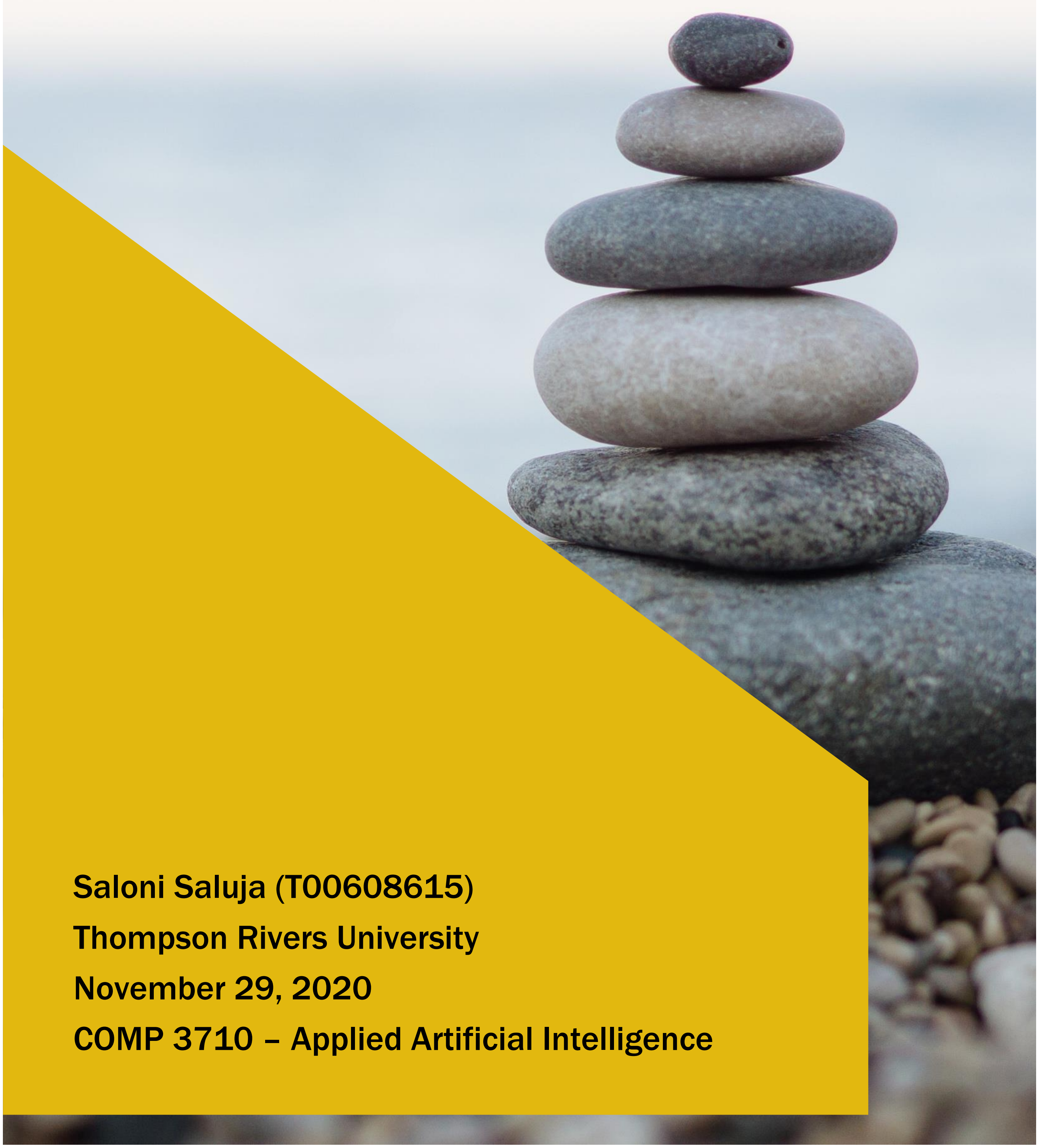
Machine Learning Project

Part II: Scientific Research Experiment Report



Saloni Saluja (T00608615)

Thompson Rivers University

November 29, 2020

COMP 3710 – Applied Artificial Intelligence

World Happiness Report

Introduction

Background Research

The World Happiness Report is a survey of the state of global happiness. The dataset I chose for this project was published in 2019 by the United Nations on March 20, 2019. March 20 is celebrated as the International Day of Happiness. This report continues to gain recognition by the world. Policy makers are increasingly using the happiness indicators suggested by this report. Leading experts from across the globe describe how we can assess the progress of nations by the well-being of their people. The dataset reviews the state of happiness across the world and shows how the new science of happiness explains personal and national variations in happiness. [1]

Purpose

The purpose of this project is to use the 2019 dataset and predict happiness score by making a Machine Learning prediction model.

Research Questions

For this project, we will try to find answers to the following research questions:

1. How can we describe the data in the 2019 dataset?
2. Can we make a machine learning model that helps us predict the happiness score based on multiple fields?
3. How do each of the features described in the data set affect the happiness score?

Summary

In this project, we use various Machine Learning techniques such as Train-Test Split and Linear Regression to analyze the 2019 World Happiness Report Dataset. The language being used for this project is Python and the corresponding libraries being used to analyze data include Pandas, NumPy, Matplotlib, Seaborn, Math, and Scikit Learn. To run the code, we are using Jupyter Notebook. The first step is to import all the libraries. The next step is to load the dataset into our Notebook. The dataset is a CSV file. Then, we use some commands to display the first and last few records of the dataset. After that, we clean the data by removing any null values from it to feed it into our Machine Learning Model. After preparing the data, we arrange it into Features Matrix and Target Vector, which is needed as input for using the Scikit learn library. Further, we split our dataset into training set and testing set and build a Linear Regression Model based on our training data. Further down the road, we use this model to make predictions on our test data. Using Seaborn and Matplotlib, we plot linear regression lines to analyze the relationship between each of our selected features and how they affect the happiness score. We also use a correlation table and build a heatmap to analyze the correlation between each of these variables. Ultimately, we describe some of the future works that can be done to improve the accuracy of our model.

Project Details

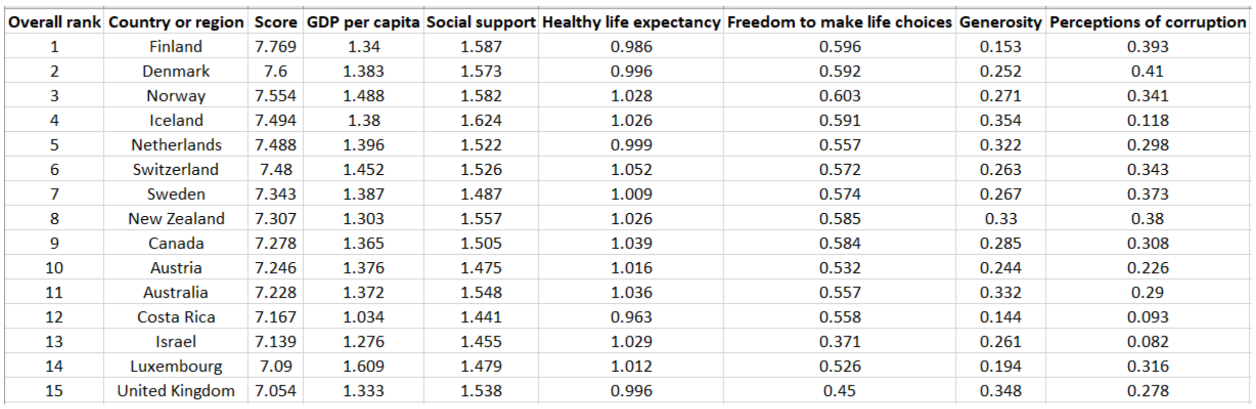
Description of Chosen Dataset

This dataset ranks 155 countries by their happiness levels. The happiness scores described in the dataset are gathered from Gallup World Poll. These scores are gathered from participants' answers given in a poll. The questions in the poll were based on main life evaluation. The metric used for evaluation is called "Cantril ladder". The respondents were asked to think of a ladder with the best possible life for them being a 10 and the worst possible life being a 0. They were supposed to rate their life on this metric. [1]

Description of Features and Predictions

There are a total of 9 fields in the dataset. The 9 fields are as follows:

1. **Overall Rank**: This field describes the overall rank of the country based on the score.
2. **Country or region:** This field describes the name of the country as per their overall rank.
3. **Score:** This field describes the happiness score of the respective country. Greater the score, more will be the overall rank.
4. **GDP per capita:** GDP per capita is a measure of a country's economic output that accounts for its number of people. [2]
5. **Social support:** This refers to having friends and family who are there for help in the time of need or crisis. They are people who can give you a broader focus and positive self-image. Social support enhances the quality of life and provides a buffer against adverse life events. [2]
6. **Health life expectancy:** This refers to the average number of years that a newborn can expect to live in "full health" without being hampered by disabling illnesses or injuries. [2]
7. **Freedom to make life choices:** This field describes an individual's opportunity and autonomy to perform an action selected from at least two available options, unconstrained by external parties. [2]
8. **Perceptions of corruption:** The Corruption Perceptions Index (CPI) is an index published annually by Transparency International since 1995. It ranks countries by their perception levels of public sector corruption, as determined by experts and opinion surveys. [2]



Overall rank	Country or region	Score	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Generosity	Perceptions of corruption
1	Finland	7.769	1.34	1.587	0.986	0.596	0.153	0.393
2	Denmark	7.6	1.383	1.573	0.996	0.592	0.252	0.41
3	Norway	7.554	1.488	1.582	1.028	0.603	0.271	0.341
4	Iceland	7.494	1.38	1.624	1.026	0.591	0.354	0.118
5	Netherlands	7.488	1.396	1.522	0.999	0.557	0.322	0.298
6	Switzerland	7.48	1.452	1.526	1.052	0.572	0.263	0.343
7	Sweden	7.343	1.387	1.487	1.009	0.574	0.267	0.373
8	New Zealand	7.307	1.303	1.557	1.026	0.585	0.33	0.38
9	Canada	7.278	1.365	1.505	1.039	0.584	0.285	0.308
10	Austria	7.246	1.376	1.475	1.016	0.532	0.244	0.226
11	Australia	7.228	1.372	1.548	1.036	0.557	0.332	0.29
12	Costa Rica	7.167	1.034	1.441	0.963	0.558	0.144	0.093
13	Israel	7.139	1.276	1.455	1.029	0.371	0.261	0.082
14	Luxembourg	7.09	1.609	1.479	1.012	0.526	0.194	0.316
15	United Kingdom	7.054	1.333	1.538	0.996	0.45	0.348	0.278

Figure 1: Few rows of the dataset

The fields that will form a part of the features matrix are as follows:

1. **GDP per capita**
2. **Social support**
3. **Healthy life expectancy**
4. **Freedom to make life choices**
5. **Generosity**
6. **Perceptions of corruption**

The field that will act as the target vector is **Score.**

From the above-mentioned features, we will try to predict the value of score based on machine learning algorithms. For instance, we can build a model that finds the correlation between GDP per capita and score. Similarly, we can use other fields to decipher their relationship with our target vector, i.e. Score.

Problem being tried to solve

The goal of using this dataset is to use the features matrix described above to predict the target vector, which is the happiness score. The second goal is to find out the effect of different factors such as GDP per capita, Social support, Healthy life expectancy, Freedom to make life choices, Generosity, and Perceptions of corruption on happiness score. For our model, we first divide the dataset into features matrix 'X' and target vector y. Further, we split them into X_train, X_test, y_train, and y_test. About 75 to 80% of the data was used to train the model and about 20 to 25% data was used for testing the model. To find out the effect of each of these variables, we will plot regression lines for these features against Score. If the graph shows a line at an angle with one of the axes, this would mean that the Score increases steadily with an increase in that feature. If we get a line which is almost parallel to one of the axes, this would mean that the variable is (almost) constant does not have much effect on the happiness Score.

