Project 2_week 6_Using Python-milestone 4

October 26, 2020

0.0.1 project 2- DSC680

0.0.2 Happiness 2019

soukhna Wade 10/10/2020

0.0.3 Introduction

There are three parts to my report as follows:

** Cleaning ** Visualization ** Random Forest

The purpose of choosing this work is to find out which factors are more important to live a happier life. As a result, people and countries can focus on the more significant factors to achieve a higher happiness level. We also will implement several machine learning algorithms to predict the happiness score and compare the result to discover which algorithm works better for this specific dataset.

https://www.kaggle.com/pinarkaya/world-happiness-eda-visualization-ml#2019-Data

0.0.4 Import necessary Libraries

[164]:
```
# Standard library import-Python program# for some basic operations
import pandas as pd
import numpy as np                  # linear algebra

import matplotlib.pyplot as plt     # for graphics
import seaborn as sns               # for visualizations
plt.style.use('fivethirtyeight')

import seaborn as seabornInstance
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics

# Use to configure display of graph
%matplotlib inline

#stop unnecessary warnings from printing to the screen
import warnings
warnings.simplefilter('ignore')
```

```
# for interactive visualizations
import plotly.offline as py
from plotly.offline import init_notebook_mode, iplot
import plotly.graph_objs as go
init_notebook_mode(connected = True)
```

0.0.5 Import and read Dataset from local library

[158]:
```
#The following command imports the CSV dataset using pandas:
import pandas as pd

happyness_2019 = pd.read_csv("happyness_2019.csv")

happyness_2019.head(3)
```

[158]:

	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

[293]:
```
happyness_2019.columns
```

[293]:
```
Index(['Overall rank', 'Country or region', 'Score', 'GDP per capita',
       'Social support', 'Healthy life expectancy',
       'Freedom to make life choices', 'Generosity',
       'Perceptions of corruption'],
      dtype='object')
```

Looking at the current shape of the dataset under consideration

[294]:
```
# Looking at the current shape of the dataset under consideration
#df.shape

# Step 2:  check the dimension of the table or the size of dataframe

print("The dimension of the table is: ",happyness_2019.shape)
```

```
The dimension of the table is:  (156, 9)
```

0.0.6 Cleaning - Is threre any missing or null Values in this dataset (happyness_2019)?

In this section, we load our dataset and see the structure of happiness variables. Our dataset is pretty clean, and we will implement a few adjustments to make it looks better.

[295]:
```
#check for any missing values or null values (NA or NaN)
happyness_2019 .isnull().sum()
#df.isnull().head(6)
```

[295]:
```
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
```

** Note that the above result no missing values so, the dataset is pretty cleaned.**

[296]:
```
# Print a list datatypes of all columns

#happyness_2019 .dtypes
```

1 Exploratory Data Analysis

Prints information of all columns:

[297]:
```
happyness_2019 .info() # Prints information of all columns:
```

```
<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
```

Display some statistical summaries of the numerical columns data. To see the statistical details of the dataset, we can use describe():

[298]:
```
happyness_2019 .describe().head(2)     # display some statistical summaries of␣
↪the numerical columns data.
```

[298]:
```
       Overall rank       Score  GDP per capita  Social support  \
count         156.0  156.000000      156.000000      156.000000
mean           78.5    5.407096        0.905147        1.208814

       Healthy life expectancy  Freedom to make life choices  Generosity  \
count                156.000000                    156.000000  156.000000
mean                   0.725244                      0.392571    0.184846

       Perceptions of corruption
count                 156.000000
mean                    0.110603
```

[299]:
```
happyness_2019.columns              # display the list of the columns
```

[299]:
```
Index(['Overall rank', 'Country or region', 'Score', 'GDP per capita',
       'Social support', 'Healthy life expectancy',
       'Freedom to make life choices', 'Generosity',
       'Perceptions of corruption'],
      dtype='object')
```

1.0.1 Let us examine the data for the county very happy and the one that is not

[300]:
```
maxSupport=np.max(happyness_2019["Social support"])
maxSupport
```

[300]:
```
1.624
```

[301]:
```
maxEconomy=np.max(happyness_2019["GDP per capita"])
maxEconomy
```

[301]:
```
1.6840000000000002
```

[302]:
```
happyness_2019[happyness_2019['Score']==np.
↪max(happyness_2019['Score'])]['Country or region']
```

[302]:
```
0    Finland
Name: Country or region, dtype: object
```

[303]:
```
maxSupport=np.min(happyness_2019["Social support"])
maxSupport
```

[303]:
```
0.0
```

[304]:
```
minEconomy=np.min(happyness_2019["GDP per capita"])
minEconomy
```

[304]:
```
0.0
```

[305]:
```
happyness_2019[happyness_2019['Score']==np.
↪min(happyness_2019['Score'])]['Country or region']
```

[305]:
```
155    South Sudan
Name: Country or region, dtype: object
```

1.0.2 To rename columns

[306]:
```
happyness_2019 .rename(columns={"Country or region":"Country",
                       "GDP per capita":"Economy",
                      "Healthy life expectancy":"Health",
                      "Freedom to make life choices":"Freedom",
                      "Overall rank":"Happiness Rank"},inplace=True)
happyness_2019 .columns
```

[306]:
```
Index(['Happiness Rank', 'Country', 'Score', 'Economy', 'Social support',
       'Health', 'Freedom', 'Generosity', 'Perceptions of corruption'],
      dtype='object')
```

Removing unnecessary columns (Freedom to make life choices and Healthy life expectancy)

[160]:
```
''' drop multiple column based on name in pandas'''

df_new = happyness_2019 .drop(['Perceptions of corruption', 'Generosity'], axis␣
↪= 1)
df_new
df_new.shape
```

[160]:
```
(156, 7)
```

[161]:
```
df_new.columns
```

[161]:
```
Index(['Overall rank', 'Country or region', 'Score', 'GDP per capita',
       'Social support', 'Healthy life expectancy',
       'Freedom to make life choices'],
      dtype='object')
```

[162]:
```
df_new.head(1)
```

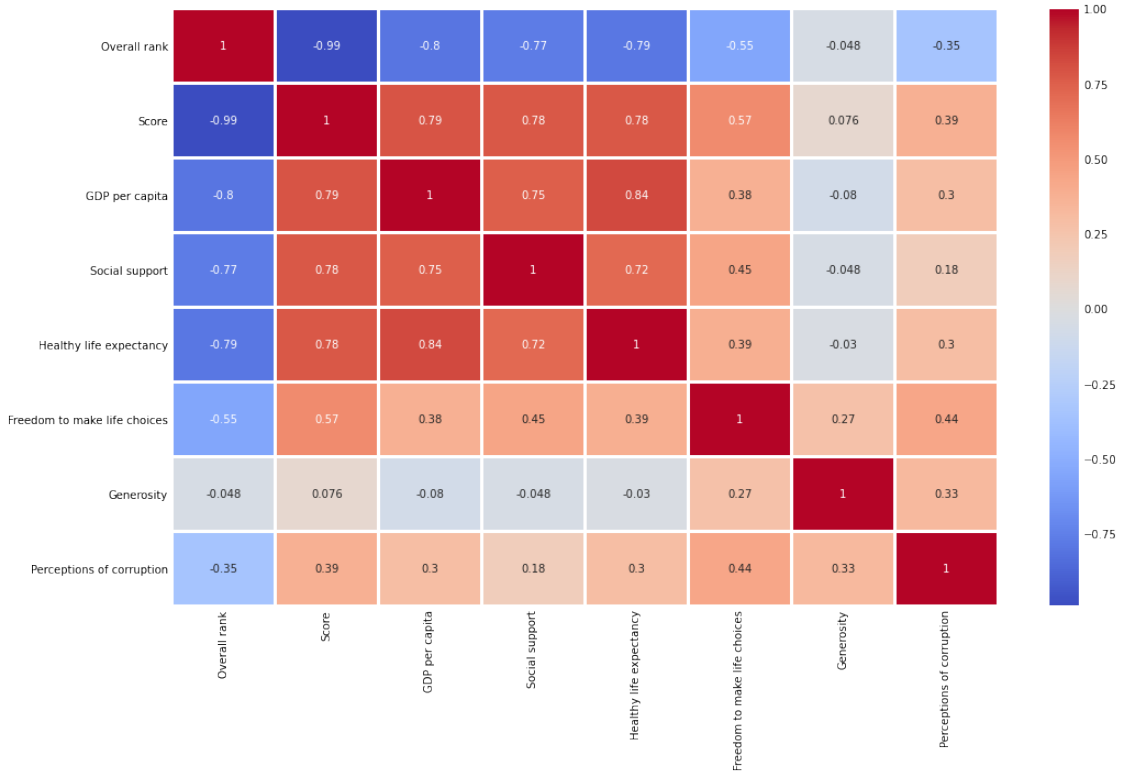
[162]:
```
   Overall rank Country or region  Score  GDP per capita  Social support  \
0             1           Finland  7.769            1.34           1.587

   Healthy life expectancy  Freedom to make life choices
0                    0.986                         0.596
```

