Міністерство освіти і науки, молоді та спорту
України Національний технічний університет
України "Київський політехнічний інститут імені
Ігоря Сікорського" Фізико - технічний інститут

Курсова робота

Аналіз даних за допомогою Python та бібліотеки Pandas

Виконав:

Студент групи ФІ-21, НН ФТІ

Мелоян Мирослав

Перевірив:

Частина перша

Мета:

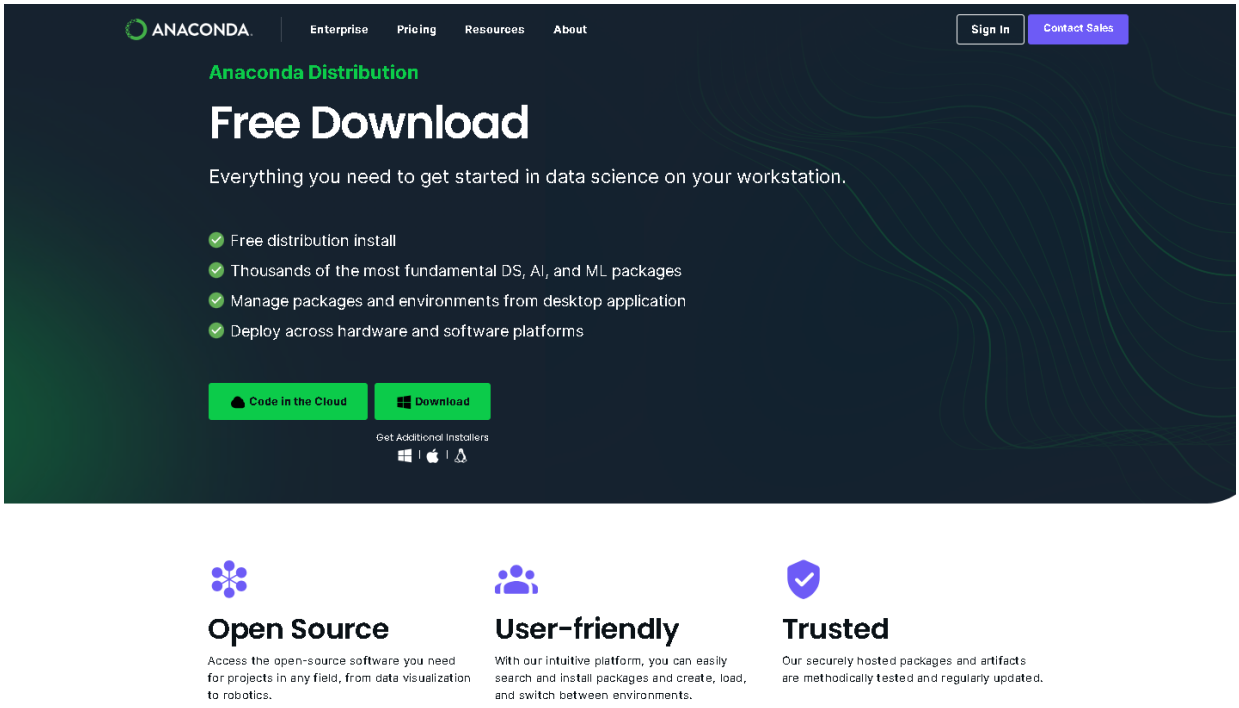
Вивчення основних принципів роботи та використання програмних компонентів Anaconda.

Загальна характеристика Anaconda

Anaconda - це інтегроване середовище розробки для мов програмування Python та R, призначене для аналізу даних, машинного навчання та візуалізації. Воно включає потужний пакетний менеджер, багато корисних бібліотек, IPython для інтерактивного програмування, інструменти для візуалізації даних та підтримку мови R. Anaconda є популярним інструментом для спеціалістів у сфері Data Science.

Встановлення Anaconda

Перейдіть на веб-сайт [Free Download Anaconda] і завантажте Anaconda



Після завантаження відкрийте **Anaconda Navigator** для подальшого встановлення **Jupyter Notebook**

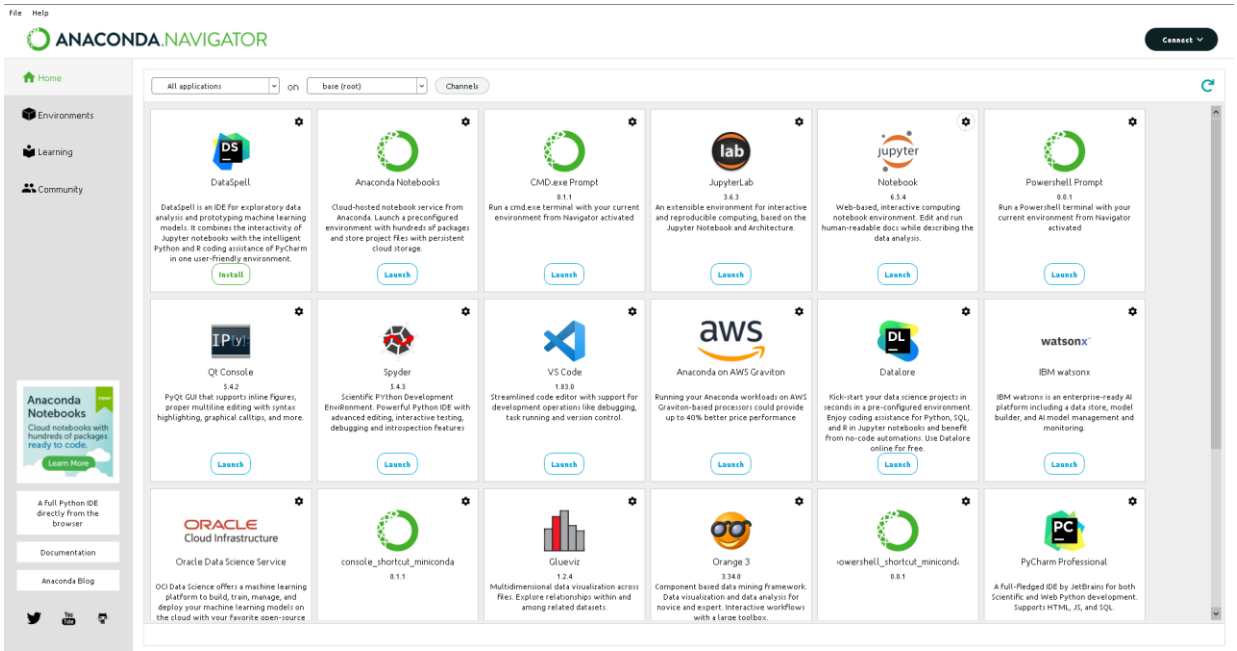
Загальна характеристика IPython Notebook

Ipython Notebook — це середовище розробки, що дає можливості для інтерактивної та ітеративної розробки проектів, пов’язаних з Data Science. Notebook інтегрує код та результати його виконання в один файл разом з графіками, текстом, математичними виразами та іншими можливими видами представлення інформації.

Jupyter Notebook — розширення IPython Notebook, яке підтримує не лише Python, але й інші мови програмування. Він також дозволяє створювати і виконувати інтерактивні блокноти, об'єднуючи код та результати його виконання з текстовими описами та графіками. Jupyter Notebook користується великою популярністю в галузі Data Science та наукових досліджень.

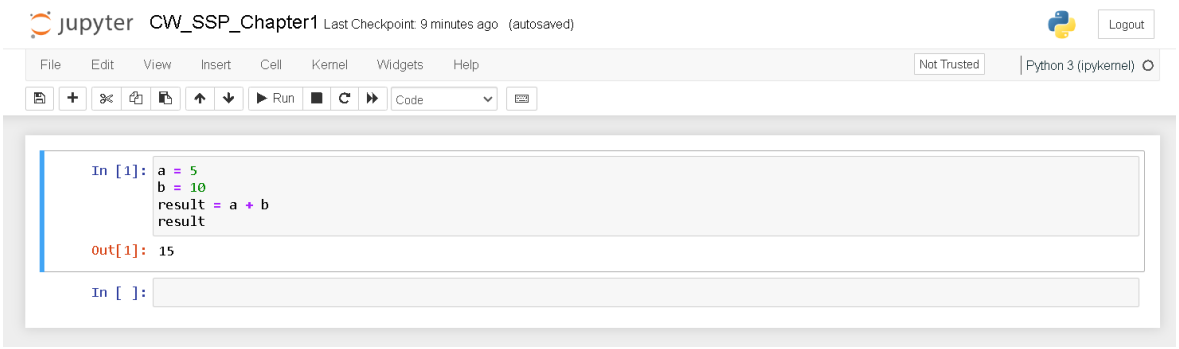
Встановлення Jupyter Notebook

Відкривши Anaconda Navigator Натисіть **«Install»** під Jupyter Notebook. У мене він вже був встановлений:



