A CAREFUL EXPLORATORY DATA ANALYSIS OF "WORLD HAPPINESS REPORT OF 2023"

Author: José Bento Ferreira Montenegro

Jul/02/2024

bentomontenegro@gmail.com

1. Introduction

Exploratory Data Analysis (EDA) is a fundamental step of machine learning workflows. It is a powerful toolbox that allows us to efficiently extract statistical information about the data even before making a model, as well as to select which attributes as characteristics of the data are essential for the modeling process.

In the present work we analyze the data set "World Happiness Report of 2023", conducting all the best-practices steps of EDA, as well as a formal hypothesis test to exemplify the major role of statistical analysis in today's industrial scenario. More specifically, it is well known that raw data sets almost never come ideally suited for modeling purposes. We then show through code snippets and figures how the process of cleaning, removing and renaming columns, re-scaling attributes, handling outliers, analyze correlations and testing hypotheses can be implemented on real data science workflows.

This is actually my final project for the EDA module of the IBM Machine Learning Professional Certificate course on Coursera's platform. I hope the reader will enjoy it.

2. Brief description of the data

As said above, we shall analyze the data set "World Happiness Report of 2023". This data set contains information about the happiness scores reported from population samples of several countries around the world, as well as economic and political factors specific of each country. Of course, taking "Hapiness score" as target and such factors as features, the main purpose of the data set is to understand what exactly might influence individual reports at each country.

Let us have a look at the initial attributes of the data set.

```python
import pandas as pd

filepath = 'data/World Happiness Report 2023.csv'
data = pd.read_csv(filepath)

data.info()
```

▸ **Output:**

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 137 entries, 0 to 136
Data columns (total 19 columns):
 #   Column                                      Non-Null Count  Dtype
---  ------                                      --------------  -----
 0   Country name                                137 non-null    object
 1   Ladder score                                137 non-null    float64
 2   Standard error of ladder score              137 non-null    float64
 3   upperwhisker                                137 non-null    float64
 4   lowerwhisker                                137 non-null    float64
 5   Logged GDP per capita                       137 non-null    float64
 6   Social support                              137 non-null    float64
 7   Healthy life expectancy                     136 non-null    float64
 8   Freedom to make life choices                 137 non-null    float64
 9   Generosity                                   137 non-null    float64
 10  Perceptions of corruption                    137 non-null    float64
 11  Ladder score in Dystopia                     137 non-null    float64
 12  Explained by: Log GDP per capita             137 non-null    float64
 13  Explained by: Social support                 137 non-null    float64
 14  Explained by: Healthy life expectancy        136 non-null    float64
 15  Explained by: Freedom to make life choices   137 non-null    float64
 16  Explained by: Generosity                     137 non-null    float64
 17  Explained by: Perceptions of corruption      137 non-null    float64
 18  Dystopia + residual                          136 non-null    float64
dtypes: float64(18), object(1)
memory usage: 20.5+ KB
```

As our goal in the sections to come is to make these attributes clearer and well-suited for modeling, it is good to make an initial plan for data exploration.

3. Initial plan for data exploration

As suggested by the list of attributes above, previous data analysis has been set forth on this data set. In order to make it more suited for the purposes of the present work, I suggest the following steps, which will be explored in the sections to come:

1. Remove columns that refer to the previous data analysis
2. Rename certain columns more concisely
3. Identify null values and decide what to do with them
4. Re-scale attributes, if necessary
5. Identify outliers and decide what to do with them
6. Plot correlations between features and target
7. Extract information from the correlation plot
8. Propose three relevant statistical tests for this data
9. Explore one of such tests using formal statistical analysis

4. Data cleaning and feature engineering

4.1 Removing columns

First of all, we would like to conduct a brand new EDA (Exploratory Data Analysis) on this data. To this end, it is useful to begin by removing all columns that make reference to any previous analysis conducted on it. Such columns are:

```
1. Upperwhisker
2. Lowerwhisker
3. Standard error of ladder score
4. Ladder score in Dystopia
5. All the "Explained by" columns
6. Dystopia + residuals
```

We remove them at once using `pandas.DataFrame.drop`.