Preparing the Data

Machine Learning helps us to find patterns in data. These patterns can be used to make predictions about new data points. In order to get correct predictions out of our model, we must clean the raw data available in our dataset before feeding it into our machine learning prediction model. Data preparation is one of the most complex steps in any Machine Leaning Project.

Scikit learn, which is one of the libraries used for this project requires the data to be free of missing values. If we don't remove or impute missing values from our dataset, we will get an error message. To pre-process the data for our Machine Learning Model, first of all we find out the total number of entries (total number of rows and columns contained in the dataset). After that, we check the total number of null values present. If the number of null values comes out to be more than one, we can either remove the rows that contain a null value, or we can replace the null value with the average of other values in the column. To do this, we need to check the data type of each column. Columns with integer datatype that have missing values will be replaced with integer. Similarly, columns with float or double datatype that have missing values will be replaces by float/double values. After the raw data becomes free of any missing values, we can proceed on to the next steps.

Description of Machine Leaning Platforms (Packages / APIs / Libraries) Used in this Project

The programming language that have be used to build the machine learning model is **Python**.

The packages/APIs/libraries that have be used are as under:

1. **Pandas**
   Pandas is a Python package that provides data structures that make working with "labelled" data easy and intuitive. It is one of the most powerful and flexible open source data analysis / manipulation tool available. Pandas is built on top of the Numpy package and its key data structure is called "DataFrame". Data frames allow us to store and manipulate tabular data in rows of observations and columns of variables. [3]
2. **NumPy**
   NumPy is a python library which is used to work with arrays. It also has functions that can be used for working with linear algebra, fourier transformation, and matrices. [4]
   Unlike lists, NumPy arrays are stored at one continuous place in the memory. Hence, the processes can access and manipulate them very efficiently. NumPy is also optimized to work with the latest CPU architectures. [4]

3. **<u>Matplotlib</u>**
Matplotlib is an open source library. It was created by John D. Hunter. It is a low-level graph plotting library in python that serves as visualization utility. [5]

Matplotlib comes with a variety of 2D / 3D plots. These plots help us in understanding the trends, patterns, and to make correlations. One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots such as line, bar, scatter, histogram, etc. [5]

4. **<u>Seaborn</u>**
Seaborn helps us to explore and understand our data. It is a library that uses Matplotlib underneath to plot graphs. It is used to visualize random distributions. The plotting functions offered by Seaborn operate on data frames and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots. [6]

5. **<u>Scikit learn</u>**
Scikit learn provides a range of supervised and unsupervised learning algorithms. This library is built upon the SciPy (Scientific Python) library that must be installed before we can use scikit learn. This stack includes: [7]
   - NumPy: Base n-dimensional array package [7]
   - SciPy: Fundamental library for scientific computing [7]
   - Matplotlib: Comprehensive 2D / 3D plotting [7]
   - IPython: Enhances interactive console [7]
   - Sympy: Symbolic mathematics [7]
   - Pandas: Data structures and analysis [7]

The prime focus of the scikit library is on modelling data. It's main focus does not include loading, manipulating, or summarizing data. Some popular groups of models provided by scikit learn include the following: [7]
   - Clustering: This is used for grouping unlabeled data such as KMeans. Since out model is based on supervised learning, therefore, we will not be using this in the scope of this project. [7]
   - Cross Validation: This is used for estimating the performance of models build on supervised learning on unseen data. [7]
   - Datasets: Scikit learn has some inbuilt datasets for investigating model behaviors. [7]
   - Ensemble Methods: This is for combining the predictions of multiple supervised models. [7]
   - Feature Extraction: This is for defining attributes in images and text data. [7]
   - Feature selection: This is for identifying meaningful attributes using which we can create models based on supervised learning. [7]

- Parameter tuning: This is for getting the most out of supervised models. [7]
- Manifold Learning: This is for summarizing and depicting complex multi-dimensional data. [7]
- Supervised Models: A vast variety of array not limited to generalized linear models, discriminate Analysis, Naïve Bayes, Lazy Methods, Neural Networks, Support Vector Machines, and Decision Trees. [7]

Methods

Machine Learning Techniques

The machine learning models and techniques used in this project are listed below:

1. **Train-Test Split**
   This procedure involves splitting a data set and dividing it into two subsets. The first subset is used to fit the model and it known as **training set.** The second subset is used to evaluate the fit machine learning model and is known as the **test dataset.** [8]
   <u>Reason for choosing train-test split</u>
   Train-test split has been used to determine the performance of our model when it is being used to make predictions on new data which has not been used for training.

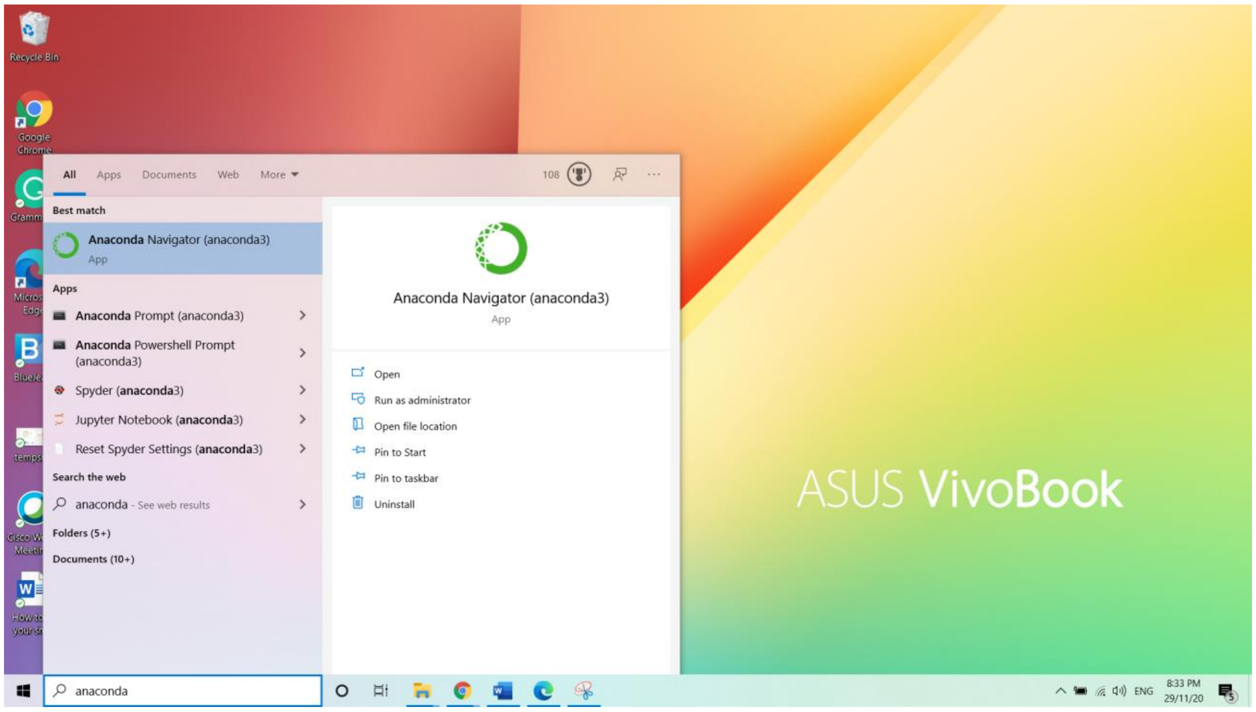
2. **Linear Regression**
   The goal of linear regression is to find the best fit line that can accurately predict the output for continuous dependent variable. If a single independent variable is used for prediction, then the model is called Simple Linear Regression. If more than two independent variables are used, then the model is known as Multiple Linear Regression. By finding the best fit line, the algorithm establishes a relationship between dependent variable and independent variable. This relationship has a "linear" nature. The output of linear regression should only be continuous values. [9]
   <u>Reason for choosing Linear Regression</u>
   This technique has been used to predict the score from the features. Score is a continuous value. Therefore, Linear Regression has been used for the project.

Experiments

Instructions to load Jupyter Notebook on local machine

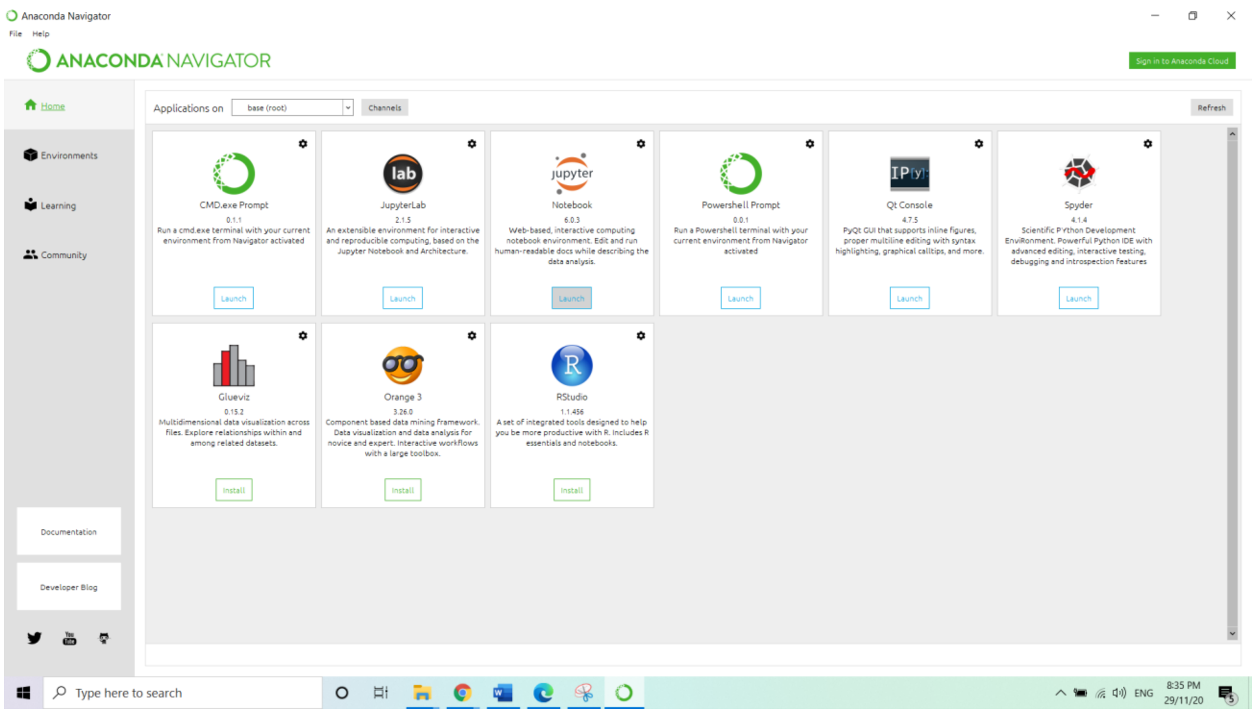
To run the code on your local machine, you need to install Anaconda. Anaconda Navigator comes with the Anaconda distribution of Python. Please download the notebook from the root folder to your computer.

Step 1: Open Anaconda Navigator using the Windows start menu.


All
Apps
Documents
Web
More
Best match
Anaconda Navigator (anaconda3)
App
Apps
Anaconda Prompt (anaconda3)
Anaconda Powershell Prompt (anaconda3)
Spyder (anaconda3)
Jupyter Notebook (anaconda3)
Reset Spyder Settings (anaconda3)
Search the web
anaconda - See web results
Folders (5+)
Documents (10+)
Anaconda Navigator (anaconda3)
App
Open
Run as administrator
Open file location
Pin to Start
Pin to taskbar
Uninstall
ASUS VivoBook
anaconda

The Anaconda Navigator window will open.