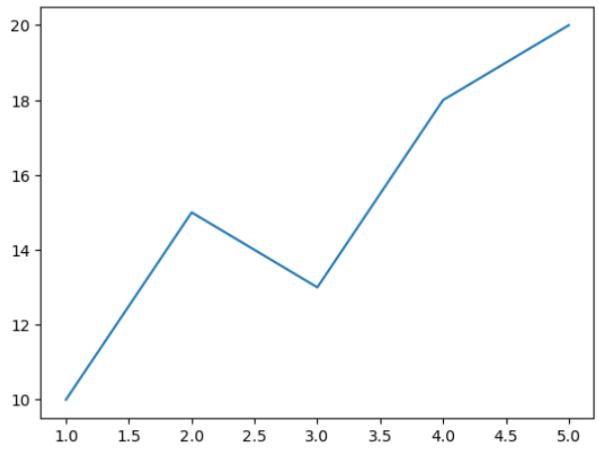
Можливості Jypiter Notebook

- Інтерактивне програмування Python



- Відображення результатів виконаного коду безпосередньо у нотатку.

```
In [2]: import matplotlib.pyplot as plt
In [3]: x = [1, 2, 3, 4, 5]
In [4]: y = [10, 15, 13, 18, 20]
In [5]: plt.plot(x, y)
Out[5]: [<matplotlib.lines.Line2D at 0x1e59a451090>]
```

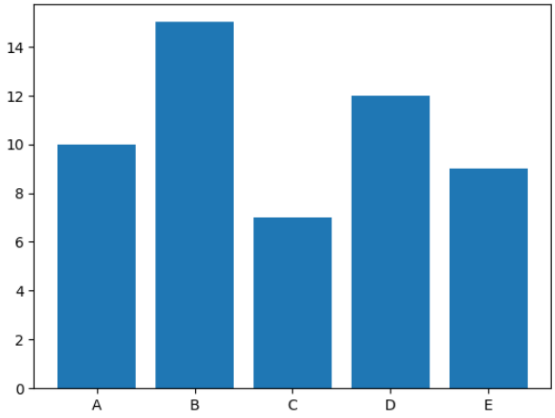


- Візуалізація даних у вигляді графіків, діаграм, та таблиць.

```
In [6]: import pandas as pd
In [7]: data = {'Name': ['alice', 'bob', 'jonh', 'david', 'tom'],'Age': [25, 30, 22, 35, 28]}
In [8]: df = pd.DataFrame(data)
In [9]: df
Out[9]:
```

	Name	Age
0	alice	25
1	bob	30
2	jonh	22
3	david	35
4	tom	28

```
In [10]: import matplotlib.pyplot as plt

In [11]: categories = ['A', 'B', 'C', 'D', 'E']

In [12]: values = [10, 15, 7, 12, 9]

In [13]: plt.bar(categories, values)

Out[13]: <BarContainer object of 5 artists>
```



- Вставка текстових коментарів, формул, зображень та інших мультимедійних елементів за допомогою Markdown-синтаксису

```
In [15]: #Ipython Notebook is a development environment
```

```
This is a text comment. You can add *cursive*, **bold text**, [link](https://www.kaggle.com/datasets/kumarajarshi/life-expectancy-who/data).

This is Einstein's formula: $E=mc^2$.

This is the square root formula: $\sqrt{a^2 + b^2}$

<img src="https://www.redfin.com/blog/wp-content/uploads/2021/07/Kitty1.jpg" alt="and this is the kitty">
```

This is a text comment. You can add *cursive*, **bold text**, link.

This is Einstein's formula: $E = mc^2$.

This is the square root formula: $\sqrt{a^2 + b^2}$

Markdown - це легкий мовний формат для форматування тексту, який широко використовується у Jupyter Notebook для створення зручних та читабельних розміток. Зазвичай, Markdown використовується для створення текстових коментарів, формул, списків, посилань і багатьох інших елементів в Jupyter Notebook.

Частина друга

Мета:

Отримати досвід використання основних засобів аналізу та візуалізації даних у середовищі IPython Notebook на модельних наборах даних.

Аналіз обраного датасету

В роботі використовується публічний набір даних на платформі Kaggle - **Life Expectancy (WHO)**.

Датасет стосується факторів, що впливають на тривалість життя населення в різних країнах. Дослідження включає такі аспекти, як імунізація, фактори смертності, економічні та соціальні фактори. Датасет збирає дані з року 2000 до 2015 року для 193 країн та містить інформацію про рівень імунізації (наприклад, гепатит B, поліо, дифтерія), фактори смертності та економічні показники.

Мета дослідження полягає в тому, щоб визначити фактори, які впливають на тривалість життя населення та надати рекомендації країнам для покращення тривалості життя їхнього населення.

Обробка даних

Імпортуємо необхідні бібліотеки та відкриваємо набір даних

Life Expectancy (WHO)

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

Opening a dataset with pandas

In [2]:
```
df = pd.read_csv (r'C:\\Users\\Rock4\\Downloads\\Life Expectancy Data.csv')
```

In [3]:
```
df.head()
```

Out[3]:

	Country	Year	Status	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles	...	Polio	Total expenditure	Diphtheria	HIV/AIDS
0	Afghanistan	2015	Developing	65.0	263.0	62	0.01	71.279624	65.0	1154	...	6.0	8.16	65.0	0.1
1	Afghanistan	2014	Developing	59.9	271.0	64	0.01	73.523582	62.0	492	...	58.0	8.18	62.0	0.1
2	Afghanistan	2013	Developing	59.9	268.0	66	0.01	73.219243	64.0	430	...	62.0	8.13	64.0	0.1
3	Afghanistan	2012	Developing	59.5	272.0	69	0.01	78.184215	67.0	2787	...	67.0	8.52	67.0	0.1
4	Afghanistan	2011	Developing	59.2	275.0	71	0.01	7.097109	68.0	3013	...	68.0	7.87	68.0	0.1

5 rows × 22 columns

Перевірка даних на наявність пропущених значень та NaN

Check for missing values and NaN.

```
In [4]: df.isnull().values.any()
Out[4]: True

In [5]: df.isnull().sum()
Out[5]: Country                          0
        Year                             0
        Status                           0
        Life expectancy                 10
        Adult Mortality                 10
        infant deaths                    0
        Alcohol                        194
        percentage expenditure           0
        Hepatitis B                    553
        Measles                          0
         BMI                            34
        under-five deaths                0
        Polio                           19
        Total expenditure              226
        Diphtheria                      19
         HIV/AIDS                        0
        GDP                            448
        Population                     652
         thinness  1-19 years           34
         thinness 5-9 years             34
        Income composition of resources  167
        Schooling                      163
        dtype: int64
```

Заповнення пропущених значень

Data Cleaning

```
In [6]: df = df.fillna(method='bfill')

In [7]: df.isnull().values.any()
Out[7]: False
```

Відображення 10 елементів для 4-ох ознак

Matrix representation

```
In [8]: selected_features = df[['Life expectancy ', 'Adult Mortality', 'Alcohol', 'percentage expenditure']].head(10)
        selected_features
```

Out[8]:

	Life expectancy	Adult Mortality	Alcohol	percentage expenditure
0	65.0	263.0	0.01	71.279624
1	59.9	271.0	0.01	73.523582
2	59.9	268.0	0.01	73.219243
3	59.5	272.0	0.01	78.184215
4	59.2	275.0	0.01	7.097109
5	58.8	279.0	0.01	79.679367
6	58.6	281.0	0.01	56.762217
7	58.1	287.0	0.03	25.873925
8	57.5	295.0	0.02	10.910156
9	57.3	295.0	0.03	17.171518

Створення матриці. Виділення зеленим кольором максимальні елементи в таблиці, а червоним мінімальні для цих 10-ти елементів

In [9]:
```
# find the maximum and minimum values for each feature
max_values = selected_features.max()
min_values = selected_features.min()

# create a matrix of maximum and minimum values for each feature
matrix = pd.concat([max_values, min_values], axis=1)
matrix.columns = ['max', 'min']

# function for highlighting with color
def highlight_max_min(val):
    color = ['background-color: green' if val == max_value else
         'background-color: red' if val == min_value else
         '' for val, max_value, min_value in zip(val, max_values, min_values)]
    return color

# display the matrix with the maximum and minimum values highlighted in color
styled_matrix = matrix.style.apply(highlight_max_min, axis=0)
styled_matrix
```

Out[9]:

	max	min
Life expectancy	65.000000	57.300000
Adult Mortality	295.000000	263.000000
Alcohol	0.030000	0.010000
percentage expenditure	79.679367	7.097109

Математичне сподівання та дисперсія для набору даних

Math expectation

In [10]:
```
numeric_columns = df.select_dtypes(include=[float, int])
mean_values = numeric_columns.mean()
mean_values
```

Out[10]:
```
Year                               2.007519e+03
Life expectancy                    6.923410e+01
Adult Mortality                    1.646726e+02
infant deaths                      3.030395e+01
Alcohol                            4.493087e+00
percentage expenditure             7.382513e+02
Hepatitis B                        8.137304e+01
Measles                            2.419592e+03
 BMI                               3.857723e+01
under-five deaths                  4.203574e+01
Polio                              8.264159e+01
Total expenditure                  5.976076e+00
Diphtheria                         8.241695e+01
 HIV/AIDS                          1.742103e+00
GDP                                7.393906e+03
Population                         1.139037e+07
 thinness  1-19 years              4.808645e+00
 thinness 5-9 years                4.838325e+00
Income composition of resources    6.373210e-01
Schooling                          1.221467e+01
dtype: float64
```