2 Visualization

2.0.1 The correlation of the variables of the dataset

[347]:
```
fig, ax = plt.subplots()
fig.set_size_inches(15, 10)
sns.heatmap(happyness_2019 .
 ↪corr(),cmap='coolwarm',ax=ax,annot=True,linewidths=2)
```

[347]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x203d5119550>
```

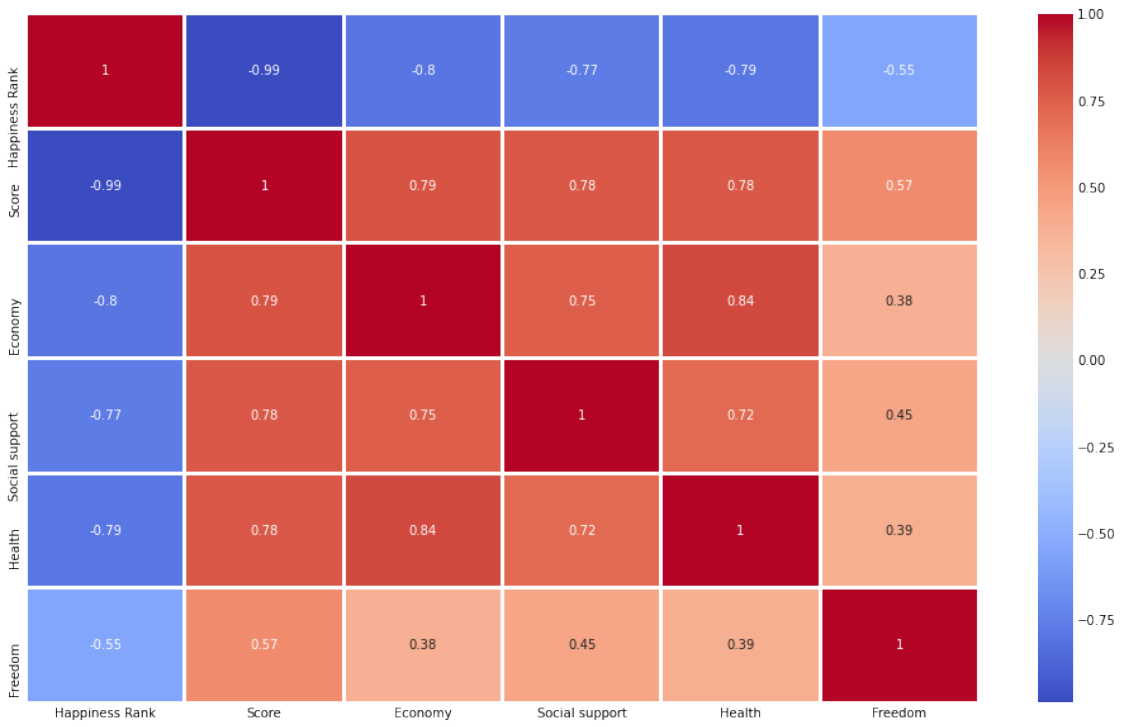


There is an inverse correlation between “overall Rank” and all the other numerical variables. The lower the happiness(overall) rank, the higher the score, and the higher the other six factors that contribute to happiness.

2.0.2 The correlation of the new dataset after droping columns - (Perceptions of corruption,and Generosity)

[312]:
```
#The correlation of the new dataset
fig, ax = plt.subplots()
fig.set_size_inches(15, 10)
sns.heatmap(df_new.corr(),cmap='coolwarm',ax=ax,annot=True,linewidths=2)
```

[312]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x203ca391820>
```



According to the above correlation plot, Economy, social support, and health play the most significant role in contributing to happiness. Freedom, generosity and perceptions of corruption have the lowest impact on the happiness score.

[313]:
```
happyness_2019.columns
```

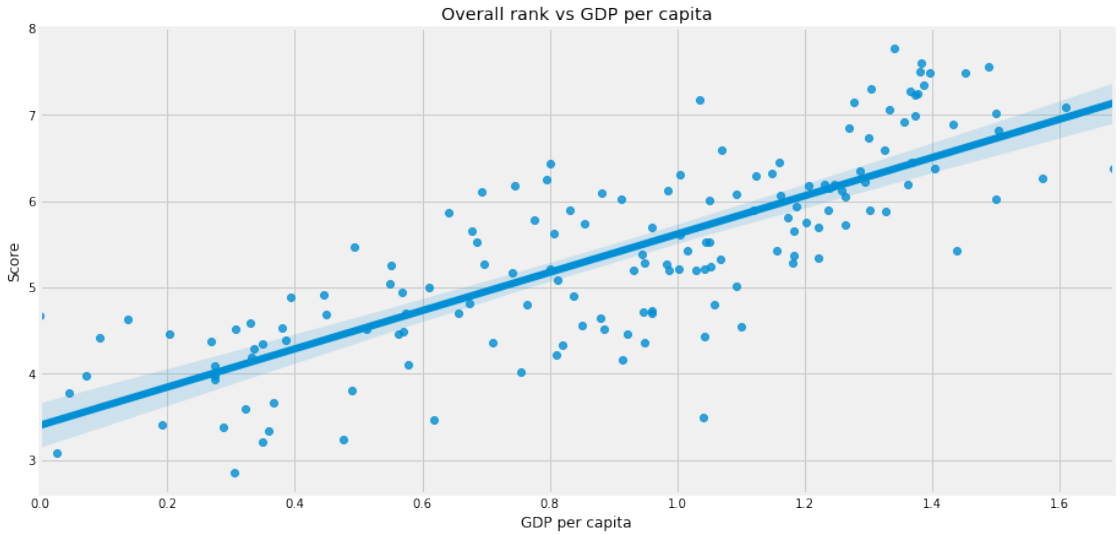
[313]:
```
Index(['Happiness Rank', 'Country', 'Score', 'Economy', 'Social support',
       'Health', 'Freedom', 'Generosity', 'Perceptions of corruption'],
      dtype='object')
```

3 Let's see relationship between different features with happiness score.

3.0.1 1. GDP per capita

Relationship between GDP per capita(Economy of country) has postive strong relationship with happiness score. So If GDP per Capita of a country is high than Happiness Score of that country also more likely to high.

