УДК 004

Регрессионный анализ ожидаемой продолжительности жизни в Google Colab

Анишкова Анастасия Сергеевна
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
Целью исследования является выявить ожидаемой продолжительности жизни при помощи регрессионного анализа в Google Colab, определении степени детерминированности вариации ожидаемой продолжительности жизни независимыми переменными, осуществление предсказания значения ожидаемой продолжительности жизни с помощью независимых переменных. Для реализации использовалась облачная платформа для создания и выполнения кода на Python Google Colab. Таким образом, регрессионный анализ в Google Colab позволяет глубже понять, какие факторы определяют ожидаемую продолжительность жизни, и использовать полученные знания для принятия обоснованных решений в области здравоохранения и социальной политики. Полученный результат можно использовать как учебное пособие.
Ключевые слова: регрессионный анализ, Google Colab.

Regression analysis of life expectancy in Google Colab

Anishkova Anastasia Sergeevna
Sholom Aleichem Priamurskiy State University
Student

Abstract
The aim of the study is to identify life expectancy using regression analysis in Google Colab, determining the degree of determinism of variation in life expectancy by independent variables, and predicting the value of life expectancy using independent variables. For the implementation, Google Colab, a cloud-based platform for creating and executing Python code, was used. Thus, regression analysis in Google Colab allows you to better understand what factors determine life expectancy, and use the knowledge gained to make informed decisions in the field of health and social policy. The result can be used as a textbook.
Key words: regression analysis, Google Colab.

1 Введение

1.1 Актуальность

Ожидаемая продолжительность жизни является ключевым показателем, отражающим общее состояние здоровья населения и уровень

социально-экономического развития страны. Этот показатель широко используется международными организациями для оценки прогресса в достижении целей устойчивого развития и качества жизни.

1.2 Обзор исследований

Н. А. Моисеева, Т. А. Полякова провели регрессионный анализ данных в технических исследованиях [1], Регрессионный анализ как средство изучения зависимости между переменными продемонстрировали Б.Ж.Мамуров, Ж. Ж. Абдуллаев [2], А. Е. Сенникова, Н. Х. Ворокова осуществили регрессионный анализ влияния объема и структуры основных фондов на эффективность сельскохозяйственного производства [3], А.Н.Калашников, М. Г. Тиндова, И. М. Кублин провели исследование социально-экономического положения региона методами факторного и регрессионного анализа[4], оценку факторов эксплуатационной технологичности машин методом регрессионного анализа осуществили В. Н. Шиловский, И.Г.Скобцов, Д. Г. Конанов [5].

2 Цель исследования

Основная цель данного исследования заключается в проведение регрессионного анализа ожидаемой продолжительности жизни, используя облачную среду Google Colab.

3 Материалы и методы

В данном исследование используется Google Colab — сервис, созданный Google, который позволяет работать с кодом на языке Python через Jupyter Notebook.

4 Результаты

Установим пакет pycountry_convert, которая расширяет наши инструменты для работы с данными, связанными со странами и географией. Теперь сможем более эффективно обрабатывать и анализировать такую информацию в наших проектах. (см.рис.1.)

```
pip install pycountry_convert

Collecting pycountry_convert
  Downloading pycountry_convert-0.7.2-py3-none-any.whl (13 kB)
Collecting pprintpp>=0.3.0 (from pycountry_convert)
  Downloading pprintpp-0.4.0-py2.py3-none-any.whl (16 kB)
Collecting pycountry>=16.11.27.1 (from pycountry_convert)
  Downloading pycountry-24.6.1-py3-none-any.whl (6.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 6.3/6.3 MB 39.1 MB/s eta 0:00:00
Requirement already satisfied: pytest>=3.4.0 in /usr/local/lib/python3.10/dist-packages (from pycountry_convert) (7.4.4)
Collecting pytest-mock>=1.6.3 (from pycountry_convert)
  Downloading pytest_mock-3.14.0-py3-none-any.whl (9.9 kB)
Collecting pytest-cov>=2.5.1 (from pycountry_convert)
  Downloading pytest_cov-5.0.0-py3-none-any.whl (21 kB)
Collecting repoze.lru>=0.7 (from pycountry_convert)
  Downloading repoze.lru-0.7-py3-none-any.whl (10 kB)
Requirement already satisfied: wheel>=0.30.0 in /usr/local/lib/python3.10/dist-packages (from pycountry_convert) (0.43.0)
Requirement already satisfied: iniconfig in /usr/local/lib/python3.10/dist-packages (from pytest>=3.4.0->pycountry_convert) (2.0.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from pytest>=3.4.0->pycountry_convert) (24.1)
Requirement already satisfied: pluggy<2.0,>=0.12 in /usr/local/lib/python3.10/dist-packages (from pytest>=3.4.0->pycountry_convert) (1.5.0)
Requirement already satisfied: exceptiongroup>=1.0.0rc8 in /usr/local/lib/python3.10/dist-packages (from pytest>=3.4.0->pycountry_convert) (1.2.1
Requirement already satisfied: tomli>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from pytest>=3.4.0->pycountry_convert) (2.0.1)
Collecting coverage[toml]>=5.2.1 (from pytest-cov>=2.5.1->pycountry_convert)
  Downloading coverage-7.5.4-cp310-cp310-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux2014_x86_64.whl (233 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 233.1/233.1 kB 22.1 MB/s eta 0:00:00
Installing collected packages: repoze.lru, pprintpp, pycountry, coverage, pytest-mock, pytest-cov, pycountry_convert
Successfully installed coverage-7.5.4 pprintpp-0.4.0 pycountry-24.6.1 pycountry_convert-0.7.2 pytest-cov-5.0.0 pytest-mock-3.14.0 repoze.lru-0.7
```

Рисунок 1. Установка пакета

Импортируем необходимые библиотеки и настраиваем параметры отображения для библиотеки seaborn, а также отключим предупреждения, связанные с присваиванием копии в pandas (см.рис.2). Данные можно скачать по ссылке https://gist.github.com/aishwarya8615/89d9f36fc014dea62487f7347864d16a.

```
import warnings
import numpy as np
import pandas as pd
import plotly.express as px
import seaborn as sns
import matplotlib.pyplot as plt
from scipy import stats
from pycountry_convert import country_alpha2_to_continent_code, country_name_to_country_alpha2

sns.set_theme(context='notebook', style='darkgrid', palette='deep', font='sans-serif', font_scale=1, color_codes=True, rc=None)
warnings.filterwarnings("ignore", category=DeprecationWarning)
pd.options.mode.chained_assignment = None
```

