Name: Ishita More
UID: 2019130039 (A)
TE Comps
Subject: Data Analytics Lab

Exp No.: 01

Aim:

Exploratory Data Analysis

Objective:

Perform EDA such as number of data samples, number of features, number of classes, number of data samples per class, removing missing values, conversion to numbers, using a seaborn library to plot different graphs.

Theory:

Exploratory data analysis (EDA):

Exploratory data analysis (EDA) is used by data scientists to analyze and investigate data sets and summarize their main characteristics, often employing data visualization methods. It helps determine how best to manipulate data sources to get the answers you need, making it easier for data scientists to discover patterns, spot anomalies, test a hypothesis, or check assumptions.

EDA is primarily used to see what data can reveal beyond the formal modeling or hypothesis testing task and provides a provides a better understanding of data set variables and the relationships between them. It can also help determine if the statistical techniques you are considering for data analysis are appropriate.

Objectives of EDA:

- Maximize insight into the database/understand the database structure.
- Visualize potential relationships (direction and magnitude) between exposure and outcome variables.
- Detect outliers and anomalies (values that are significantly different from the other observations).
- Develop parsimonious models (a predictive or explanatory model that performs with as few exposure variables as possible) or preliminary selection of appropriate models.
- Extract and create clinically relevant variables.

Implementation:

DataSet: https://happiness-report.s3.amazonaws.com/2021/DataPanelWHR2021C2.xls

Importing the required libraries and Loading the data into the data frame:

```
import pandas as pd
import matplotlib as plt

data = pd.read_csv('DataPanelWHR2021C2.csv')
data.info()
len(data)
```

.head() displays the first 5 rows of the table:

```
data.head()
```

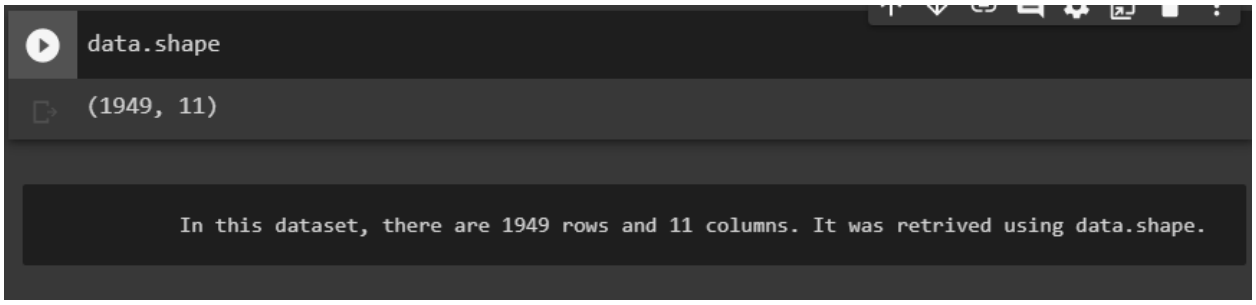
	Country name	year	Life Ladder	Log GDP per capita	Social support	Healthy life expectancy at birth	Freedom to make life choices	Generosity	Perceptions of corruption
0	Afghanistan	2008	3.724	7.370	0.451	50.80	0.718	0.168	0.882
1	Afghanistan	2009	4.402	7.540	0.552	51.20	0.679	0.190	0.850
2	Afghanistan	2010	4.758	7.647	0.539	51.60	0.600	0.121	0.707
3	Afghanistan	2011	3.832	7.620	0.521	51.92	0.496	0.162	0.731
4	Afghanistan	2012	3.783	7.705	0.521	52.24	0.531	0.236	0.776