[314]:
```
#https://www.kaggle.com/dgtech/world-happiness-with-basic-visualization-and-eda
import matplotlib.pyplot as plt
import seaborn as sb
import warnings
warnings.filterwarnings('ignore')
import pandas as pd

happyness_2019 = pd.read_csv("happyness_2019.csv")

plt.figure(figsize=(14,7))

plt.title("Overall rank vs GDP per capita")
sb.regplot(data=happyness_2019, x='GDP per capita', y='Score');
```

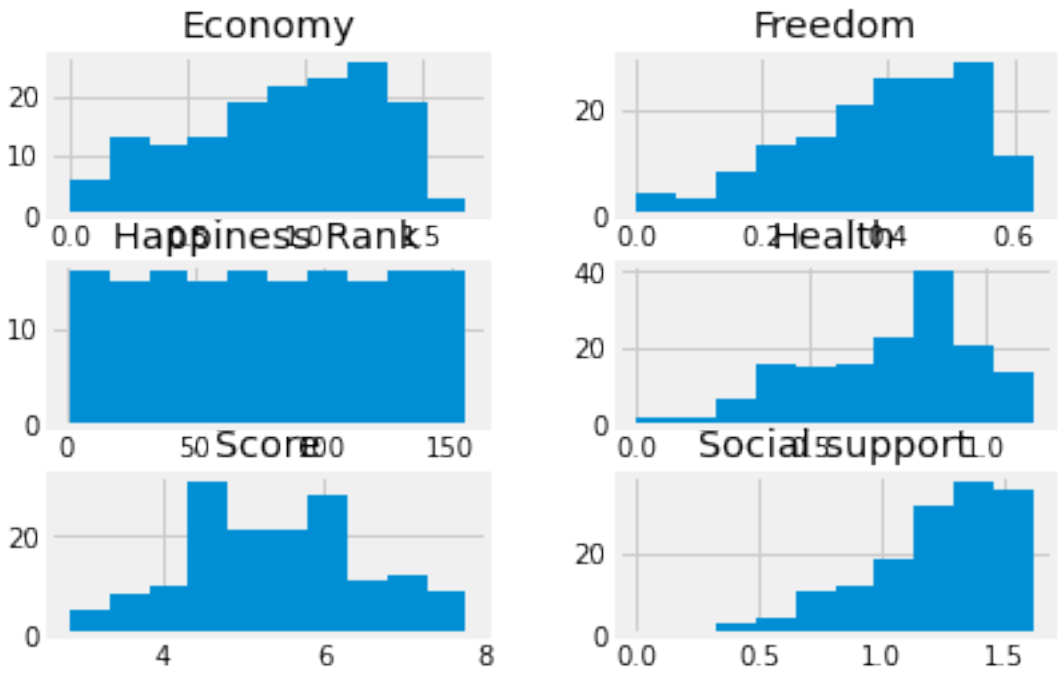


Using the histogram helps us to make the decision making process a lot more easy to handle by viewing the data that was collected

[315]:
```
df_new.hist()
```

[315]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x00000203D27FED30>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000203D03A5BB0>],
```

```
 [<matplotlib.axes._subplots.AxesSubplot object at 0x00000203CF8B78B0>,
  <matplotlib.axes._subplots.AxesSubplot object at 0x00000203CA0DA130>],
 [<matplotlib.axes._subplots.AxesSubplot object at 0x00000203CFA541F0>,
  <matplotlib.axes._subplots.AxesSubplot object at 0x00000203D287C5B0>]],
dtype=object)
```



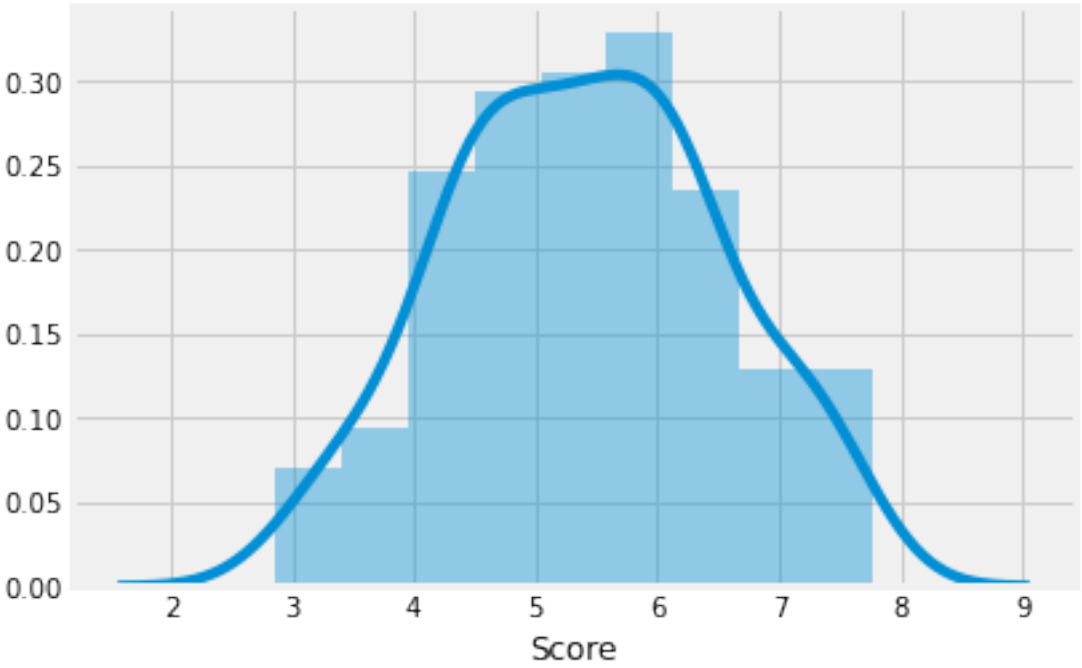
3.0.2 How is the Happiness Score is distributed?

As you can see below happiness score has values above 2.86 and below 7.77. So there is no single country which has happiness score above 8.

[316]:
```