```
data = data.drop(['Standard error of ladder score',
                  'upperwhisker',
                  'lowerwhisker',
                  'Ladder score in Dystopia',
                  'Explained by: Log GDP per capita',
                  'Explained by: Healthy life expectancy',
                  'Explained by: Healthy life expectancy',
                  'Explained by: Freedom to make life choices',
                  'Explained by: Generosity',
                  'Explained by: Perceptions of corruption',
                  'Explained by: Social support',
                  'Dystopia + residual'],
                  axis = 1)

 data.info()
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 137 entries, 0 to 136
Data columns (total 8 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   Country name                  137 non-null    object 
 1   Ladder score                  137 non-null    float64
 2   Logged GDP per capita         137 non-null    float64
 3   Social support                137 non-null    float64
 4   Healthy life expectancy       136 non-null    float64
 5   Freedom to make life choices  137 non-null    float64
 6   Generosity                    137 non-null    float64
 7   Perceptions of corruption     137 non-null    float64
dtypes: float64(7), object(1)
memory usage: 8.7+ KB
```

4.2 Renaming columns

Going further, we can use `pandas.DataFrame.rename` to make the following change of names:

1. Country name → Country
2. Ladder score → Happiness score
3. Logged GDP per capita → GDP/capita
4. Freedom to make life choices → Freedom of choice

Let us do it at once:

```
data = data.rename(columns = {'Country name': 'Country',
                              'Ladder score': 'Happiness score',
                              'Logged GDP per capita': 'GDP/capita',
                              'Freedom to make life choices': 'Freedom of
choice'})

data.info()
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 137 entries, 0 to 136
Data columns (total 8 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   Country                    137 non-null    object 
 1   Happiness score            137 non-null    float64
 2   GDP/capita                 137 non-null    float64
 3   Social support            137 non-null    float64
 4   Healthy life expectancy   136 non-null    float64
 5   Freedom of choice         137 non-null    float64
 6   Generosity                137 non-null    float64
 7   Perceptions of corruption 137 non-null    float64
dtypes: float64(7), object(1)
memory usage: 8.7+ KB
```

Now that we selected and cleaned our list of attributes, it is time to deal with feature engineering, namely, looking for null values, re-scaling the data and handling outliers.

4.3 Null values, re-scaling and outliers

4.3.1 Null values

Now, as suggested by the last output, there is only one missing value that we have to deal with. It is relative to the feature “Healthy life expectancy” for the country “State of Palestine”, as can be confirmed below.

```
data[data.isnull().any(axis = 1)]
```

▸ **Output:**

	Country	Happiness score	GDP/capita	Social support	Healthy life expectancy	Freedom of choice	Generosity	Perceptions of corruption
98	State of Palestine	4.908	8.716	0.859	NaN	0.694	-0.132	0.836

There are several ways to deal with null values, such as removing the entire row containing them from the data set, or replacing them for the mean value of the attribute they belong to. The first method has the downside of compromising the accuracy of future models that involve other attributes of the set, and the latter has the downside of not representing actual data very well. So, we have to make a decision here.

As today's scientific development of health care has reached a wide range of countries, I think that, for "Healthy life expectancy" specifically, it is reasonable to replace this null value for the mean of this attribute. If this, for instance, compromises our model, we can always get back to this point and try a different technique.

We use `pandas.DataFrame.fillna` to do this job.

```
palestine_hle_mean = data['Healthy life expectancy'].mean()
data = data.fillna(value = palestine_hle_mean)

data[data['Country'] == 'State of Palestine']
```

▸ **Output:**

	Country	Happiness score	GDP/capita	Social support	Healthy life expectancy	Freedom of choice	Generosity	Perceptions of corruption
98	State of Palestine	4.908	8.716	0.859	64.967632	0.694	-0.132	0.836

4.3.2 Re-scaling

Now we check if any attribute of the data needs re-scaling, as making sure that all attributes have similar scales is a good practice for regression modeling.

We start by looking at the first few lines.

