STATISTICAL ANALYSIS TO PREDICT HAPPINESS SCORE FOR A COUNTRY IN A UPCOMING YEAR

A Project Report submitted to

EduBridge Learning Pvt. Ltd.

FOR THE PARTIAL FULFILLMENT OF THE POST GRADUATE CERTIFICATE IN DATA ANALYTICS

SUBMITTED BY,
Mr. Nikam Pritam Anandrao
Mr. saravanan E
Ms. Rajia Khatun

Under the guidance of
Mrs.Amruta Kedar Chimote

2021-2022

CONTENTS

ABSTRACT

- Happiness is an emotional state characterized by feelings of joy, satisfaction, contentment, and full-fillment. While happiness has many different definitions, it is often described as involving positive emotions and life satisfaction.
- When most people talk about happiness, they might be talking about how they feel in the present moment, or they might be referring to a more general sense of how they feel about life overall.
- Because happiness tends to be such a broadly defined term, psychologists and other social scientists typically use the term 'subjective well-being' when they talk about this emotional state. Just as it sounds, subjective well-being tends to focus on an individual's overall personal feelings about their life in the present.
- So, we'll get the most awaited answer for the question "Can our country will be happy in upcoming year? Let's find out the solutions to the questions in this report!!!

INTRODUCTION

- The **World Happiness Report** is a publication of the United Nations Sustainable Development Solutions Network. It contains articles and rankings of national happiness, based on respondent ratings of their own lives,[1] which the report also correlates with various (quality of) life factors
- The report uses six key variables to measure happiness differences: “income, healthy life expectancy, having someone to count on in times of trouble, generosity, freedom and trust, with the latter measured by the absence of corruption in business and government.”

OBJECTIVES OF THE PROJECT

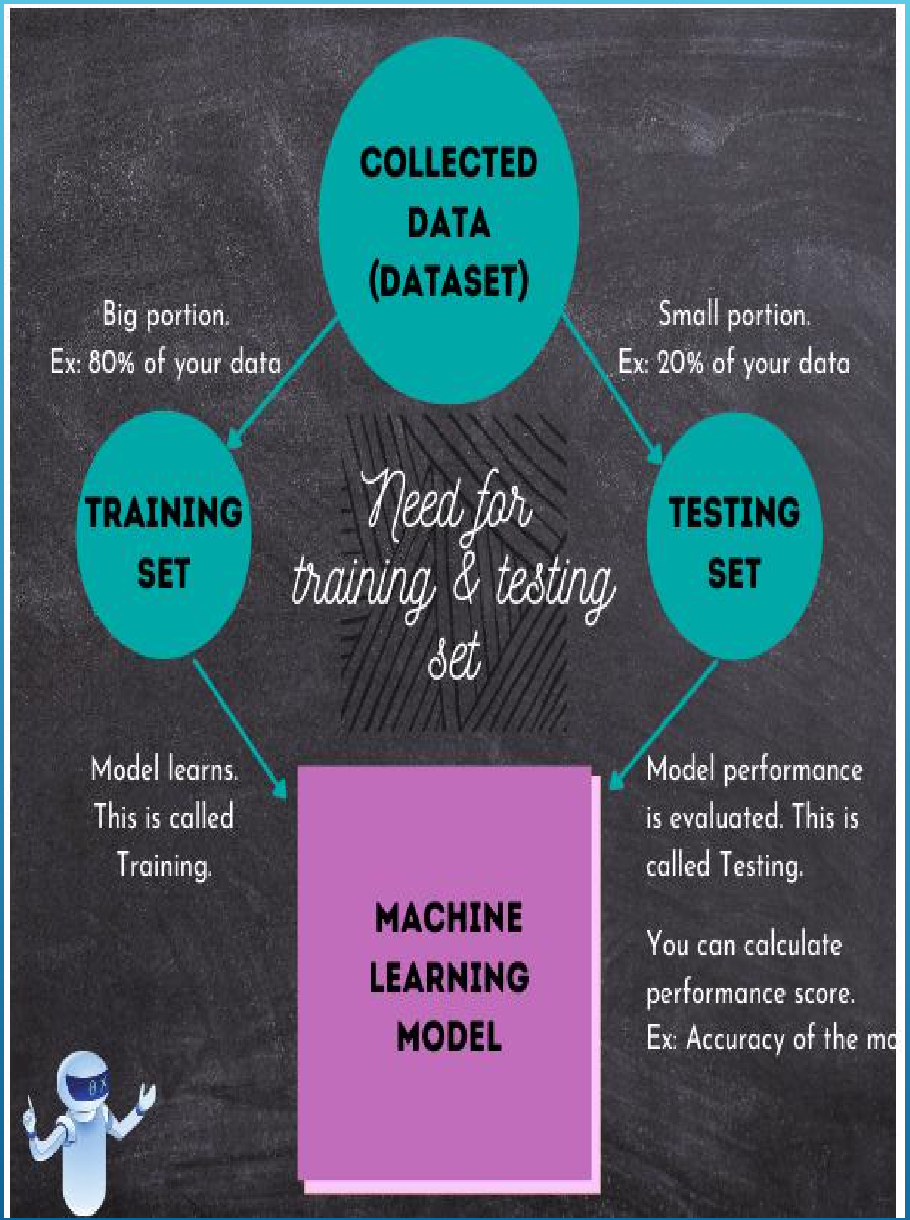
- To find the world's happiest country, year and continent.
- How to deal with missing values?
- To know the important features that affect a happiness score.
- To predict a happiness score for a particular country for an upcoming year.

ABOUT DATA

- We are going to analyse about the happiness score in this project. Obviously, we collected data from the secondary sources. The data for the happiness score taken from the source kaggle.com. The data consists of happiness score from year 2005 to 2020. We have daily, monthly and yearly data. The report uses six key variables to measure happiness differences: “income, healthy life expectancy, having someone to count on in times of trouble, generosity, freedom and trust, with the latter measured by the absence of corruption in business and government.”

STATISTICAL TOOLS AND TECHNIQUES USED

- In this project, we first visualize using different visualization techniques such as line chart, bar plot, etc. with the help of Tableau then we find out the impact of different factors on happiness score. For this, we use the heat map to see the relationship between them and also from that heat map we got the important variables that affect the happiness score and finally we use simple linear regression and SVR for predicting happiness score.
- We are doing this project using the Python language and Tableau.