Dispersion

In [11]:
```
dispersion = df.var(numeric_only=True)
dispersion
```

Out[11]:
```
Year                               2.128753e+01
Life expectancy                    9.056044e+01
Adult Mortality                    1.542285e+04
infant deaths                      1.390666e+04
Alcohol                            1.659412e+01
percentage expenditure             3.951805e+06
Hepatitis B                        6.032302e+02
Measles                            1.314983e+08
 BMI                               4.040515e+02
under-five deaths                  2.574277e+04
Polio                              5.466693e+02
Total expenditure                  6.331895e+00
Diphtheria                         5.602411e+02
 HIV/AIDS                          2.578390e+01
GDP                                1.783400e+08
Population                         2.923765e+15
 thinness  1-19 years              1.942120e+01
 thinness 5-9 years                2.021024e+01
Income composition of resources    4.412145e-02
Schooling                          1.177008e+01
dtype: float64
```

Стандартизовані дані

Standard data

In [12]:
```
numeric_columns = df.select_dtypes(include=[float, int])
mean_values = numeric_columns.mean()
std_deviation = numeric_columns.std()
standardized_data = (numeric_columns - mean_values) / std_deviation
df[numeric_columns.columns] = standardized_data
```

In [13]:
```
df[numeric_columns.columns]
```

Out[13]:

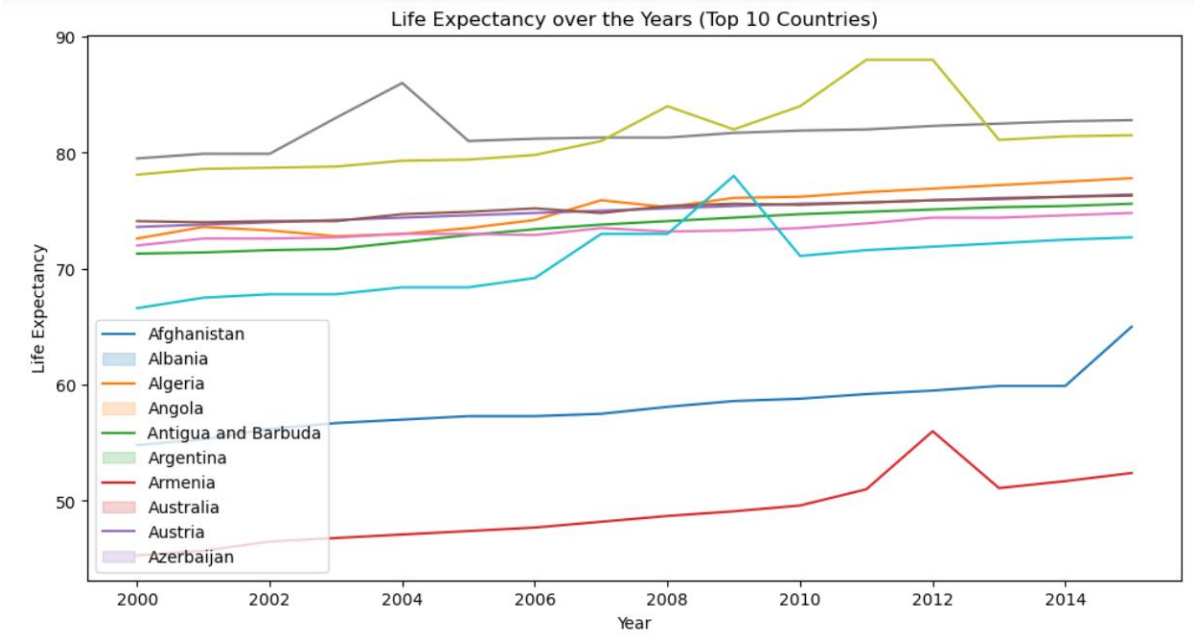
	Year	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles	BMI	under-five deaths	Polio	Total expenditure	Diphther
0	1.621486	-0.444931	0.791758	0.268778	-1.100525	-0.335513	-0.666635	-0.110366	-0.968967	0.255316	-3.277949	0.867902	-0.7358
1	1.404747	-0.980852	0.856176	0.285738	-1.100525	-0.334384	-0.788781	-0.168095	-0.993841	0.274014	-1.053917	0.875850	-0.8625
2	1.188008	-0.980852	0.832019	0.302697	-1.100525	-0.334537	-0.707350	-0.173502	-1.018715	0.292712	-0.882838	0.855980	-0.7780
3	0.971269	-1.022885	0.864228	0.328137	-1.100525	-0.332040	-0.585204	0.032040	-1.043590	0.317642	-0.668989	1.010968	-0.6513
4	0.754530	-1.054410	0.888385	0.345097	-1.100525	-0.367800	-0.544488	0.051748	-1.063489	0.342573	-0.626219	0.752655	-0.6090
...	...	...	...	...	...	...	...	...	...	...	...	...	...
2933	-0.762644	-2.620140	4.495797	-0.028017	-0.032671	-0.371370	-0.544488	-0.208296	-0.570977	-0.000223	-0.668989	0.458575	-0.7358
2934	-0.979384	-2.599124	4.431378	-0.036497	-0.106316	-0.371370	-3.028126	-0.123970	-0.590877	-0.006455	-3.235179	0.216158	-0.6090
2935	-1.196123	-2.567599	-0.738171	-0.044977	-0.015487	-0.371370	-0.340912	-0.184490	-0.610776	-0.012688	-0.412369	0.220132	-0.4823
2936	-1.412862	-2.515058	4.197863	-0.044977	-0.680748	-0.371370	-0.218765	-0.164869	-0.630676	-0.018921	-0.284060	0.073092	-0.3133
2937	-1.629601	-2.441500	4.028766	-0.053457	-0.690567	-0.371370	-0.096619	-0.081675	-0.650575	-0.018921	-0.198520	0.446653	-0.1866

Візуалізація даних

Data Visualization

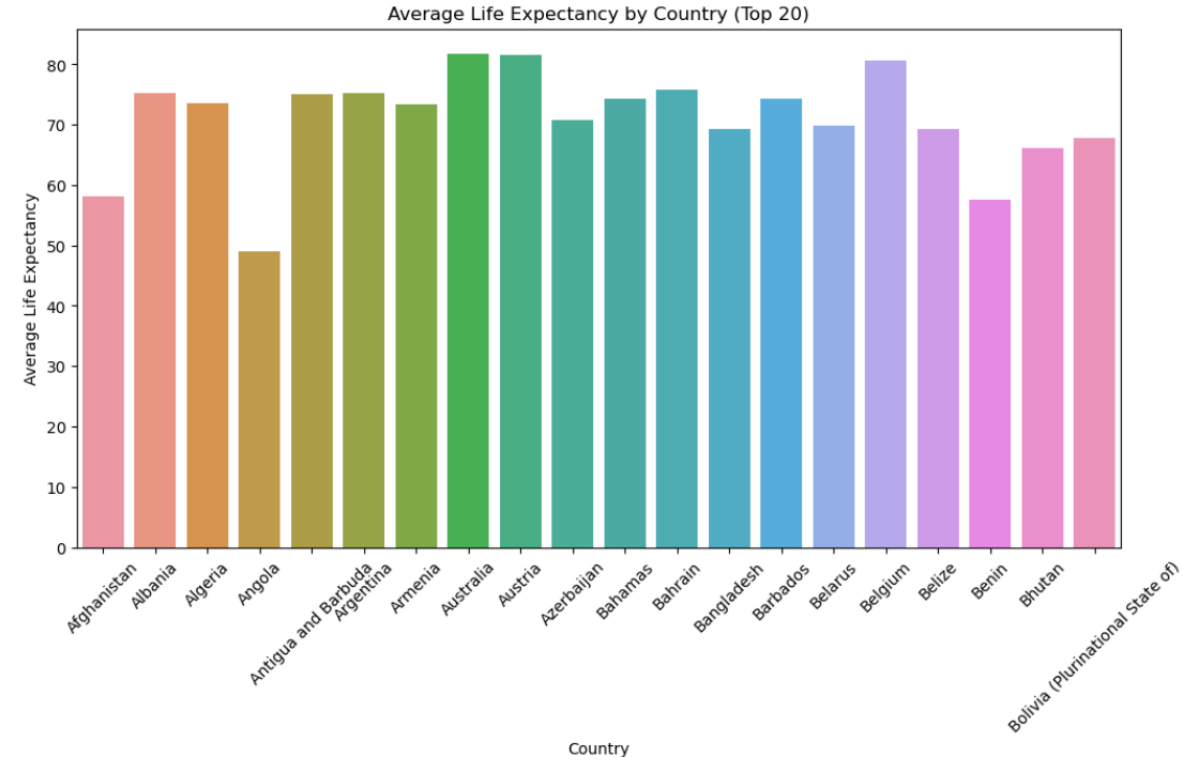
Line Plot

In [13]:
```
selected_countries = df['Country'].unique()[:10]

plt.figure(figsize=(12, 6))
for country in selected_countries:
    sns.lineplot(data=df[df['Country'] == country], x='Year', y='Life expectancy ')
plt.xlabel('Year')
plt.ylabel('Life Expectancy')
plt.legend(selected_countries)
plt.title('Life Expectancy over the Years (Top 10 Countries)')
plt.show()
```