```
data.head(10)
```

▸ **Output:**

	Country	Happiness score	GDP/capita	Social support	Healthy life expectancy	Freedom of choice	Generosity	Perceptions of corruption
0	Finland	7.804	10.792	0.969	71.150	0.961	-0.019	0.182
1	Denmark	7.586	10.962	0.954	71.250	0.934	0.134	0.196
2	Iceland	7.530	10.896	0.983	72.050	0.936	0.211	0.668
3	Israel	7.473	10.639	0.943	72.697	0.809	-0.023	0.708
4	Netherlands	7.403	10.942	0.930	71.550	0.887	0.213	0.379
5	Sweden	7.395	10.883	0.939	72.150	0.948	0.165	0.202
6	Norway	7.315	11.088	0.943	71.500	0.947	0.141	0.283
7	Switzerland	7.240	11.164	0.920	72.900	0.891	0.027	0.266
8	Luxembourg	7.228	11.660	0.879	71.675	0.915	0.024	0.345
9	New Zealand	7.123	10.662	0.952	70.350	0.887	0.175	0.271

We can see that the attributes “Social support”, “Healthy life expectancy”, “Freedom of choice”, “Generosity" and “Perceptions of corruption” are not on the same scale as the target variable “Happiness score”.

The scales are not drastically far from each other, and we could use several methods to fix this, such as log-scaling and min-max scaling. However, as we shall see in the "correlations" sections, several attributes of the data seem to have a linear dependence with one another. This suggests that linear regression models could be a good choice for modeling. Thus, as such models work best when the target variable (which in our case would be naturally taken as "Happiness score") is normally distributed, z-scale normalization will be our choice of scaling. This choice of scale will also be very useful for handling outliers, as we shall see in the next section.

```
from sklearn.preprocessing import StandardScaler

## Sampling numeric columns
numeric_columns = data.drop('Country', axis=1).columns

## Rescaling numeric columns
scaler = StandardScaler()
rescaled_data = data.copy()
rescaled_data[numeric_columns]                                        =
scaler.fit_transform(rescaled_data[numeric_columns])

rescaled_data.head(10)
```

▸ **Output:**

	Country	Happiness score	GDP/capita	Social support	Healthy life expectancy	Freedom of choice	Generosity	Perceptions of corruption
0	Finland	1.993557	1.115818	1.319824	1.083056	1.550602	-0.293442	-3.082101
1	Denmark	1.801616	1.257145	1.203319	1.100574	1.309445	0.790215	-3.002695
2	Iceland	1.752309	1.202277	1.428562	1.240722	1.327309	1.335585	-0.325573
3	Israel	1.702123	0.988624	1.117882	1.354067	0.192978	-0.321773	-0.098699
4	Netherlands	1.640490	1.240518	1.016911	1.153130	0.889654	1.349751	-1.964743
5	Sweden	1.633446	1.191469	1.086814	1.258241	1.434490	1.009780	-2.968664
6	Norway	1.563009	1.361893	1.117882	1.144371	1.425558	0.839794	-2.509243
7	Switzerland	1.496974	1.425074	0.939241	1.389629	0.925380	0.032363	-2.605664
8	Luxembourg	1.486408	1.837415	0.620794	1.175028	1.139742	0.011115	-2.157587
9	New Zealand	1.393959	1.007745	1.187785	0.942908	0.889654	1.080607	-2.577305

4.3.3 Outliers

Now that we re-scaled our data, it is a good practice to look for outliers, which might significantly bias our future analyses. As always, there are several methods available for this, such as box plots and z-score analysis. As in the last section we already converted our data points to their z-scores, the latter method can save us some time.

In summary, z-score analysis consists in calculating the z-score ((x - μ)/σ) for every point x of a given attribute with mean μ and standard deviation σ, and comparing it to a certain *threshold value* T. More specially, the z-score measures how far x is from μ in units of σ. If x has a z score that is less than -T or greater than T, it is considered an outlier. It is a common practice in statistics to choose T = 3.

Below we create a function that identifies the outliers of a given column and apply it to all numeric columns of our data set.