- For any machine learning problem, the first step would be data collection. This data that you have collected for your machine learning task is called the “**Dataset”.**

- The next step in machine learning would be, building a **model.** This is actually a program written to instruct machine to learn by itself..

- **The train test split** technique can be used for classification and regression problems to test machine learning algorithms. The procedure takes the given dataset and splits it into two subsets:

- **Training datasets** are fed to machine learning algorithms to teach them how to make predictions or perform a desired task.

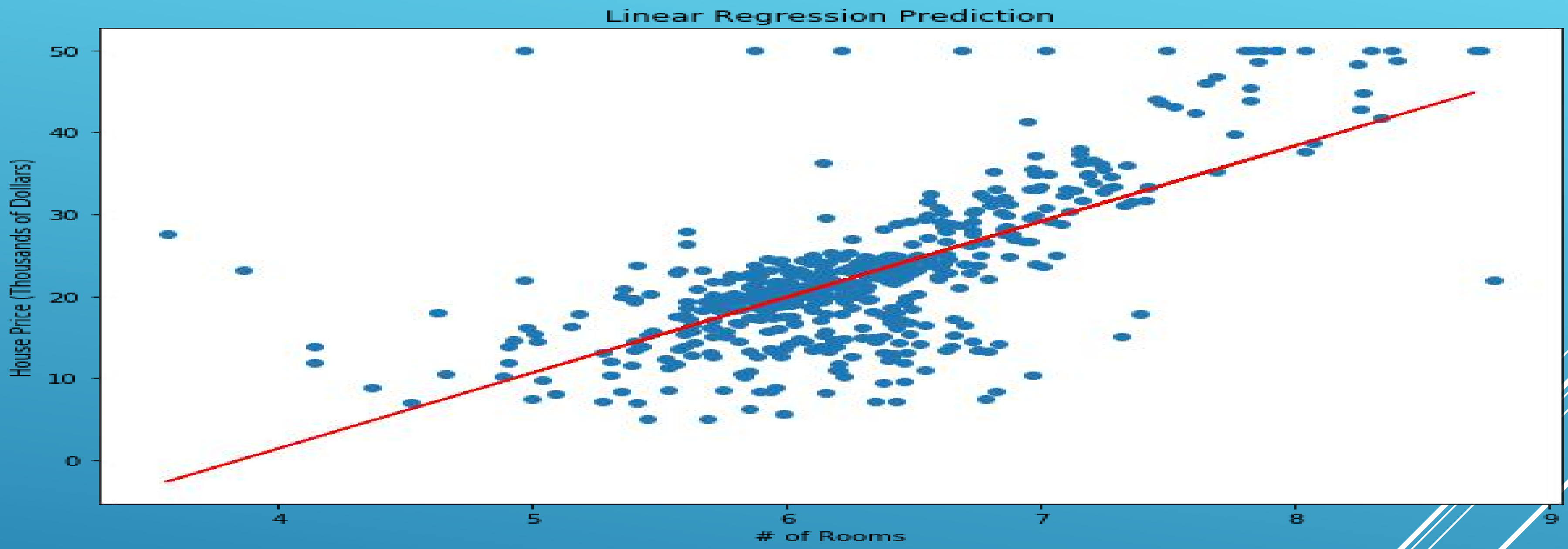
- Once the model completes learning on the training set, it is time to evaluate the performance of the model. For this, we use the smaller portion of the data that we have already set aside. This data which the model has never seen, is called the **Testing set**. Because, this data is what the model will be tested on.

Simple Linear Regression

In most linear regression models, the objective is to minimize the sum of squared errors. Take Ordinary Least Squares (OLS) for example. The objective function for OLS with one predictor (feature) is as follows:

$$MIN \sum_{i=1}^{n} (y_i - w_i x_i)^2$$

where y_i is the target, w_i is the coefficient, and x_i is the predictor (feature).



Lasso, Ridge, and ElasticNet are all extensions of this simple equation, with an additional penalty parameter that aims to minimize complexity and/or reduce the number of features used in the final model. Regardless, the aim — as with many models — is to reduce the error of the test set.

However, what if we are only concerned about reducing error to a certain degree? What if we don't care how large our errors are, as long as they fall within an acceptable range?

Take housing prices for example. What if we are okay with the prediction being within a certain dollar amount — say $5,000? We can then give our model some

SVR FTW

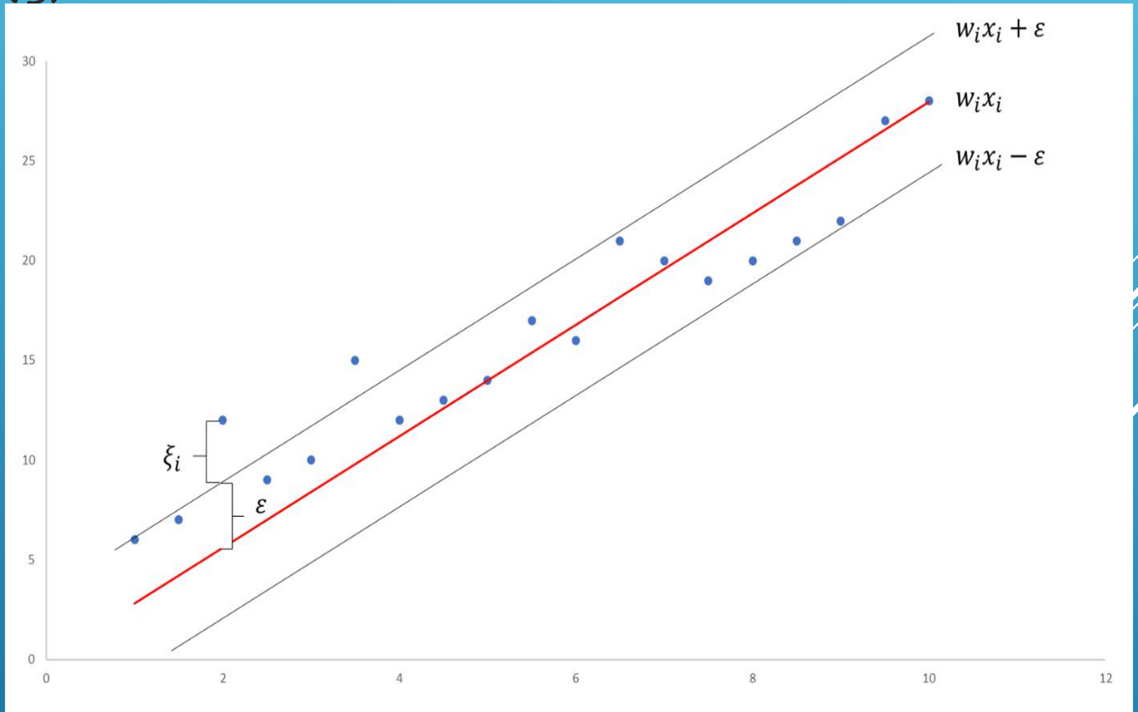
Enter Support Vector Regression. SVR gives us the flexibility to define how much error is acceptable in our model and will find an appropriate line (or hyperplane in higher dimensions) to fit the data. In contrast to OLS, the objective function of SVR is to minimize the coefficients — more specifically, the l2-norm of the coefficient vector — not the squared error. The error term is instead handled in the constraints, where we set the absolute error less than or equal to a specified margin, called the maximum error, ϵ (epsilon). We can tune epsilon to gain the desired accuracy of our model. Our new objective function and constraints are as follows:

$$MIN\ \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}|\xi_i|$$

Constraints:

$$|y_i - w_i x_i| \leq \varepsilon + |\xi_i|$$

Illustrative Example:



In simple regression we try to minimise the error rate. While in SVR we try to fit the error within a certain threshold.

SVR is a powerful algorithm that allows us to choose how tolerant we are of errors, both through an acceptable error margin(ϵ) and through tuning our tolerance of falling outside that acceptable error rate.

DATA ANALYSIS

Happiness score prediction by using a Linear regression:

1.Let's firstly deal with a missing values:

```
In [1]: #loading a libraries requied for the analysis

        import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
        #import pandas_profiling
        %matplotlib inline
        from sklearn.impute import SimpleImputer
        import seaborn as sns
        from sklearn.model_selection import train_test_split
        from sklearn import metrics
        from sklearn.linear_model import LinearRegression

In [2]: #loading a data
        happy=pd.read_csv("world-happiness-report.csv")
        happy.head()
        #to get a data upto year 2019
        happy=happy.loc[happy.year<=2019,:]
        happy.describe()
        #to check any null value present or not
        happy.isnull().sum()

Out[2]: Country name                          0
        year                                  0
        Life Ladder                           0
        Log GDP per capita                   29
        Social support                       13
        Healthy life expectancy at birth     52
        Freedom to make life choices         31
        Generosity                           82
        Perceptions of corruption           104
        Positive affect                      21
        Negative affect                      15
        dtype: int64
```

FROM GIVEN OUTPUT WE CLEARLY SEE THAT THERE ARE SOME MISSING VALUES IN A DATASET.

So, to remove these missing values we use the sklearn.impute library and from that we import SimpleImputer and we have following output:

In [3]:
```
#to remove the null values by using a simple imputer we want only the numeric data that's why we split a data.
x=happy.iloc[:,1:]
y=happy.iloc[:,0]
```

In [4]:
```
#removing a null values with the help of simple imputer
imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
imputer = imputer.fit(x)
x= imputer.transform(x)
x=pd.DataFrame(x)
happy1=x
#to give the column names to the happy1 dataframe
happy1.columns =['year','Life Ladder','Log GDP per capita','Social support','Healthy life expectancy at birth','Freedom to make l
happy1.head()
```

Out[4]:

	year	Life Ladder	Log GDP per capita	Social support	Healthy life expectancy at birth	Freedom to make life choices	Generosity	Perceptions of corruption	Positive affect	Negative affect
0	2008.0	3.724	7.370	0.451	50.80	0.718	0.168	0.882	0.518	0.258
1	2009.0	4.402	7.540	0.552	51.20	0.679	0.190	0.850	0.584	0.237
2	2010.0	4.758	7.647	0.539	51.60	0.600	0.121	0.707	0.618	0.275
3	2011.0	3.832	7.620	0.521	51.92	0.496	0.162	0.731	0.611	0.267
4	2012.0	3.783	7.705	0.521	52.24	0.531	0.236	0.776	0.710	0.268

In [5]:
```
happy1.isnull().sum()
```

Out[5]:
```
year                                0
Life Ladder                         0
Log GDP per capita                  0
Social support                      0
Healthy life expectancy at birth    0
Freedom to make life choices        0
Generosity                          0
Perceptions of corruption           0
Positive affect                     0
Negative affect                     0
dtype: int64
```

SO, FROM GIVEN OUTPUT WE CAN SEE THAT THERE IS NO NULL VALUE IN A GIVEN DATA. HENCE WE CAN USE THAT DATA FOR THE PREDICTION OF THE HAPPINESS SCORE

Let's fit the regression model with all the features:

Linear regression with all features

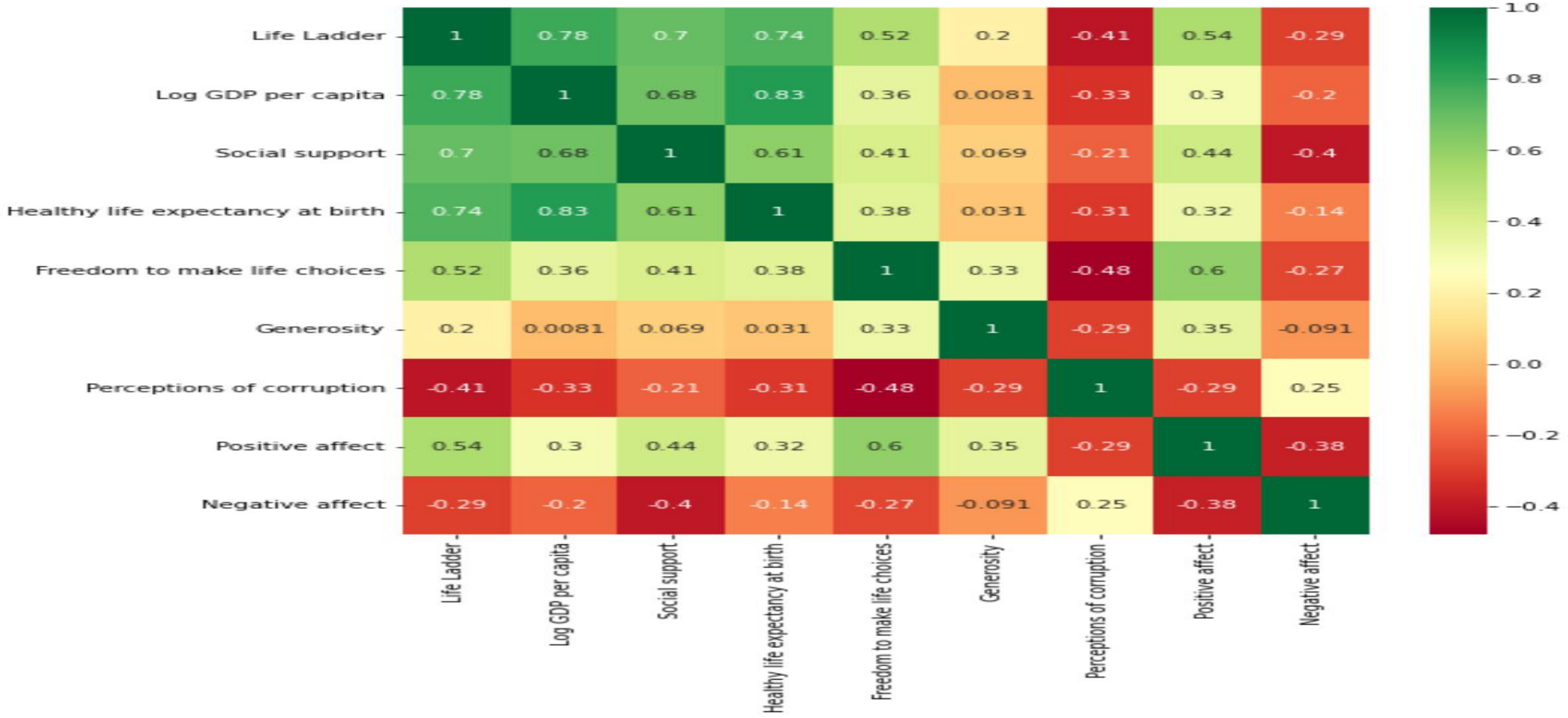
```
#loading a data
happy=pd.read_csv("world-happiness-report.csv")
happy.head()
#to get a data upto year 2019
happy=happy.loc[happy.year<=2019,:]
happy.describe()
#to check any null value present or not
happy.isnull().sum()
#to remove the null values by using a simple imputer we want only the numeric data that's why we split a data.
x=happy.iloc[:,1:]
y=happy.iloc[:,0]
#removing a null values with the help of simple imputer
imputer = SimpleImputer(missing_values=np.nan, strategy='median')
imputer = imputer.fit(x)
x= imputer.transform(x)

x=pd.DataFrame(x)
happy1=x
#to give the column names to the happy1 dataframe
happy1.columns =['year','Life Ladder','Log GDP per capita','Social support','Healthy life expectancy at birth',
                 'Freedom to make life choices','Generosity','Perceptions of corruption','Positive affect','Negative affect']
happy1.head()
happy1.isnull().sum()
happy1= happy1.drop(columns = ['year']) #dropping a column which are not require
y = np.array(happy1['Life Ladder']).reshape(-1,1) #dependent variable i.e. we have to predict life ladder score
X=happy1.iloc[:,2:]#independents variables i.e variables from which we predict a life ladder score
# Feature Scaling
sc_X = StandardScaler()
sc_y = StandardScaler()
X = sc_X.fit_transform(X)
y = sc_y.fit_transform(y)
#splitting a data into train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=101, shuffle = False)
#print(X_train)
#applying a linear regression model
lm = LinearRegression()
#fitting a model on train data
lm.fit(X_train,y_train)
#to see the regression coefficients
print('Coefficients: \n', lm.coef_)
#predicted value of X_test data
predictions = lm.predict(X_test)
#to find the root mean square error to find the model is fit or not.
print('MAE:', metrics.mean_absolute_error(y_test, predictions))
print('MSE:', metrics.mean_squared_error(y_test, predictions))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predictions)))
print("The R^2 score is: ", metrics.r2_score(y_test,predictions))
```

```
Coefficients:
 [[ 0.32488054  0.41627323  0.0427359   0.06933588 -0.11614603  0.16621955
   0.00746243]]
MAE: 0.4573718483991971
MSE: 0.35564965028474654
RMSE: 0.5963636896095759
The R^2 score is:  0.7007621852915417
```

FROM ABOVE OUTPUT WE HAVE THAT RMSE FOR THIS MODEL IS 0.5964 AND R-SQUARE IS 70% WHICH IS LARGE SO FOR THE FURTHER PROCESS WE HAVE TO ELIMINATES NUMBER OF FEATURES TO IMPROVE THE ACCURACY OF THE MODEL.

To eliminate features we use the heatmap:

```
In [9]: #get correlations of each features in dataset
        corrmat = happy1.corr()
        #print(corrmat)
        top_corr_features = corrmat.index
        #print(top_corr_features)
        plt.figure(figsize=(9,9))
        #plot heat map
        g=sns.heatmap(happy1[top_corr_features].corr(),annot=True,cmap="RdYlGn")
```



FROM THE ABOVE HEATMAP WE HAVE LIFE LADDER IS HIGHLY CORRELATED WITH A 'LOG GDP PER CAPITA', 'SOCIAL SUPPORT',' HEALTHY LIFE EXPECTANCY AT BIRTH', 'FREEDOM TO MAKE LIFE CHOICES',' POSITIVE AFFECT', 'PERCEPTIONS OF CORRUPTION' AND LESS CORRELATED WITH A 'GENEROSITY', 'NEGATIVE AFFECT'. SO, FOR THE FURTHER PROCESS WE USE ONLY THOSE FEATURES WHICH ARE HIGHLY CORRELATED WITH A LIFE LADDER SCORE.