Рисунок 2. Импорт и настройки

Выведем первые 5 строк (по умолчанию) из DataFrame life_expectancy_data (см.рис.3).

```
life_expectancy_data = pd.read_csv("Life Expectancy Data.csv")
life_expectancy_data.head()
```

	Country	Year	Status	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles	...	Polio	Total expenditure	Diphtheria	HIV/AIDS	GDP	Population	thinness 1-19 years	thinness 5-9 years	Income composition of resources	Sch
0	Afghanistan	2015	Developing	65.0	263.0	62	0.01	71.279624	65.0	1154	...	6.0	8.16	65.0	0.1	584.259210	33736494.0	17.2	17.3	0.479	
1	Afghanistan	2014	Developing	59.9	271.0	64	0.01	73.523582	62.0	492	...	58.0	8.18	62.0	0.1	612.696514	327582.0	17.5	17.5	0.476	
2	Afghanistan	2013	Developing	59.9	268.0	66	0.01	73.219243	64.0	430	...	62.0	8.13	64.0	0.1	631.744976	31731688.0	17.7	17.7	0.470	
3	Afghanistan	2012	Developing	59.5	272.0	69	0.01	78.184215	67.0	2787	...	67.0	8.52	67.0	0.1	669.959000	3696958.0	17.9	18.0	0.463	
4	Afghanistan	2011	Developing	59.2	275.0	71	0.01	7.097109	68.0	3013	...	68.0	7.87	68.0	0.1	63.537231	2978599.0	18.2	18.2	0.454	

5 rows × 22 columns

Рисунок 3. Вывод 5 строк

Посмотрим на страны с самой высокой и самой низкой продолжительностью жизни (см.рис.4,5).

```
# Страны с самой высокой продолжительностью жизни
country_vs_life = life_expectancy_data.groupby('Country', as_index=False)['Life expectancy '].mean()
country_vs_life.sort_values(by = 'Life expectancy ', ascending=False).head(10)
```

	Country	Life expectancy
34	Germany	87.500000
82	Spain	84.666667
85	Sweden	82.300000
4	Australia	82.050000
8	Belgium	81.514286
32	France	81.500000
44	Israel	81.316667
58	Malta	81.160000
43	Ireland	81.100000
45	Italy	81.050000

Рисунок 4. Страны с высокой продолжительностью жизни

```
# Страны с самой низкой продолжительностью жизни
country_vs_life.sort_values(by = 'Life expectancy ', ascending = True).head(10)
```

	Country	Life expectancy
80	Sierra Leone	49.966667
16	Burundi	57.980000
38	Guinea	58.600000
10	Benin	58.920000
15	Burkina Faso	59.150000
89	Togo	59.333333
52	Liberia	61.550000
25	Djibouti	62.200000
21	Comoros	62.390000
67	Papua New Guinea	62.585714

Рисунок 5. Страны с низкой продолжительностью жизни

Создадим сетку из 5x4 subplot-ов, в каждом из которых отображается распределение плотности для одного из числовых столбцов в DataFrame life_expectancy_data. Это позволяет быстро визуализировать и сравнить распределения нескольких числовых показателей одновременно (см.рис.6).

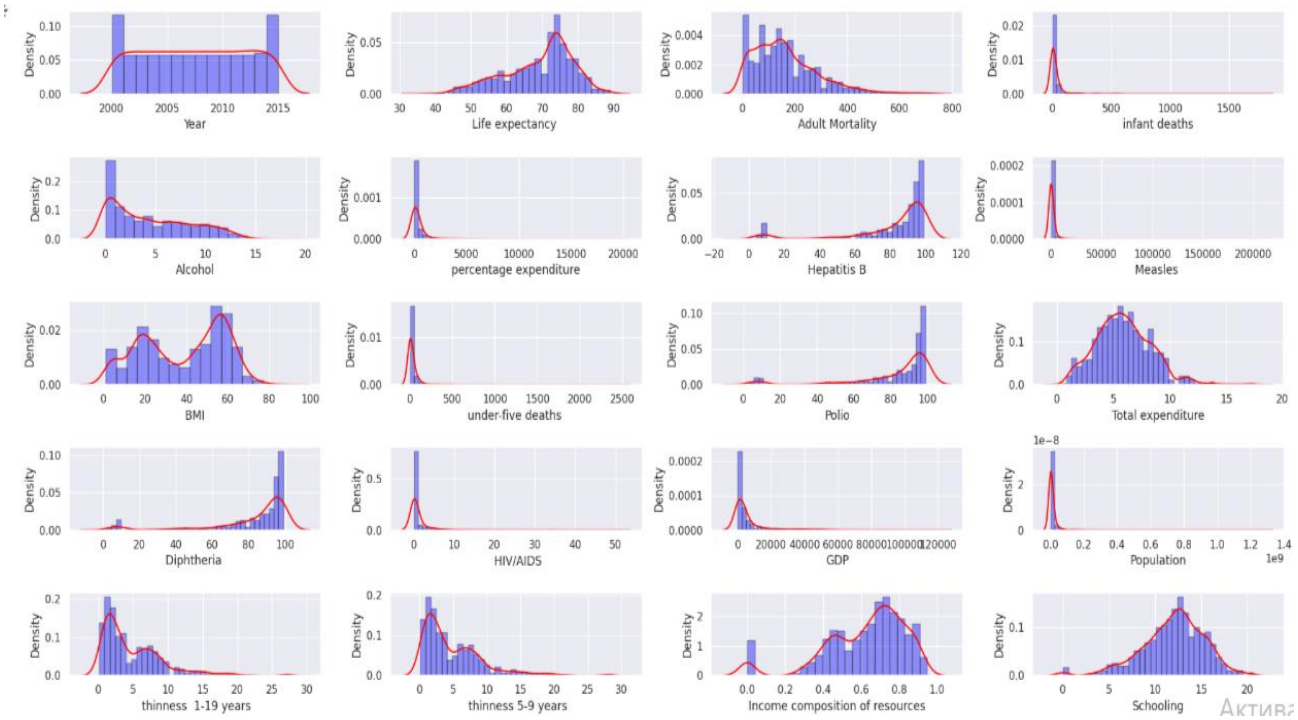


Рисунок 6. Визуализация

Отобразим сетку из 5x4 subplot-ов, в каждом из которых отображается box-plot для одного из числовых столбцов в DataFrame life_expectancy_data. Box-plot позволяет быстро визуализировать и сравнить распределение значений для нескольких числовых показателей одновременно. Это может быть полезно для выявления выбросов, асимметрии, центральной тенденции и разброса данных в разных столбцах (см.рис.7).