Step 2: In the middle of the page, in the Jupyter notebook tile, click the Launch button.



Run a cmd.exe terminal with your current environment from Navigator activated

An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Run a Powershell terminal with your current environment from Navigator activated

PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

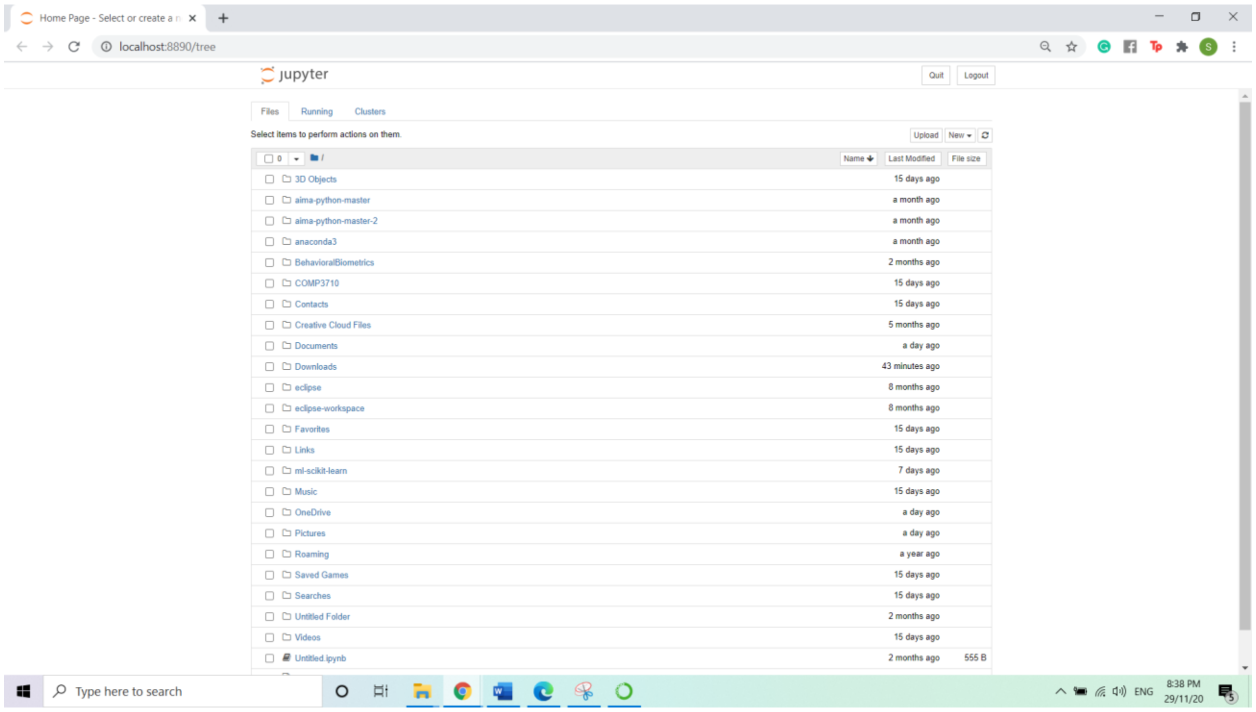
Multidimensional data visualization across files. Explore relationships within and among related datasets.

Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.

A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.

A Jupyter file browser will open in a web browser tab.

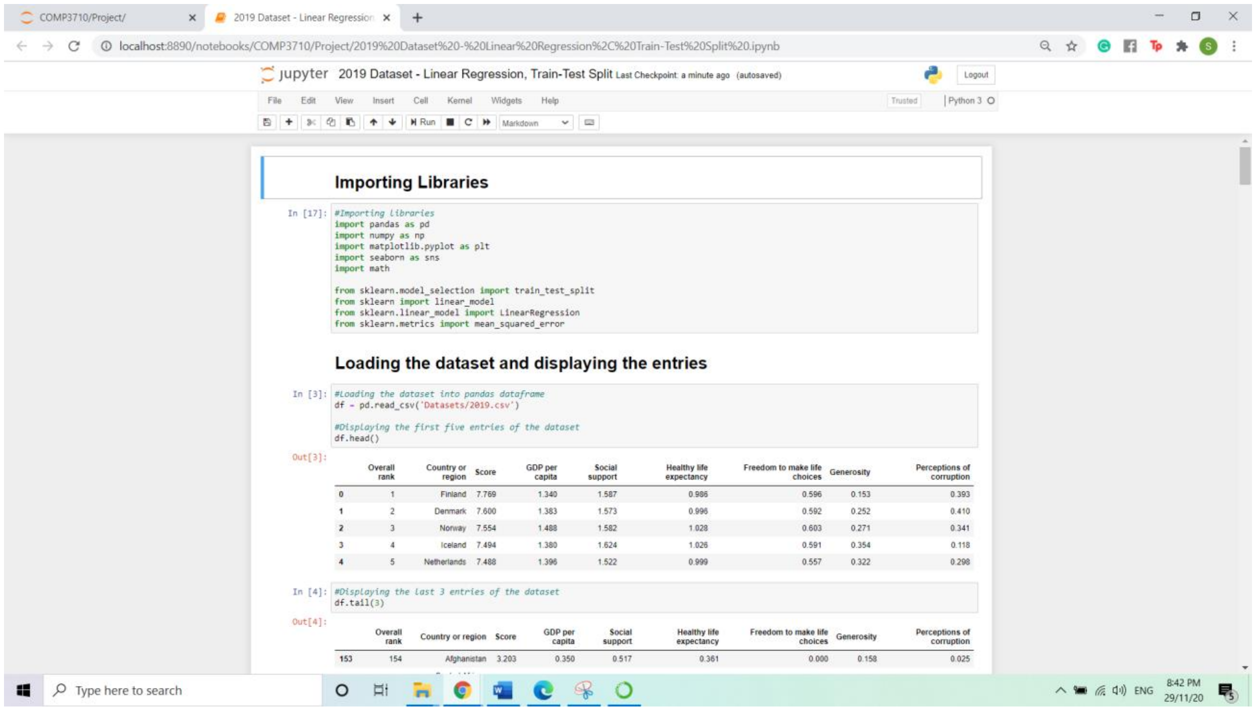
Step 3: This screen displays all the files in the root folder. Switch to the folder where you have saved the notebook.



Select items to perform actions on them.

Name	Last Modified	File size
3D Objects	15 days ago	
aima-python-master	a month ago	
aima-python-master-2	a month ago	
anaconda3	a month ago	
BehavioralBiometrics	2 months ago	
COMP3710	15 days ago	
Contacts	15 days ago	
Creative Cloud Files	5 months ago	
Documents	a day ago	
Downloads	43 minutes ago	
eclipse	8 months ago	
eclipse-workspace	8 months ago	
Favorites	15 days ago	
Links	15 days ago	
ml-scikit-learn	7 days ago	
Music	15 days ago	
OneDrive	a day ago	
Pictures	a day ago	
Roaming	a year ago	
Saved Games	15 days ago	
Searches	15 days ago	
Untitled Folder	2 months ago	
Videos	15 days ago	
Untitled.ipynb	2 months ago	555 B

Step 4: Open up the 2019 Dataset – Linear Regression, Train-Test Split.ipynb notebook.


COMP3710/Project/
2019 Dataset - Linear Regression
localhost:8890/notebooks/COMP3710/Project/2019%20Dataset%20-%20Linear%20Regression%2C%20Train-Test%20Split%20.ipynb
jupyter
2019 Dataset - Linear Regression, Train-Test Split Last Checkpoint: a minute ago (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3
Run
Markdown
Type here to search
ENG
8:42 PM
29/11/20

Importing Libraries

```
In [17]: #Importing libraries
         import pandas as pd
         import numpy as np
         import matplotlib.pyplot as plt
         import seaborn as sns
         import math

         from sklearn.model_selection import train_test_split
         from sklearn import linear_model
         from sklearn.linear_model import LinearRegression
         from sklearn.metrics import mean_squared_error
```

Loading the dataset and displaying the entries

```
In [3]: #Loading the dataset into pandas dataframe
        df = pd.read_csv('Datasets/2019.csv')

        #Displaying the first five entries of the dataset
        df.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]: #Displaying the last 3 entries of the dataset
        df.tail(3)
```

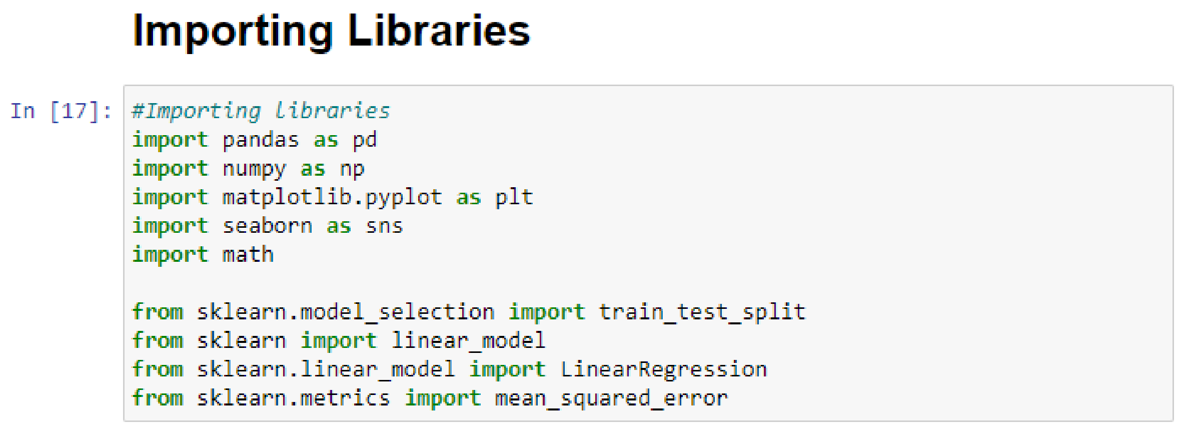
Out[4]:

	Overall rank	Country or region	Score	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Generosity	Perceptions of corruption
153	154	Afghanistan	3.203	0.350	0.517	0.361	0.000	0.158	0.025

Procedure

Step 1: Importing Libraries

First, we imported all the libraries into our project in Jupyter Notebook, including Pandas, NumPy, Matplotlib, Seaborn, and Scikit learn. From Scikit learn, we imported train test split, which is used to split the dataset into training set and testing set.