sns.distplot(df_new['Score'])
df_new.head(1)
```

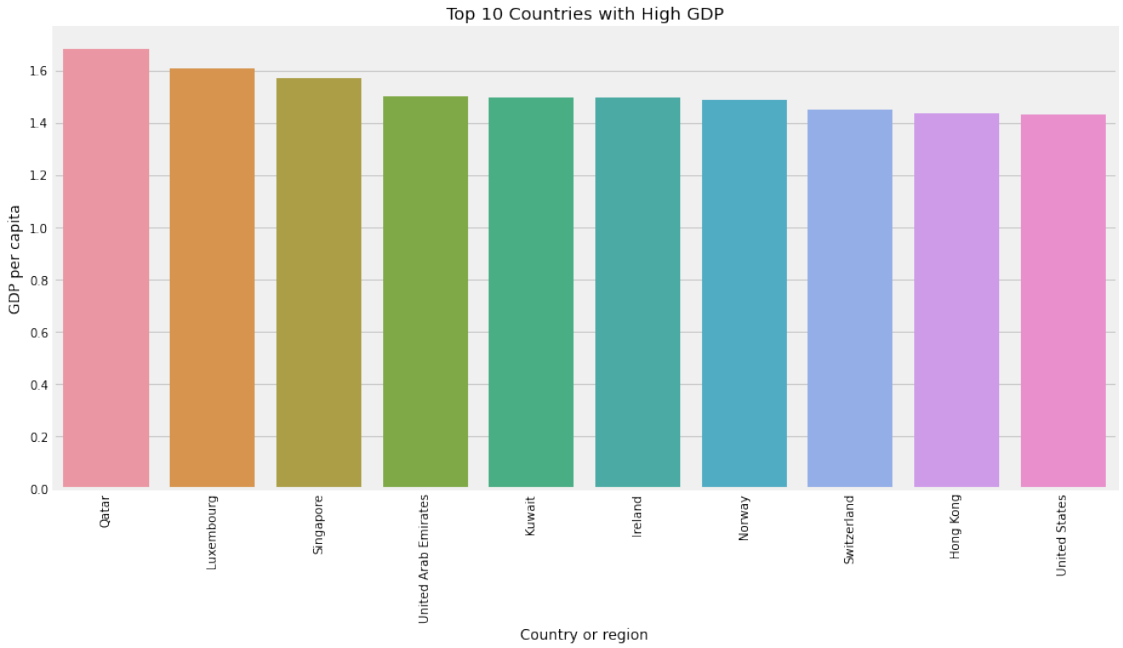
[316]:

	Happiness Rank	Country	Score	Economy	Social support	Health	Freedom
0	1	Finland	7.769	1.34	1.587	0.986	0.596



3.0.3 Top 10 Countries with high GDP (Economy)

[317]:
```
plt.figure(figsize=(14,7))
plt.title("Top 10 Countries with High GDP")
sb.barplot(data = happyness_2019.sort_values('GDP per capita', ascending=␣
 ↪False).head(10), y='GDP per capita', x='Country or region')
plt.xticks(rotation=90);
```



3.0.4 2. Perceptions of corruption

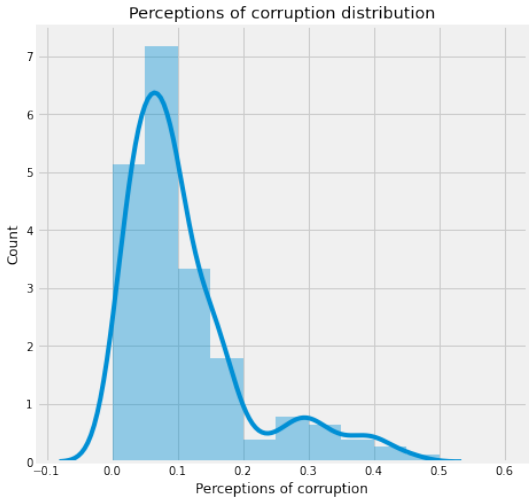
Distribution of Perceptions of corruption rightly skewed, which means very less number of country has high perceptions of corruption. That means most of the country has corruption problem.

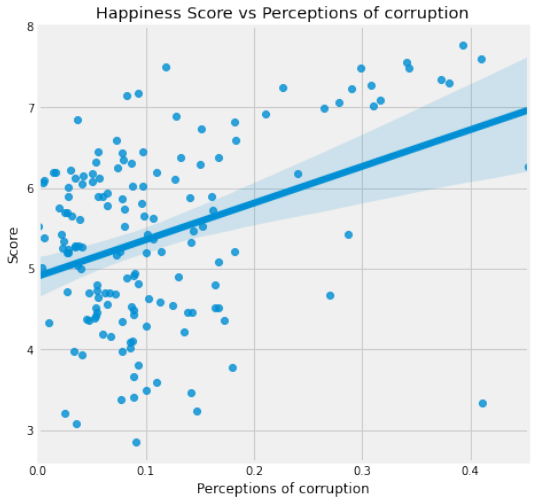
Corruption is a very big problem for the word. How corruption can impact on Happiness Score?

Perceptions of corruption data is highly skewed no wonder why the data has weak linear relationship, but as you can see in scatter plot most of the data points are on left side and most of the countries with low perceptions of corruption has happiness score between 4 to 6.

Countries with high perception score has high happiness score above 7.

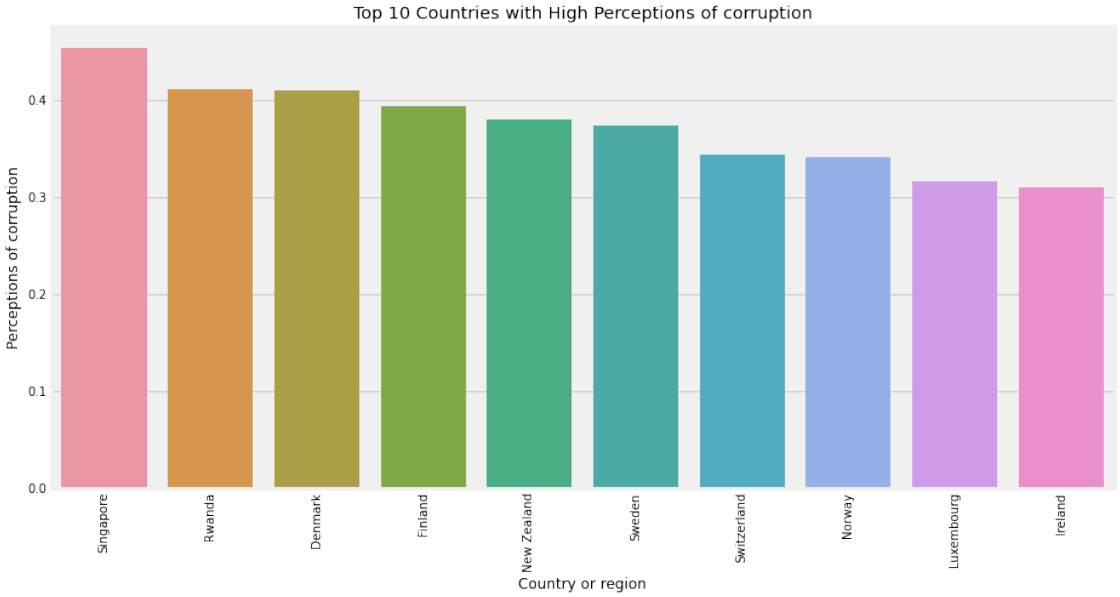
[318]:
```
plt.figure(figsize= (15,7))