```python
def detect_outliers_z_score(df_column, threshold):

    '''
    df_column: column of a data frame with standard-scaled values
    threshold: threshold value
    '''

    ## Identifying outliers
    outliers = []
    for score in df_column:
        index = list(df_column).index(score)
        if score < -threshold:
            outliers.append(df_column[index])
        elif score > threshold:
            outliers.append(df_column[index])

    return outliers

## Applying the function
threshold = 3
outliers_dict = {}
for column in numeric_columns:
    outliers = detect_outliers_z_score(rescaled_data[column], threshold)
    outliers_dict[column] = outliers

print('Outliers:')
outliers_dict
```

▸ **Output:**

```
Outliers:

{'Happiness score': [-3.240819380486641],
 'GDP/capita': [-3.261147548346835],
 'Social support': [-3.557856157681139],
 'Healthy life expectancy': [],
 'Freedom of choice': [-3.62087544144848],
 'Generosity': [3.6020584537617704, 3.3187493237187913],
 'Perceptions of corruption': [-3.0821012297790795,
  -3.002695080909882,
  -3.286288469728444]}
```

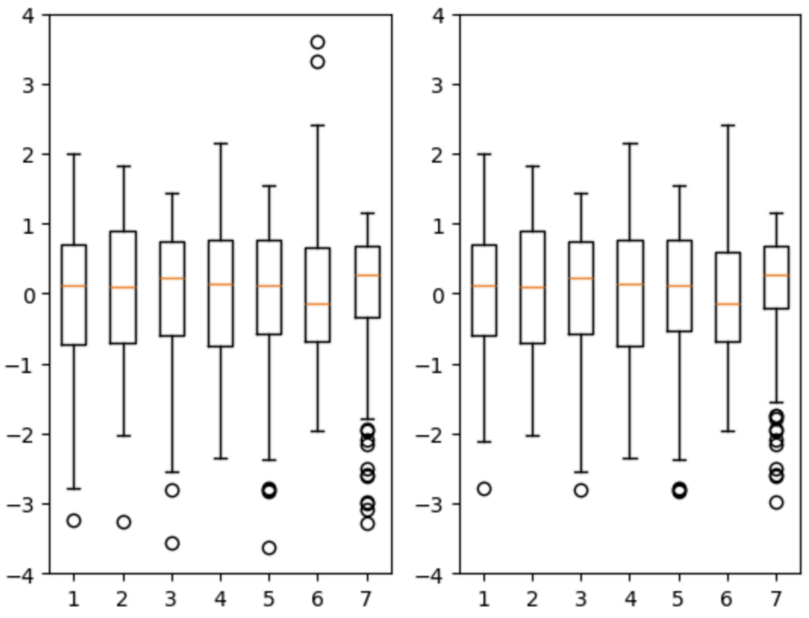
Once the outliers are identified, we must decide what to do with them. One option would be replacing them by the mean value of their respective attributes, or even removing the entire row containing them. If the presence of the outlier is not due to systematic errors, the first method would have the downside of not representing the data very well. The second, naturally, has the downside of possibly compromising modeling precision.

Looking at the z-score of the outliers, we can notice that they are not absurdly higher than 3 standard deviations. Thus, it does not seem that their presence within the data set is due to systematic errors during the process of obtaining or reporting the data. Therefore, I think that replacing them for the mean value of their respective attributes is a good choice in our case. This method will also be proven useful in the last section, where we will analyze a hypothesis test that depends on sampling all countries from Europe and Asia. Thus, replacing outliers by mean values will not reduce the sample of any continent.

Below we implement the replacements and make two box plots for comparison.

```python
import numpy as np

clean_data = rescaled_data.copy()

## Replacing outliers by mean values
for column in numeric_columns:
    for outlier in outliers_dict[column]:
        index = list(rescaled_data[column]).index(outlier) # Identifying the index of the outlier
        clean_data.at[index, column] = rescaled_data[column].mean()