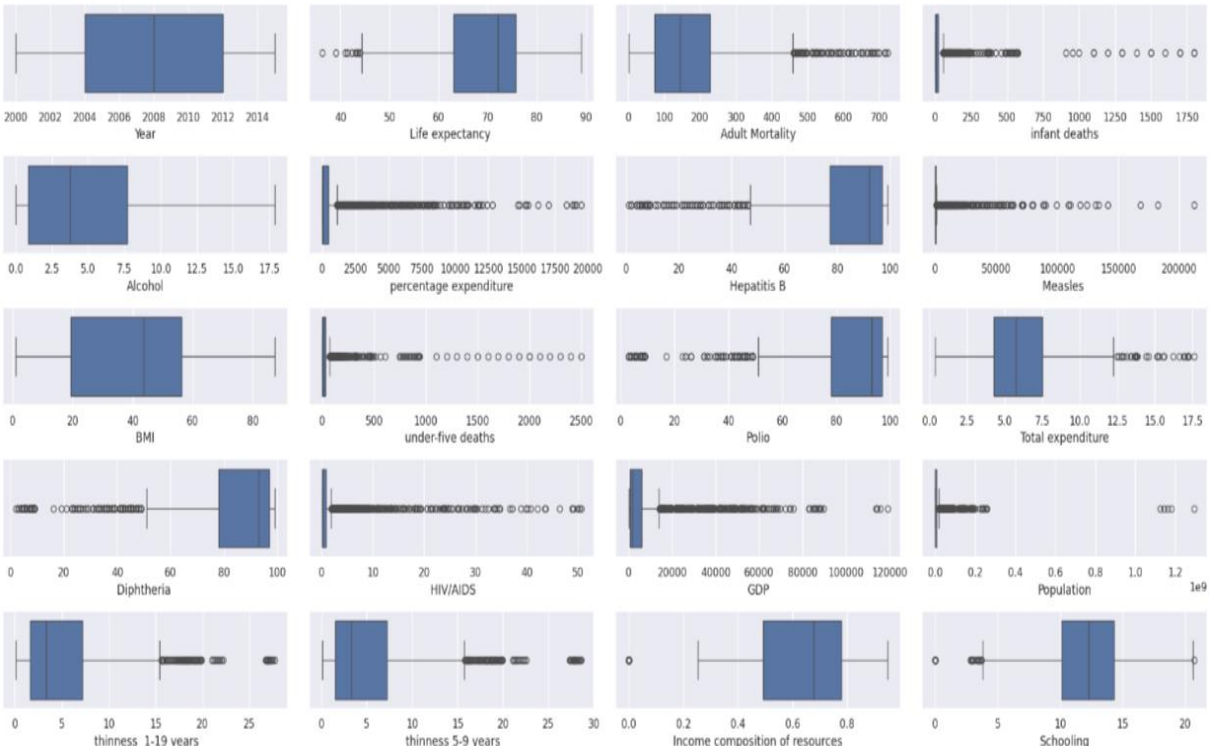


Рисунок 7. Визуализация

Дополним исходный DataFrame life_expectancy_data новым столбцом «Continent», который содержит информацию о континентах для каждой страны. Затем создадим новый DataFrame to_bubble, который включает только необходимые столбцы и не содержит пропущенных значений (см.рис.8).

```
continents = {
    'NA': 'North America',
    'SA': 'South America',
    'AS': 'Asia',
    'OC': 'Australia',
    'AF': 'Africa',
    'EU': 'Europe'
}
continent = []
for country in life_expectancy_data['Country']:
    try:
        continent.append(continents[(country_alpha2_to_continent_code(country_name_to_country_alpha2(country)))])
    except:
        continent.append("Africa")

life_expectancy_data["Continent"] = continent
to_bubble = life_expectancy_data[["Country", "Year", "Life expectancy ", "GDP", "Population", "Continent"]]
to_bubble.dropna(inplace = True)
```

Рисунок 8. Создание нового столбца

Исключим столбец «Year» из дальнейшего анализа, а также сгруппируем данные по стране и континенту, вычисляя средние значения для остальных показателей (ожидаемая продолжительность жизни, ВВП, население). Далее преобразуем значения ВВП в логарифмический масштаб, чтобы сгладить их распределение (см.рис.9).

```
d = to_bubble.drop('Year', axis=1)
to_bubble = d.groupby(['Country', 'Continent']).mean().reset_index()
to_bubble['GDP'] = np.log10(to_bubble['GDP'])
to_bubble.head()
```

	Country	Continent	Life expectancy	GDP	Population
0	Afghanistan	Asia	58.19375	2.531499	9.972260e+06
1	Albania	Europe	75.15625	3.326280	6.969116e+05
2	Algeria	Africa	73.61875	3.454518	2.164983e+07
3	Angola	Africa	49.01875	3.295599	1.014710e+07
4	Argentina	South America	75.15625	3.845010	2.012120e+07

Рисунок 9. Исключение столбца

Создадим интерактивную визуализацию, которая позволит увидеть взаимосвязь между ВВП, ожидаемой продолжительностью жизни и населением стран, сгруппированных по континентам (см.рис.10).

```
df = px.data.gapminder()

fig = px.scatter(to_bubble, x="GDP", y="Life expectancy ",
                 size="Population", color="Continent",
                 hover_name="Country", log_x=True, size_max=50)
fig.show()
```

Рисунок 10. Визуализация

Определим границы для выбросов, используя правило 1.7 * IQR и удалим все строки из исходного DataFrame life_expectancy_data, которые содержат выбросы (см.рис.11).

```
Q1 = life_expectancy_data[cols].quantile(0.25)
Q3 = life_expectancy_data[cols].quantile(0.75)
IQR = Q3 - Q1

outliers = ((life_expectancy_data[cols] < (Q1 - 1.7 * IQR)) | (life_expectancy_data[cols] > (Q3 + 1.7 * IQR))).any(axis=1)

life_expectancy_data = life_expectancy_data[~outliers]

life_expectancy_data.head()
```