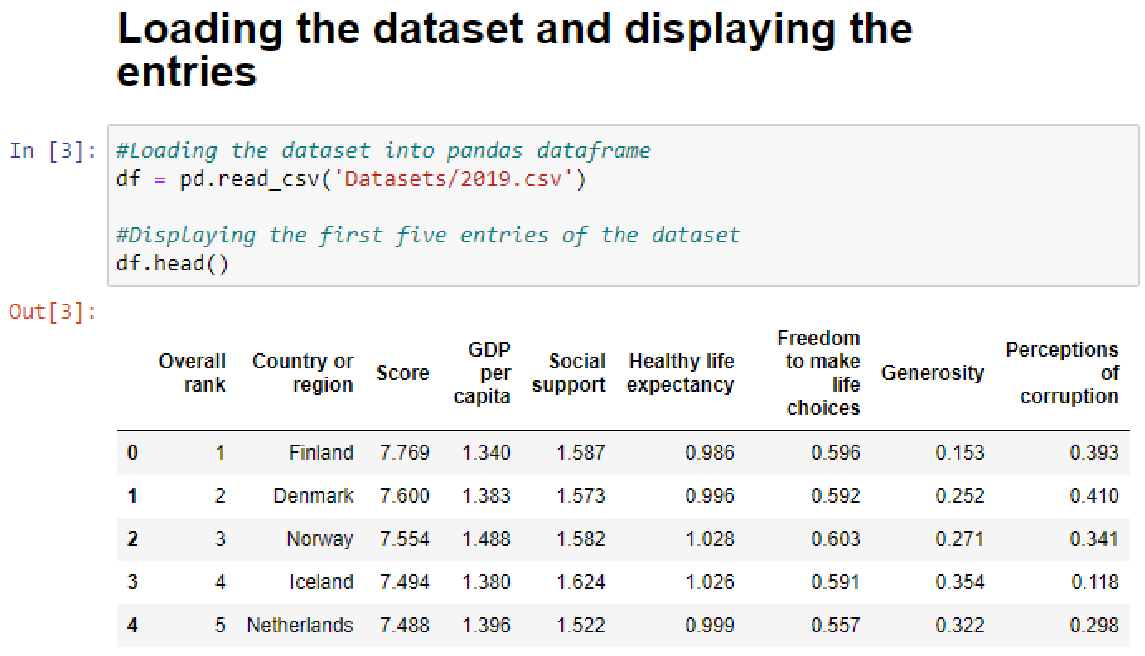


Importing Libraries

In [17]:
```python
#Importing libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import math

from sklearn.model_selection import train_test_split
from sklearn import linear_model
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
```

Step 2: Loading the dataset and displaying the entries

In this step, we loaded the 2019 World Happiness Report dataset into the Pandas Dataframe. We used head() function to display the first 5 entries of the dataset and tail(3) to display the last 3 entries of the dataset.

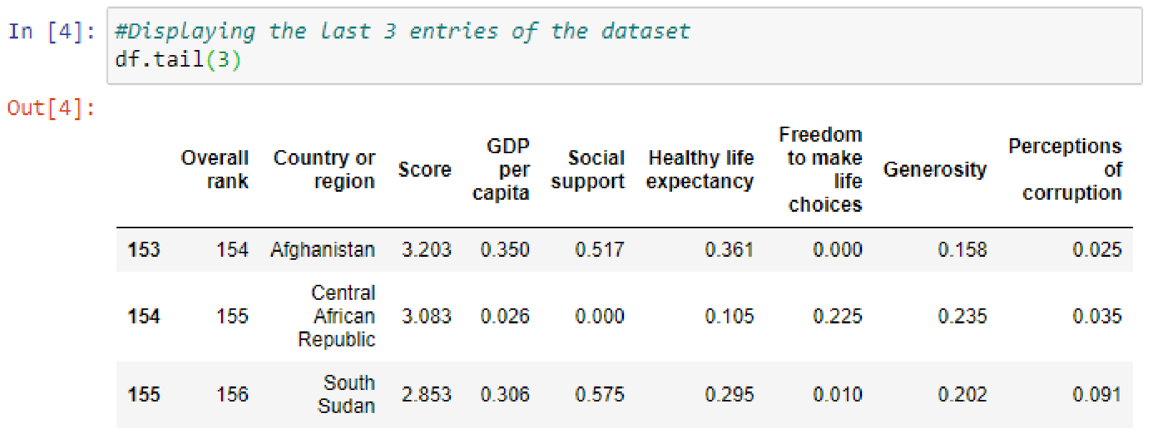


Loading the dataset and displaying the entries

In [3]:
```python
#Loading the dataset into pandas dataframe
df = pd.read_csv('Datasets/2019.csv')

#Displaying the first five entries of the dataset
df.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

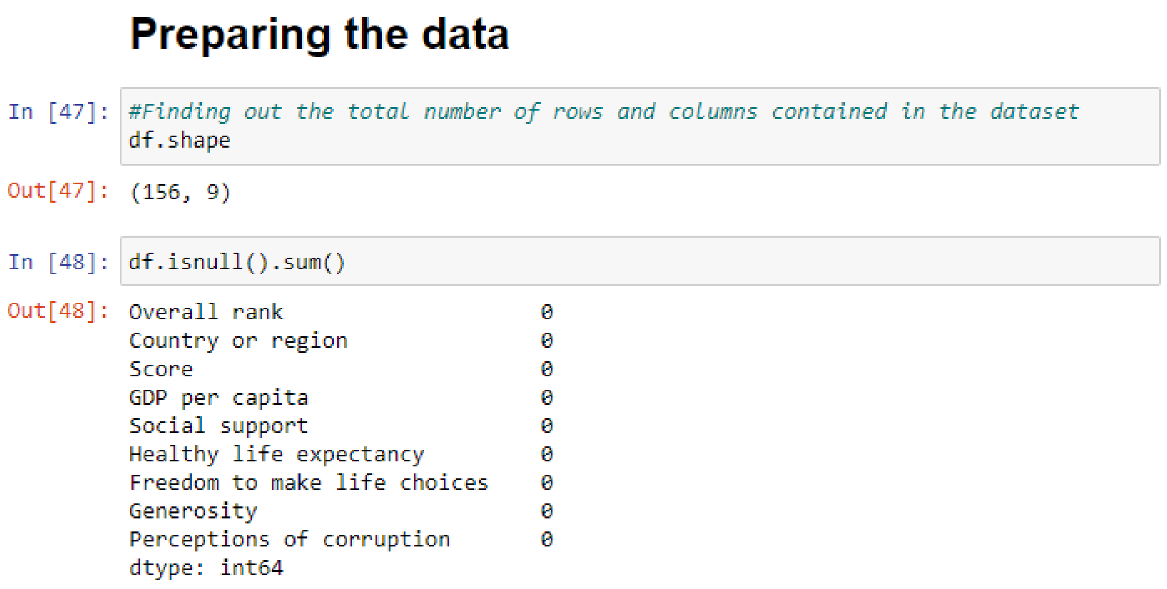


```
In [4]:  #Displaying the last 3 entries of the dataset
         df.tail(3)

Out[4]:
```

	Overall rank	Country or region	Score	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Generosity	Perceptions of corruption
153	154	Afghanistan	3.203	0.350	0.517	0.361	0.000	0.158	0.025
154	155	Central African Republic	3.083	0.026	0.000	0.105	0.225	0.235	0.035
155	156	South Sudan	2.853	0.306	0.575	0.295	0.010	0.202	0.091

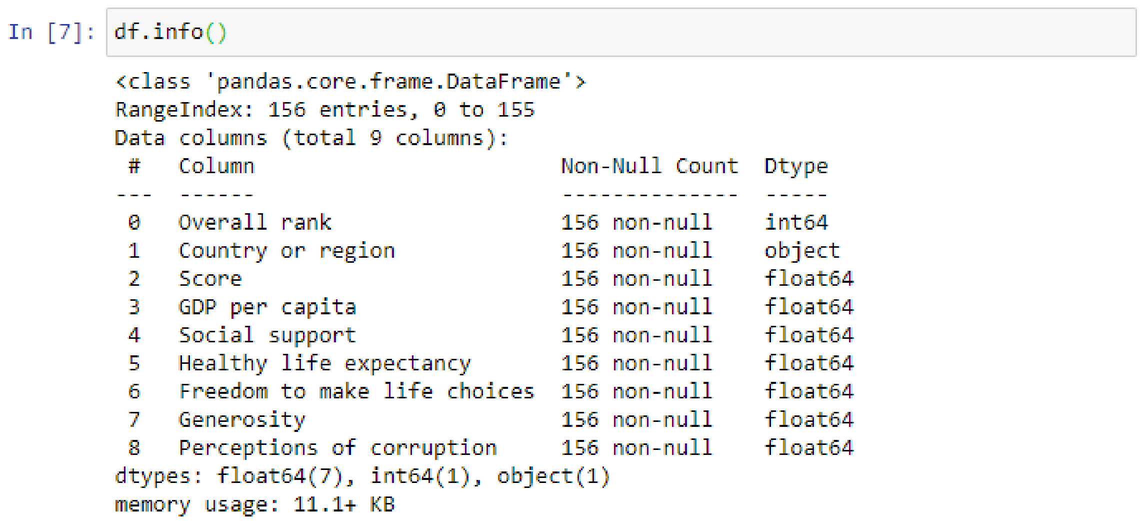
Step 4: Preparing the Data

Scikit lean requires the data to be free of all missing values. In this step, we cleaned our raw dataset to feed it to our Linear Regression Model. For that, we attempted to find the number of rows and columns contained in the dataset. On using the shape command, the output that we received was (156, 9). This means that our dataset contains 156 rows and 9 columns. Further, we used info() to find out the data type of each column and isnull().sum() function to find out the total number of entries in each column that contain a null value.



Preparing the data

```
In [47]:  #Finding out the total number of rows and columns contained in the dataset
          df.shape

Out[47]:  (156, 9)

In [48]:  df.isnull().sum()

