месяцам и годам, как показано на Рисунке 7. Видно, что выработка постепенно увеличивается год от года, и что уровень выработки выше в более холодные месяцы.

```
# 12. Из таблицы, полученной в ячейке 13, видно, что у данных нет заголовка, а дата записана просто стройкой.
# Нужно дать столбцам имена и выделить из даты номер месяца и года.
# Для разделение строки на части используется функция split, которой нужно указать разделитель. В данном случае это точка
df_date = df_init[0].str.split('.', expand = True)
print(df_date.head())
print()

# Создание новой таблицы, изначально пустой.
df_gen = pd.DataFrame()
df_gen['Generation, mln MWt*h'] = df_init[1] / 1000 # перевод кВтч в МВтч
df_gen['Month_ID'] = df_date[1].astype(int) # указание, что этот столбец хранит уже не строки, а целые числа (int - integer)
df_gen['Year'] = df_date[2].astype(int)
print(df_gen.head())

# 13. Статистика суммарной генерации по годам.
year_stat = df_gen.groupby(['Year'])['Generation, mln MWt*h'].sum()
print(year_stat)
print()
year_stat.plot(kind = 'bar')
print()

# 14. Статистика суммарной генерации по месяцам.
month_stat = df_gen.groupby(['Month_ID'])['Generation, mln MWt*h'].sum()
print(month_stat)
print()
```

```
month_stat.plot(kind = 'bar')
print()

# 15. Сортировка по возрастанию
month_stat.sort_values().plot(kind = 'bar')
print()
```