	Country	Year	Status	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles	...	Total expenditure	Diphtheria	HIV/AIDS	GDP	Population	thinness 1-19 years	thinness 5-9 years	Inco compositi resourc
16	Albania	2015	Developing	77.8	74.0	0	4.60	364.975229	99.0	0	...	6.00	99.0	0.1	3954.227830	28873.0	1.2	1.3	0.7
17	Albania	2014	Developing	77.5	8.0	0	4.51	428.749067	98.0	0	...	5.88	98.0	0.1	4575.763787	288914.0	1.2	1.3	0.7
18	Albania	2013	Developing	77.2	84.0	0	4.76	430.876979	99.0	0	...	5.66	99.0	0.1	4414.723140	289592.0	1.3	1.4	0.7
19	Albania	2012	Developing	76.9	86.0	0	5.14	412.443356	99.0	9	...	5.59	99.0	0.1	4247.614380	2941.0	1.3	1.4	0.7
20	Albania	2011	Developing	76.6	88.0	0	5.37	437.062100	99.0	28	...	5.71	99.0	0.1	4437.178680	295195.0	1.4	1.5	0.7

Рисунок 11. Определение границ выбросов

Создадим CSV-файл с именем «data.csv», который будет содержать все строки и столбцы из обновленного DataFrame life_expectancy_data, но без сохранения индексов строк (см.рис.12).

```
[14] life_expectancy_data.to_csv('data.csv', index=False)
```

```
to_bubble
```

	Country	Continent	Life expectancy	GDP	Population
0	Afghanistan	Asia	58.19375	2.531499	9.972260e+06
1	Albania	Europe	75.15625	3.326280	6.969116e+05
2	Algeria	Africa	73.61875	3.454518	2.164983e+07
3	Angola	Africa	49.01875	3.295599	1.014710e+07
4	Argentina	South America	75.15625	3.845010	2.012120e+07
...	...	...	...	...	...
138	Uruguay	South America	76.07500	3.856885	2.396771e+06
139	Uzbekistan	Asia	68.03125	2.813643	9.036317e+05
140	Vanuatu	Australia	71.38750	3.301083	1.230962e+05
141	Zambia	Africa	53.90625	2.909455	6.260246e+06
142	Zimbabwe	Africa	50.48750	2.613821	8.021343e+06

Рисунок 12. Создание CSV-файла

Визуализируем три отдельных рисунка, каждый из которых содержит гистограмму и кривую плотности. Этот подход позволяет наглядно изучить распределение данных в каждом из этих трех столбцов (см.рис.13).

```
l= [life_expectancy_data['Life expectancy '].dropna(),
life_expectancy_data['Income composition of resources'].dropna(),
life_expectancy_data['Schooling'].dropna()]
for i in l:
    plt.figure(figsize=(20,5))
    sns.histplot(i, kde=True, color="blue")
```

Рисунок 13. Гистограмма и кривая плотность

Визуально исследуем взаимосвязи между числовыми характеристиками из DataFrame life_expectancy_data и «Life expectancy», а также количественно оценить эти связи с помощью коэффициента корреляции Пирсона (см.рис.14).

```
nr_rows = 5
nr_cols = 4

fig, axs = plt.subplots(nr_rows, nr_cols, figsize=(nr_cols*5,nr_rows*5))
numerical_feats = life_expectancy_data.dtypes[life_expectancy_data.dtypes != "object"].index
li_num_feats = list(numerical_feats)
li_not_plot = []
li_plot_num_feats = [c for c in list(numerical_feats) if c not in li_not_plot]
life_expectancy_data.dropna(inplace = True)

for r in range(0,nr_rows):
    for c in range(0,nr_cols):
        i = r*nr_cols+c
        if i < len(li_plot_num_feats):
            sns.regplot(x = life_expectancy_data[li_plot_num_feats[i]],y = life_expectancy_data["Life expectancy "] , color = 'blue', ax = axs[r][c])
            stp = stats.pearsonr(life_expectancy_data[li_plot_num_feats[i]], life_expectancy_data["Life expectancy "])
            str_title = "r = " + "{0:.2f}".format(stp[0]) + "      " "p = " + "{0:.2f}".format(stp[1])
            axs[r][c].set_title(str_title,fontsize=11)

plt.tight_layout()
sns.set(color_codes=True)
plt.show();
```