plt.subplot(1, 2, 1)
plt.boxplot(rescaled_data[numeric_columns])
plt.ylim(-4,4)
plt.subplot(1, 2, 2)
plt.boxplot(clean_data[numeric_columns])
plt.ylim(-4,4)
```

▸ **Output:**



Note that although both figures contain outliers (white circles), this is not really a problem, as it is clear that most outliers of the left plot are not outliers for the right plot. In fact, the right plot only shows points as outliers because after the replacement, our new data set has new statistical parameters (standard deviation, mean, etc.). It is also clear that the lines dividing the 50% quantile in the right plot are somewhat better centralized than the lines of the left plot.

4.5 Correlations

Now that we organized, cleaned and re-scaled our data, it is useful to check how its attributes correlate among themselves. This is a valuable step in modeling, where we have to decide which features to use and which features are really independent from each other.

A simple way to start is to make a pairplot and a heatmap plot, as shown below. The pairplot provides us a visual method to infer relevant information on how attributes correlate, and the heatmap provides us a good way to formally quantify such correlations, which is an essential step for modeling. As a matter of interpretation, the correlation values shown in the heatmap quantify how linearly one variable may depend on the other. For ex. a positive (negative) correlation means that if we increase one variable, the other will increase (decrease) proportionally to that correlation value.