.tail() displays the last 5 rows of the table:

```
data.tail()
```

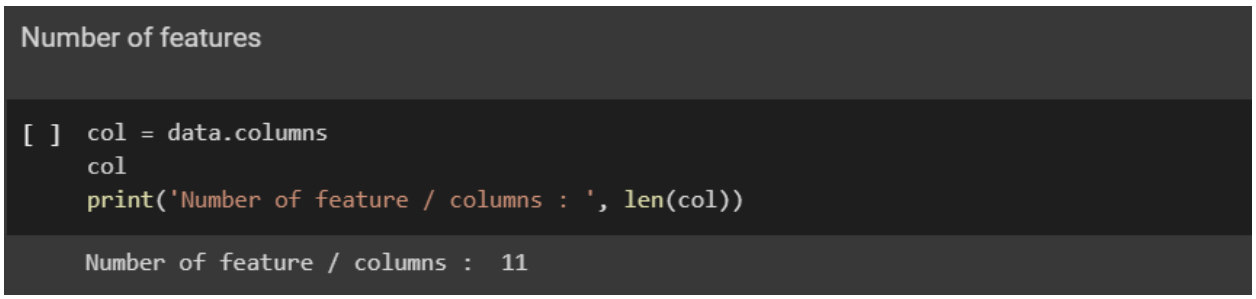
	Country name	year	Life Ladder	Log GDP per capita	Social support	Healthy life expectancy at birth	Freedom to make life choices	Generosity	Percepti corrupt
1944	Zimbabwe	2016	3.735	7.984	0.768	54.4	0.733	-0.095	0.
1945	Zimbabwe	2017	3.638	8.016	0.754	55.0	0.753	-0.098	0.
1946	Zimbabwe	2018	3.616	8.049	0.775	55.6	0.763	-0.068	0.
1947	Zimbabwe	2019	2.694	7.950	0.759	56.2	0.632	-0.064	0.
1948	Zimbabwe	2020	3.160	7.829	0.717	56.8	0.643	-0.009	0.

Under Country Name Type, Name of the different countries are present. In year type, years are shown. Corresponding to the year the various other features like life ladder social support, etc are present.

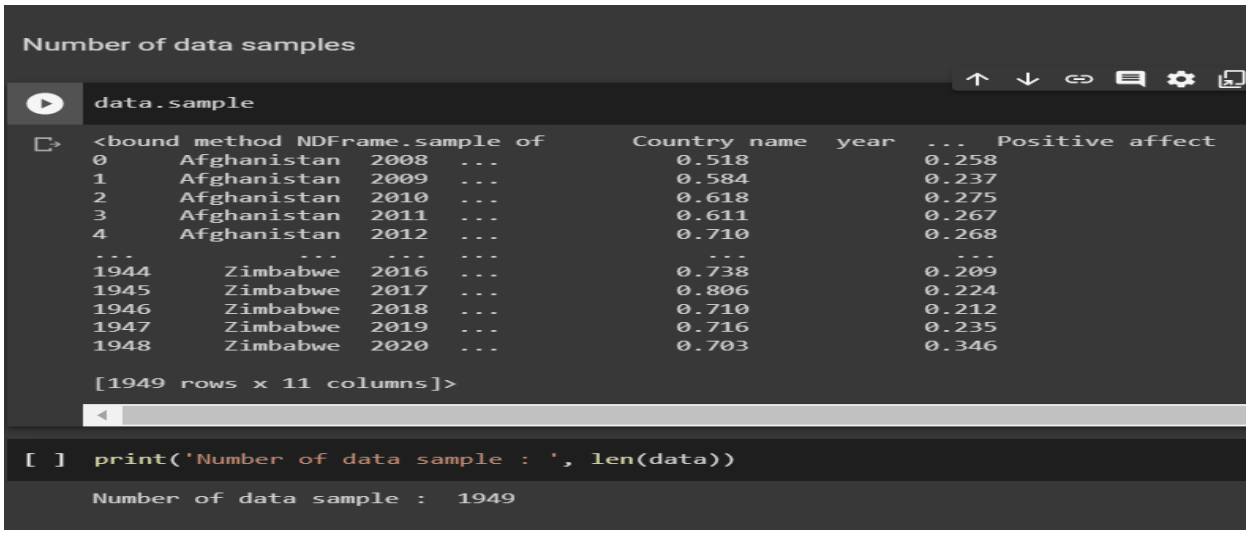


Using .shape() we can get information about the number of rows and columns of the dataset:

Number of features



Number of data samples



.nunique() helps to find number of unique elements in the object. Like in this dataset there are 166 unique country names are present.

```
data.nunique()
```

```
Country name                          166
year                                   16
Life Ladder                          1553
Log GDP per capita                   1500
Social support                        455
Healthy life expectancy at birth      828
Freedom to make life choices          535
Generosity                            609
Perceptions of corruption             572
Positive affect                       431
Negative affect                       374
dtype: int64
```

.groupby() function is used to split the data into groups based on some criteria.

```
[ ]  data.groupby(['year','Country name'])['Life Ladder'].count()
```

```
year  Country name
2005  Australia         1
      Belgium           1
      Brazil            1
      Canada            1
      Czech Republic    1
                      ..
2020  United States     1
      Uruguay           1
      Venezuela         1
      Zambia            1
      Zimbabwe          1
Name: Life Ladder, Length: 1949, dtype: int64
```