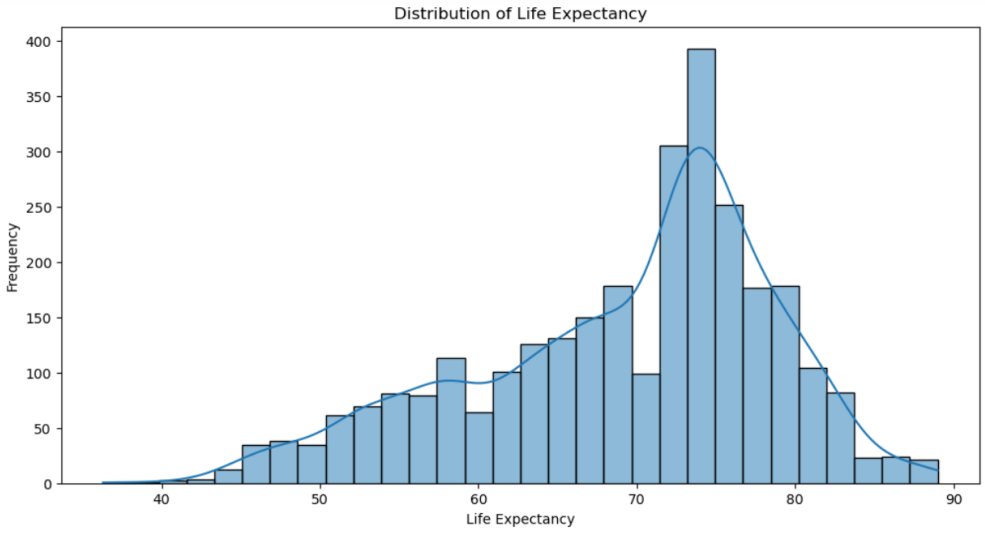
Bar Plot

In [14]:
```
plt.figure(figsize=(12, 6))
avg_life_expectancy = df.groupby('Country')['Life expectancy '].mean().head(20)  # Вибрати перші 10 країн
sns.barplot(x=avg_life_expectancy.index, y=avg_life_expectancy)
plt.xlabel('Country')
plt.ylabel('Average Life Expectancy')
plt.title('Average Life Expectancy by Country (Top 20)')
plt.xticks(rotation=45)
plt.show()
```



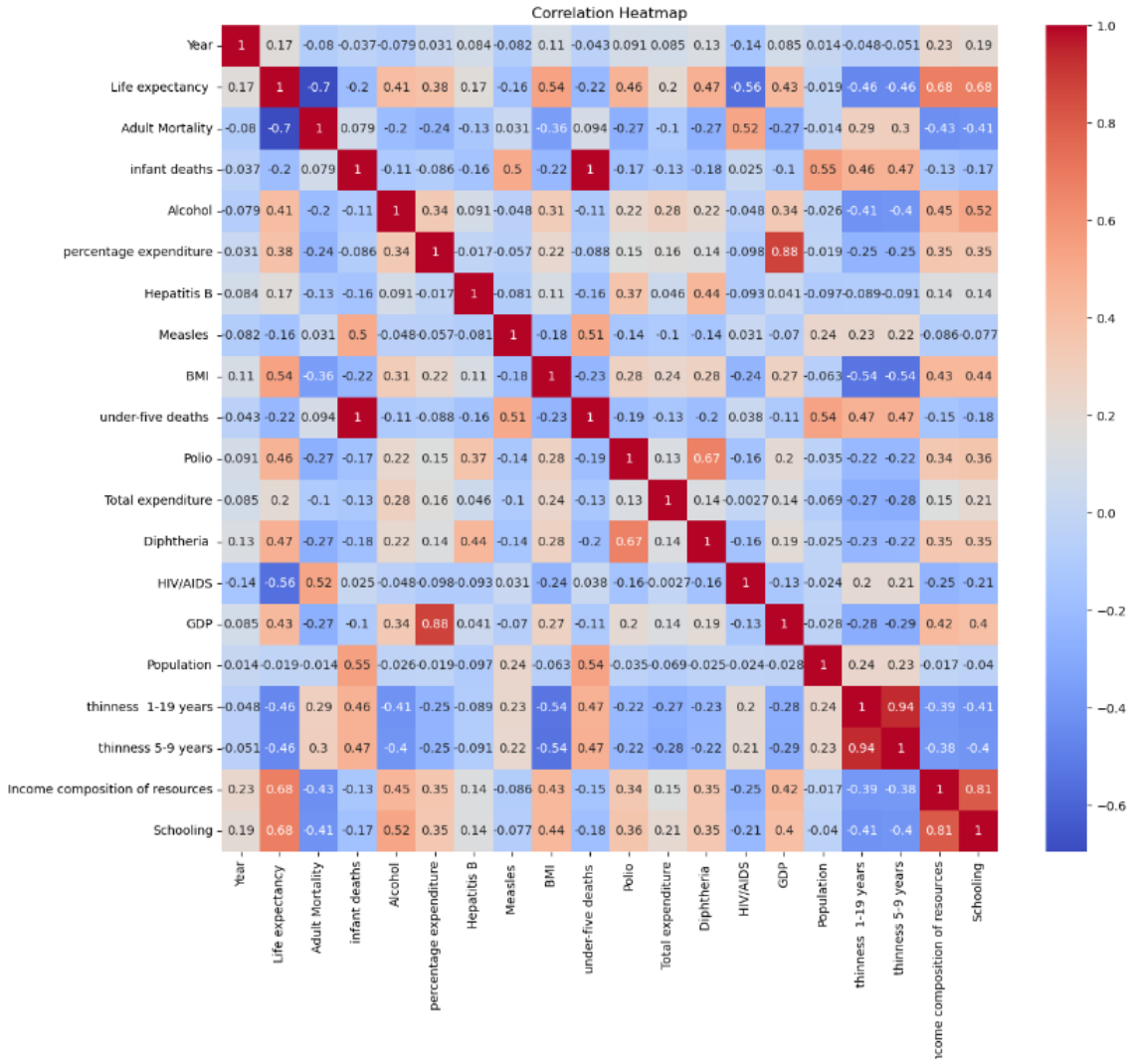
Histogram

In [15]:
```
plt.figure(figsize=(12, 6))
sns.histplot(df['Life expectancy '], bins=30, kde=True)
plt.xlabel('Life Expectancy')
plt.ylabel('Frequency')
plt.title('Distribution of Life Expectancy')
plt.show()
```



Heatmap

```
In [16]: plt.figure(figsize=(14, 12))
numeric_columns = df.select_dtypes(include=[float, int])
correlation_matrix = numeric_columns.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```



Лінійний графік демонструє, як змінюється тривалість життя з роками для різних країн:

- Легко порівняти середню тривалість життя між різними країнами.
- Визначити країни з вищою або нижчою середньою тривалістю життя.
- Використовуйте цю інформацію для ранжування або порівняння країн за тривалістю життя.

Стовпчаста діаграма відображає середню тривалість життя для різних країн:

- Зрозуміти частоту і розкид значень тривалості життя в різних країнах.
- Визначити, чи дані розподілені нормально чи мають викривлення.
- Виявити загальні діапазони значень тривалості життя в наборі даних.

Гістограма представляє розподіл значень тривалості життя:

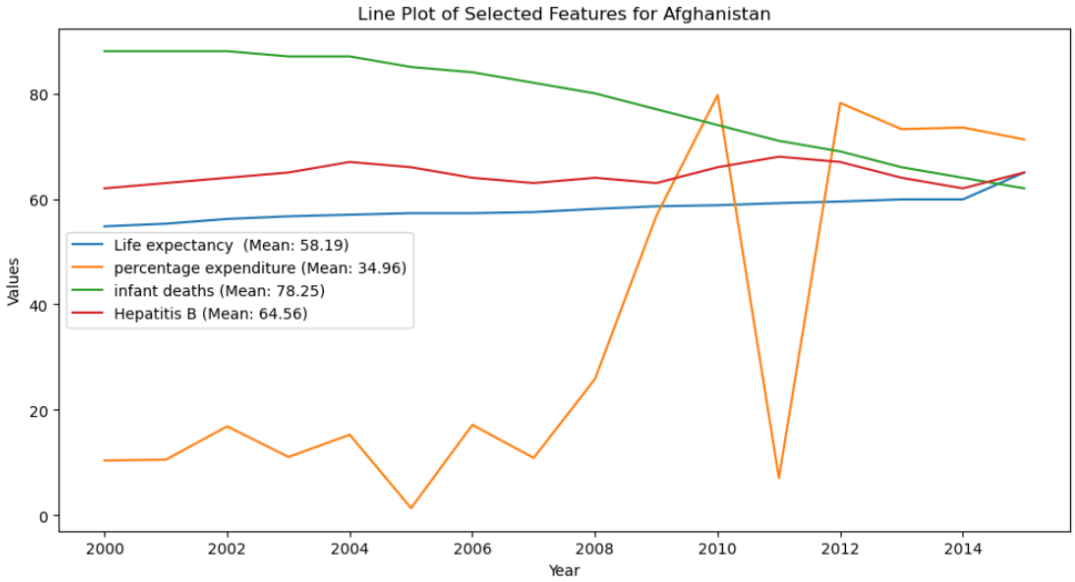
- Зрозуміти частоту і розкид значень тривалості життя в різних країнах.
- Визначити, чи дані розподілені нормально чи мають викривлення.
- Виявити загальні діапазони значень тривалості життя в наборі даних.

Теплова карта відображає кореляції між різними факторами здоров'я та економіки в наборі даних:

- Визначити фактори, які мають сильну позитивну або негативну кореляцію.
- Виявити зв'язки між різними змінними, такими як ВВП, освіта та тривалість життя.
- Використовуйте цю інформацію для розуміння того, які фактори пов'язані із змінами тривалості життя.

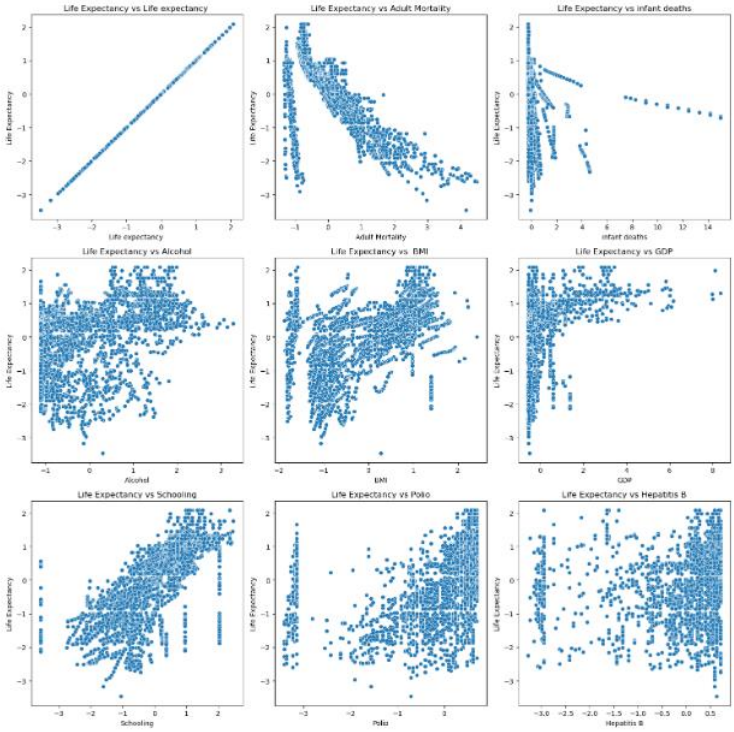
Лінійний графік з 4-ма лініями, які взяті з ознак:

In [18]:
```
# Filter the data for Afghanistan
features = ['Life expectancy ', 'percentage expenditure', 'infant deaths', 'Hepatitis B']
country_selected = "Afghanistan"
df_filtered_country = df[df['Country'] == country_selected]