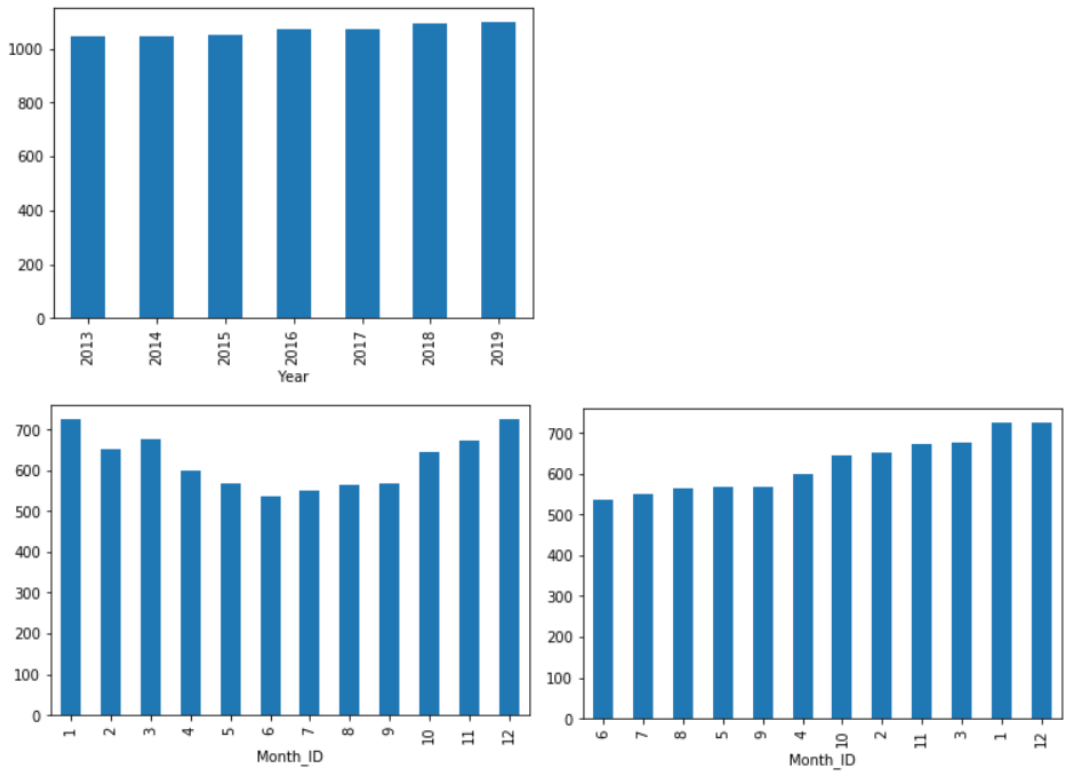

Рисунок 7. Статистика по годам и месяцам

Последний шаг - построение интерактивного графика, который покажет изменение выработки и по годам, и по месяцам, как показано на Рисунке 8.

```
# 16. Подключение библиотек для интерактивного графика
from bokeh.plotting import figure, output_file, show
from bokeh.io import output_notebook
import numpy as np

output_notebook() # чтобы график показывался прямо в блокноте

# 17. Построение графиков помесячной выработки за каждый год
p = figure(plot_width = 800, plot_height = 400)

# Список цветов по количеству лет.
colors = ['magenta', 'red', 'yellow', 'green', 'cyan', 'blue', 'black']
```

```
# Для каждого года свой график.
for year in range(2013, 2020):

  # Выбор из таблицы данных по выработке за нужный год.
  values = df_gen[df_gen['Year'] == year]['Generation, mln MWt*h'].values

  # Построение линий.
  p.line(np.arange(12), values, color = colors[year - 2013], legend_label =
str(year), line_width = 3)

  # Построение маркеров.
  p.circle(np.arange(12), values, color = colors[year - 2013], alpha=0.5, size
= 8)


show(p)
```

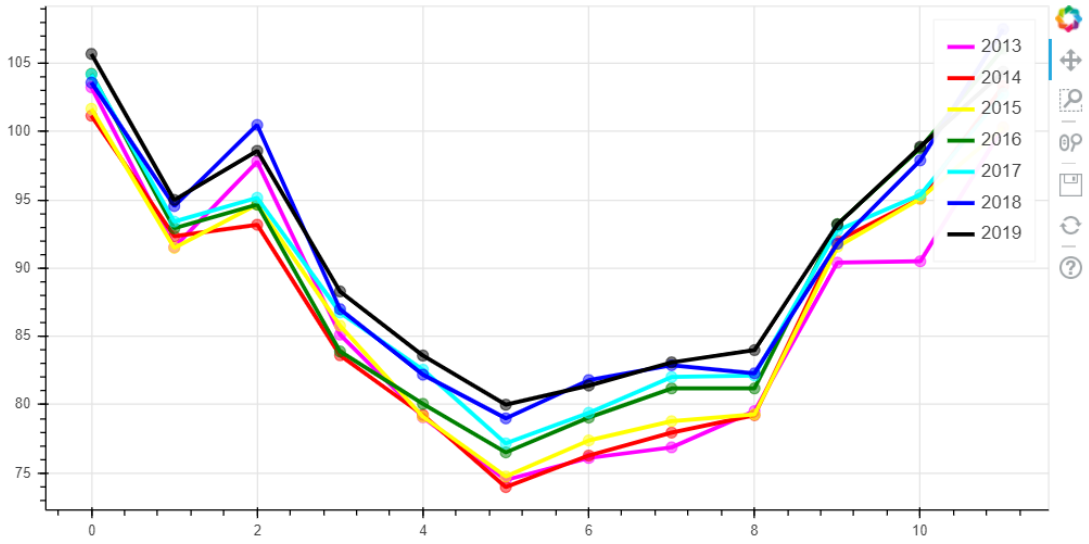

Рисунок 8. Интерактивный график выработки по годам и месяцам

5. Индивидуальные задания

1. Найти в открытых источниках сайт, предоставляющий таблицы с данными, связанными с энергетикой. Примеры сайтов: so-ups.ru/, Атомная_энергетика_России, Solar_power, Тепловая_энергетика_России Гидроэнергетика, rosseti.ru/investment/dzo/long/, https://www.kommersant.ru/doc/2645543
2. Создать новый блокнот в Google Colab.
3. Реализовать web-scraping таблицы с данными, для сайта https://minenergo.gov.ru/activity/statistic можно скачать данные и разместить на Google Drive.
4. Выполнить обработку данных таблицы, продемонстрировать указанные ниже приемы.
    4.1. Сортировка данных.
    4.2. Добавление к таблице столбца с новыми данными, полученным путем преобразований имеющихся.
    4.3. Выбор фрагмента таблицы по условию.
    4.4. Выбор фрагмента таблицы по номерам строк и столбцов.
    4.5. Расчеты статистических показателей по сгруппированным данным.
    4.6. Построение гистограммы по данным.
    4.7. Построение круговой диаграммы по данным.
    4.8. Построение интерактивного графика по данным.
5. Добавить созданный блокнот в свой GitHub репозиторий, созданный в первой лабораторной работе или в новый репозиторий.

6. Отчет один на бригаду

- Титульный лист.
- Цель работы.
- Имя GitHub репозитория, в который сохранен код программы.
- Python код программы.
- Полученные графики или таблицы из пунктов задания 4.3, 4.5 - 4.8.