.isnull() helps to find missing values in the given series object and .notnull() helps to find existing values in the dataset

```
data.isnull().sum()
```

```
Country name                          0
year                                  0
Life Ladder                           0
Log GDP per capita                   36
Social support                       13
Healthy life expectancy at birth     55
Freedom to make life choices         32
Generosity                           89
Perceptions of corruption           110
Positive affect                      22
Negative affect                      16
dtype: int64
```

Removing missing values

Removing missing values

Dropping rows with at least 1 null value

```
#any: if any NA values are present, drop that label
data.dropna(how = 'any').shape
```

```
(1708, 11)
```

Dropping rows if all values in that row are missing

```
[ ] #all: if all values ate NA, drop that label
    data.dropna(how = 'all').shape
```

```
(1949, 11)
```

```
[ ] data.dropna(subset = ['Perceptions of corruption'], how='all').shape
```

```
(1839, 11)
```

Dropping columns with at least 1 null value

```
data.dropna(axis = 1, how ='any')
```

Null values adversely affect the performance and accuracy of any machine learning algorithm. Therefore using dropna() function, remove rows and columns with Null/NaN values.

Replacing Null value with Mean

```
[ ]  data['Social support'].fillna(float(data['Social support'].mean()), inplace = True)
     data['Log GDP per capita'].fillna(float(data['Log GDP per capita'].mean()), inplace = T
     data['Healthy life expectancy at birth'].fillna(float(data['Healthy life expectancy at
     data['Social support'].fillna(float(data['Social support'].mean()), inplace = True)
     data['Freedom to make life choices'].fillna(float(data['Freedom to make life choices'].
     data['Generosity'].fillna(float(data['Generosity'].mean()), inplace = True)
     data['Perceptions of corruption'].fillna(float(data['Perceptions of corruption'].mean()
     data['Positive affect'].fillna(float(data['Positive affect'].mean()), inplace = True)
     data['Negative affect'].fillna(float(data['Negative affect'].mean()), inplace = True)
```

Conversion to Numbers

```
[ ]  data['Life Ladder'] = data['Life Ladder'].astype('int')
```

```
[ ]  data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1949 entries, 0 to 1948
Data columns (total 11 columns):
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   Country name                      1949 non-null   object
 1   year                              1949 non-null   int64
 2   Life Ladder                       1949 non-null   int64
```

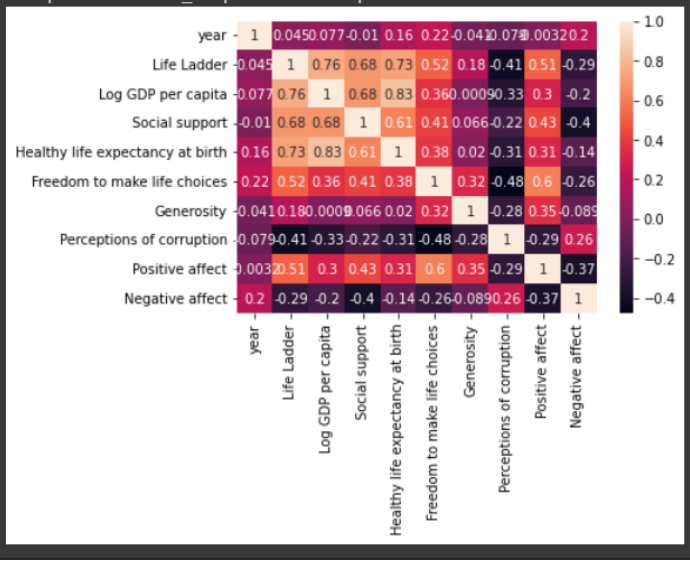
Heat Map:

We can find the dependent variables using heat map.

```
corelation = gra.corr()
```

```
sns.heatmap(corelation, xticklabels=corelation.columns, yticklabels=corelation.columns,annot=True)
```

