Федеральное государственное бюджетное образовательное учреждение
высшего образования «Московский Авиационный Институт»
(Национальный Исследовательский Университет)
Институт № 8
«Информационные технологии и прикладная математика»

Реферат по курсу «Практикум на ЭВМ»
Семестр 2
«Анализ данных: Pandas, NumPy, Seaborn»

Студент: Хайруллина Ясмин Алмазовна
Группа: М8О-103Б-21
Руководитель: Севастьянов Виктор Сергеевич
Дата сдачи:

Москва, 2022

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Если вы занимаетесь спортом, то наверняка знаете о бейсбольной статистике Moneyball и революции в профессиональном бейсболе, которую позволил совершить анализ данных об эффективности действий отдельных игроков. Если вы увлекаетесь сетевыми компьютерными играми, то наверняка знаете, что разнообразные сведения о вашем игровом поведении накапливают и анализируют компании Zynga и Electronic Arts. Любите кино? Возможно, слышали о методике, применяемой компанией Netflix для прогнозирования предпочтений в области кино. Может быть, вы не знаете, что некоторые голливудские киностудии (например, Relativity Media) используют похожие методики, принимая решение о том, какие кинопроекты финансировать.

Стремитесь ли вы создать дополнительную ценность для потребителей или оптимизировать принимаемые решения – все это задачи для аналитиков. И, учитывая рост объема информации почти в каждой сфере общественной деятельности, можно без труда прийти к заключению о том, насколько актуальна в наше время данная тема.

КРАТКО ОБ АНАЛИЗЕ ДАННЫХ

Анализ данных — это область математики и информатики, занимающаяся построением и исследованием наиболее общих математических методов и вычислительных алгоритмов извлечения знаний из экспериментальных (в широком смысле) данных; процесс исследования, фильтрации, преобразования и моделирования данных с целью извлечения полезной информации и принятия решений.

Говоря чуть более простым языком, я бы предложил понимать под анализом данных совокупность методов и приложений, связанных с алгоритмами обработки данных и не имеющих четко зафиксированного ответа на каждый входящий объект.

Анализ данных можно описать как процесс, состоящий из нескольких шагов, в которых сырые данные превращаются и обрабатываются с целью создать визуализации и сделать предсказания на основе математической модели.

Анализ данных — это всего лишь последовательность шагов, каждый из которых играет ключевую роль для последующих. Этот процесс похож на цепь последовательных, связанных между собой этапов:

1. Определение проблемы;
2. Извлечение данных;
3. Подготовка данных — очистка данных;
4. Подготовка данных — преобразование данных;
5. Исследование и визуализация данных;
6. Предсказательная модель;
7. Проверка модели, тестирование;
8. Развертывание — визуализация и интерпретация результатов;
9. Развертывание — развертывание решения.

ЯЗЫК ПРОГРАММИРОВАНИЯ PYTHON ДЛЯ АНАЛИЗА ДАННЫХ

Существуют различные инструменты для анализа данных: Python, Microsoft Excel, Tableau, SaS и т.д. Data Science-специалистам для работы необходим простой, но в тоже время, функциональный язык. Многие опытные аналитики делают выбор в пользу Python. В рассматриваемой сфере он имеет множество сильных сторон.

1. Высокая продуктивность разработки

   У Python простой синтаксис. Это позволяет писать код быстрее, чем на других языках программирования (например, Java или C). При этом, код на Python получается читабельным и легко интерпретируемым.

2. Низкий порог входа для изучения

   Анализом данных обычно занимаются те, кто участвуют в управлении бизнесом — предприниматели, аналитики, экономисты и т.п. Когда возникает потребность изучения языка программирования, остается мало времени на решение важных задач. Поэтому отказываются от Java, C и подобных — так как их изучение занимает много времени.

3. «Интерактивность» языка

   У Python есть встроенный интерпретатор, позволяющий кодить на ходу. То есть, аналитики могут проверять многочисленные гипотезы в интерактивном режиме. Работая с другими языками программирования, добиться аналогичного результата сложнее.