Let’s fit a regression model on highly correlated features:

Linear regression with highly correlated features

```
happy=pd.read_csv("world-happiness-report.csv")
happy=happy.loc[happy.year<=2019,:]
happy.head()
happy.describe()
happy.isnull().sum()
x=happy.iloc[:,1:]
y=happy.iloc[:,0]
imputer = SimpleImputer(missing_values=np.nan, strategy='median')
imputer = imputer.fit(x)
x= imputer.transform(x)
x=pd.DataFrame(x)
happy1=x
happy1.head()
happy1.columns =['year','Life Ladder','Log GDP per capita','Social support','Healthy life expectancy at birth',
                 'Freedom to make life choices','Generosity','Perceptions of corruption','Positive affect','Negative affect']
happy1.head()
happy1.isnull().sum()
happy1= happy1.drop(columns = ['year'])
happy1= happy1.drop(columns = ['Generosity','Negative affect'])
y = np.array(happy1['Life Ladder']).reshape(-1,1)
X=happy1.iloc[:,1:]
# Feature Scaling
sc_X = StandardScaler()
sc_y = StandardScaler()
X = sc_X.fit_transform(X)
y = sc_y.fit_transform(y)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=101, shuffle = False)
#Scaling numeric features using sklearn StandardScalar

lm = LinearRegression()
lm.fit(X_train,y_train)
print('Coefficients: \n', lm.coef_)
predictions = lm.predict(X_test)

print('MAE:', metrics.mean_absolute_error(y_test, predictions))
print('MSE:', metrics.mean_squared_error(y_test, predictions))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predictions)))
print("The R^2 score is: ", metrics.r2_score(y_test,predictions))
```

```
Coefficients:
 [[ 0.34864683  0.20831247  0.17516457  0.04103833 -0.10150848  0.225727  ]]
MAE: 0.3822093198742215
MSE: 0.2754350224285944
RMSE: 0.5248190377916891
The R^2 score is:  0.7682534647799631
```

FROM ABOVE OUTPUT WE CLEARLY SEE THAT WHEN WE USE THE HIGHLY CORRELATED FEATURES WE GET LOWER RMSE 0.5248 AND R-SQUARE IS 77%. SO, FOR THE FURTHER PROCESS WE USE ONLY HIGHLY CORRELATED FEATURES. BUT DUE TO LARGE RMSE VALUE WE USE THE SUPPORT VECTOR REGRESSION(SVR) FOR THE FURTHER ANALYSIS.

Let’s fit a SVR model on 2020 dataset with highly correlated features:

SVR with highly correlated features for a 2020 dataset

In [6]:

```python
#loading a data
happy=pd.read_csv("world-happiness-report.csv")
happy.head()
#to get a data upto year 2019
happy=happy.loc[happy.year>2019,:]
happy.describe()
#to check any null value present or not
happy.isnull().sum()

x=happy.iloc[:,1:]
y=happy.iloc[:,0]
#removing a null values with the help of simple imputer
imputer = SimpleImputer(missing_values=np.nan, strategy='median')
imputer = imputer.fit(x)
x= imputer.transform(x)

x=pd.DataFrame(x)
happy1=x
happy1.columns =['year','Life Ladder','Log GDP per capita','Social support','Healthy life expectancy at birth',
                 'Freedom to make life choices','Generosity','Perceptions of corruption','Positive affect','Negative affect']
happy1= happy1.drop(columns = ['year'])
happy1= happy1.drop(columns = ['Generosity','Negative affect'])
happy1.head()

y = np.array(happy1['Life Ladder']).reshape(-1,1)
X=happy1.iloc[:,1:]
# Feature Scaling
sc_X = StandardScaler()
sc_y = StandardScaler()
X = sc_X.fit_transform(X)
y = sc_y.fit_transform(y)
#splitting a dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=101, shuffle = False)

#to obtain the value of C and epsilon
grid = {
    'C': np.linspace(0.01, 10),
    'epsilon': np.linspace(0.01, 10)
}
svr_gridsearch =  LinearSVR(fit_intercept=True, max_iter=10000)
grid_svr = GridSearchCV(svr_gridsearch, grid, scoring='neg_mean_absolute_error', cv=5)
grid_svr.fit(X_train, y_train)
best_grid_svr_mae = grid_svr.best_estimator_
best_grid_svr_mae.fit(X_train, y_train)
```

Out[6]:

```
LinearSVR(C=0.6216326530612245, epsilon=0.21387755102040817, max_iter=10000)
```

```python
from sklearn.svm import SVR
#applying SVR model
regressor = SVR(kernel="linear",C=0.6216326530612245, epsilon=0.21387755102040817)
regressor.fit(X_train,y_train)
coeffecients = pd.DataFrame( (regressor.coef_),columns =['Log GDP per capita','Social support','Healthy life expectancy at birth'
                 'Freedom to make life choices','Perceptions of corruption','Positive affect',])

print(coeffecients)
#coeffecients.columns = ['Coeffecient']


predictions = regressor.predict(X_test)

from sklearn.metrics import accuracy_score
#to find the root mean square error to find the model is fit or not.
print('MAE:', metrics.mean_absolute_error(y_test, predictions))
print('MSE:', metrics.mean_squared_error(y_test, predictions))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predictions)))
print("The R^2 score is: ", metrics.r2_score(y_test,predictions))
#print("accuracy score:",metrics.accuracy_score(y_test,predictions))
```

```
   Log GDP per capita  Social support  Healthy life expectancy at birth  \
0            0.280361        0.310744                          0.215691

   Freedom to make life choices  Perceptions of corruption  Positive affect
0                      0.151156                  -0.132989         0.055851
MAE: 0.35211428481461443
MSE: 0.22626807133120666
RMSE: 0.47567643554332883
The R^2 score is:  0.847615860509661
```

FROM ABOVE OUTPUT WE CAN EASILY SEE THAT THE RMSE GOES ON DECREASING AND WE HAVE R-SQUARE IS ABOUT 85%. SO, THIS MODEL CAN WE CAN USE FOR THE PREDICTION. HENCE, WE WILL USE THIS MODEL TO PREDICT THE HAPPINESS SCORE FOR THE PARTICULAR COUNTRY FROM THE GIVEN DATASET.