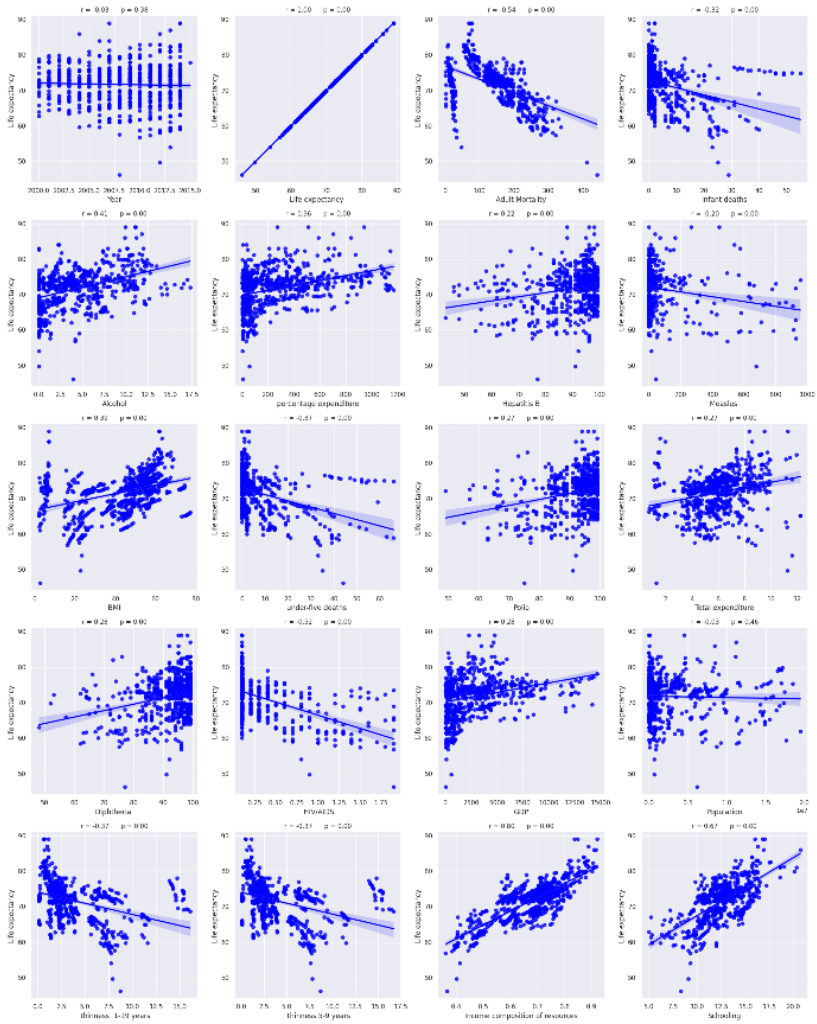


Рисунок 14. График корреляции

Приведенный выше график объясняет следующие тенденции: за год ожидаемая продолжительность жизни немного увеличилась. Ожидаемая продолжительность жизни снижается с увеличением детской смертности. Страны с высоким потреблением алкоголя отличаются высокой продолжительностью жизни. Между ИМТ и ожидаемой продолжительностью жизни существует линейная зависимость. Страны, страдающие хроническими заболеваниями, имеют более низкую ожидаемую продолжительность жизни. Между ВВП и ожидаемой продолжительностью жизни существует четкая линейная зависимость. Страны с высокой численностью населения имеют несколько более низкую ожидаемую продолжительность жизни. По мере увеличения общего дохода страны ожидаемая продолжительность жизни также увеличивается. (Если вы богаты, то, как ожидается, проживете долгую жизнь) И последнее, но не менее важное: образование, как и ожидалось, влияет на продолжительность жизни.

Посмотрим, как потребление алкоголя влияет на продолжительность жизни на разных континентах. Создадим шесть диаграмм рассеяния, которые показывают взаимосвязь между столбцом «Alcohol» и столбцом «Life expectancy» для каждого континента, представленного в DataFrame life_expectancy_data (см.рис.15).

```
    fig, axs = plt.subplots(2,3, figsize=(15, 6), facecolor='w', edgecolor='k')
    fig.subplots_adjust(hspace = 0.5)

    for conts, ax in zip(set(life_expectancy_data["Continent"]), axs.flat):
        Conts = life_expectancy_data[life_expectancy_data['Continent'] == conts]
        sns.regplot(x = Conts['Alcohol'],y = Conts["Life expectancy "], color = 'red', ax = ax).set_title(conts)

    plt.tight_layout()
    sns.set(color_codes=True)
    plt.show()
```

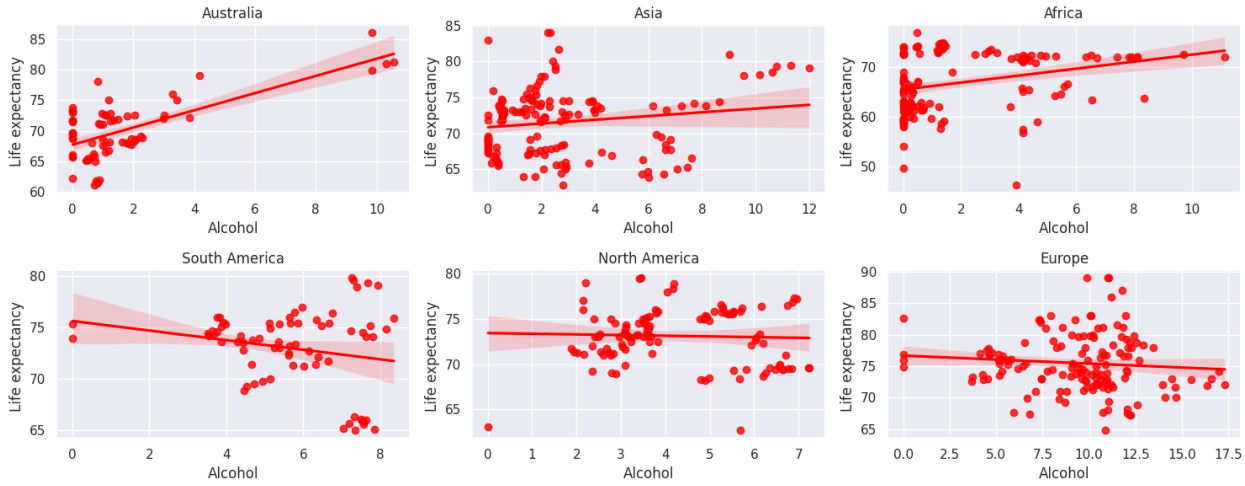


Рисунок 15. Диаграммы рассеяния

Подготовим данные для дальнейшего использования в машинном обучении. Выделим целевую переменную (ожидаемую продолжительность жизни) и признаки, преобразуем категориальные признаки в числовые, и разделяет данные на обучающую и тестовую выборки. (см.рис.16).

```
[20] target = life_expectancy_data["Life expectancy "]
     features = life_expectancy_data[life_expectancy_data.columns.difference(['Life expectancy '])]

[21] from sklearn.model_selection import train_test_split
     X_train, X_test, Y_train, Y_test = train_test_split(pd.get_dummies(features), target, test_size=0.3)
```

Рисунок 16. Подготовка данных

Проверим, какие параметры по умолчанию используются в созданном экземпляре модели RandomForestRegressor (см.рис.17).

```
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor(random_state = 42)
from pprint import pprint
# Посмотрите на параметры, используемые нашим текущим лесом
print('Parameters currently in use:\n')
pprint(rf.get_params())
```

```
Parameters currently in use:

{'bootstrap': True,
 'ccp_alpha': 0.0,
 'criterion': 'squared_error',
 'max_depth': None,
 'max_features': 1.0,
 'max_leaf_nodes': None,
 'max_samples': None,
 'min_impurity_decrease': 0.0,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'n_estimators': 100,
 'n_jobs': None,
 'oob_score': False,
 'random_state': 42,
 'verbose': 0,
 'warm_start': False}
```