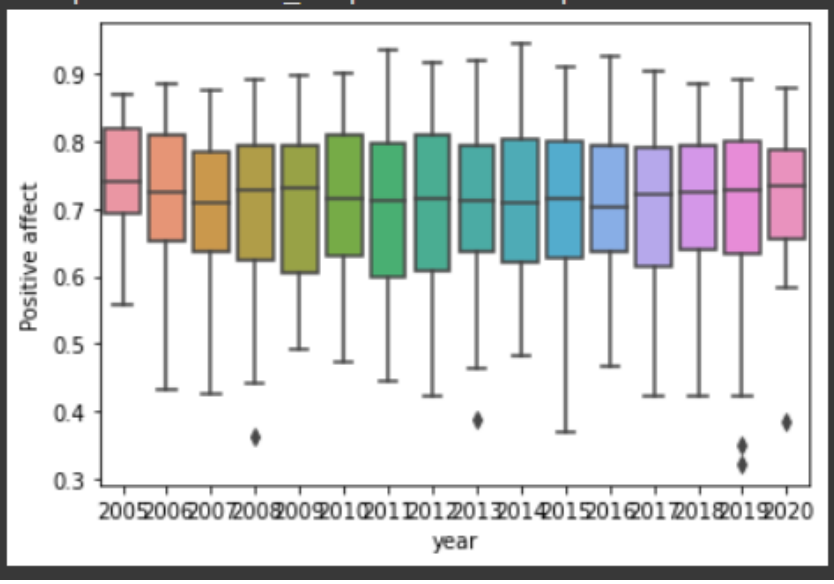
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f868fa70f10>
```



Box Plot

```
sns.boxplot(x=gra['year'], y=gra['Positive affect'])
```

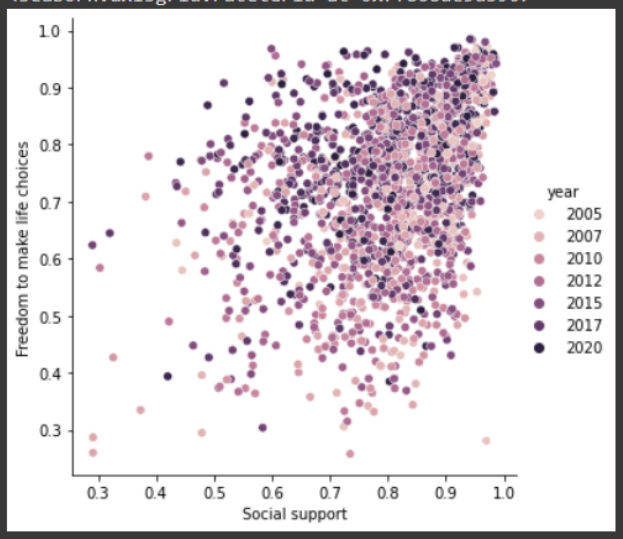
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f868af48710>
```



RelPlot:

```
sns.relplot(x= 'Social support', y= 'Freedom to make life choices', hue= 'year', data=gra)
```

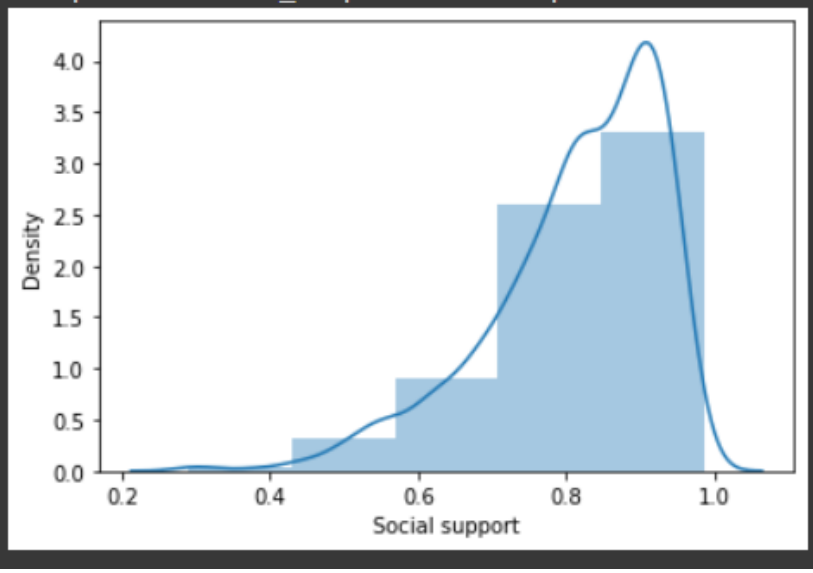
```
<seaborn.axisgrid.FacetGrid at 0x7f868de5d390>
```



DisPlot

```
[74] sns.distplot(gra['Social support'], bins=5)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7f868d45d7d0>
```



Conclusion:

I learnt the concept of Exploratory Data Analysis. Also, how to perform EDA on a dataset using various Python libraries like pandas, seaborn, matplotlib which helps to visualize the data using plots like boxplot, heat map, etc.