# Creating a linear plot for the selected country across all years
plt.figure(figsize=(12, 6))
for feature in ['Life expectancy ', 'percentage expenditure', 'infant deaths', 'Hepatitis B']:
    mean_value = df_filtered_country[feature].mean()
    plt.plot(df_filtered_country['Year'], df_filtered_country[feature], label=f'{feature} (Mean: {mean_value:.2f})')

plt.xlabel('Year')
plt.ylabel('Values')
plt.title(f'Line Plot of Selected Features for {country_selected}')
plt.legend()
plt.show()
```



Графіки комбінування ознак:

```
In [20]: # adjusting the feature names based on the actual column names
subplot_features = ['Life expectancy ', 'Adult Mortality', 'infant deaths', 'Alcohol', ' BMI ', 'GDP', 'Schooling', 'Polio'

# recreating the 9 subplots with the corrected feature names
fig, axes = plt.subplots(3, 3, figsize=(15, 15))
for i, ax in enumerate(axes.flatten()):
    sns.scatterplot(ax=ax, data=df, x=subplot_features[i], y='Life expectancy ')
    ax.set_title(f'Life Expectancy vs {subplot_features[i]}')
    ax.set_xlabel(subplot_features[i])
    ax.set_ylabel('Life Expectancy')

plt.tight_layout()
plt.show()
```

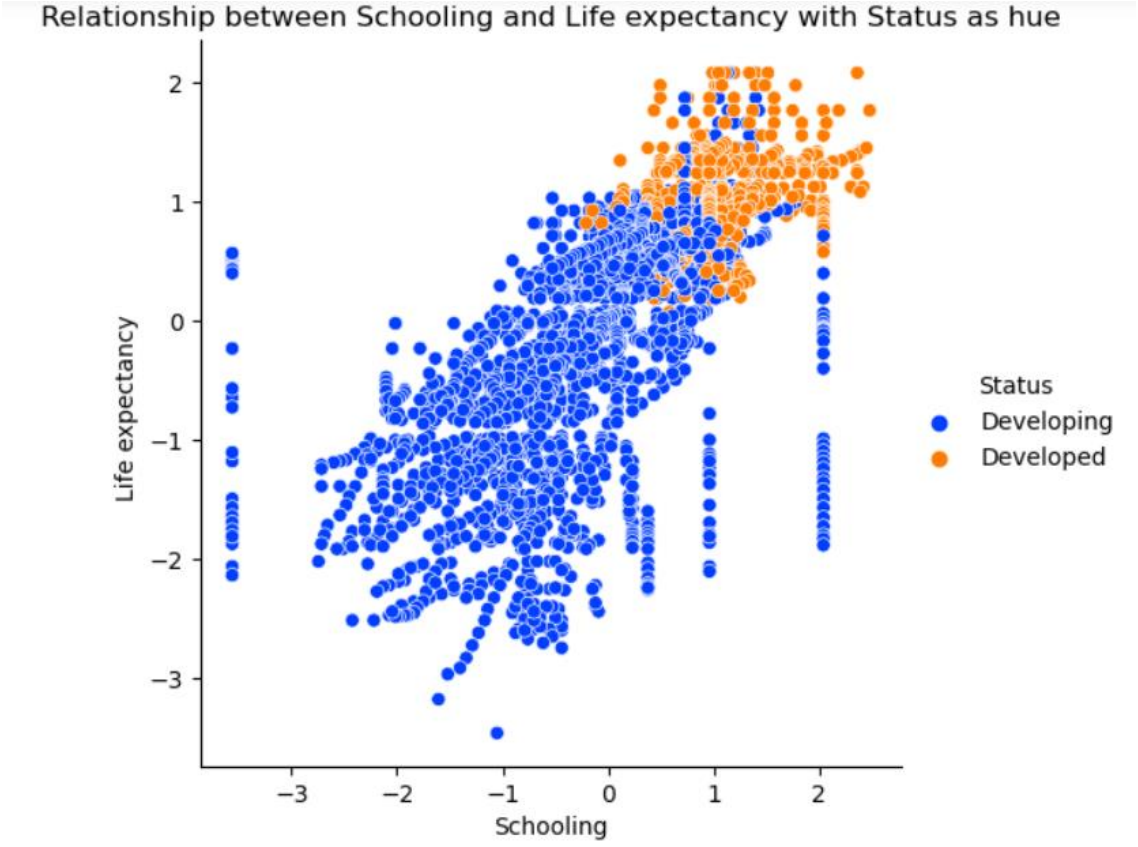


Коефіцієнт Пірсона для кожної пари ознак:

```
In [25]: selected_features = ['Life expectancy ', 'Adult Mortality', 'infant deaths', 'Alcohol', ' BMI ', 'GDP', 'Schooling', 'Polio
         # calculating Pearson correlation coefficients
         pearson_correlations = {}
         for feature in selected_features:
             if feature != 'Life expectancy ':
                 corr = df['Life expectancy '].corr(df[feature], method='pearson')
                 pearson_correlations[f'Life expectancy vs {feature}'] = corr

         # the pearson_correlations dictionary now contains the Pearson correlation coefficient for each pair
         pearson_correlations

Out[25]: {'Life expectancy vs Adult Mortality': -0.6958257409710517,
          'Life expectancy vs infant deaths': -0.1966034337928464 2,
          'Life expectancy vs Alcohol': 0.4068095090955223,
          'Life expectancy vs  BMI ': 0.538652211389222,
          'Life expectancy vs GDP': 0.42543128375376227,
          'Life expectancy vs Schooling': 0.67755663261605,
          'Life expectancy vs Polio': 0.45886712069867314,
          'Life expectancy vs Hepatitis B': 0.16901720774365264}
```

Relplot для двох ознак, що мають найбільшу кореляцію:

```
In [26]: # correcting the relplot to use 'Life expectancy ' on the y-axis and 'Schooling' on the x-axis.
         # we will also choose a categorical variable for hue. In the absence of a clear categorical feature
         # from the dataset's description, we can use 'Status' which indicates whether a country is 'Developing' or 'Developed'.
         # this will provide a clear categorical distinction in the data.