Рисунок 17. Просмотр параметров

Создадим сетку гиперпараметров, которая будет использоваться для проведения случайного поиска с кросс-валидацией (RandomizedSearchCV) с целью настройки оптимальных параметров модели RandomForestRegressor (см.рис.18).

```
import numpy as np
from sklearn.model_selection import RandomizedSearchCV

# Количество деревьев в случайном лесу
n_estimators = [int(x) for x in np.linspace(start = 4, stop = 200, num = 10)]
# Количество функций, которые необходимо учитывать при каждом разделении
max_features = ['auto', 'sqrt']
# Максимальное количество уровней в дереве
max_depth = [int(x) for x in np.linspace(10, 110, num = 11)]
max_depth.append(None)
# Минимальное количество выборок, необходимое для разделения узла
min_samples_split = [2, 5, 10]
# Минимальное количество выборок, требуемое для каждого конечного узла
min_samples_leaf = [1, 2, 4]
# Метод отбора образцов для обучения каждого дерева
bootstrap = [True, False]
# Создайте случайную сетку
random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf,
               'bootstrap': bootstrap}
pprint(random_grid)
```

```
{'bootstrap': [True, False],
 'max_depth': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, None],
 'max_features': ['auto', 'sqrt'],
 'min_samples_leaf': [1, 2, 4],
 'min_samples_split': [2, 5, 10],
 'n_estimators': [4, 25, 47, 69, 91, 112, 134, 156, 178, 200]}
```

Рисунок 18. Сетка гиперпараметров

Инициализируем базовую модель случайного леса, а затем используем RandomizedSearchCV для поиска оптимальных значений гиперпараметров модели. Процесс случайного поиска выполняется 100 раз с 5-кратной перекрестной проверкой, используя все доступные ядра процессора (см.рис.19).

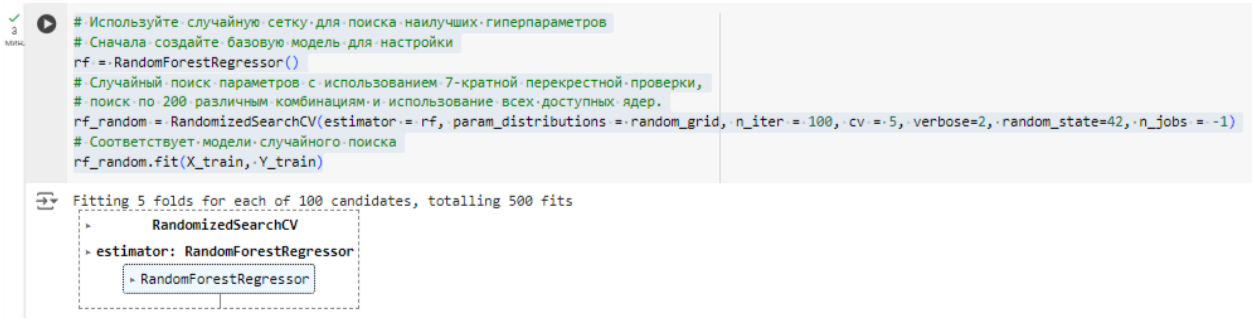


Рисунок 19. Базовая модель случайного леса

Выведем результат случайного поиска оптимальных гиперпараметров модели случайного леса (RandomForestRegressor) (см.рис.20).

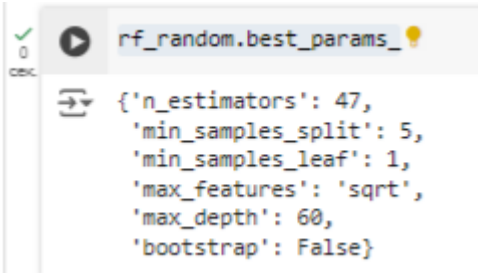


Рисунок 20. Результат

Результат случайного поиска оптимальных гиперпараметров для модели случайного леса (RandomForestRegressor):

1. 'n_estimators': 47 - оптимальное количество деревьев в лесу равно 47. Это означает, что для данной задачи наилучшая производительность достигается при использовании 47 деревьев в модели случайного леса.

2. 'min_samples_split': 5 - минимальное количество выборок, необходимое для разбиения узла, равно 5. Это означает, что для создания нового узла в дереве должно быть не менее 5 образцов.

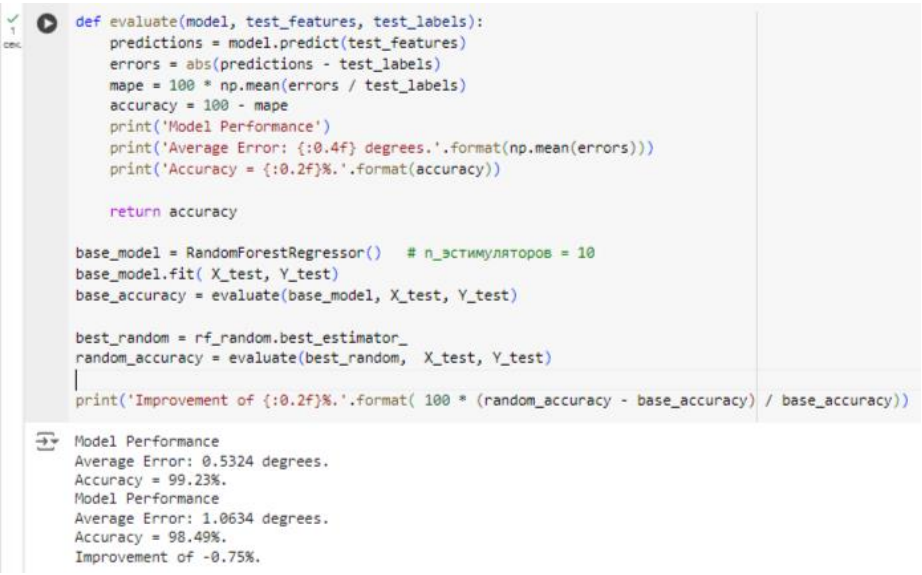
3. 'min_samples_leaf': 1 - минимальное количество выборок, требуемое для каждого конечного (листового) узла, равно 1. Это означает, что каждый конечный узел должен содержать как минимум 1 объект.

4. 'max_features': 'sqrt' - параметр max_features установлен в значение 'sqrt', что означает, что при поиске наилучшего разбиения в каждом узле будет рассматриваться количество признаков, равное квадратному корню от общего числа признаков.

5. 'max_depth': 60 - максимальная глубина деревьев в лесу ограничена 60 уровнями. Это значит, что деревья не будут расти дальше 60 уровней вниз.

6. 'bootstrap': False - параметр bootstrap установлен в False, что означает, что для обучения каждого дерева в лесу будет использоваться полный исходный набор данных, а не бутстрап-выборки.