Let's predict a happiness score for a particular country:

To see the predicted value for a particular country :

```
#Loading a data
happy=pd.read_csv("world-happiness-report.csv")
happy.head()
happy.describe()
#to check any null value present or not
happy.isnull().sum()
x=happy.iloc[:,2:]
b=happy.iloc[:,1]
a=happy.iloc[:,0]
#removing a null values with the help of simple imputer
imputer = SimpleImputer(missing_values=np.nan, strategy='median')
imputer = imputer.fit(x)
x= imputer.transform(x)
x=pd.DataFrame(x)
happy1=x
happy1.columns =['Life Ladder','Log GDP per capita','Social support','Healthy life expectancy at birth',
                 'Freedom to make life choices','Generosity','Perceptions of corruption','Positive affect','Negative affect']
#mean of the life ladder
life_ladder_mean=happy['Life Ladder'].mean()
#standard deviation of life ladder
life_ladder_std=happy['Life Ladder'].std()
Y= np.array(happy1['Life Ladder']).reshape(-1,1)
X=happy1.iloc[:,1:]
# Feature Scaling
sc_X = StandardScaler()
sc_y = StandardScaler()
X = sc_X.fit_transform(X)
y = sc_y.fit_transform(Y)
dataset = pd.DataFrame({'Column1': X[:, 0], 'Column2': X[:, 1],'Column3': X[:, 2],'Column4': X[:, 3],'Column5': X[:, 4],
                   'Column6': X[:, 5],'Column7': X[:, 6],'Column8': X[:, 7]})
dataset.columns =['Log GDP per capita','Social support','Healthy life expectancy at birth',
                  'Freedom to make life choices','Generosity','Perceptions of corruption','Positive affect','Negative affect']
dataset['Life Ladder']=y
dataset['country']=a
dataset['year']=b
```

```
dataset=dataset.loc[dataset.year>2019,:]
dataset= dataset.drop(columns = ['Generosity','Negative affect'])
dataset=dataset.loc[dataset.country=='India',:]
dataset.head()
```

	Log GDP per capita	Social support	Healthy life expectancy at birth	Freedom to make life choices	Perceptions of corruption	Positive affect	Life Ladder	country	year
746	-0.583605	-1.657534	-0.339008	1.157534	0.163764	0.393163	-1.113214	India	2020

```
predict= (dataset['Log GDP per capita']*0.280361)+( dataset['Social support']* 0.310744 )+
( dataset['Healthy life expectancy at birth'] * 0.215691)+(dataset['Freedom to make life choices']*0.151156)-
(dataset['Perceptions of corruption']*0.132989 ) +(dataset['Positive affect']* 0.055851)

predict
```

```
746   -0.576662
dtype: float64
```

```
When we apply a standardScalar the original value get transformed to the
y=(x-mean)/std
where,
y=transformed value
x=original value
mean=mean of all values
std=standard deviation of all values
```

```
original=(dataset['Life Ladder']*life_ladder_std)+life_ladder_mean
print(original)
```

```
746    4.224681
Name: Life Ladder, dtype: float64
```

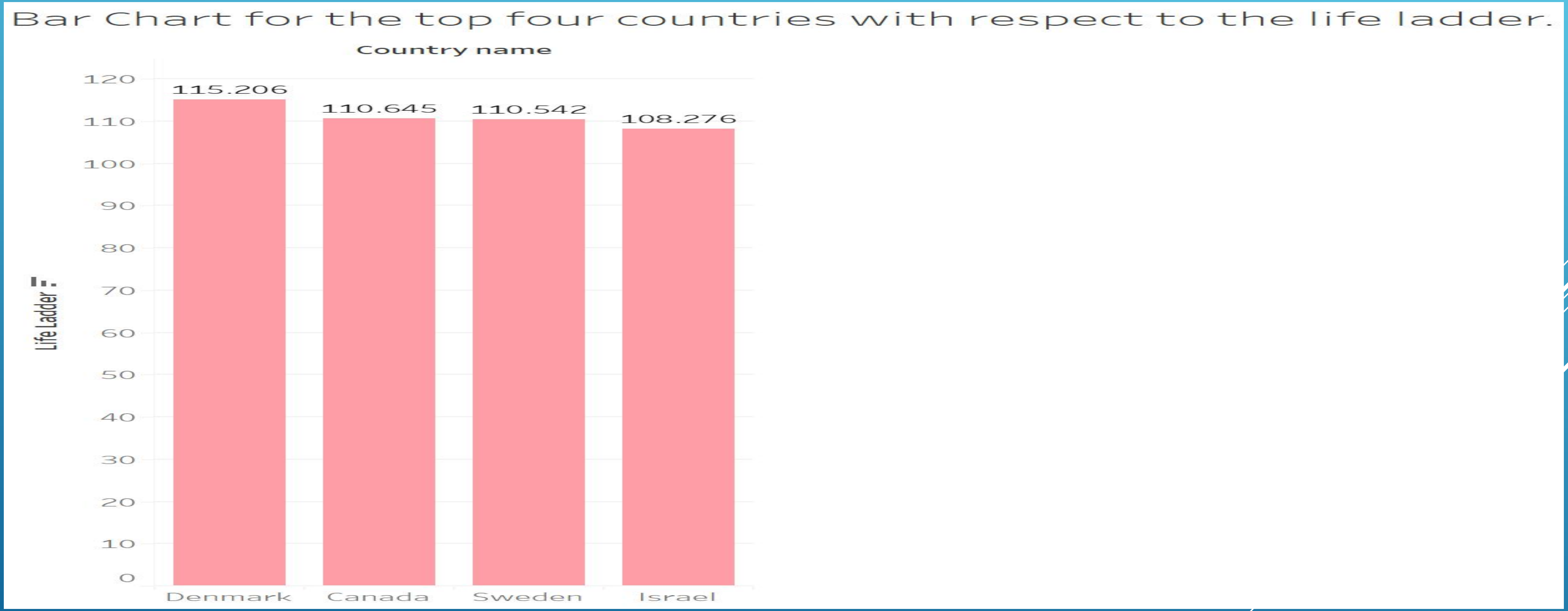
```
predicted=(predict*life_ladder_std)+life_ladder_mean
print(predicted)
```

```
746    4.823318
dtype: float64
```

FROM THE ABOVE OUTPUT WE PREDICT HAPPINESS SCORE FOR THE COUNTRY INDIA AND IT IS 4.823318.