         sns.relplot(data=df, x='Schooling', y='Life expectancy ', hue='Status', kind='scatter', palette='bright')
         plt.title('Relationship between Schooling and Life expectancy with Status as hue')
         plt.show()
```



Позитивний зв'язок: Є чіткий позитивний тренд між кількістю років навчання та очікуваною тривалістю життя. Зі збільшенням кількості років

навчання, очікувана тривалість життя здається вищою.

Вплив розвитку країни: Розвинені країни, як правило, мають вищі показники як для навчання, так і для очікуваної тривалості життя порівняно з країнами що розвиваються.

Розкид даних: Для країн що розвиваються даних більше та вони мають більший розкид за обома ознаками, що може свідчити про різноманітність у рівнях освіти та здоров'я в цих країнах.

Видалення всіх колонок з NaN значеннями

Data Cleaning

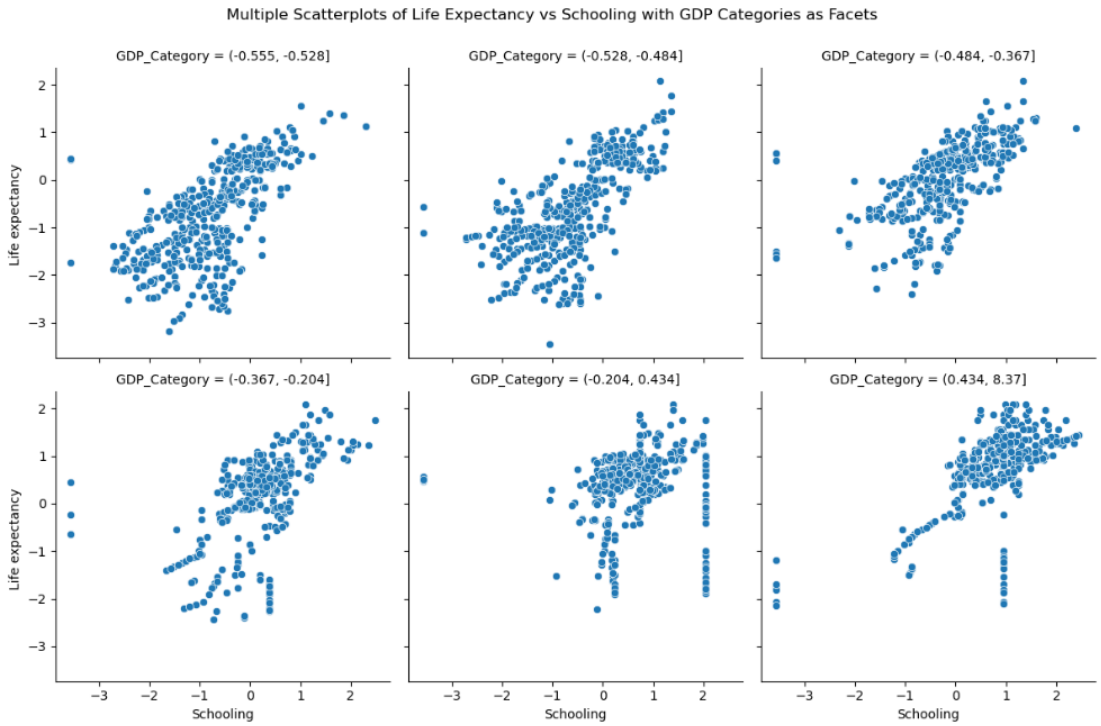
In [6]:
```
# dropping all columns with any NaN values from the dataset
cleaned_data = df.dropna(axis=1, how='any')

# displaying the first few rows of the cleaned dataset to verify
cleaned_data.head()
```

Out[6]:

	Country	Year	Status	infant deaths	percentage expenditure	Measles	under-five deaths	HIV/AIDS
0	Afghanistan	2015	Developing	62	71.279624	1154	83	0.1
1	Afghanistan	2014	Developing	64	73.523582	492	86	0.1
2	Afghanistan	2013	Developing	66	73.219243	430	89	0.1
3	Afghanistan	2012	Developing	69	78.184215	2787	93	0.1
4	Afghanistan	2011	Developing	71	7.097109	3013	97	0.1

Видалиня випадкових значень з ознак що мають найбільшу кореляцію у кількості 10% для обох. Побудова multipple scatterplot за допомогою FacetGrid

In [23]:
```
# dropping NaN values from 'Life expectancy ' and 'Schooling'
life_expectancy_data = df.dropna(subset=['Life expectancy ', 'Schooling'])

# calculate the number of rows to remove (10% of the data)
rows_to_remove = int(len(life_expectancy_data) * 0.10)

# removing 10% of random rows from the dataset for both 'Life expectancy ' and 'Schooling'
cleaned_data = life_expectancy_data.sample(frac=1, random_state=1).iloc[rows_to_remove:]

# calculate the new Pearson correlation for the cleaned data
new_corr = cleaned_data[['Life expectancy ', 'Schooling']].corr(method='pearson').iloc[0, 1]

# ensuring 'GDP' column does not have NaN values for the FacetGrid
cleaned_data = cleaned_data.dropna(subset=['GDP'])

# since 'GDP' is a continuous variable, we'll create a categorical version of it by binning into quantiles for clearer Face
cleaned_data['GDP_Category'] = pd.qcut(cleaned_data['GDP'], q=6)

# creating the FacetGrid
g = sns.FacetGrid(cleaned_data, col='GDP_Category', col_wrap=3, height=4)
g.map_dataframe(sns.scatterplot, x='Schooling', y='Life expectancy ')

# adding a main title and adjusting the plots
plt.subplots_adjust(top=0.9)
g.fig.suptitle('Multiple Scatterplots of Life Expectancy vs Schooling with GDP Categories as Facets')


plt.show()
new_corr
```




```
Out[23]: 0.6796421533826298
```

Кожен підграфік показує розкид точок для певного діапазону ВВП, демонструючи зв'язок між освітою та очікуваною тривалістю життя в різних економічних умовах.

Позитивна кореляція: У всіх категоріях ВВП існує позитивний зв'язок між кількістю років навчання та очікуваною тривалістю життя. Це вказує на те, що незалежно від економічного статусу країни, більш висока освіта, як правило, асоціюється з більш високою очікуваною тривалістю життя.

Економічний статус: Графіки демонструють, що країни з вищим ВВП мають тенденцію до вищих значень як для "Schooling", так і для "Life expectancy ". Це підкреслює важливість економічного благополуччя у підвищенні рівня освіти та покращенні здоров'я населення.

Розкид даних: На графіках видно, що розкид даних у категоріях з нижчим ВВП ширший, ніж у категоріях з вищим ВВП. Це може свідчити про більшу нерівномірність доступу до освіти та охорони здоров'я в бідніших країнах.

Частина третя

Мета:

Виконання та протоколювання додаткових завдань. Використання реального датасету

Додаткові завдання

Збудувати регресійні моделі прогнозування однієї ознаки на основі іншої. Для цього використаємо дві ознаки, що мають найбільшу кореляцію:

Task 1

```
In [29]: # ensure that only numeric columns are included for correlation calculation
numeric_df = df.select_dtypes(include=[np.number])

# calculate the Pearson correlation coefficient matrix on the numeric dataframe
correlation_matrix = numeric_df.corr(method='pearson')

# find the two features with the highest absolute correlation (ignoring the diagonal)
highest_corr = correlation_matrix.unstack().sort_values(kind="quicksort", ascending=False).drop_duplicates()
highest_corr_pair = highest_corr.index[1]  # index[0] is the diagonal, which is always 1
correlation_matrix, highest_corr_pair
```

```
Adult Mortality                  -0.430516  -0.411500
infant deaths                    -0.130803  -0.167416
Alcohol                           0.447385   0.524134
percentage expenditure            0.353530   0.345258
Hepatitis B                       0.143300   0.139000
Measles                          -0.086209  -0.076606
 BMI                              0.434298   0.436232
under-five deaths                -0.148003  -0.181655
Polio                             0.342367   0.358676
Total expenditure                 0.146646   0.210101
Diphtheria                        0.346929   0.347409
 HIV/AIDS                        -0.247070  -0.214802
GDP                               0.421795   0.396998
Population                       -0.017377  -0.039886
 thinness  1-19 years            -0.389668  -0.412555
 thinness 5-9 years              -0.380053  -0.404040
Income composition of resources   1.000000   0.807309
Schooling                         0.807309   1.000000  ,
('under-five deaths ', 'infant deaths'))
```

Найбільша кореляція у ознак “under-five-deaths" та “infant death”, але для більш наглядної візуалізації оберу ознаки з попередніх завдань

Необхідно протестувати такі регресійні моделі:

a. Linear regression

```
In [26]: # prepare the data
X = df[['Life expectancy ']].values
y = df['Schooling'].values

# split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
In [27]: # implementing the Linear Regression model
linear_model = LinearRegression()
linear_model.fit(X_train, y_train)
y_pred_linear = linear_model.predict(X_test)

# calculate metrics for Linear Regression
linear_mse = mean_squared_error(y_test, y_pred_linear)
linear_r2 = r2_score(y_test, y_pred_linear)

# return the metrics for Linear Regression
linear_metrics = {
    'Linear Regression': {'MSE': linear_mse, 'R2': linear_r2}
}
linear_metrics
```

```
Out[27]: {'Linear Regression': {'MSE': 0.5019089425149128, 'R2': 0.504896621914349}}
```

b. Random Forest Regression

In [28]:
```
# implementing the Random Forest Regression model
random_forest_model = RandomForestRegressor(random_state=42)
random_forest_model.fit(X_train, y_train)
y_pred_rf = random_forest_model.predict(X_test)

# calculate metrics for Random Forest Regression
rf_mse = mean_squared_error(y_test, y_pred_rf)
rf_r2 = r2_score(y_test, y_pred_rf)