In particular, some conclusions we can immediately draw from the figures are.

- The GDP/capita of individuals seems to grow when they are happier, have high healthy life expectancy and social support.

- The GDP/capita does not seem significantly correlated with how much freedom of choice individuals report to have, nor with their generosity and perception of corruption. This is due to the fact that the data points are mostly dispersed in such plots and do not seem to follow any regular pattern.

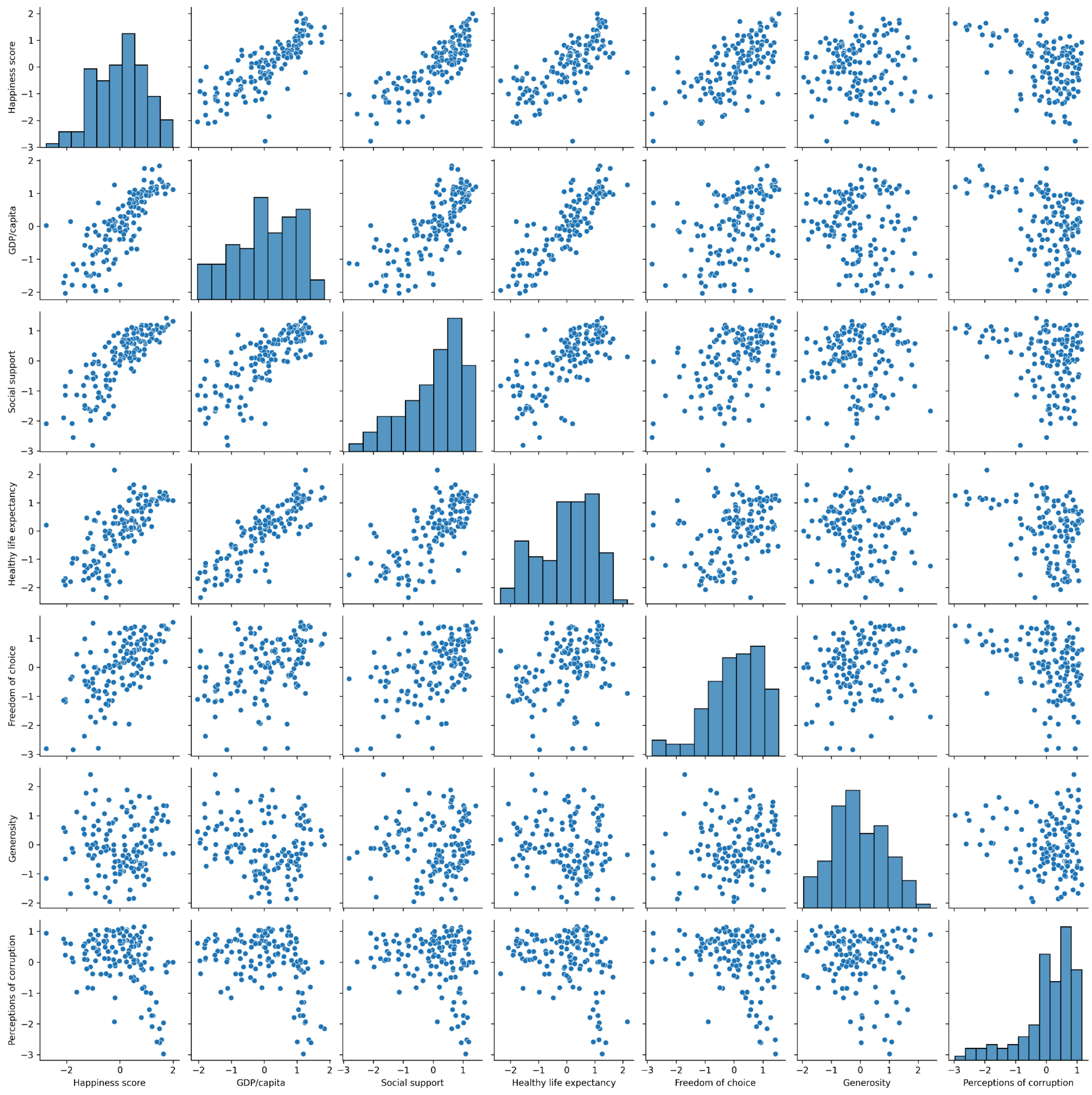
Since “Happiness score” is the natural target variable for modeling, it is also valuable to write down some explicit conclusions relative to this quantity:

- It seems to grow almost linearly with "GDP/capita", "Social Support", "Healthy life expectancy" and "Freedom of choice”, suggesting that a linear regression model may be a good fit for prediction and interpretation.

- It seems to have no regularity alongside “Generosity”, suggesting that individuals do not necessarily feel happier as they feel more generous.
- Although at first it seems to decrease with “Perception of corruption", there is a certain point where this behavior simply diffuses. This suggests that a linear regression model might be a good fit only when we deal with populations that perceive less corruption around them.

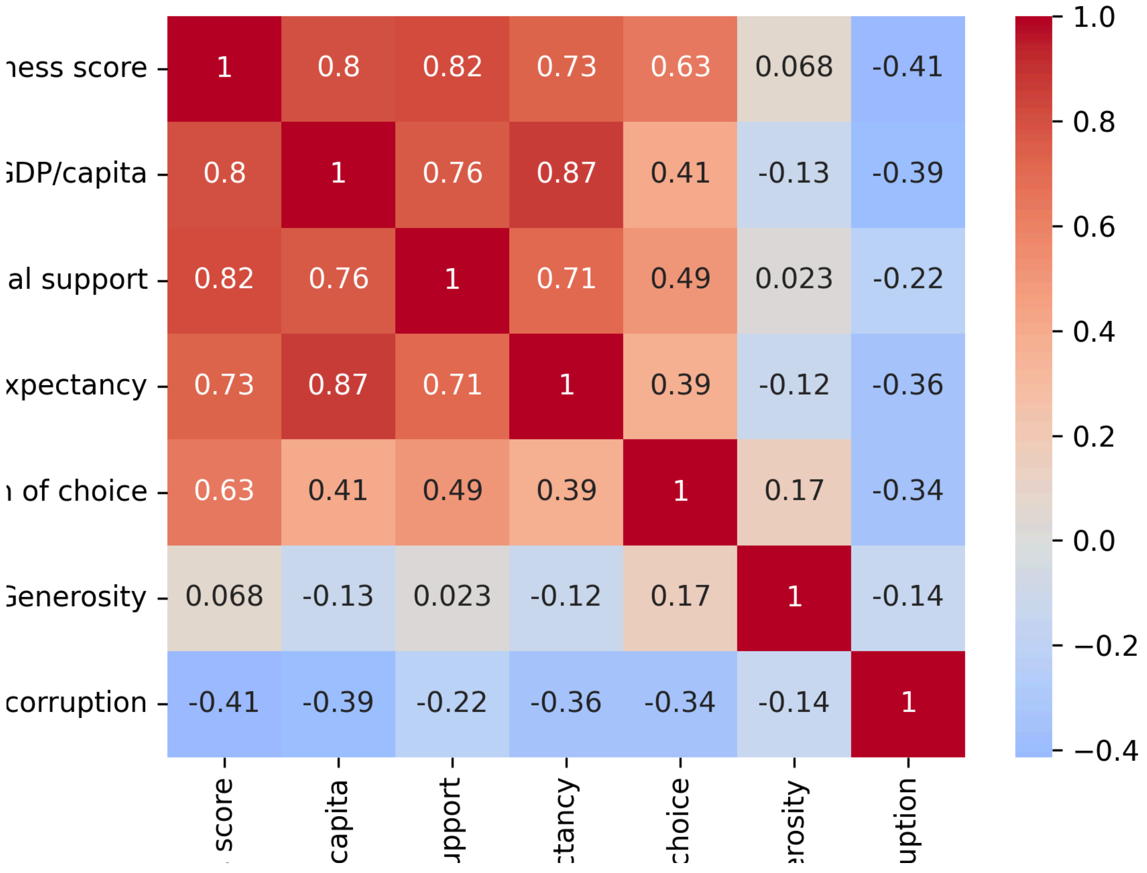
```
import seaborn as sns
import matplotlib.pyplot as plt