4. Интегрированные возможности для оптимизации кода

   Встроенный интерпретатор пригодится специалистам, работающим с Big Data. Кроме анализа данных, приходится часто улучшать алгоритмы их обработки. И так как Python предлагает неявную и динамическую типизацию данных, сходу определить оптимизацию не получится — это можно сделать только в процессе исполнения кода.

   Для этого и нужен интерпретатор. Он автоматически преобразует исходный код в машинные инструкции. На основе этого генерируются идеи по оптимизации. Например, если сравнить две инструкции, выполняющие одинаковые функции — получится быстро понять, почему одна работает быстрее другой.

5. Динамичное развитие языка

   Java и C++ сильно отстают по скорости развития от Python. Последний больше открыт для комьюнити: любой разработчик может предлагать свои идеи, которые впоследствии могут быть добавлены в обновление. Благодаря этому с каждой новой версией производительность языка повышается, а синтаксис совершенствуется.

10 библиотек Python, которые помогают в области Data Science:

1. Pandas
2. NumPy
3. SciPy
4. Matplotlib
5. Seaborn
6. Scikit Learn
7. TensorFlow
8. Keras
9. Statsmodels
10. Plotly

PANDAS

Pandas — это пакет Python с открытым исходным кодом, который предоставляет высокоэффективные, простые в использовании структуры данных и инструменты анализа для помеченных данных на языке программирования Python. Она построена поверх более низкоуровневого пакета NumPy и используется для обработки и анализа данных.

Уэс Мак-Кинни приступил к созданию Pandas в 2008 году, нуждаясь в быстром и гибком инструменте для количественного анализа финансовой информации. Сейчас эта высокоуровневая библиотека для анализа данных на Python считается одной из наиболее динамично развивающихся.

Pandas - это идеальный инструмент для обработки данных. Он предназначен для быстрой и простой обработки данных, чтения, агрегирования и визуализации.

Он позволяет строить сводные таблицы, выделять колонки, использовать фильтры, выполнять группировку по параметрам, запускать функции (сложение, нахождение медианы, среднего, минимального, максимального значений), объединять таблицы и многое другое. В Pandas можно создавать и многомерные таблицы.

Библиотека позволяет решить практически любую задачу, которая может возникнуть при работе с данными. При этом Pandas позволяет работать с огромными массивами данных разного формата — и работать достаточно быстро. Вот лишь некоторые из возможностей библиотеки:

1. умеет работать с разными источниками данных: файлы Excel, текстовые файлы табличных данных (csv, txt), табличные данные веб-страниц, данные в формате JSON, данные из СУБД и даже из буфера обмена;

2. позволяет очень быстро работать с файлами больших объемов. Файлы на десятки и сотни мегабайт или даже гигабайты данных — для библиотеки не проблема;

3. позволяет просто и быстро объединять данные из нескольких источников в единый массив для дальнейшей обработки;

4. умеет не только читать различные форматы, но и преобразовывать данные, сохраняя их в файлы различных форматов;

5. предоставляет возможности по группировке данных, накладыванию различных фильтров, построению сводных таблиц и многое другое.

Это делает Pandas фундаментальной библиотекой в изучении Python для Data Science.

NUMPY

NumPy - один из самых фундаментальных пакетов в Python - универсальный пакет для обработки массивов. Он предоставляет высокопроизводительные объекты многомерных массивов и инструменты для работы с массивами. NumPy - это эффективный контейнер универсальных многомерных данных.

Основной объект NumPy - это однородный многомерный массив. Это таблица элементов или чисел одного и того же типа данных, проиндексированная набором натуральных чисел. В NumPy размеры называются осями, а число осей называется рангом. Класс массива NumPy называется ndarray, он же array.