plt.subplot(1,2,1)
plt.title("Perceptions of corruption distribution")
sb.distplot(a=happyness_2019['Perceptions of corruption'], bins =np.arange(0, 0.
↪45+0.2,0.05))
plt.ylabel('Count')

plt.subplot(1,2,2)
plt.title("Happiness Score vs Perceptions of corruption")
sb.regplot(data=happyness_2019, x='Perceptions of corruption', y='Score');
```





3.0.5 Top 10 Countries with high Perceptions of corruption

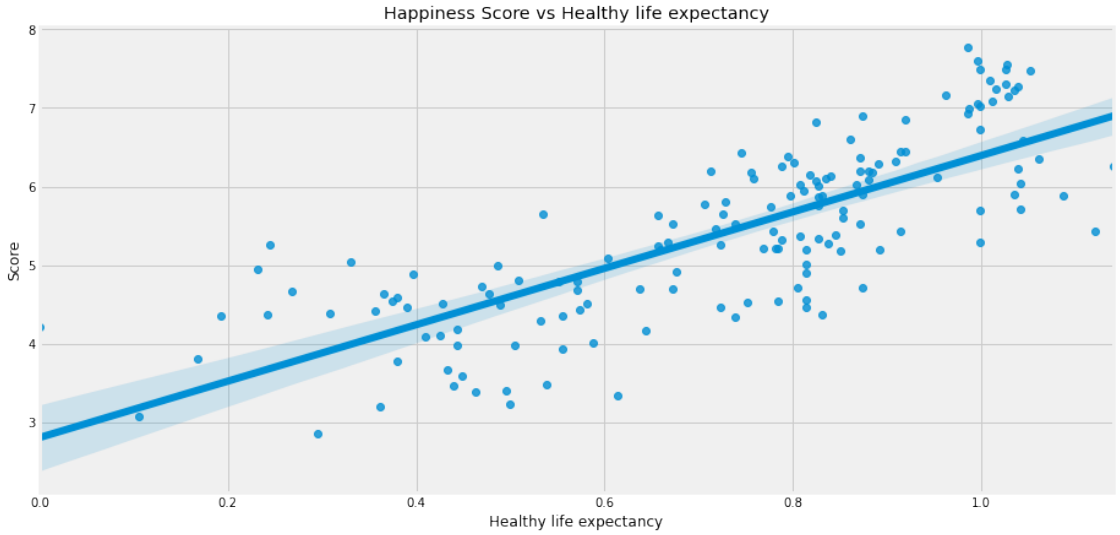
[319]:
```
plt.figure(figsize=(14,7))
plt.title("Top 10 Countries with High Perceptions of corruption")
sb.barplot(data =happyness_2019.sort_values('Perceptions of corruption',␣
 ↪ascending= False).head(10), x='Country or region', y='Perceptions of␣
 ↪corruption')
plt.xticks(rotation=90);
```



3.0.6 3. Healthy life expectancy

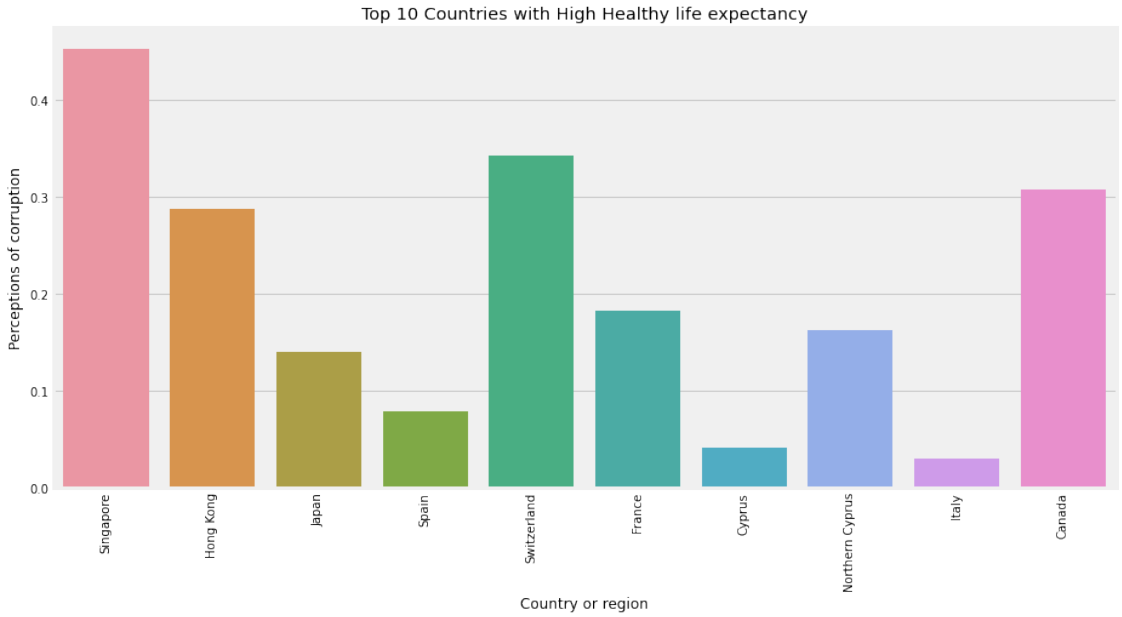
A healthy life expectancy has a strong and positive relationship with a happiness score. If a country has a high life expectancy that means it can also have a high happiness score. It makes sense because anyone who has a very long healthy life he/she is happy. Everyone likes to get a healthy and long life aren't you.

[320]:
```
plt.figure(figsize=(14,7))

plt.title("Happiness Score vs Healthy life expectancy")
sb.regplot(data=happyness_2019, x='Healthy life expectancy', y='Score');
```



3.0.7 Top 10 Countries with high Healthy life expectancy

[321]:
```
plt.figure(figsize=(14,7))
plt.title("Top 10 Countries with High Healthy life expectancy")
sb.barplot(data = happyness_2019.sort_values('Healthy life expectancy',␣
 ↪ascending= False).head(10), x='Country or region', y='Perceptions of␣
 ↪corruption')
plt.xticks(rotation=90);
```

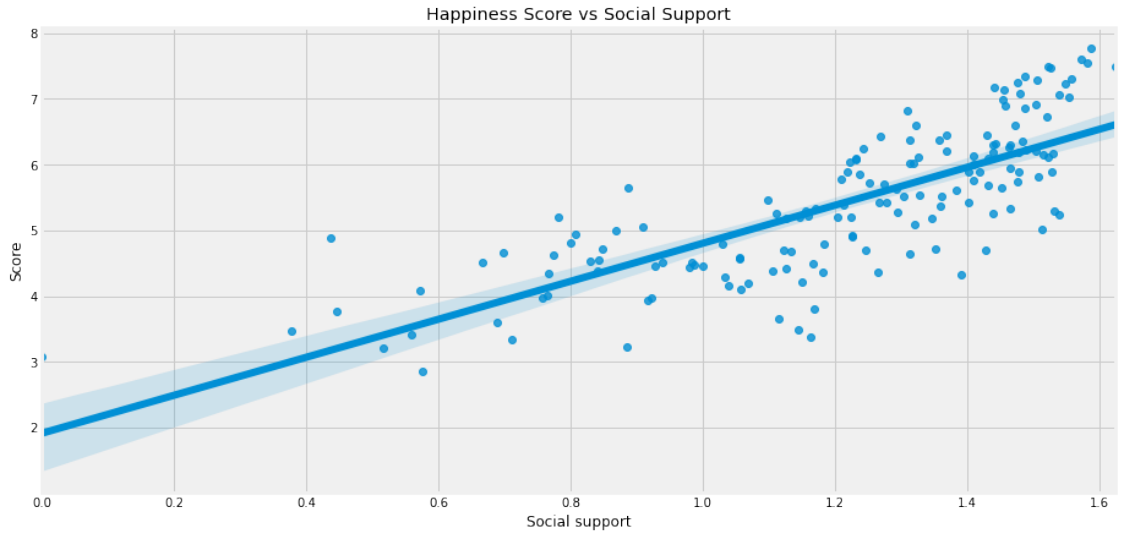


3.0.8 4. Social Support

Social support of countries also has a strong and positive relationship with happiness scores. Also, the relationship with happiness score needs to be strong because the more you will help socially more you will be happy.

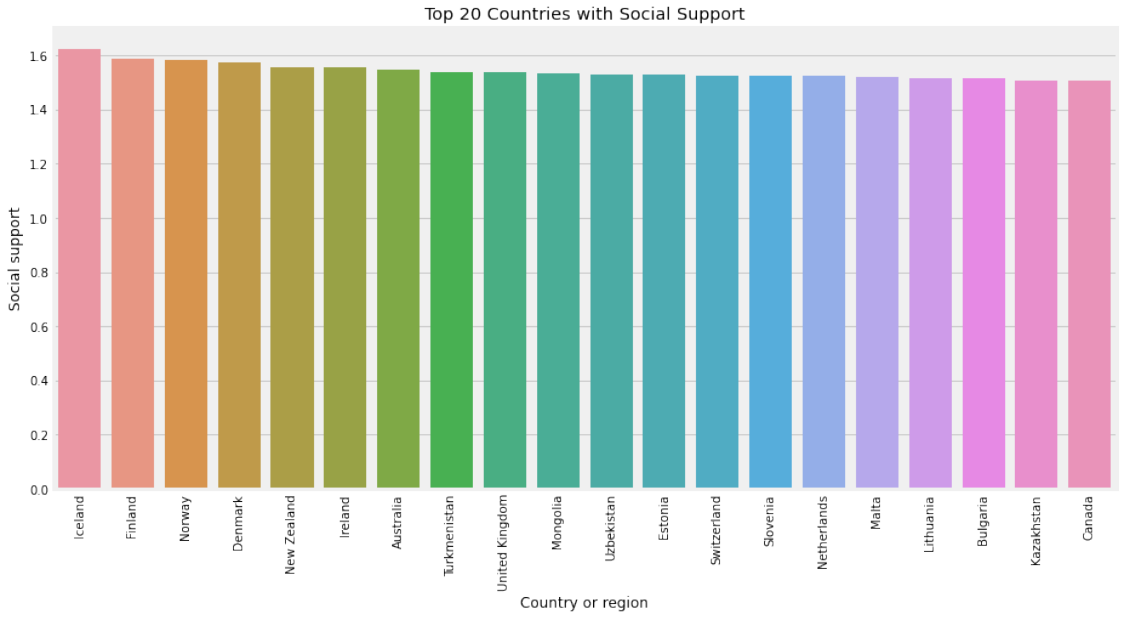
Social support measures the perception that one has assistance available, the received assistance, or the degree to which a person can integrate into a social network. Support can come from many sources, such as family, friends, pets, neighbors, coworkers,

[322]:
```
import matplotlib.pyplot as plt
import seaborn as sb
import warnings
import pandas as pd