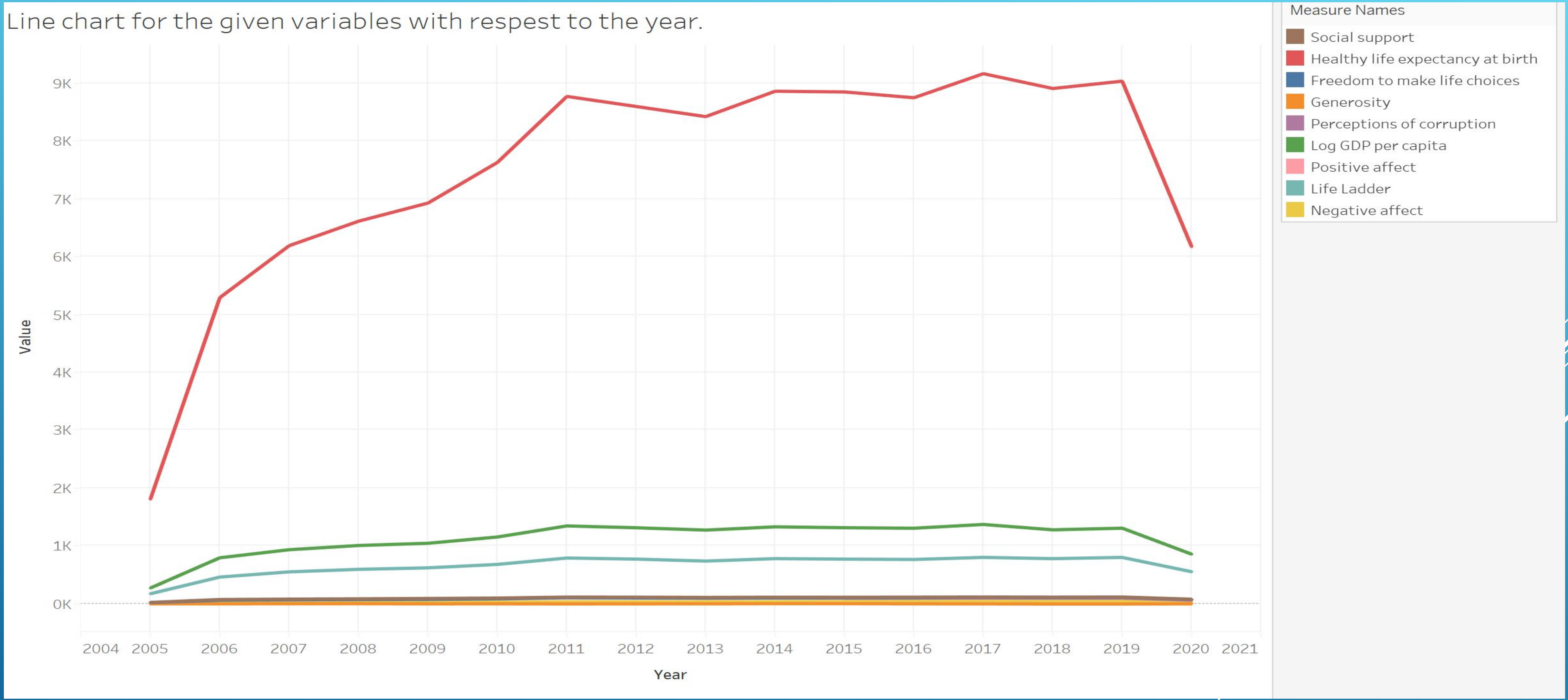
Data visualization –

- Let's look at Bar plot to know the top four happiest country in the world.



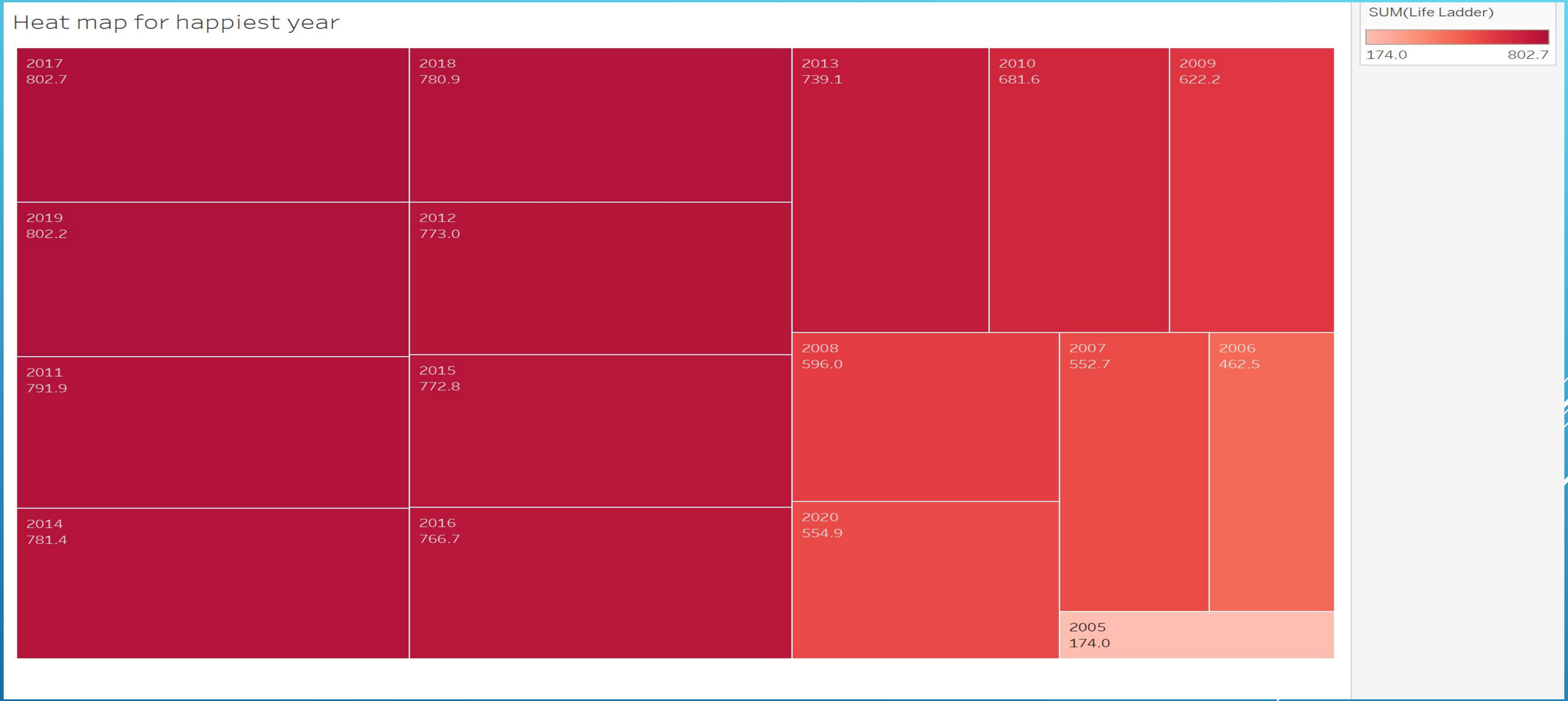
FROM ABOVE BAR CHART WE HAVE TOP FOUR HAPPIEST COUNTRY SUCH AS DENMARK, SWEDEN, CANADA AND ISRAEL.