sns.pairplot(clean_data)
```

▶ **Output:**



```
corr_matrix = clean_data[numeric_columns].corr()
sns.heatmap(corr_matrix, annot=True, cmap="coolwarm", center=0)
```

▶ **Output:**



6. Hypothesis testing

This is a rich data set and many hypotheses can be formulated for it. Three examples are:

1. On average, Asians are as happy as Europeans
2. The GDP/capita has a very significant influence on Happiness scores
3. The variance of samples of Middle-East is higher than the variance of samples of Asia

We can always use formal statistical analysis to test such hypothesis. For example, we could use a t-test to test the first one, a z-test (or even a regression model) to test the second, and a f-test to test the third. Unfortunately, explaining all these procedures is out of the scope of the present text, and we encourage the reader to find more about them online.

As an illustration, let us test the first hypothesis. The null and the alternative hypotheses can be formulated as:

H_0: $\mu_{Europe} - \mu_{Asia} = 0$ (*The Happiness scores of Asia and Europe are significantly different*)
H_1: $\mu_{Europe} - \mu_{Asia} \neq 0$ (*The Happiness scores of Asia and Europe are not significantly different*)

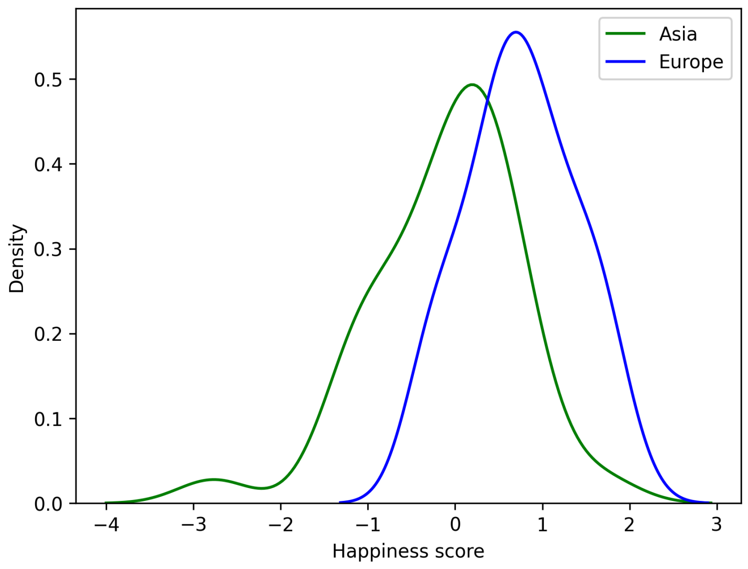
Now, let us fix the significance level $\alpha = 0.05$ (5%), as usual. If the p-value of the test is less than this value, we must reject the null hypothesis and conclude that the two means are, in fact, significantly different.

Let us start by sampling the countries of the two continents into two separate data frames and having a look at their mean values, as well as at their respective distributions.

```python
## Asia
Asia = [
    "Afghanistan", "Armenia", "Azerbaijan", "Bahrain", "Bangladesh",
    "Bhutan", "Cambodia", "China", "Cyprus", "Georgia",
    "India", "Indonesia", "Iran", "Iraq", "Israel", "Japan", "Jordan",
    "Kazakhstan", "Kuwait", "Kyrgyzstan", "Laos", "Lebanon", "Malaysia",
    "Maldives", "Mongolia", "Myanmar", "Nepal", "Pakistan", "Philippines",
    "Qatar", "Russia", "Saudi Arabia", "Singapore", "South Korea", "Sri Lanka",
    "Syria", "Taiwan", "Tajikistan", "Thailand", "Timor-Leste", "Turkey",
    "Turkmenistan", "United Arab Emirates", "Uzbekistan", "Vietnam", "Yemen"
]

Asia_data = clean_data[clean_data["Country"].isin(Asia)]
Asia_hs = Asia_data["Happiness score"]