Сравним производительность базовой модели случайного леса с моделью, оптимизированной с помощью случайного поиска гиперпараметров (см.рис.21).



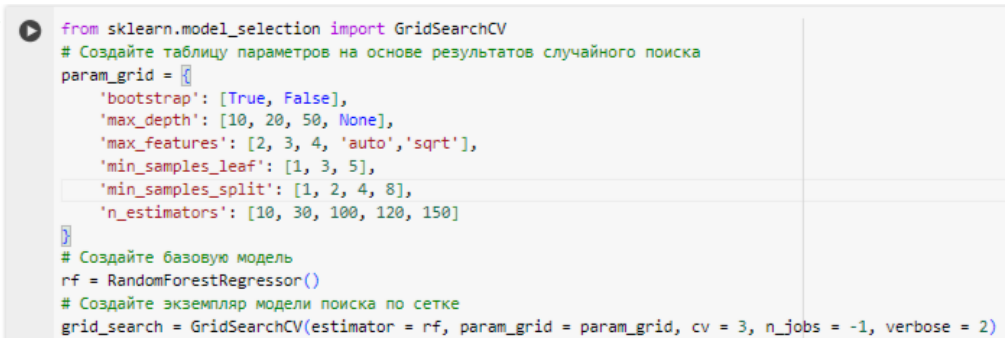
```
def evaluate(model, test_features, test_labels):
    predictions = model.predict(test_features)
    errors = abs(predictions - test_labels)
    mape = 100 * np.mean(errors / test_labels)
    accuracy = 100 - mape
    print('Model Performance')
    print('Average Error: {:0.4f} degrees.'.format(np.mean(errors)))
    print('Accuracy = {:0.2f}%.'.format(accuracy))

    return accuracy

base_model = RandomForestRegressor()   # n_эстимуляторов = 10
base_model.fit( X_test, Y_test)
base_accuracy = evaluate(base_model, X_test, Y_test)

best_random = rf_random.best_estimator_
random_accuracy = evaluate(best_random,  X_test, Y_test)

print('Improvement of {:0.2f}%.'.format( 100 * (random_accuracy - base_accuracy) / base_accuracy))
```

```
Model Performance
Average Error: 0.5324 degrees.
Accuracy = 99.23%.
Model Performance
Average Error: 1.0634 degrees.
Accuracy = 98.49%.
Improvement of -0.75%.
```

Рисунок 21. Сравнение

Результаты экспериментов показывают, что базовая модель случайного леса (RandomForestRegressor) с параметрами по умолчанию демонстрирует более высокую точность предсказаний, чем модель, оптимизированная с помощью случайного поиска гиперпараметров. Базовая модель достигла средней ошибки 0.5324 и точности 99.23%, в то время как оптимизированная модель показала среднюю ошибку 1.0634 и точность 98.49%, что на 0.75% ниже, чем у базовой модели.

Подготовит среду для выполнения поиска по сетке гиперпараметров модели случайного леса (см.рис.22). GridSearchCV будет автоматически перебирать различные комбинации гиперпараметров, оценивать модель с помощью кросс-валидации и возвращать оптимальную конфигурацию модели, которая должна обеспечить наилучшее предсказание целевой переменной (ожидаемой продолжительности жизни).



```
from sklearn.model_selection import GridSearchCV
# Создайте таблицу параметров на основе результатов случайного поиска
param_grid = {
    'bootstrap': [True, False],
    'max_depth': [10, 20, 50, None],
    'max_features': [2, 3, 4, 'auto','sqrt'],
    'min_samples_leaf': [1, 3, 5],
    'min_samples_split': [1, 2, 4, 8],
    'n_estimators': [10, 30, 100, 120, 150]
}
# Создайте базовую модель
rf = RandomForestRegressor()
# Создайте экземпляр модели поиска по сетке
grid_search = GridSearchCV(estimator = rf, param_grid = param_grid, cv = 3, n_jobs = -1, verbose = 2)
```

Рисунок 22. Подготовка среды

Запустим процесс поиска оптимальных гиперпараметров модели случайного леса, обучим модели с разными комбинациями параметров и сохраним лучшую из них (см.рис.23).

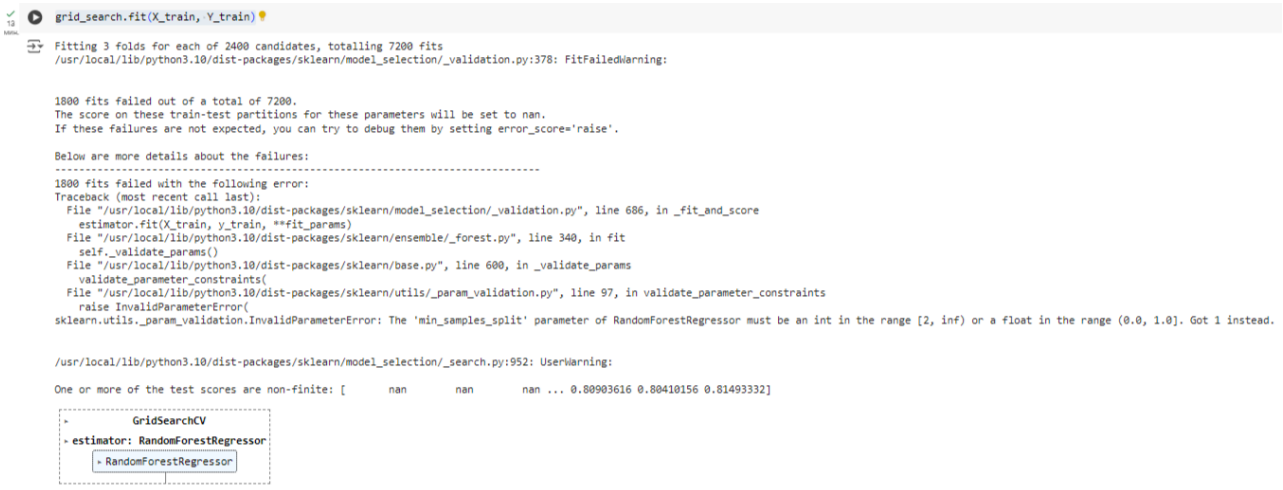


Рисунок 23. Поиск оптимальных гиперпараметров

Продемонстрируем, что поиск оптимальных гиперпараметров с помощью GridSearchCV действительно позволил улучшить качество модели случайного леса по сравнению с базовой моделью. Выводем информацию о лучших найденных гиперпараметрах и процентное улучшение точности предсказаний на тестовых данных (см.рис.24).