# return the metrics for Random Forest Regression
rf_metrics = {
    'Random Forest Regression': {'MSE': rf_mse, 'R2': rf_r2}
}
rf_metrics
```

Out[28]:
```
{'Random Forest Regression': {'MSE': 0.5388648980066927,
  'R2': 0.4684417655559783}}
```

c. Multi Layer Perceptron Regression

In [29]:
```
# implementing the Multi Layer Perceptron Regression model
mlp_model = MLPRegressor(random_state=42, max_iter=1000)
mlp_model.fit(X_train, y_train)
y_pred_mlp = mlp_model.predict(X_test)

# calculate metrics for MLP Regression
mlp_mse = mean_squared_error(y_test, y_pred_mlp)
mlp_r2 = r2_score(y_test, y_pred_mlp)

# return the metrics for MLP Regression
mlp_metrics = {
    'MLP Regression': {'MSE': mlp_mse, 'R2': mlp_r2}
}
mlp_metrics
```

Out[29]:
```
{'MLP Regression': {'MSE': 0.46403638165453887, 'R2': 0.5422556550584305}}
```

Вибір моделі з найкращим результатом:

Task 2

In [30]:
```
# calculate RMSE for each model
linear_rmse = np.sqrt(linear_mse)
rf_rmse = np.sqrt(rf_mse)
mlp_rmse = np.sqrt(mlp_mse)

# compile RMSE into a dictionary for comparison
rmse_metrics = {
    'Linear Regression': linear_rmse,
    'Random Forest Regression': rf_rmse,
    'MLP Regression': mlp_rmse
}

# identify the model with the lowest RMSE
best_model_name = min(rmse_metrics, key=rmse_metrics.get)
best_model_rmse = rmse_metrics[best_model_name]

rmse_metrics, best_model_name, best_model_rmse
```

Out[30]:
```
({'Linear Regression': 0.7084553214670017,
  'Random Forest Regression': 0.7340741774553119,
  'MLP Regression': 0.681202159167555},
 'MLP Regression',
 0.681202159167555)
```

Найкращою у данному випадку виявилась Multi Layer Perceptron Regression

MSE - це абревіатура для "Mean Squared Error", що українською "Середньоквадратична помилка". Це метрика для оцінки якості моделі регресії, яка вимірює середню величину квадратів помилок, тобто середню відстань між прогнозованими та фактичними значеннями. Математично це визначається як середнє від квадратів різниць між фактичними та прогнозованими значеннями.

R^2 (або коефіцієнт детермінації) - це статистичний показник, який використовується для оцінки сили зв'язку між залежною змінною (тобто тим, що ми хочемо передбачити) та однією або кількома незалежними змінними (тобто предикторами або вхідними даними) у регресійній моделі.

- $R^2 = 0$: Модель не надає жодного пояснення варіації залежної змінної. Це означає, що предиктори моделі не дають інформації для передбачення значень залежної змінної краще, ніж би ми могли очікувати випадковим чином.
- $R^2 = 1$: Модель пояснює всю варіацію залежної змінної. Це означає, що всі точки лежать на лінії регресії, і модель передбачає залежну змінну без помилок.

Візуалізація результуючої регресійної функції та результату прогнозування за допомогою matplotlib та seaborn:

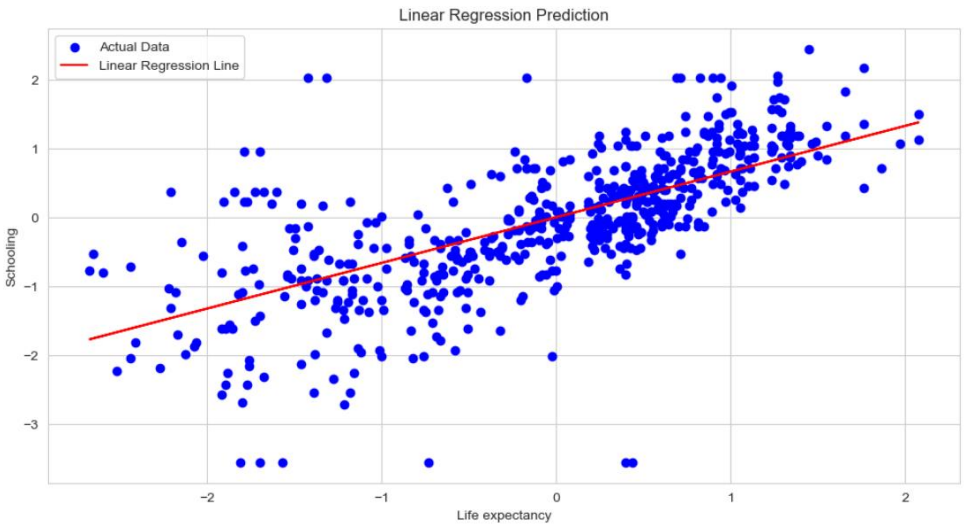
a. Linear regression

Цю модель можна зобразити як лінію на графіку розсіювання:

Task 4

In [31]:
```
# set the style for seaborn plots
sns.set_style("whitegrid")
```

In [40]:
```
# plot for Linear Regression
plt.figure(figsize=(12, 6))
plt.scatter(X_test, y_test, color='blue', label='Actual Data')
plt.plot(X_test, y_pred_linear, color='red', label='Linear Regression Line')
plt.title('Linear Regression Prediction')
plt.xlabel('Life expectancy')
plt.ylabel('Schooling')
plt.legend()
plt.show()
```



b. Random Forest Regression

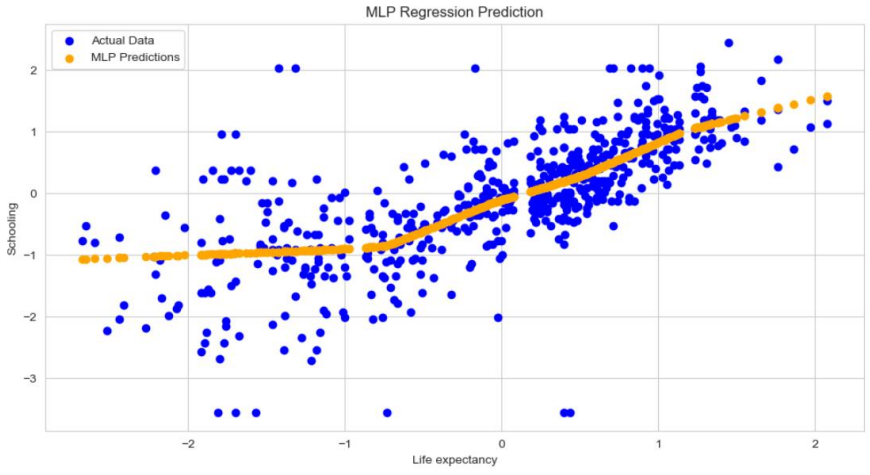
Оскільки це ансамблева модель, її не можна зобразити однією лінією. Натомість, я покажу передбачення моделі порівняно з фактичними даними:

```
In [41]: # Plot for Random Forest Regression
         plt.figure(figsize=(12, 6))
         plt.scatter(X_test, y_test, color='blue', label='Actual Data')
         plt.scatter(X_test, y_pred_rf, color='green', label='Random Forest Predictions')
         plt.title('Random Forest Regression Prediction')
         plt.xlabel('Life expectancy')
         plt.ylabel('Schooling')
         plt.legend()
         plt.show()
```

c. Multi Layer Perceptron Regression

Ця модель також не має простої візуалізації як лінія, тому я знову використаю метод порівняння передбачених та фактичних значень.

```
In [42]: # Plot for MLP Regression
         plt.figure(figsize=(12, 6))
         plt.scatter(X_test, y_test, color='blue', label='Actual Data')
         plt.scatter(X_test, y_pred_mlp, color='orange', label='MLP Predictions')
         plt.title('MLP Regression Prediction')
         plt.xlabel('Life expectancy')
         plt.ylabel('Schooling')
         plt.legend()
         plt.show()
```



Піднесення до квадрату ознаки, що використовується в якості предиктора для регресійної моделі та використання її квадрату ознаки як другу ознаку:

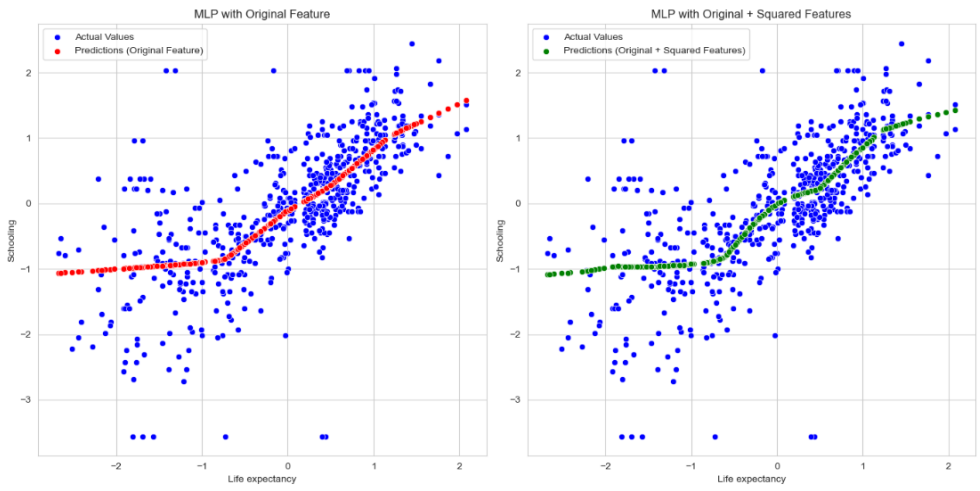
Task 5

In [53]:
```
# Preparing data with the original feature and its square
X = df[['Life expectancy ']].copy()
X['Life expectancy squared'] = X['Life expectancy '] ** 2
y = df['Schooling']