Out[48]:  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
```



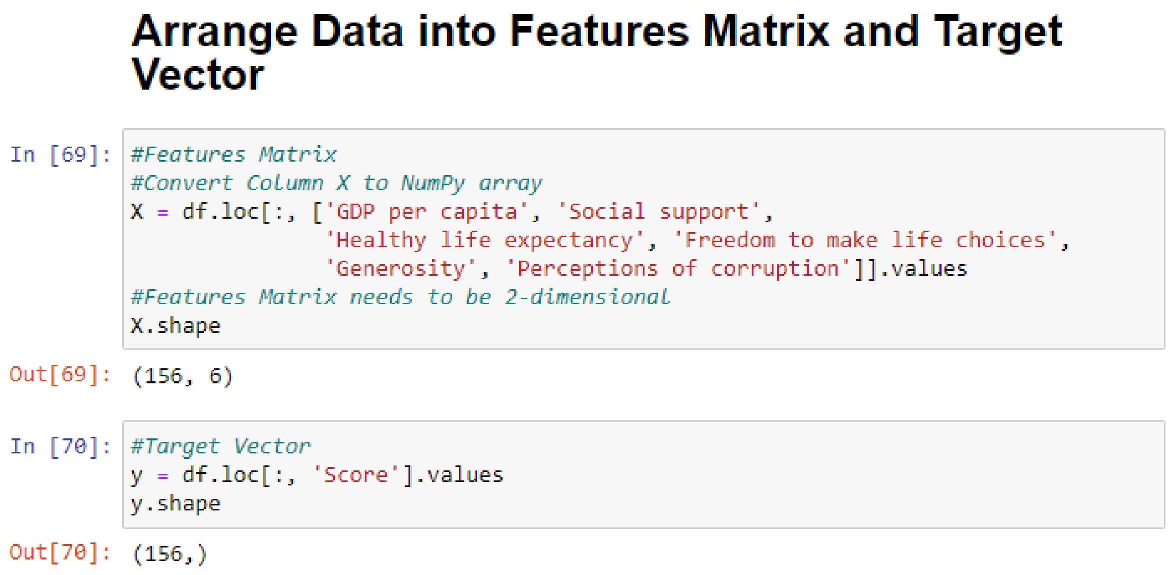
```
In [7]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 156 entries, 0 to 155
Data columns (total 9 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   Overall rank                  156 non-null    int64  
 1   Country or region             156 non-null    object 
 2   Score                         156 non-null    float64
 3   GDP per capita                156 non-null    float64
 4   Social support                156 non-null    float64
 5   Healthy life expectancy       156 non-null    float64
 6   Freedom to make life choices  156 non-null    float64
 7   Generosity                    156 non-null    float64
 8   Perceptions of corruption     156 non-null    float64
dtypes: float64(7), int64(1), object(1)
memory usage: 11.1+ KB
```

Step 5: Arranging Data into Features Matrix and Target Vector

This step involves arranging columns into features matrix and target vector. Following columns were included in the features matrix:

1. GDP per capita
2. Social support
3. Healthy life expectancy
4. Freedom to make life choices
5. Generosity
6. Perceptions of corruption

All these features were used to predict the value of the target vector **Score.**

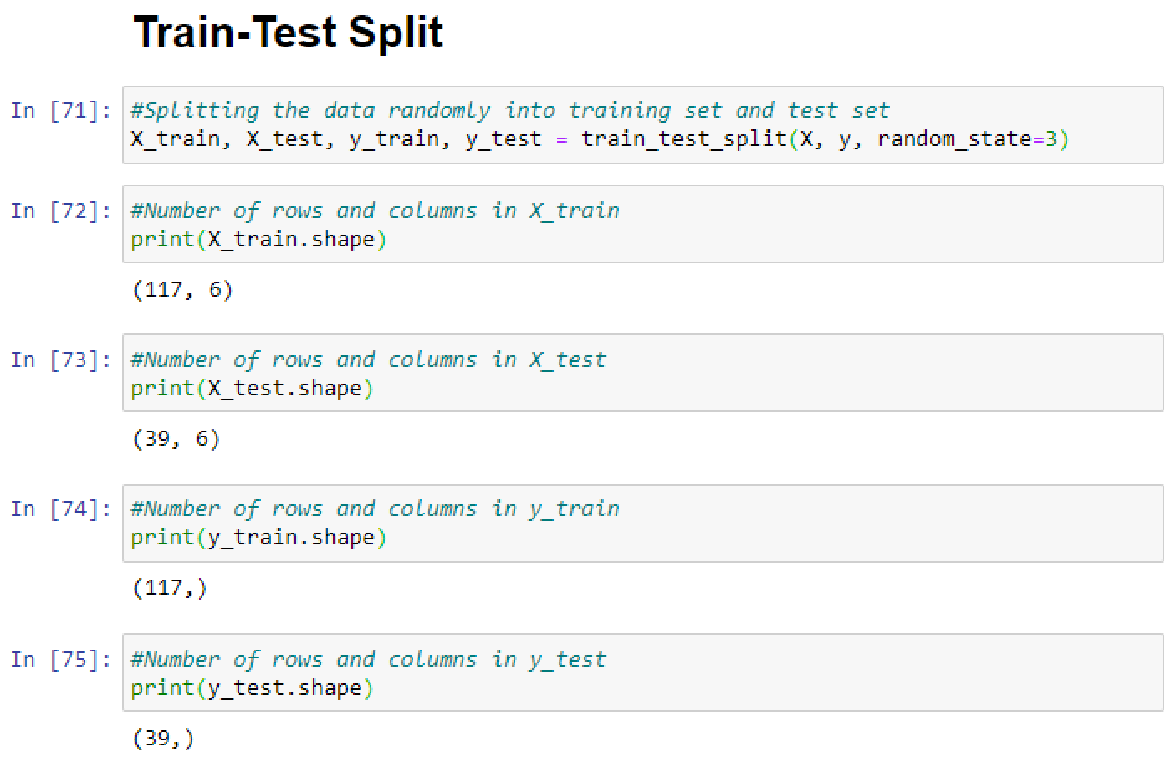


Arrange Data into Features Matrix and Target Vector

```
In [69]: #Features Matrix
         #Convert Column X to NumPy array
         X = df.loc[:, ['GDP per capita', 'Social support',
                        'Healthy life expectancy', 'Freedom to make life choices',
                        'Generosity', 'Perceptions of corruption']].values
         #Features Matrix needs to be 2-dimensional
         X.shape

Out[69]: (156, 6)

In [70]: #Target Vector
         y = df.loc[:, 'Score'].values
         y.shape

Out[70]: (156,)
```

Step 6: Train-Test Split

Further, we randomly split the records in the dataset into X_train, X_test, y_train, and y_test.

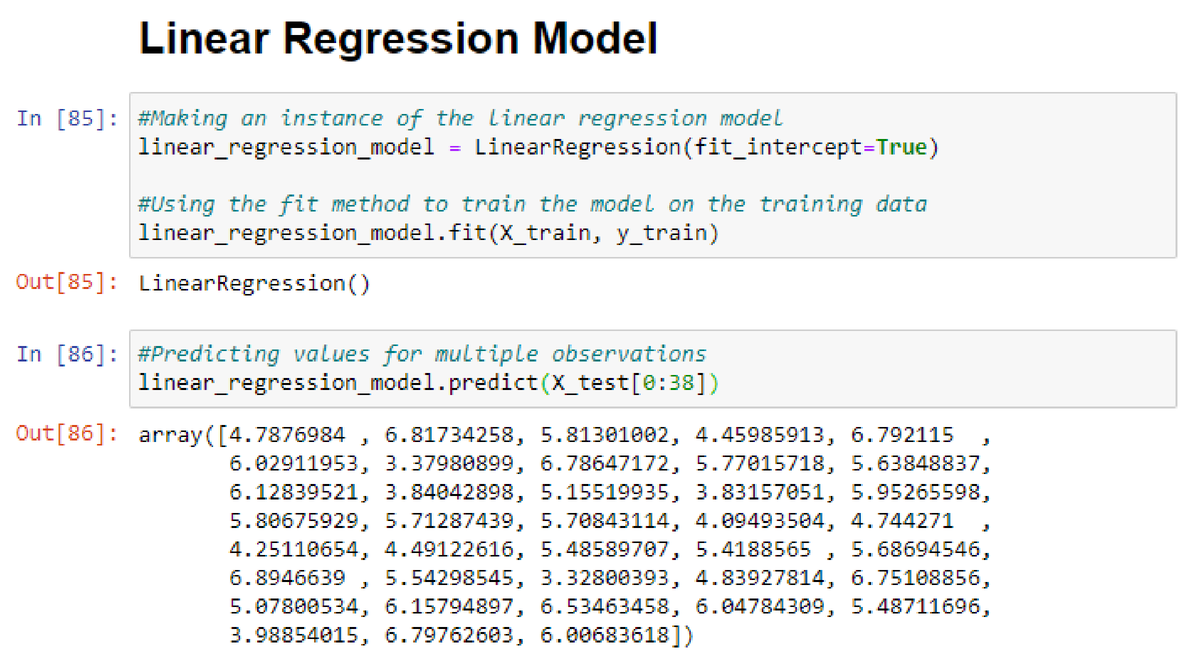


Train-Test Split

```
In [71]: #Splitting the data randomly into training set and test set
         X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=3)

In [72]: #Number of rows and columns in X_train
         print(X_train.shape)

         (117, 6)

In [73]: #Number of rows and columns in X_test
         print(X_test.shape)

         (39, 6)

In [74]: #Number of rows and columns in y_train
         print(y_train.shape)

         (117,)

In [75]: #Number of rows and columns in y_test
         print(y_test.shape)

         (39,)
```

Step 7: Building Linear Regression Model

The next step is to build a Linear Regression Model using Scikit learn. At the very beginning of the notebook, we already imported the linear regression model from scikit learn library. From here, we can make an instance of the model. Then, we can use the fit() method to train the model on the training dataset, which is X_train and Y_train. X_train contains the features and y_train contains the score values based on features in X_train. From here, we can predict values of for our testing data.

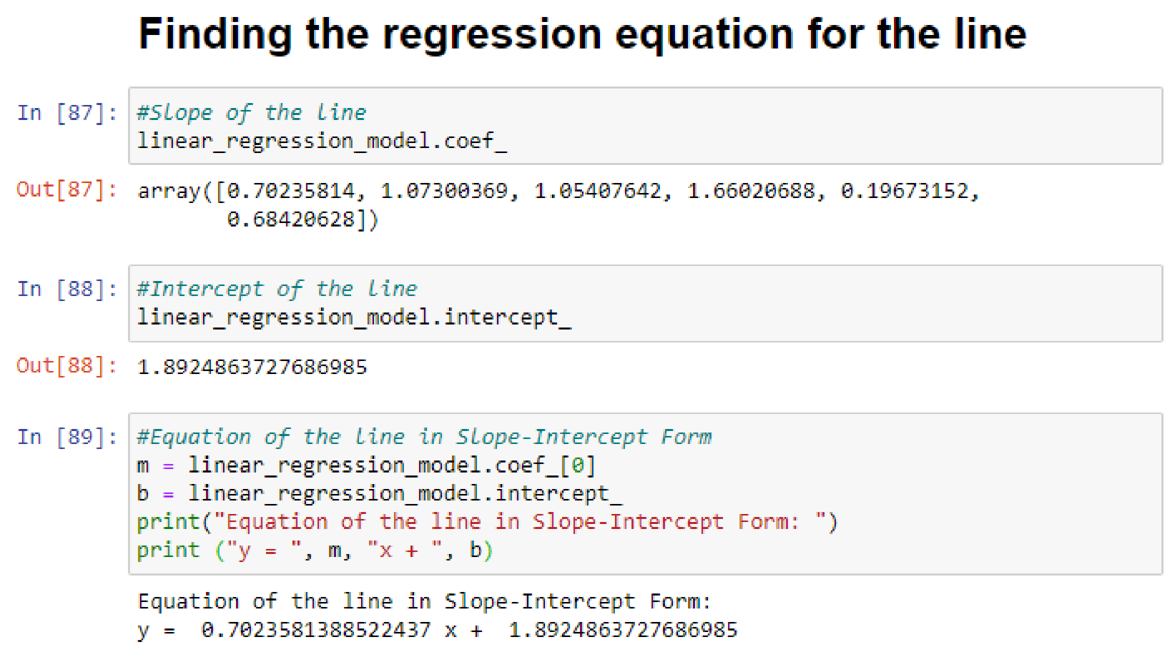


Linear Regression Model

```
In [85]: #Making an instance of the linear regression model
         linear_regression_model = LinearRegression(fit_intercept=True)

         #Using the fit method to train the model on the training data
         linear_regression_model.fit(X_train, y_train)

Out[85]: LinearRegression()

In [86]: #Predicting values for multiple observations
         linear_regression_model.predict(X_test[0:38])

Out[86]: array([4.7876984 , 6.81734258, 5.81301002, 4.45985913, 6.792115  ,
                6.02911953, 3.37980899, 6.78647172, 5.77015718, 5.63848837,
                6.12839521, 3.84042898, 5.15519935, 3.83157051, 5.95265598,
                5.80675929, 5.71287439, 5.70843114, 4.09493504, 4.744271  ,
                4.25110654, 4.49122616, 5.48589707, 5.4188565 , 5.68694546,
                6.8946639 , 5.54298545, 3.32800393, 4.83927814, 6.75108856,
                5.07800534, 6.15794897, 6.53463458, 6.04784309, 5.48711696,
                3.98854015, 6.79762603, 6.00683618])
```

Step 8: Finding Regression Equation for the Line

Further, I conducted some experiments to find the equation of the regression line. It is in the form of y = mx + c, where m = slope or co-efficient and c = intercept.



Finding the regression equation for the line

```
In [87]: #Slope of the line
         linear_regression_model.coef_

Out[87]: array([0.70235814, 1.07300369, 1.05407642, 1.66020688, 0.19673152,
                0.68420628])

In [88]: #Intercept of the line
         linear_regression_model.intercept_

Out[88]: 1.8924863727686985

In [89]: #Equation of the line in Slope-Intercept Form
         m = linear_regression_model.coef_[0]
         b = linear_regression_model.intercept_
         print("Equation of the line in Slope-Intercept Form: ")
         print ("y = ", m, "x + ", b)