happyness_2019 = pd.read_csv("happyness_2019.csv")
warnings.filterwarnings('ignore')
plt.figure(figsize=(14,7))

plt.title("Happiness Score vs Social Support")
sb.regplot(data=happyness_2019, x='Social support', y='Score');
```



4 Top 20 Countries with high Social Support

[323]:
```
plt.figure(figsize=(14,7))
plt.title("Top 20 Countries with Social Support")
sb.barplot(data = happyness_2019.sort_values('Social support', ascending=␣
 ↪False).head(20), x='Country or region', y='Social support')
plt.xticks(rotation=90);
```

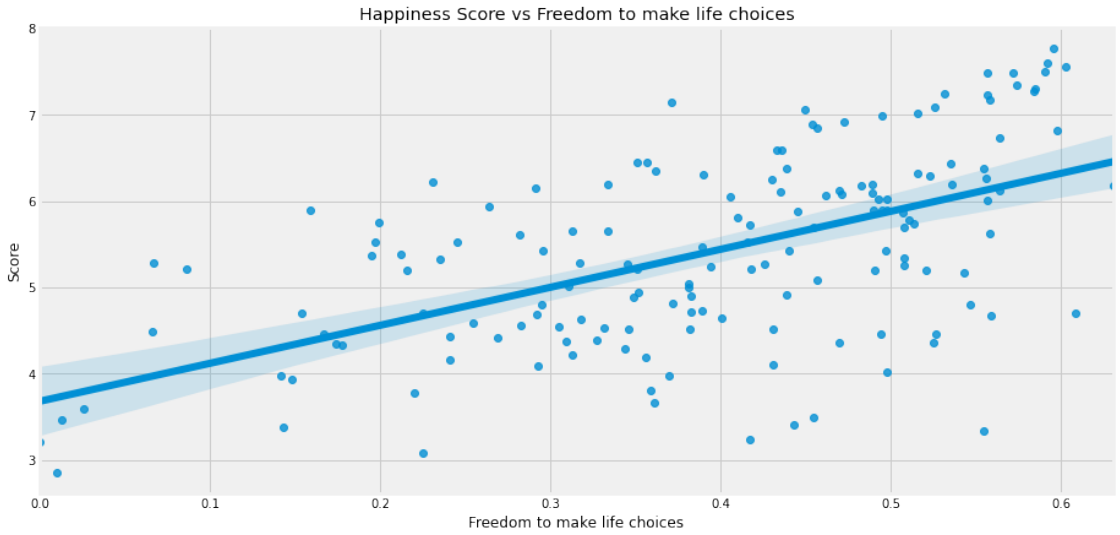


5 Freedom to make life choices

"Freedom to make life choices" is the national average of responses to the question "Are you satisfied or dissatisfied with your freedom to choose what you do with your life?"

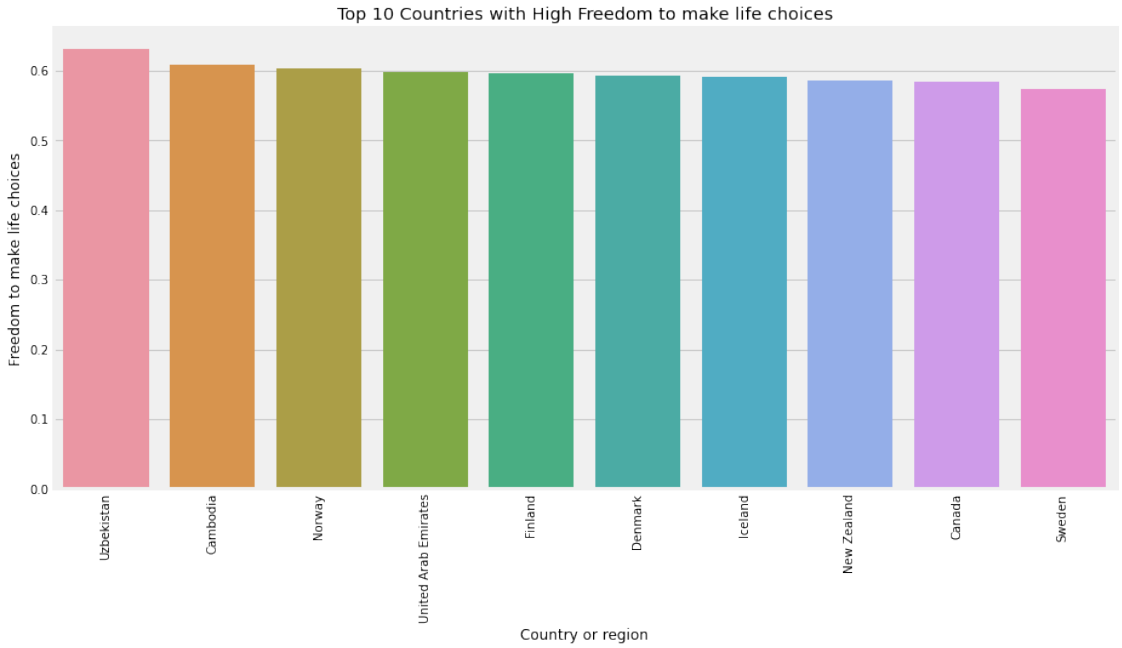
Freedom to make life choices has some positive relationships with happiness scores. This relation makes sense because the more you will get free to make decisions about your life, the more you will be happy.

[324]:
```
plt.figure(figsize=(14,7))