## Europe
Europe = [
    "Albania", "Andorra", "Armenia", "Austria", "Azerbaijan",
    "Belarus", "Belgium", "Bosnia and Herzegovina", "Bulgaria", "Croatia",
    "Cyprus", "Czech Republic", "Denmark", "Estonia", "Finland", "France",
    "Georgia", "Germany", "Greece", "Hungary", "Iceland", "Ireland", "Italy",
    "Kazakhstan", "Kosovo", "Latvia", "Lithuania", "Luxembourg", "Malta",
    "Moldova", "Montenegro", "Netherlands", "North Macedonia", "Norway",
    "Poland", "Portugal", "Romania", "Russia", "Serbia", "Slovakia", "Slovenia",
    "Spain", "Sweden", "Switzerland", "Turkey", "Ukraine", "United Kingdom"
]

Europe_data = clean_data[clean_data["Country"].isin(Europe)]
Europe_hs = Europe_data["Happiness score"]

print("Asia:")
print("Number of countries in the sample: ", Asia_hs.shape[0], "of 48")
print("Mean Happiness score: ", Asia_hs.mean())
print("Std: ", Asia_hs.std())
print("\n")
print("Europe:")
print("Number of countries in the sample: ", Europe_hs.shape[0], "of 50")
print("Mean Happiness score: ", Europe_hs.mean())
print("Std: ", Europe_hs.std())

## Plotting distributions
sns.kdeplot(data=Asia_hs, color = "green", fill=False, label = "Asia")
sns.kdeplot(data=Europe_hs, color = "blue", fill=False, label = "Europe")
plt.legend(loc="upper right")
```

▸ **Output:**

```
Asia:
Number of countries in the sample:  35 of 48
Mean Happiness score:  -0.11307841492366978
Std:  0.8357108426092205


Europe:
Number of countries in the sample:  42 of 50
Mean Happiness score:  0.7502959765962951
Std:  0.6353833308393687
```



Note that the two means are not incredibly far from each other. Nonetheless, considering the value of the standard deviation of the two distributions, it is not that obvious that the two means can be statistically equal. Because of that, we use `scipy.stats` to efficiently calculate the t and the p-values, so that we can formally test our hypothesis.

It should be also noted that the sample of Asian countries in our data set is somewhat restricted. This suggests that, although we may have data about the most populated countries, and, thus, the test may represent a good portion of this continent, more data is highly desirable to support its conclusion. We must keep this in mind when interpreting our result.

```
from scipy import stats

## t and p-values
t_value, p_value = stats.ttest_ind(Europe_hs, Asia_hs)

print("\n>> Test results: \n")
print("t-value = ",t_value)
print("p-value = ", p_value)
```

▸ **Output:**

```
Test results:

t-value =  5.146360064489117
p-value =  2.0636714146474547e-06
```

Conclusion of the test: As the p-value of the test is very low, we conclude that we must reject the null hypothesis. Thus, the two mean Happiness scores are, in fact, significantly different. In particular, our samples suggest that Europeans tend to feel happier than Asians.

Hopefully this test illustrates the power of hypothesis testing in exploratory data analysis: we may infer significant statistical conclusions even before making a model. In particular, the more data we have about the samples, the more robust our conclusions will be.

7. Conclusion

In the present work, we performed an exploratory data analysis on the data set “World Happiness Report of 2023”. We followed the well-known best-practices of data cleaning and feature engineering, namely: dealing with null values, renaming the data attributes more concisely, re-scaling data attributes for future modeling, looking for correlations between data attributes, interpreting these correlations, and conducting a formal hypothesis test to exemplify how statistical analysis can be a powerful tool to extract predictions from the data before modeling.

In particular, from our analysis we could determine insightful characteristics about the happiness scores reported from countries around the world as well as how they correlate with attributes specific to each country. For example, we saw that happiness scores seem to be significantly correlated with net GDPs/capita and levels of Social Support. We also found evidence, via a formal hypothesis test, that Asia and Europe have, on average, significantly different happiness scores, though more data about Asian countries is highly desirable to support this conclusion.

It should be also noted that our replacement of outliers by mean values may not be the best choice depending on the models we want to implement with this data. Thus, when modeling, it is also desirable that other methods are tested in as many models as possible, such as replacing outliers by medians, or simply removing their respective rows from the data.

We hope the reader enjoyed our presentation. If you have any questions, please, feel free to email the author.