Когда использовать? NumPy используется для обработки массивов, в которых хранятся значения одного и того же типа данных. NumPy облегчает математические операции над массивами и их векторизацию. Это значительно повышает производительность и, соответственно, ускоряет время выполнения.

Что можно делать с помощью NumPy?

1. Основные операции с массивами: добавление, умножение, срез, выравнивание, изменение формы, индексирование массивов;

2. Расширенные операции с массивами: стековые массивы, разбиение на секции, широковещательные массивы;

3. Работа с DateTime или линейной алгеброй;

4. Основные нарезки и расширенное индексирование в NumPy Python.

SEABORN

Seaborn – это библиотека визуализации данных на основе Matplotlib, предоставляющем высокоуровневый интерфейс для изображения интересных и информативных статистических графиков. Проще говоря, Seaborn - это расширение Matplotlib с дополнительными возможностями.

Так в чем разница между Matplotlib и Seaborn? Matplotlib используется для основного построения столбцовых, круговых, линейных, точечных диаграмм и пр., в то время как Seaborn предоставляет множество шаблонов визуализации с меньшим количеством синтаксических правил, причем более простых.

Что можно делать с помощью Seaborn?

1. Определять отношения между несколькими переменными (корреляция);

2. Соблюдать качественные переменные для агрегированных статистических данных;

3. Анализировать одномерные или двумерные распределения и сравнивать их между различными подмножествами данных;

4. Построить модели линейной регрессии для зависимых переменных;

5. Обеспечить многоуровневые абстракции, многосюжетные сетки.

Seaborn — это отличный вариант для библиотек визуализации R, таких как *corrplot* и *ggplot.*

ПРИМЕР АНАЛИЗА ДАННЫХ НА PYTHON

Для примера нам потребуется какой-нибудь датасет. Для выбора подходящего датасета воспользуемся kaggle - популярной платформой для соревнований по Data Science; системой организации конкурсов по исследованию данных, а также социальной сетью специалистов по обработке данных и машинному обучению.

О датасете - World Happiness Report

Контекст
Доклад о мировом счастье - это эпохальный обзор состояния глобального счастья. Первый отчет был опубликован в 2012 году, второй - в 2013 году, третий - в 2015 году, а четвертый - в обновлении за 2016 год. Всемирный рейтинг счастья за 2017 год, в котором 155 стран ранжируются по уровню счастья, был опубликован в Организации Объединенных Наций на мероприятии, посвященном Международному дню счастья 20 марта. Доклад продолжает получать всемирное признание, поскольку правительства, организации и гражданское общество все чаще используют показатели счастья для обоснования своих политических решений. Ведущие эксперты в различных областях – экономике, психологии, анализе опросов, национальной статистике, здравоохранении, государственной политике и многом другом - описывают, как измерения благосостояния могут быть эффективно использованы для оценки прогресса стран. В отчетах рассматривается состояние счастья в современном мире и показано, как новая наука о счастье объясняет личные и национальные различия в счастье.

Содержание
Оценки и рейтинги счастья основаны на данных Всемирного опроса Gallup. Оценки основаны на ответах на основной вопрос об оценке жизни, заданный в опросе. Этот вопрос, известный как лестница Кантрила, просит респондентов подумать о лестнице, в которой наилучшая возможная жизнь для них равна 10, а наихудшая возможная жизнь равна 0, и оценить свою собственную текущую жизнь по этой шкале. Оценки получены из национальных репрезентативных выборок за 2013-2016 годы, и для репрезентативности оценок используются веса Гэллапа. Столбцы, следующие за показателем счастья, оценивают степень, в которой каждый из шести факторов – экономическое производство, социальная поддержка, ожидаемая продолжительность жизни, свобода, отсутствие коррупции и щедрость – способствуют повышению оценок жизни в каждой стране по сравнению с Антиутопией, гипотетической страной, ценности которой равны мировым. самые низкие средние показатели по стране для каждого из шести факторов.