         Equation of the line in Slope-Intercept Form:
         y =  0.7023581388522437 x +  1.8924863727686985
```

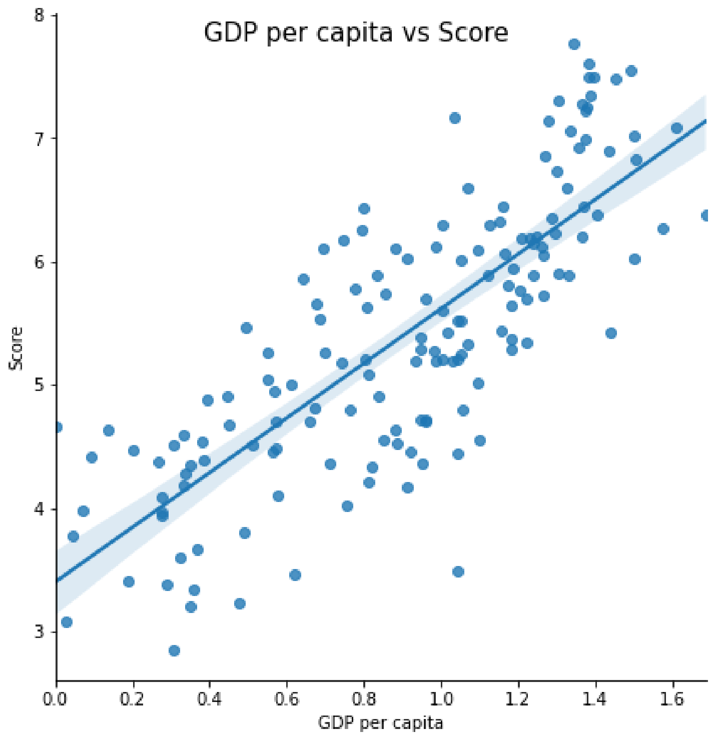
Results

In the upcoming steps, we attempt to find out the relationship between the Score vs each of the different fields in our dataset by using Seaborn for drawing graphs. The blue dots represent the data points and the blue line is the best fit regression line. Linear regression involves finding the best line to fit two variables so that one variable can be used to predict the other.

Relationship between GDP per capita and Score

```
In [90]: sns.pairplot(df, x_vars='GDP per capita', y_vars='Score', height=6, kind='reg')
         plt.suptitle("GDP per capita vs Score", fontsize=15)
```

```
Out[43]: Text(0.5, 0.98, 'GDP per capita vs Score')
```

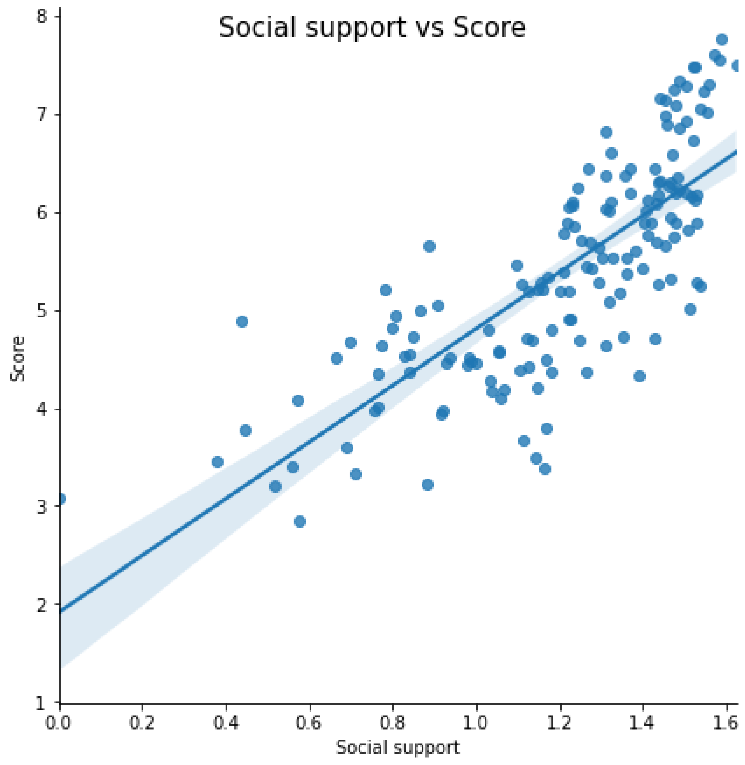


From the above graph, it is clear that happiness sore increases with an increase in GDP per capita. The graph also depicts a linear relation with positive correlation. Increase in GDP per capita pushes happiness score upward. Also, the data points are distributed equally throughout the graph.

Relationship between Social Support and Score

```
In [91]: sns.pairplot(df, x_vars='Social support', y_vars='Score', height=6, kind='reg')
         plt.suptitle("Social support vs Score", fontsize=15)
```

```
Out[46]: Text(0.5, 0.98, 'Social support vs Score')
```

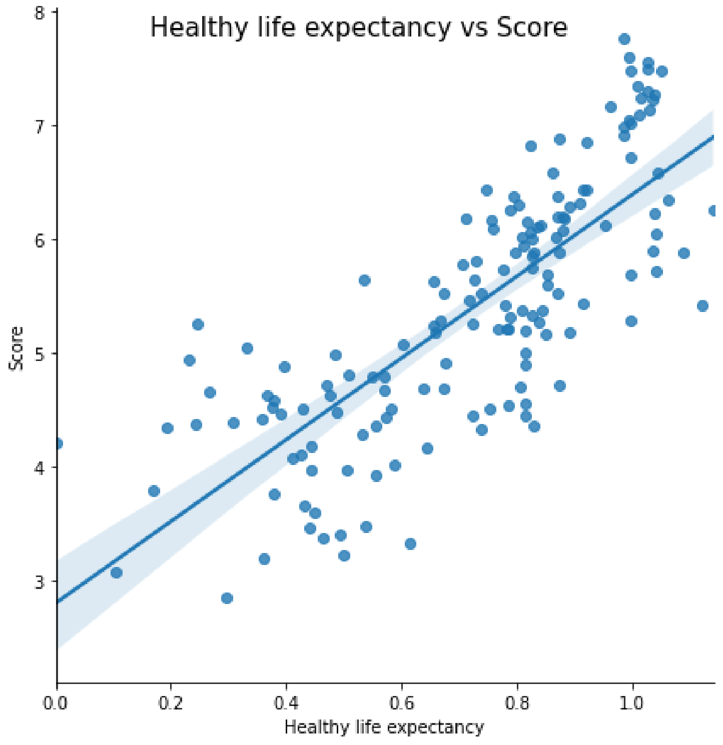


From the above graph, it is clear that the score increases with an increase in Social support. The distribution of data points is not even, though. Most of the data points are concentrated between 1.2 and 1.6. There are a few outliers between 0.2 to 0.4.

Relationship between Healthy life expectancy and Score

```
In [48]: sns.pairplot(df, x_vars='Healthy life expectancy', y_vars='Score', height=6, kin
         plt.suptitle("Healthy life expectancy vs Score", fontsize=15)
```

```
Out[48]: Text(0.5, 0.98, 'Healthy life expectancy vs Score')
```

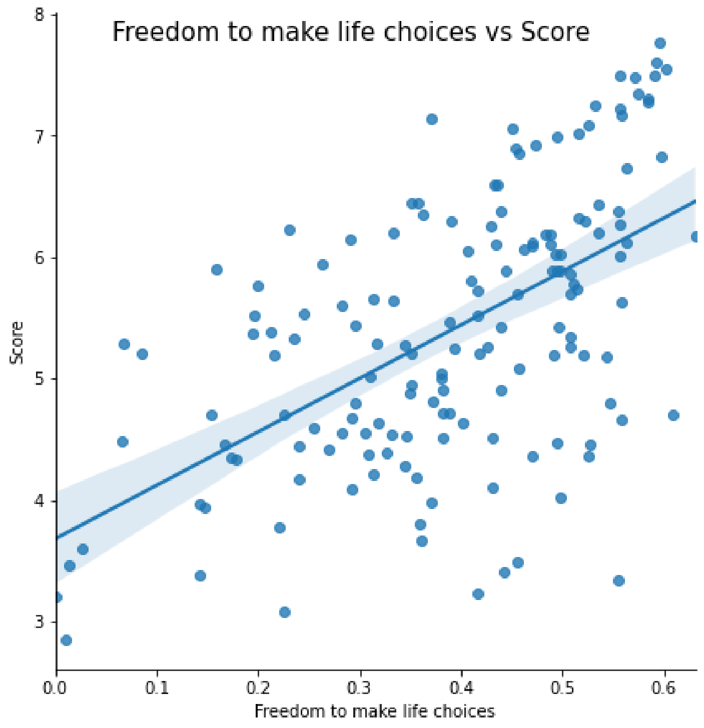


The relationship between Score and Healthy life expectancy is also linear. Score increases with an increase in healthy life expectancy. The distribution of data points is slightly uneven. There are very few data points between 0.0 – 0.4, 0.6 – 0.7. Most of the data points are concentrated between 0.4 - 0.6, 0.7 – 1.0 and so on.

Relationship between Freedom to make life choices vs Score

```
In [50]: sns.pairplot(df, x_vars='Freedom to make life choices', y_vars='Score', height=6
         plt.suptitle("Freedom to make life choices vs Score", fontsize=15)
```

```
Out[50]: Text(0.5, 0.98, 'Freedom to make life choices vs Score')
```

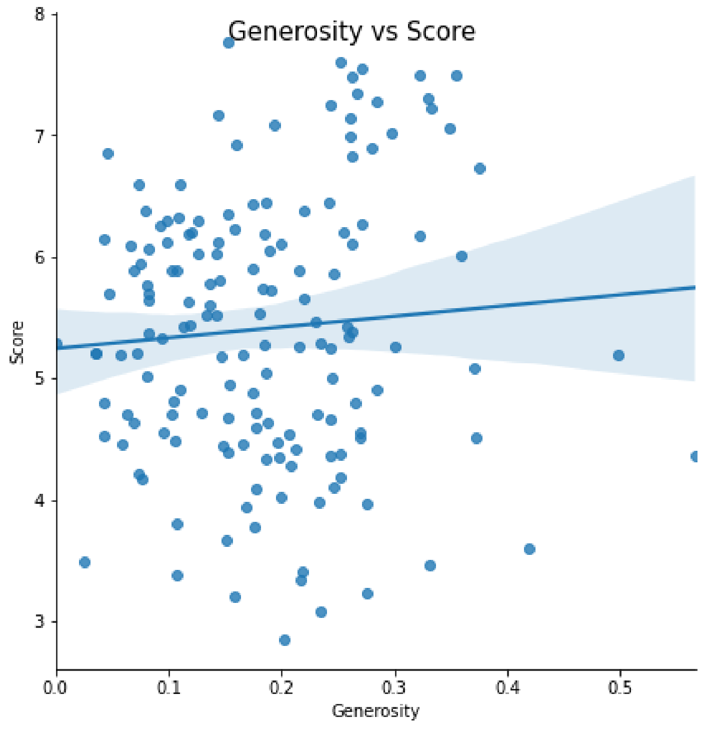


The relationship between Freedom to make life choices and Score is, again, a linear one. With an increase in the freedom to make choices in one's life, there is also an increase in the happiness score. Most of the data points are concentrated in the middle of the graph, between 0.2 to 0.6.

Relationship between Generosity and Score

In [51]:
```
sns.pairplot(df, x_vars='Generosity', y_vars='Score', height=6, kind='reg')
plt.suptitle("Generosity vs Score", fontsize=15)
```

Out[51]:
```
Text(0.5, 0.98, 'Generosity vs Score')
```

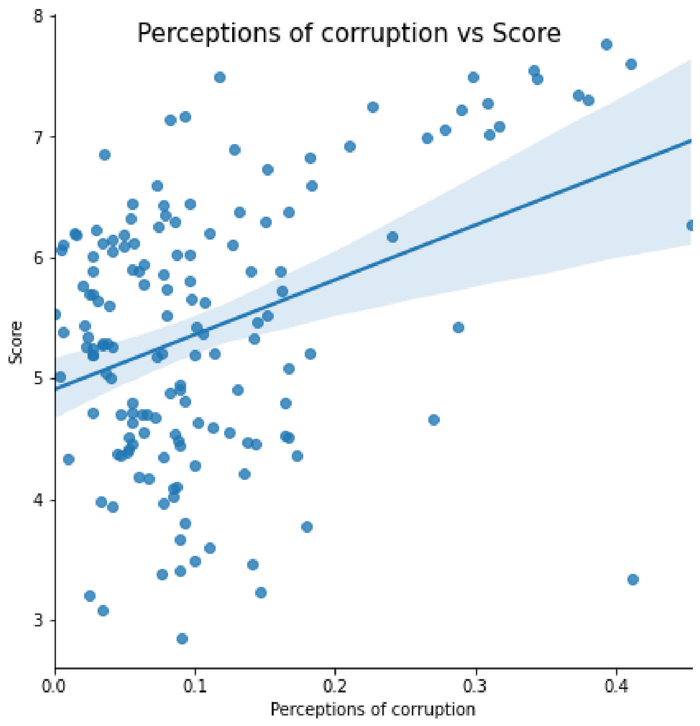


From the above graph, we can see that the relationship between Generosity and Score is almost constant. There is a very slight increase (almost negligible) in the Score with an increase in Generosity. Most of the data points are concentrated between 0.0 to 0.3.

Relationship between Perceptions of corruption vs Score