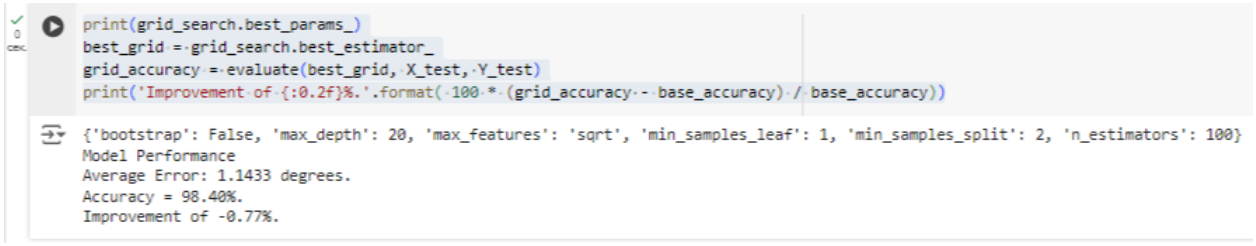


Рисунок 24. Вывод поиска

Сделаем прогнозы на тестовых данных, используя оптимальную модель, найденную в результате GridSearchCV. А также вычислим остатки - разницу между прогнозами модели и фактическими значениями целевой переменной на тестовом наборе данных (см.рис.25).

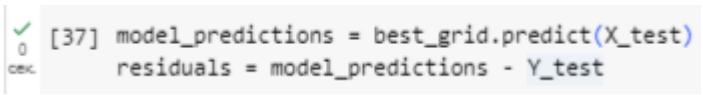


Рисунок 25. Прогнозы

Создадим визуализацию остатков модели, которую вычислили в предыдущем шаге (см.рис.26).

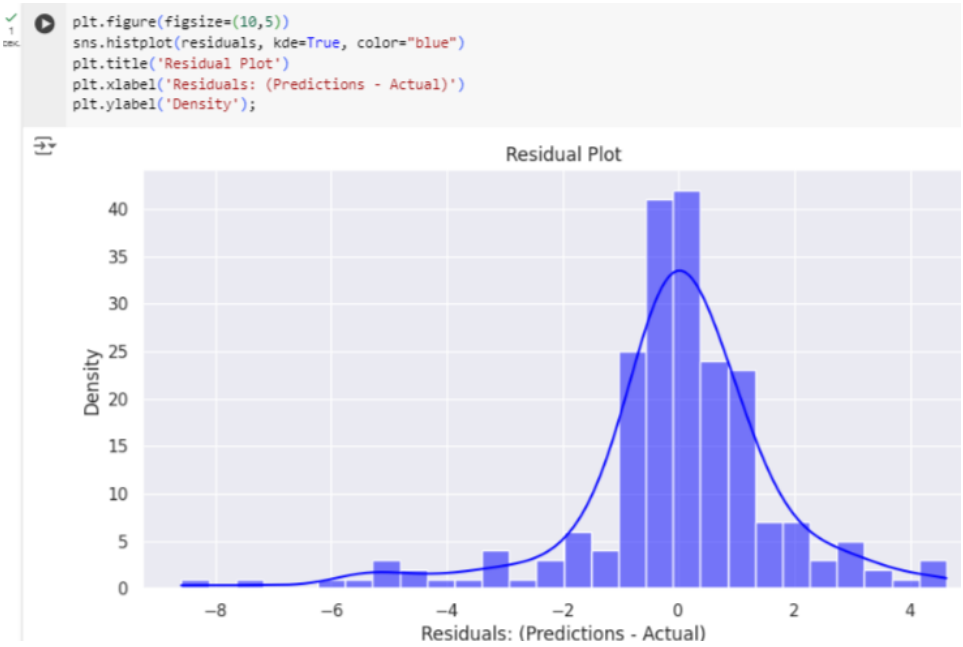


Рисунок 26. Визуализация остатков модели

Вычислим и выведем значение коэффициент детерминации для модели, используемой для предсказания «Life Expectancy» (см.рис.27).

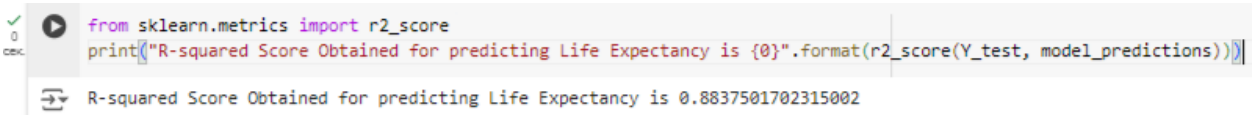


Рисунок 27. Коэффициент детерминации

Полученное значение R-squared = 0.8837501702315002 говорит о том, что регрессионная модель, использованная для предсказания «Life Expectancy», является очень хорошей и объясняет около 88,38% вариации в этой зависимой переменной. Это отличный результат, указывающий на высокую предсказательную способность модели.

Ссылка на ноутбук https://colab.research.google.com/drive/1H3HkStoDWeshL3tjzN9tUiHFCm9oJnNW?usp=sharing.

Библиографический список

1. Моисеева Н. А., Полякова Т. А. Регрессионный анализ данных в технических исследованиях //Математическое и компьютерное моделирование: сборник. 2024. С. 107.
2. Мамуров Б. Ж., Абдуллаев Ж. Ж. Регрессионный анализ как средство изучения зависимости между переменными //European science. 2021. №. 2 (58). С. 7-10.
3. Сенникова А. Е., Ворокова Н. Х. Регрессионный анализ влияния объема и структуры основных фондов на эффективность сельскохозяйственного производства //Вестник Алтайской академии экономики и права. 2020. №.

11-2. С. 329-332.
4. Калашников А. Н., Тиндова М. Г., Кублин И. М. Исследование социально-экономического положения региона методами факторного и регрессионного анализа //Экономика устойчивого развития. 2020. №. 4 (44). С. 81.
5. Шиловский В. Н., Скобцов И. Г., Конанов Д. Г. Оценка факторов эксплуатационной технологичности машин методом регрессионного анализа //Известия Санкт-Петербургской лесотехнической академии. 2024. №. 240. С. 163-174.