Они не влияют на общий балл, указанный для каждой страны, но они объясняют, почему некоторые страны занимают более высокое место, чем другие.

Что описывают столбцы, следующие за Оценкой счастья?

Следующие столбцы: ВВП на душу населения, Семья, Ожидаемая продолжительность жизни, Свобода, Щедрость, Доверие, Коррупция в правительстве описывают степень, в которой эти факторы влияют на оценку счастья в каждой стране.

jupyter Untitled1 Last Checkpoint: час назад (autosaved) Logout

File Edit View Insert Cell Kernel Help Notebook saved Not Trusted Python 3 (ipykernel)

Run Code

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

In [2]:
```
df19 = pd.read_csv('2019.csv')
```

In [3]:
```
df19.head()
```

Out[3]:

	Overall rank	Country or region	Score	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Generosity	Perceptions of corruption
0	1	Finland	7.769	1.340	1.587	0.986	0.596	0.153	0.393
1	2	Denmark	7.600	1.383	1.573	0.996	0.592	0.252	0.410
2	3	Norway	7.554	1.488	1.582	1.028	0.603	0.271	0.341
3	4	Iceland	7.494	1.380	1.624	1.026	0.591	0.354	0.118
4	5	Netherlands	7.488	1.396	1.522	0.999	0.557	0.322	0.298

In [4]:
```
df19.isnull().sum()
```

Out[4]:
```
Overall rank                  0
Country or region             0
Score                         0
GDP per capita                0
Social support                0
Healthy life expectancy       0
Freedom to make life choices  0
Generosity                    0
Perceptions of corruption     0
dtype: int64
```

In [5]:
```
top7 = df19[0:7]
```

In [6]:
```
top7.shape[0]
```

Out[6]:
```
7
```

In [7]:
```
top7.shape[1]
```

Out[7]:
```
9
```

In [8]:
```
top7.describe()
```

Out[8]:

	Overall rank	Score	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Generosity	Perceptions of corruption
count	7.000000	7.000000	7.000000	7.000000	7.000000	7.000000	7.000000	7.000000
mean	4.000000	7.532571	1.403714	1.557286	1.013714	0.583571	0.268857	0.325143
std	2.160247	0.131072	0.049708	0.047179	0.022882	0.016277	0.063012	0.098582
min	1.000000	7.343000	1.340000	1.487000	0.986000	0.557000	0.153000	0.118000
25%	2.500000	7.484000	1.381500	1.524000	0.997500	0.573000	0.257500	0.319500
50%	4.000000	7.494000	1.387000	1.573000	1.009000	0.591000	0.267000	0.343000
75%	5.500000	7.577000	1.424000	1.584500	1.027000	0.594000	0.296500	0.383000
max	7.000000	7.769000	1.488000	1.624000	1.052000	0.603000	0.354000	0.410000

In [9]:
```
top_sorted_Health = top7.sort_values(by = ['Healthy life expectancy'], ascending= False)
top_sorted_Health.head(7)
```

Out[9]:

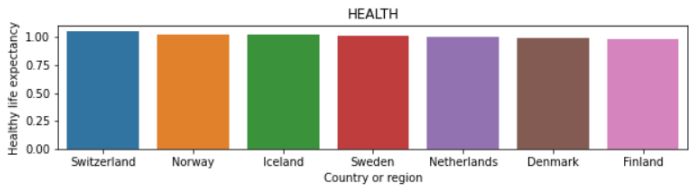
click to expand output; double click to hide output

	Overall rank	Country or region	Score	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Generosity	Perceptions of corruption
		ınd	7.480	1.452	1.526	1.052	0.572	0.263	0.343
2	3	Norway	7.554	1.488	1.582	1.028	0.603	0.271	0.341
3	4	Iceland	7.494	1.380	1.624	1.026	0.591	0.354	0.118
6	7	Sweden	7.343	1.387	1.487	1.009	0.574	0.267	0.373
4	5	Netherlands	7.488	1.396	1.522	0.999	0.557	0.322	0.298
1	2	Denmark	7.600	1.383	1.573	0.996	0.592	0.252	0.410
0	1	Finland	7.769	1.340	1.587	0.986	0.596	0.153	0.393