```
In [52]: sns.pairplot(df, x_vars='Perceptions of corruption', y_vars='Score', height=6, k
         plt.suptitle("Perceptions of corruption vs Score", fontsize=15)
```

```
Out[52]: Text(0.5, 0.98, 'Perceptions of corruption vs Score')
```



From the above graph, we can see that after a certain range, the score increases with an increase in perceptions of corruption. Most of the data points are concentrated between 0.0 to 0.2.

Evaluation

In order to evaluate the performance of our regression model, we used the following scoring methods:

1. R^2 Scoring

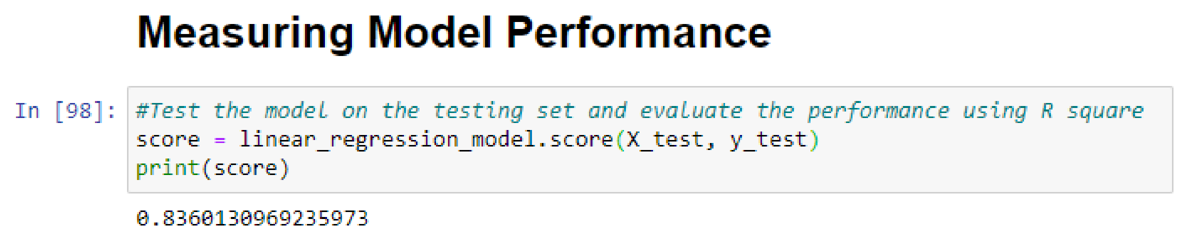
The R^2 statistic is defined as follows:

$$R^2 = 1 - \frac{RSS}{TSS}$$

Where

- RSS (Residual Sum of Squares) measures the variability left unexplained after performing the regression
- TSS measures the total variance
- R^2 statistic measures proportion of variability in Y that is explained by X using our model. [10]

R^2 can be determined using our test set and model's score method.



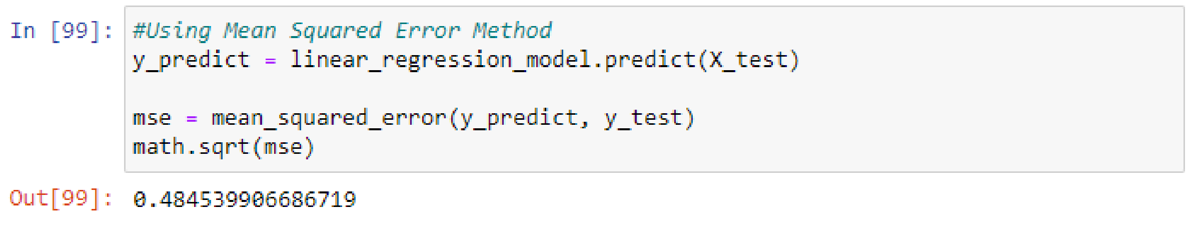
Measuring Model Performance

```
In [98]: #Test the model on the testing set and evaluate the performance using R square
         score = linear_regression_model.score(X_test, y_test)
         print(score)

         0.8360130969235973
```

The model score comes out to 83% for our test set which is pretty good.

2. Mean Square Error Method

We can also get the mean squared error using scikit learn's mean_squared_error method and comparing the prediction for the test data set with the ground truth for the data in the test set. [10]



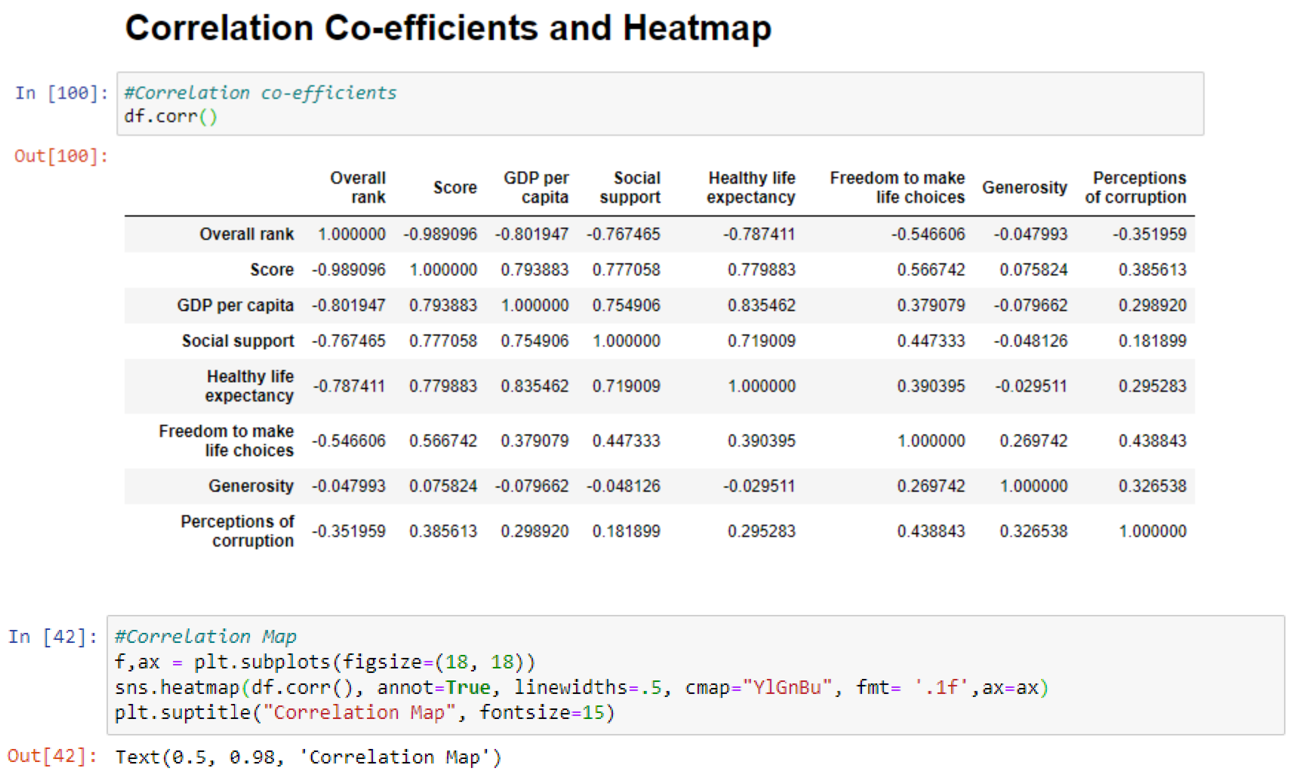
```
In [99]: #Using Mean Squared Error Method
         y_predict = linear_regression_model.predict(X_test)

         mse = mean_squared_error(y_predict, y_test)
         math.sqrt(mse)

Out[99]: 0.484539906686719
```

So, we are at an average of 0.5 units away from the ground truth when making predictions on our test set.

Discussion and Conclusion

The above metrics suggest that our model has a score of 83% for the test data set and has a mean square error rate of 0.5 units. From this, we can infer that the model can make some pretty good predictions on new, unseen data.

To find out correlation between different columns in the dataset, we used correlation function. Also, to depict these correlations visually, we made a heatmap using seaborn.



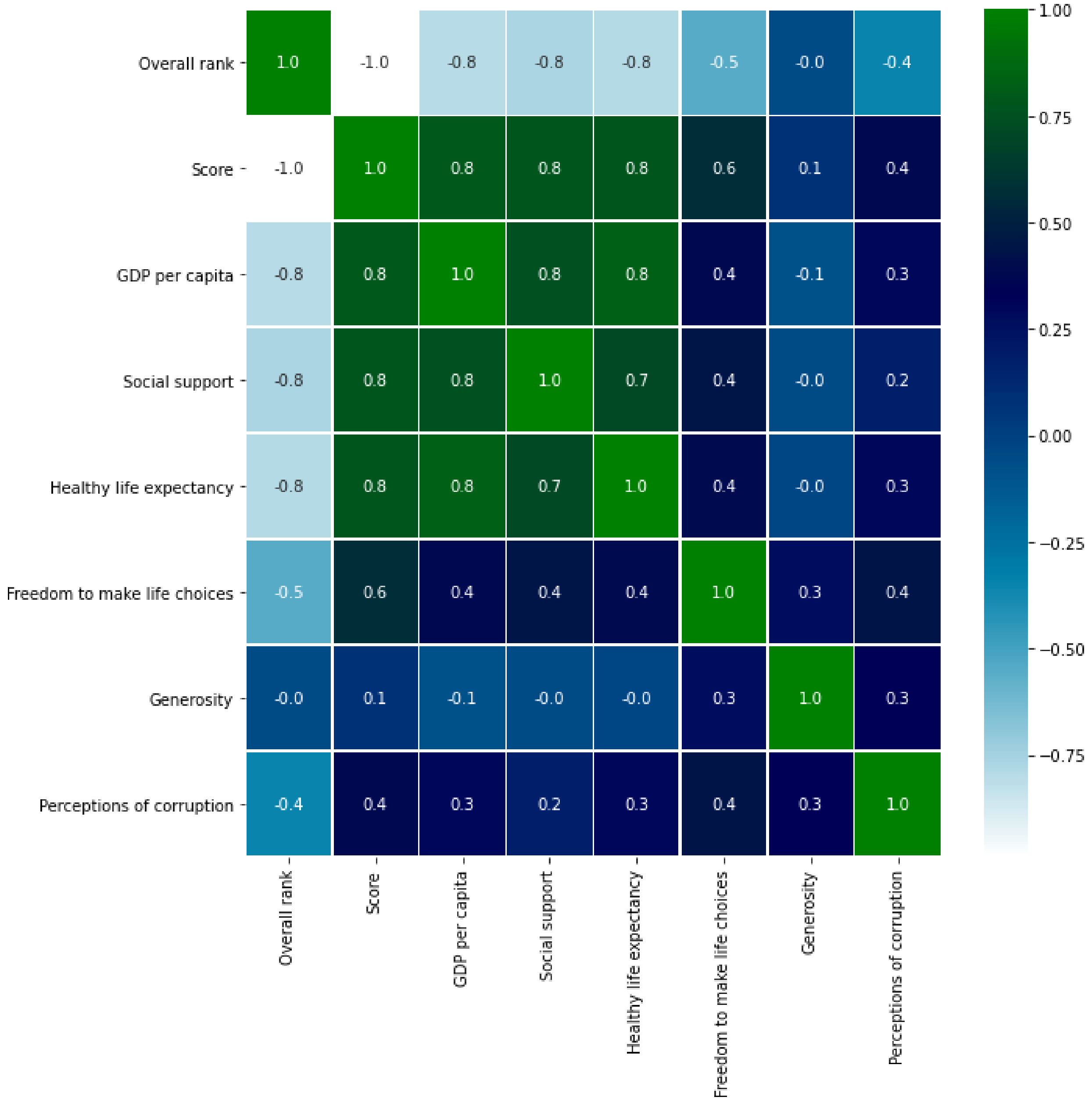
Correlation Co-efficients and Heatmap