plt.title("Happiness Score vs Freedom to make life choices")
sb.regplot(data=happyness_2019, x='Freedom to make life choices', y='Score');
```



6 Top 10 Countries with high Freedom to make life choices

[325]:
```
plt.figure(figsize=(14,7))
plt.title("Top 10 Countries with High Freedom to make life choices")
sb.barplot(data = happyness_2019.sort_values('Freedom to make life choices',␣
 ↪ascending= False).head(10), x='Country or region', y='Freedom to make life␣
 ↪choices')
plt.xticks(rotation=90);
```



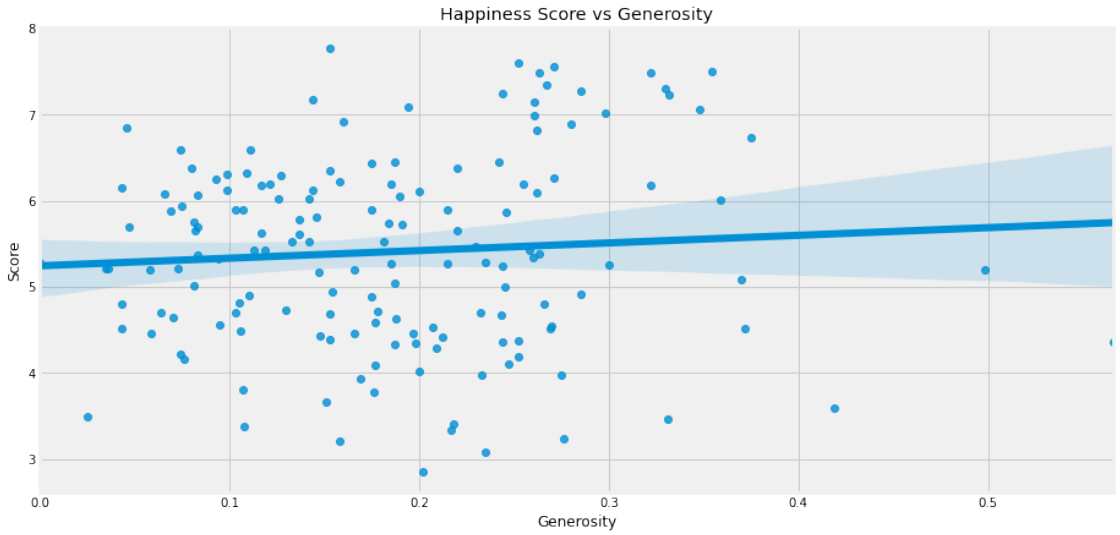
6.0.1 6. Generosity

Generosity and life expectancy are among the six variables scientists peek at when making the World Happiness Report.

Generosity has a weak linear relationship with the happiness score. One can ask a question:" Why generosity has not a linear relationship with happiness score?

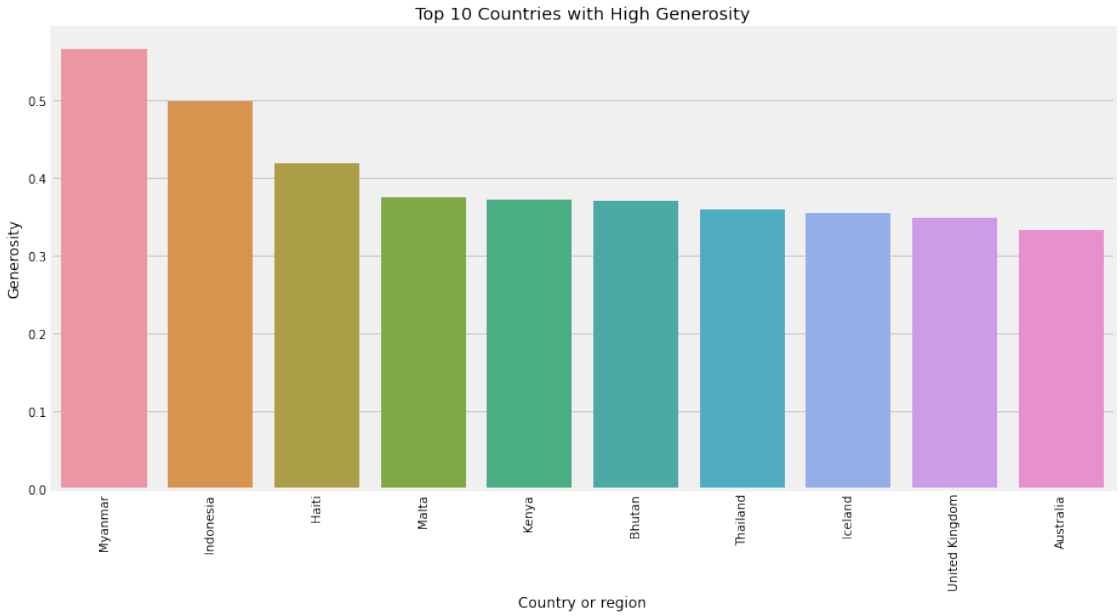
Generosity score depends on the countries that can give the most to nonprofits around the world. Countries which are not generous that does not mean they are not happy.

[326]:
```
plt.figure(figsize=(14,7))

plt.title("Happiness Score vs Generosity")
sb.regplot(data=happyness_2019, x='Generosity', y='Score');
```



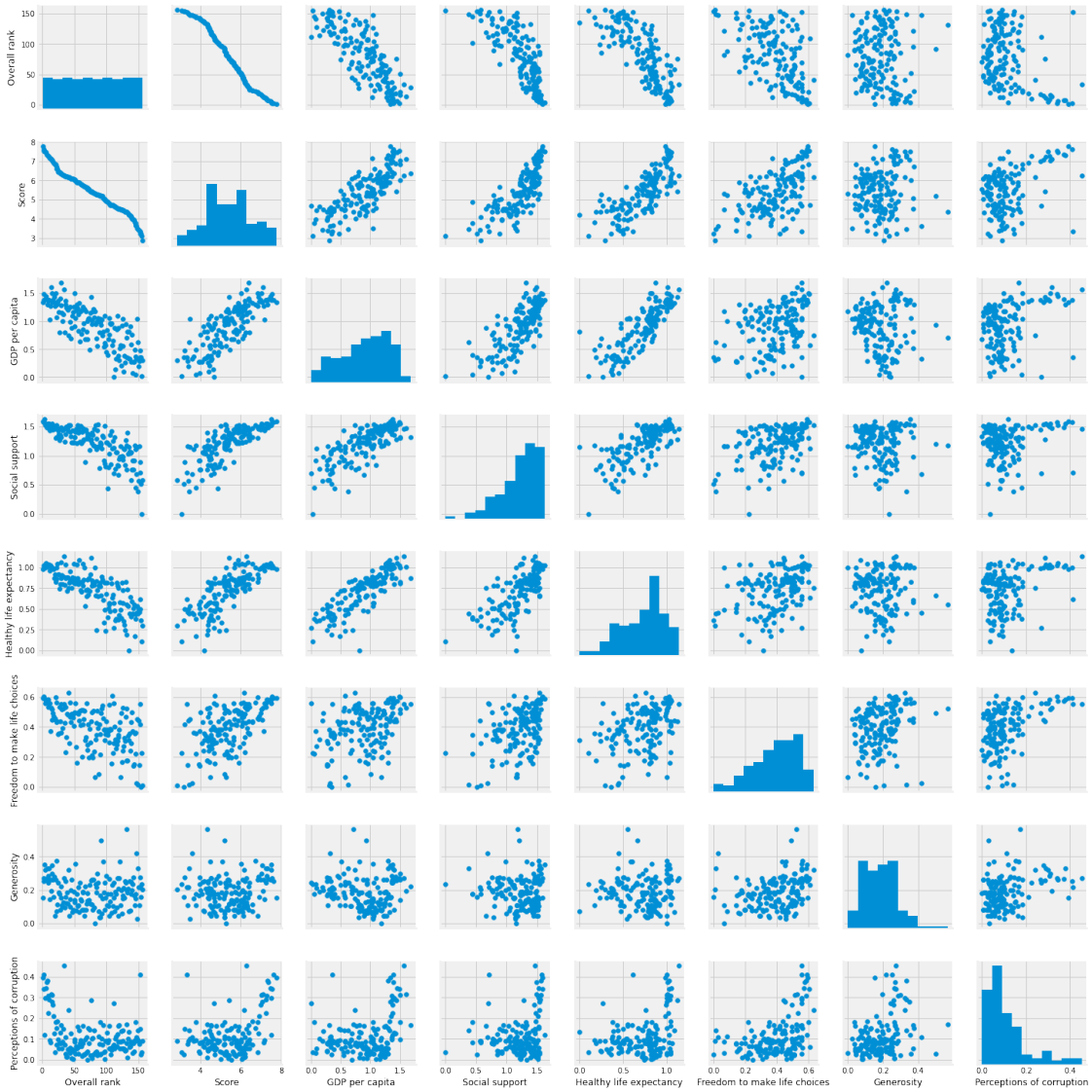
7 Top 10 Countries with high Generosity

[327]:
```
plt.figure(figsize=(14,7))