In [10]:
```
fig, axs = plt.subplots(figsize = (10,2))

sns.barplot(data = top_sorted_Health, x = 'Country or region', y = 'Healthy life expectancy')
axs.set_title('HEALTH')
```

Out[10]:
```
Text(0.5, 1.0, 'HEALTH')
```



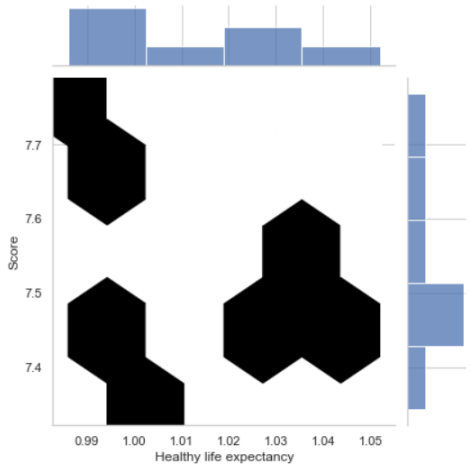
In [11]:
```
sns.set_theme(style ="whitegrid")
fig, axs = plt.subplots(figsize = (10,2))

sns.barplot(data = top_sorted_Health, x = 'Country or region', y = 'Healthy life expectancy')
axs.set_title('HEALTH')
```

Out[11]:
```
Text(0.5, 1.0, 'HEALTH')
```

In [12]:
```
sns.jointplot(x='Healthy life expectancy',y='Score',data=top7,kind='hex')
```

Out[12]:
```
<seaborn.axisgrid.JointGrid at 0x1a8b6a59090>
```



In [13]:
```
df19['Sum'] = df19['GDP per capita'] + df19['Social support']+df19['Freedom to make life choices']+df19['Healthy life expectancy
df19.head()
```

Out[13]:

	Overall rank	Country or region	Score	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Generosity	Perceptions of corruption	Sum
0	1	Finland	7.769	1.340	1.587	0.986	0.596	0.153	0.393	4.509
1	2	Denmark	7.600	1.383	1.573	0.996	0.592	0.252	0.410	4.544
2	3	Norway	7.554	1.488	1.582	1.028	0.603	0.271	0.341	4.701
3	4	Iceland	7.494	1.380	1.624	1.026	0.591	0.354	0.118	4.621
4	5	Netherlands	7.488	1.396	1.522	0.999	0.557	0.322	0.298	4.474

In [14]:
```
top7 = top7.append(top7.mean(axis=0), ignore_index = True)
```

```
C:\Users\jasmi\AppData\Local\Temp\ipykernel_10716\1878512665.py:1: FutureWarning: Dropping of nuisance columns in DataFrame red
uctions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid columns bef
ore calling the reduction.
  top7 = top7.append(top7.mean(axis=0), ignore_index = True)
C:\Users\jasmi\AppData\Local\Temp\ipykernel_10716\1878512665.py:1: FutureWarning: The frame.append method is deprecated and wil
l be removed from pandas in a future version. Use pandas.concat instead.
  top7 = top7.append(top7.mean(axis=0), ignore_index = True)
```

In [15]:
```
top7.head()
```

Out[15]:

	Overall rank	Country or region	Score	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Generosity	Perceptions of corruption
0	1.0	Finland	7.769	1.340	1.587	0.986	0.596	0.153	0.393
1	2.0	Denmark	7.600	1.383	1.573	0.996	0.592	0.252	0.410
2	3.0	Norway	7.554	1.488	1.582	1.028	0.603	0.271	0.341
3	4.0	Iceland	7.494	1.380	1.624	1.026	0.591	0.354	0.118
4	5.0	Netherlands	7.488	1.396	1.522	0.999	0.557	0.322	0.298