```
In [100]: #Correlation co-efficients
          df.corr()
```

Out[100]:

	Overall rank	Score	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Generosity	Perceptions of corruption
Overall rank	1.000000	-0.989096	-0.801947	-0.767465	-0.787411	-0.546606	-0.047993	-0.351959
Score	-0.989096	1.000000	0.793883	0.777058	0.779883	0.566742	0.075824	0.385613
GDP per capita	-0.801947	0.793883	1.000000	0.754906	0.835462	0.379079	-0.079662	0.298920
Social support	-0.767465	0.777058	0.754906	1.000000	0.719009	0.447333	-0.048126	0.181899
Healthy life expectancy	-0.787411	0.779883	0.835462	0.719009	1.000000	0.390395	-0.029511	0.295283
Freedom to make life choices	-0.546606	0.566742	0.379079	0.447333	0.390395	1.000000	0.269742	0.438843
Generosity	-0.047993	0.075824	-0.079662	-0.048126	-0.029511	0.269742	1.000000	0.326538
Perceptions of corruption	-0.351959	0.385613	0.298920	0.181899	0.295283	0.438843	0.326538	1.000000

```
In [42]: #Correlation Map
         f,ax = plt.subplots(figsize=(18, 18))
         sns.heatmap(df.corr(), annot=True, linewidths=.5, cmap="YlGnBu", fmt= '.1f',ax=ax)
         plt.suptitle("Correlation Map", fontsize=15)

Out[42]: Text(0.5, 0.98, 'Correlation Map')
```



The correlation heatmap depicts that when two independent features have a strong relationship, they are considered either positively or negatively correlated. For instance, Score has a very high correlation with Overall rank. Higher the score, higher is the overall rank of the respective country. Similary, score also has a very high correlation with GDP per capita, Social support, and Healthy life expectancy. This is followed by Freedom to make life choices and Perception of Corruption in order. Individuals have given the least importance to Generosity.

Future Works

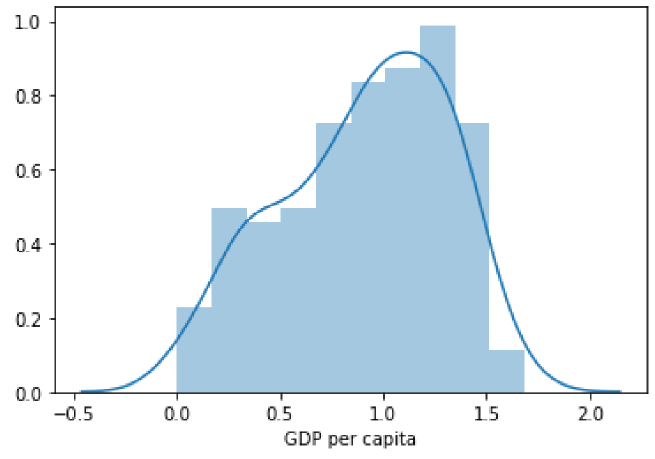
In future, we can further refine the data using techniques such as binning. Data binning or bucketing is a data pre-processing method used to minimize the effects of small observational errors. The original data values are divided into small intervals known as bins and then they are relaces by a general value calculated for that bin. This has a smoothening effect on input data and may also reduce the chances of overfitting in case of small datasets. Binning can be done using the following methods: [11]

- **Equal Frequency Binning:** Bins have equal frequency. [11]
- **Equal Width Binning**: Bins have equal width with a range of each bin defined as [min+w], [min+2w]…..[min+nw] where w = (max – min) / (no. of bins) [11]

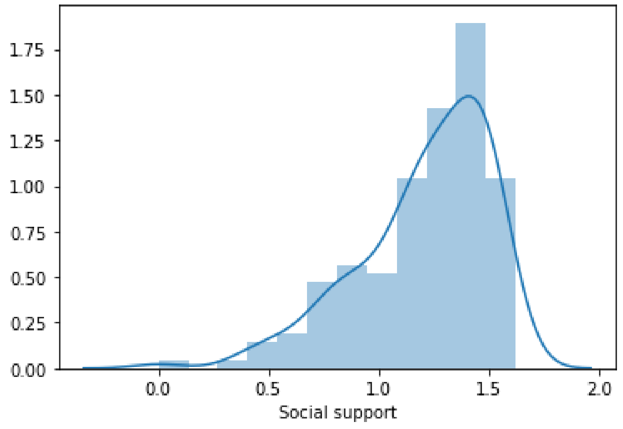
For instance, we tried creating graphs with customized number of bins for each of our field in the data set. Please refer to the figures below. Future works include using these binning techniques for data pre-processing.

Data binning

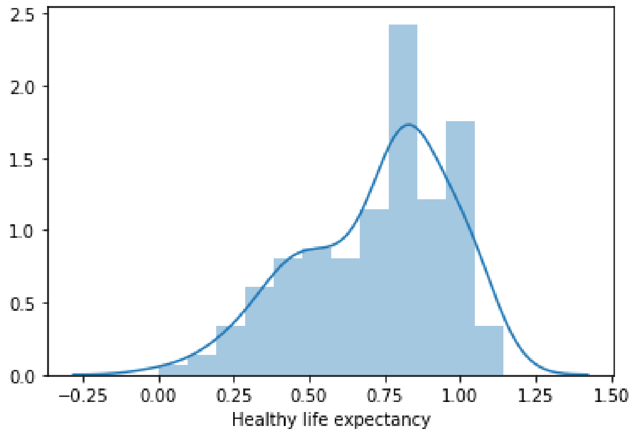
```
In [63]: #Creating a standard histogram to understand the distribution of data
         # And customizing the number of bins
         sns.distplot(df['GDP per capita'], bins=10)
         plt.show()
```



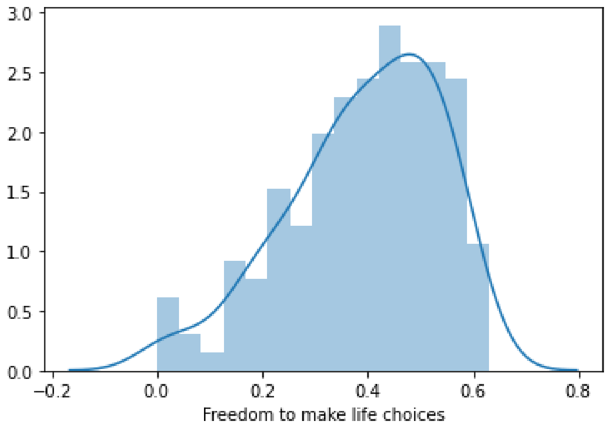
```
In [64]: sns.distplot(df['Social support'], bins = 12)
         plt.show()
```



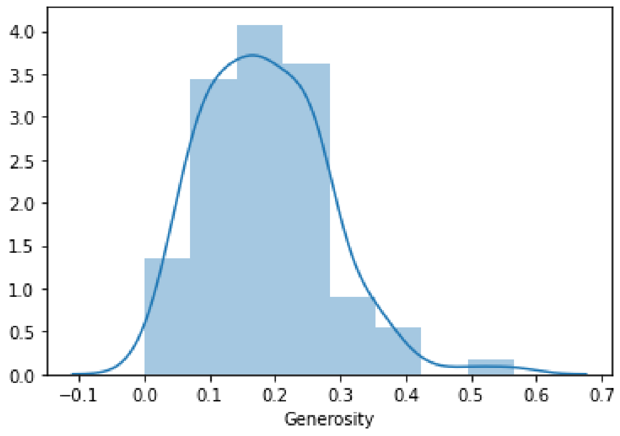
```
In [65]: sns.distplot(df['Healthy life expectancy'], bins = 12)
         plt.show()
```



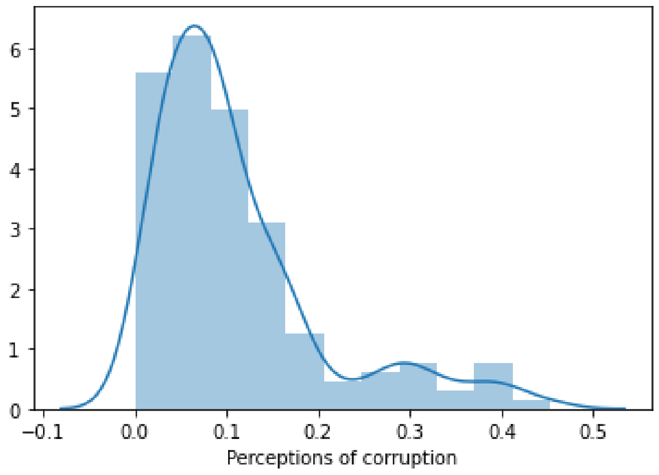
```
In [66]: sns.distplot(df['Freedom to make life choices'], bins = 15)
         plt.show()
```



```
In [67]: sns.distplot(df['Generosity'], bins = 8)
         plt.show()
```



```
In [68]: sns.distplot(df['Perceptions of corruption'], bins = 11)
         plt.show()
```



References

[1] "World Happiness Report," 2015 - 2019. [Online]. Available: https://www.kaggle.com/unsdsn/world-happiness?select=2019.csv.

[2] A. Shah, "World Happiness Report! :)," 2020. [Online]. Available: https://www.kaggle.com/avnika22/world-happiness-report-eda-clustering.

[3] "Pandas Basics," [Online]. Available: https://www.learnpython.org/en/Pandas_Basics.

[4] "NumPy Introduction," [Online]. Available: https://www.w3schools.com/python/numpy_intro.asp.

[5] "Geeks for geeks," 14 May 2018. [Online]. Available: geeksforgeeks.org/python-introduction-matplotlib/.

[6] "An Introduction to Seaborn," [Online]. Available: https://seaborn.pydata.org/introduction.html.

[7] J. Brownlee, "A Gentle Introduction to Scikit-Learn: A Python Machine Learning Library," 16 April 2014. [Online]. Available: https://machinelearningmastery.com/a-gentle-introduction-to-scikit-learn-a-python-machine-learning-library/.

[8] J. Brownlee, "Train-Test Split for Evaluating Machine Learning Algorithms," 24 July 2020. [Online]. Available: https://machinelearningmastery.com/train-test-split-for-evaluating-machine-learning-algorithms/.

[9] "Linear Regression vs Logistic Regression," [Online]. Available: https://www.javatpoint.com/linear-regression-vs-logistic-regression-in-machine-learning.

[10] B. Keen, "Linear Regression in Python Using Scikit Learn," 23 May 2017. [Online]. Available: https://benalexkeen.com/linear-regression-in-python-using-scikit-learn/.

[11] "Binning in Data Mining," Geeks for Geeks, 28 September 2020. [Online]. Available: https://www.geeksforgeeks.org/binning-in-data-mining/.