2. Let’s look at the line chart to see the what is the relationship of the other variables with a life ladder score with respect to the year.



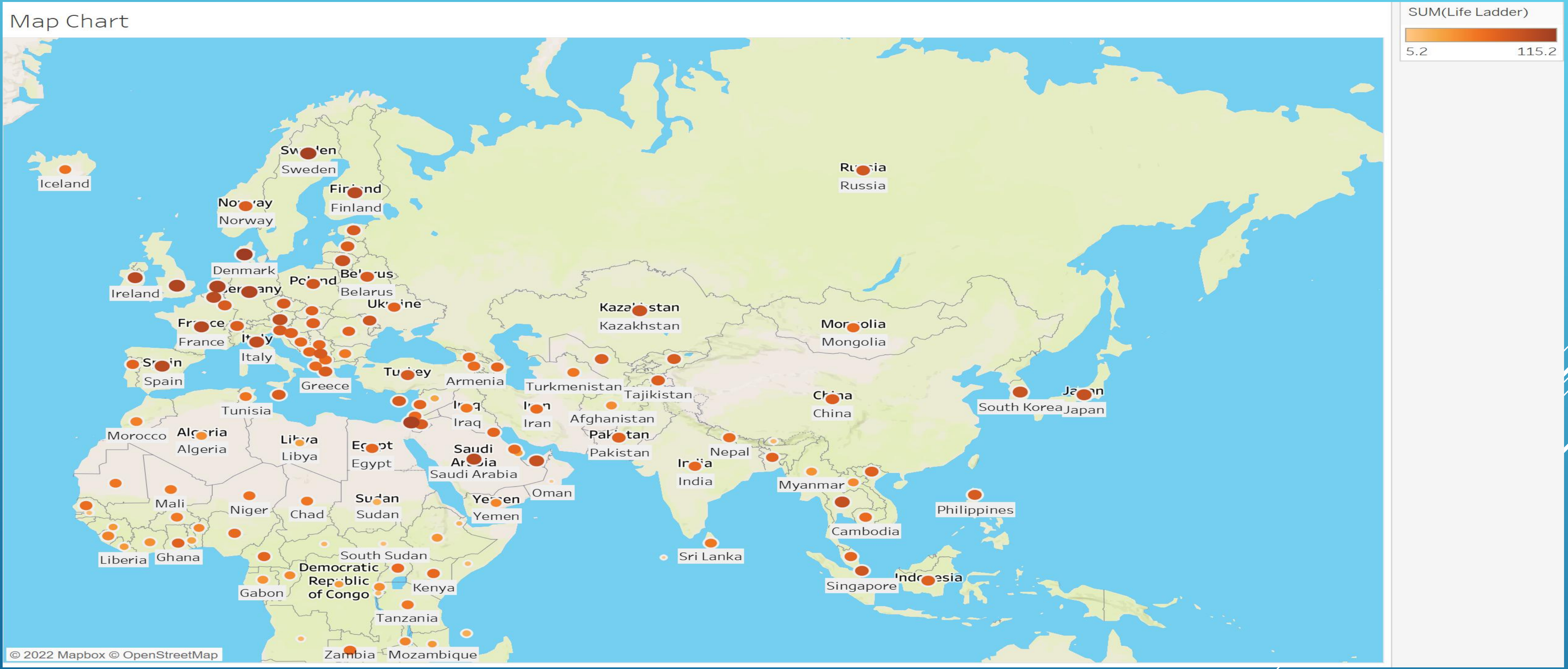
FROM THE ABOVE GRAPH WE CAN EASILY SEE THAT THE THERE IS RELATIONSHIP BETWEEN THE LIFE LADDER SCORE AND THE OTHER VARIABLES. ALSO, AFTER A COVID-19 THE HAPPINESS OF PEOPLE GOES ON DECREASING.

3. Let's look at the heat map to see the which year is most happy among others one.



FROM THE ABOVE HEAT MAP WE CAN EASILY FIND THAT THE 2017 IS THE MOST HAPPIEST YEAR AMONG THE OTHERS ONE.

4.Let’s look at the world map to see the which continent of the world is most happy.



FROM ABOVE MAP CHART WE CAN EASILY SEE THAT THE EUROPE CONTINENT IS THE MOST HAPPY THAN THE OTHERS.

MAJOR FINDINGS

1. From the data visualization we can easily see that the top four happiest country are Denmark, Sweden, Canada and Israel. Also, the happiest year is 2017 and happiest continent is Europe
2. The order of the top strongest predictors, from strongest to weakest, of World Happiness appear to be:
   a. Social Support
   b. Positive Affect
   c. Perceptions of corruption
   d. Healthy life expectancy at birth
   e. Freedom to make choices
   f. Log GDP per capita

Importantly, as both Social Support and Positive affect go up by one point, the ladder goes up two steps, making these very powerful indicators of World Happiness. To understand this better, the more a country of people has family or friends to be there for them when something bad happens in their life (Helliwell, Huang, Wang, & Norton, 2021), the happier the nation does. Further, the more people say that they are able to laugh, feel happy, joy, or a positive emotion each day throughout the week, the better at predicting the happiness of a nation. Conversely, the more people in a nation believe that there is corruption in either the government, business, or both, the amount of happiness in the country decrease by half a step. Interestingly, when people donate more money in a country, the happier they are, even more so than the freedom to choose what you want to do with your life, and the log GDP (GDP per capita).

3. We fit the regression model and a SVR model to predict the happiness score for a country in a upcoming year

References

- Basic statistics – B L Agarwal
- https://www.kaggle.com/search?q=world+happiness+report+2021+datasetFileTypes%3Acsv
- https://towardsdatascience.com/feature-selection-techniques-in-machine-learning-with-python-f24e7da3f36e
- https://scikitlearn.org/stable/modules/generated/sklearn.impute.SimpleImputer.html

THANK YOU!