In [16]:
```
top7.head(8)
```

Out[16]:

	Overall rank	Country or region	Score	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Generosity	Perceptions of corruption
0	1.0	Finland	7.769000	1.340000	1.587000	0.986000	0.596000	0.153000	0.393000
1	2.0	Denmark	7.600000	1.383000	1.573000	0.996000	0.592000	0.252000	0.410000
2	3.0	Norway	7.554000	1.488000	1.582000	1.028000	0.603000	0.271000	0.341000
3	4.0	Iceland	7.494000	1.380000	1.624000	1.026000	0.591000	0.354000	0.118000
4	5.0	Netherlands	7.488000	1.396000	1.522000	0.999000	0.557000	0.322000	0.298000
5	6.0	Switzerland	7.480000	1.452000	1.526000	1.052000	0.572000	0.263000	0.343000
6	7.0	Sweden	7.343000	1.387000	1.487000	1.009000	0.574000	0.267000	0.373000
7	4.0	NaN	7.532571	1.403714	1.557286	1.013714	0.583571	0.268857	0.325143

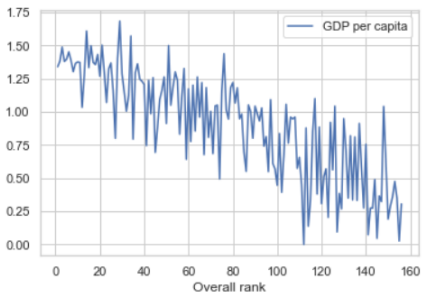
In [17]:
```
top7 = top7.drop(top7.shape[0]-1, axis = 0)
top7.head(8)
```

Out[17]:

	Overall rank	Country or region	Score	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Generosity	Perceptions of corruption
0	1.0	Finland	7.769	1.340	1.587	0.986	0.596	0.153	0.393
1	2.0	Denmark	7.600	1.383	1.573	0.996	0.592	0.252	0.410
2	3.0	Norway	7.554	1.488	1.582	1.028	0.603	0.271	0.341
3	4.0	Iceland	7.494	1.380	1.624	1.026	0.591	0.354	0.118
4	5.0	Netherlands	7.488	1.396	1.522	0.999	0.557	0.322	0.298
5	6.0	Switzerland	7.480	1.452	1.526	1.052	0.572	0.263	0.343
6	7.0	Sweden	7.343	1.387	1.487	1.009	0.574	0.267	0.373

In [18]:
```
df19.plot(x = 'Overall rank', y = 'GDP per capita')
```

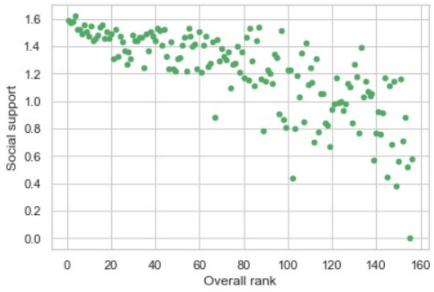
Out[18]:
```
<AxesSubplot:xlabel='Overall rank'>
```



In [19]:
```
plt.figure(figsize= (5,4))
words= "".join(df19)
final = wc.WordCloud(width = 500, height = 400, background_color ="white",min_font_size = 20, max_words = 100).generate(words)
plt.imshow(final)
plt.axis("off")
plt.show()
```

In [21]:
```
df19.plot.scatter(x = 'Overall rank', y = 'Social support',c = 'g')
```

Out[21]: <AxesSubplot:xlabel='Overall rank', ylabel='Social support'>



In [22]:
```
df19[(df19['GDP per capita'] > 1) & (df19['Social support'] > 1.5)]
```

Out[22]:

	Overall rank	Country or region	Score	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Generosity	Perceptions of corruption	Sum
0	1	Finland	7.769	1.340	1.587	0.986	0.596	0.153	0.393	4.509
1	2	Denmark	7.600	1.383	1.573	0.996	0.592	0.252	0.410	4.544
2	3	Norway	7.554	1.488	1.582	1.028	0.603	0.271	0.341	4.701
3	4	Iceland	7.494	1.380	1.624	1.026	0.591	0.354	0.118	4.621
4	5	Netherlands	7.488	1.396	1.522	0.999	0.557	0.322	0.298	4.474
5	6	Switzerland	7.480	1.452	1.526	1.052	0.572	0.263	0.343	4.602
7	8	New Zealand	7.307	1.303	1.557	1.026	0.585	0.330	0.380	4.471
8	9	Canada	7.278	1.365	1.505	1.039	0.584	0.285	0.308	4.493
10	11	Australia	7.228	1.372	1.548	1.036	0.557	0.332	0.290	4.513
14	15	United Kingdom	7.054	1.333	1.538	0.996	0.450	0.348	0.278	4.317
15	16	Ireland	7.021	1.499	1.553	0.999	0.516	0.298	0.310	4.567

In [23]:
```
df19.dropna()
```

Out[23]:

	Overall rank	Country or region	Score	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Generosity	Perceptions of corruption	Sum
0	1	Finland	7.769	1.340	1.587	0.986	0.596	0.153	0.393	4.509
1	2	Denmark	7.600	1.383	1.573	0.996	0.592	0.252	0.410	4.544
2	3	Norway	7.554	1.488	1.582	1.028	0.603	0.271	0.341	4.701
3	4	Iceland	7.494	1.380	1.624	1.026	0.591	0.354	0.118	4.621
4	5	Netherlands	7.488	1.396	1.522	0.999	0.557	0.322	0.298	4.474
...	...	...	...	...	...	...	...	...	...	...
151	152	Rwanda	3.334	0.359	0.711	0.614	0.555	0.217	0.411	2.239
152	153	Tanzania	3.231	0.476	0.885	0.499	0.417	0.276	0.147	2.277
153	154	Afghanistan	3.203	0.350	0.517	0.361	0.000	0.158	0.025	1.228
154	155	Central African Republic	3.083	0.026	0.000	0.105	0.225	0.235	0.035	0.356
155	156	South Sudan	2.853	0.306	0.575	0.295	0.010	0.202	0.091	1.186

156 rows × 10 columns

In [ ]:

Графики:

График разброса (scatter)- это тоже очень распространенный график. Как правило, именно глядя на него, начинаешь понимать, что в данных есть что-то любопытное. Диаграмма рассеяния является основой статистической визуализации. Он изображает совместное распределение двух переменных с использованием облака точек, где каждая точка представляет наблюдение в наборе данных. Это изображение позволяет глазу вывести значительный объем информации о том, существуют ли какие-либо значимые отношения между ними.

С некоторыми наборами данных вы можете захотеть понимать изменения одной переменной как функции времени или аналогично непрерывной переменной. В этой ситуации хорошим выбором будет нарисовать линию сюжета В Seaborn это можно сделать с помощью функции lineplot (), либо напрямую, либо с помощью relplot (), установив kind = ”line”.

Для других приложений вместо отображения распределения внутри каждой категории может потребоваться показать оценку центральной тенденции значений. Функция barplot () работает с полным набором данных и применяет функцию для получения оценки (принимая среднее значение по умолчанию).

ЗАКЛЮЧЕНИЕ

Тема действительно интересна и довольно обширна. Пополнять свой арсенал инструментов для работы с различными базами данных, их визуализацией и тд можно, мне кажется, бесконечно долго. В языке программирования Python для этого существует масса удобнейших библиотек, даже немного разобравшись в которых, можно сделать первый шаг в эту сферу деятельности. Была ли проделанная мной работа интересна и актуальна лично для меня? Да. Если учесть, что абсолютно все в данной работе было изучено мной фактически с нуля, то плоды ее для моего развития окажутся весьма очевидными.