# Separating data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Training a Multi Layer Perceptron model with the original feature
mlp_model_original = MLPRegressor(random_state=42, max_iter=1000)
mlp_model_original.fit(X_train[['Life expectancy ']], y_train)
y_pred_mlp_original = mlp_model_original.predict(X_test[['Life expectancy ']])

# Training a Multi Layer Perceptron model with the original feature and its square
mlp_model_with_square = MLPRegressor(random_state=42, max_iter=1000)
mlp_model_with_square.fit(X_train, y_train)
y_pred_mlp_with_square = mlp_model_with_square.predict(X_test)

# Evaluation of MLP models
mse_mlp_original = mean_squared_error(y_test, y_pred_mlp_original)
r2_mlp_original = r2_score(y_test, y_pred_mlp_original)
mse_mlp_with_square = mean_squared_error(y_test, y_pred_mlp_with_square)
r2_mlp_with_square = r2_score(y_test, y_pred_mlp_with_square)
```

Побудувати точності та графіки для обох ознак:

Task 6

In [54]:
```
# Visualisation of results
sns.set_style('whitegrid')
plt.figure(figsize=(14, 7))

# Graph for the MLP model with the original feature
plt.subplot(1, 2, 1)
sns.scatterplot(x=X_test['Life expectancy '], y=y_test, color='blue', label='Actual Values')
sns.scatterplot(x=X_test['Life expectancy '], y=y_pred_mlp_original, color='red', label='Predictions (Original Feature)')
plt.title('MLP with Original Feature')
plt.xlabel('Life expectancy')
plt.ylabel('Schooling')
plt.legend()

# Graph for the MLP model with both features
plt.subplot(1, 2, 2)
sns.scatterplot(x=X_test['Life expectancy '], y=y_test, color='blue', label='Actual Values')
sns.scatterplot(x=X_test['Life expectancy '], y=y_pred_mlp_with_square, color='green', label='Predictions (Original + Squar
plt.title('MLP with Original + Squared Features')
plt.xlabel('Life expectancy')
plt.ylabel('Schooling')
plt.legend()

plt.tight_layout()
plt.show()

# Output the metrics
print('MLP with Original Feature - MSE:', mse_mlp_original, 'R2:', r2_mlp_original)
print('MLP with Original + Squared Features - MSE:', mse_mlp_with_square, 'R2:', r2_mlp_with_square)
```




```
MLP with Original Feature - MSE: 0.46403638165453887 R2: 0.5422556550584305
MLP with Original + Squared Features - MSE: 0.460009817193299 R2: 0.5462276218794446
```

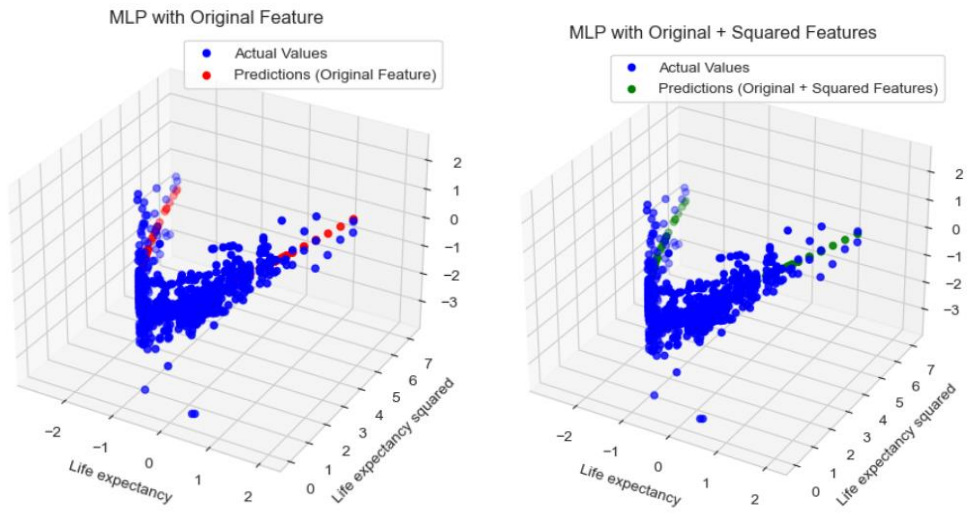
Графіки показують, що обидві моделі мають подібну точність прогнозування, що видно із схожих значень MSE та R^2. Це може вказувати на те, що додавання квадрату ознаки не призвело до значного покращення моделі у цьому конкретному випадку.

На графіках прогнозовані значення добре відповідають реальним даним для обох моделей, що свідчить про хорошу ефективність моделі Multi Layer Perceptron Regression для цього набору даних.

Візуалізація результату регресії з двома предикторами за допомогою mplot3d для кожної з регресійних моделей:

In [45]:
```
from mpl_toolkits.mplot3d import Axes3D
```

In [56]:
```
# create 3D graphics
fig = plt.figure(figsize=(14, 12))

# graph for the MLP model with the original feature
ax1 = fig.add_subplot(211, projection='3d')
ax1.scatter(X_test['Life expectancy '], X_test['Life expectancy squared'], y_test, color='blue', label='Actual Values')
ax1.scatter(X_test['Life expectancy '], X_test['Life expectancy squared'], y_pred_mlp_original, color='red', label='Predict
ax1.set_title('MLP with Original Feature')
ax1.set_xlabel('Life expectancy')
ax1.set_ylabel('Life expectancy squared')
ax1.set_zlabel('Schooling')
ax1.legend()

# graph for the MLP model with both features
ax2 = fig.add_subplot(212, projection='3d')
ax2.scatter(X_test['Life expectancy '], X_test['Life expectancy squared'], y_test, color='blue', label='Actual Values')
ax2.scatter(X_test['Life expectancy '], X_test['Life expectancy squared'], y_pred_mlp_with_square, color='green', label='Pr
ax2.set_title('MLP with Original + Squared Features')
ax2.set_xlabel('Life expectancy')
ax2.set_ylabel('Life expectancy squared')
ax2.set_zlabel('Schooling')
ax2.legend()

plt.show()
```



Висновок

Виконання лабораторної роботи "Аналіз даних за допомогою Python та бібліотеки Pandas" стало значним і цінним досвідом у моєму навчанні. Працюючи над цим завданням, я занурився у світ аналізу даних, відкрив для себе потужність Python і Pandas, та навчився використовувати ці інструменти для вирішення реальних задач.

Основні Аспекти та Виклики

Робота з Anaconda і IPython Notebook: На початковому етапі мені довелося ознайомитися з Anaconda та IPython Notebook. Це було нелегко, але вкрай корисно, оскільки ці інструменти значно спрощують процес розробки та тестування коду.

Вивчення Pandas: Pandas виявилась надзвичайно потужною бібліотекою для аналізу даних. Я зіткнувся з труднощами при обробці даних, при заповненні пропущених значень та обробці великих наборів даних, але з часом зрозумів, як ефективно використовувати її функції.

Аналіз і Візуалізація Даних: Одним з найбільш цікавих аспектів лабораторної роботи було вивчення різних методів візуалізації даних. Я використовував графіки, діаграми та теплові карти для представлення даних, що допомогло мені краще зрозуміти тенденції та закономірності в даних.

Особисті Враження та Досвід

Розвиток Аналітичних Навичок: Цей проект сприяв зміцненню моїх аналітичних здібностей, навчивши мене не лише технічним аспектам обробки даних, але й глибокому аналізу та інтерпретації результатів.

Проблеми та Виклики: В процесі роботи я стикався з численними труднощами, пов'язаними з обробкою даних. Розробка та тестування різних регресійних моделей вимагало від мене поглибленого розуміння математичних та статистичних принципів, що є важливим для будь-якого дослідника в галузі даних.

Лабораторна робота "Аналіз даних за допомогою Python та бібліотеки Pandas" ефективно демонструє застосування сучасних програмних інструментів для глибокого аналізу даних. Використання таких засобів як Anaconda, IPython Notebook, та Pandas підкреслює важливість інтеграції комп'ютерних наук і аналітичних методів в академічне дослідження.

Ця робота відзначається глибоким аналізом даних, використанням різноманітних методів візуалізації для інтерпретації результатів, а також застосуванням регресійних моделей для прогнозування, що засвідчує її високий науковий і методологічний рівень.

Посилання на джерела:

- https://www.anaconda.com/
- https://medium.com/velotio-perspectives/the-ultimate-beginners-guide-to-jupyter-notebooks-6b00846ed2af
- https://towardsdatascience.com/complete-guide-to-data-visualization-with-python-2dd74df12b5e
- https://dev.to/thalesbruno/subplotting-with-matplotlib-and-seaborn-5ei8
- https://numpy.org/doc/stable/reference/generated/numpy.corrcoef.html
- https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.corr.html
- https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html
- https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html
- https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html
- https://scikit-learn.org/stable/modules/model_evaluation.html
- https://matplotlib.org/stable/users/explain/toolkits/mplot3d.html