plt.title("Top 10 Countries with High Generosity")
sb.barplot(data = happyness_2019.sort_values('Generosity', ascending= False).
 ↪head(10), x='Country or region', y='Generosity')
plt.xticks(rotation=90);
```



8 How one feature is related to another feature?

[328]:
```
p = sb.PairGrid(happyness_2019)
p.map_diag(plt.hist)
p.map_offdiag(plt.scatter);
```

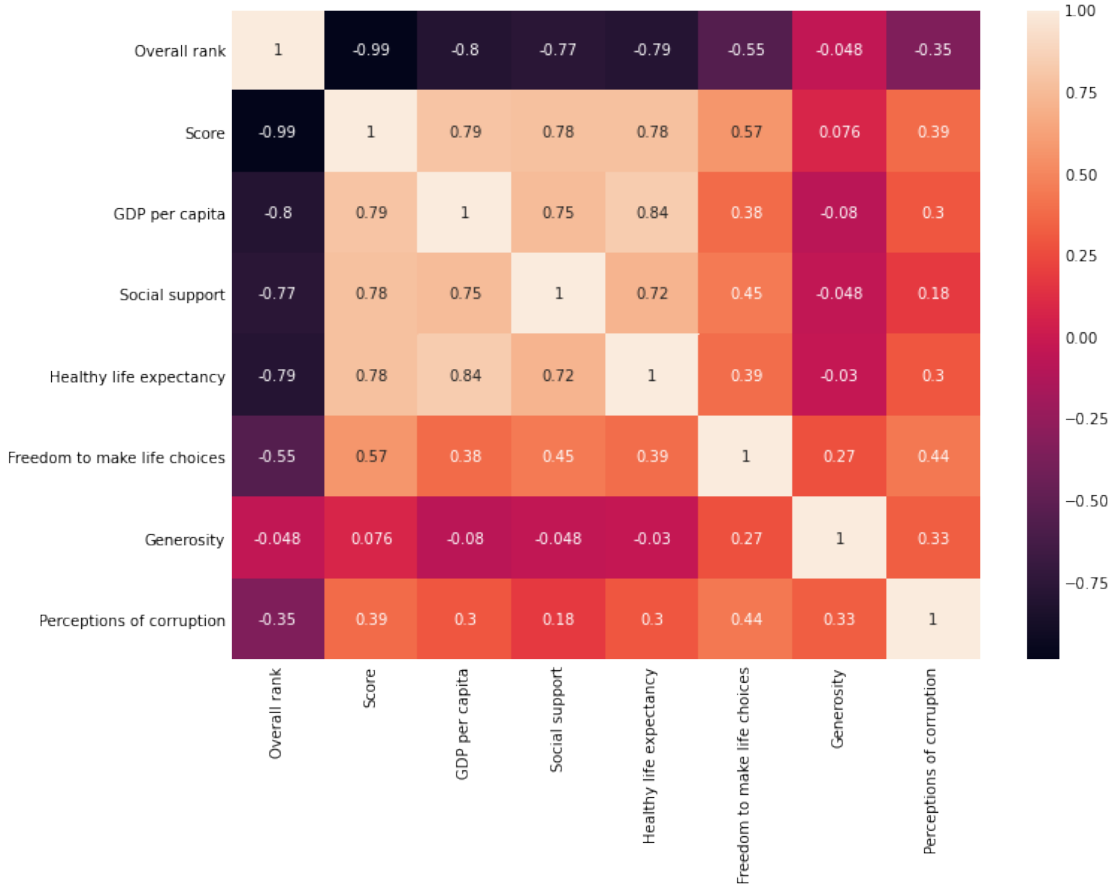


9 correlation of the entire dataset

[329]:
```
happyness_2019.corr() # correlation of the entire dataset
```

[329]:
```
                              Overall rank     Score  GDP per capita  \
Overall rank                      1.000000 -0.989096       -0.801947
Score                            -0.989096  1.000000        0.793883
GDP per capita                   -0.801947  0.793883        1.000000
Social support                   -0.767465  0.777058        0.754906
Healthy life expectancy          -0.787411  0.779883        0.835462
Freedom to make life choices     -0.546606  0.566742        0.379079
Generosity                       -0.047993  0.075824       -0.079662
Perceptions of corruption        -0.351959  0.385613        0.298920

                              Social support  Healthy life expectancy  \
Overall rank                       -0.767465                -0.787411
Score                               0.777058                 0.779883
GDP per capita                      0.754906                 0.835462
Social support                      1.000000                 0.719009
Healthy life expectancy             0.719009                 1.000000
Freedom to make life choices        0.447333                 0.390395
Generosity                         -0.048126                -0.029511
Perceptions of corruption           0.181899                 0.295283

                              Freedom to make life choices  Generosity  \
Overall rank                                    -0.546606   -0.047993
Score                                            0.566742    0.075824
GDP per capita                                   0.379079   -0.079662
Social support                                   0.447333   -0.048126
Healthy life expectancy                          0.390395   -0.029511
Freedom to make life choices                     1.000000    0.269742
Generosity                                       0.269742    1.000000
Perceptions of corruption                        0.438843    0.326538

                              Perceptions of corruption
Overall rank                                  -0.351959
Score                                          0.385613
GDP per capita                                 0.298920
Social support                                 0.181899
Healthy life expectancy                        0.295283
Freedom to make life choices                   0.438843
Generosity                                     0.326538
Perceptions of corruption                      1.000000
```

[330]:
```
plt.figure(figsize=(10,8))
sns.heatmap(happyness_2019.corr(), annot=True);
```



The lower the overall rank, the higher the happiness score, and the higher the other five factors that contribute to happiness. The above correlation plot shows that the economy, life expectancy, and family play the most significant role in contributing to happiness. Trust (perception of corruption) and generosity have the lowest impact on happiness score.

10 Random Forest

10.0.1 Regression with RandomForestRegressor in Python

Random forest is an ensemble learning algorithm based on decision tree learners. The estimator fits multiple decision trees on randomly extracted subsets from the dataset and averages their prediction.

Scikit-learn API provides the RandomForestRegressor class included in the ensemble module to implement the random forest for the regression problem.

In the following lines, we'll briefly learn how to fit and predict regression data by using the RandomForestRegressor class in Python.

We will cover:

Preparing the data Training the model **Predicting and accuracy check** Happiness dataset

prediction **visualize the original and predicted data** calculate the root mean squared error: RMS or MSE

[109]:
```
import numpy as np # linear algebra
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split,cross_val_score
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestRegressor# data processing, CSV file I/
↪O (e.g. pd.read_csv)
from sklearn.linear_model import LinearRegression
```

[143]:
```
#The following command imports the CSV dataset using pandas:

#import numpy as np
#import matplotlib.pyplot as plt
import pandas as pd

happyness_2019 = pd.read_csv("happyness_2019.csv")
happyness_2019.head(1)
```

[143]:
```
   Overall rank Country or region  Score  GDP per capita  Social support  \
0             1           Finland  7.769            1.34           1.587

   Healthy life expectancy  Freedom to make life choices  Generosity  \
0                    0.986                         0.596       0.153

   Perceptions of corruption
0                      0.393
```

10.0.2 Data Preprocessing will be done with the help of following script lines

[111]:
```
X=happyness_2019.iloc[:,3:]
y=happyness_2019["Score"]
```

10.0.3 Next, we will divide the data into train and test split. The following code will split the dataset into 70% training data and 30% of testing data.

[112]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.
↪30,random_state=42)
print(y_train.shape)
print(y_test.shape)
```

```
(109,)
(47,)
```

10.0.4 Training the model

To define the regressor model by using the RandomForestRegressor class. Here, we can use default parameters of the RandomForestRegressor class. The default values can be seen in below.

[113]:
```
from sklearn import set_config
set_config(print_changed_only=False)

rfr = RandomForestRegressor()
#print(rfr)
```

[114]:
```
# Next, train the model with the help of RandomForestRegressor class of sklearn␣
 ↪as follows

from sklearn.ensemble import RandomForestRegressor
model=RandomForestRegressor(random_state=45)
model.fit(X_train,y_train)
```

[114]:
```
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=None, max_features='auto', max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      n_estimators=100, n_jobs=None, oob_score=False,
                      random_state=45, verbose=0, warm_start=False)
```

[115]:
```
# Then, we'll fit the model on train data and check the model accuracy score.

rfr.fit(X_train, y_train)

score = rfr.score(X_train, y_train)
print("R-squared:", score)
```

```
R-squared: 0.9789401284695045
```

10.0.5 Predicting and accuracy check

Now, we can predict the test data by using the trained model. We can check the accuracy of predicted data by using MSE and RMSE metrics.

The root-mean-square deviation (RMSD) or root-mean-square error (RMSE) is a frequently used measure of the differences between values (sample or population values) predicted by a model or an estimator and the values observed.

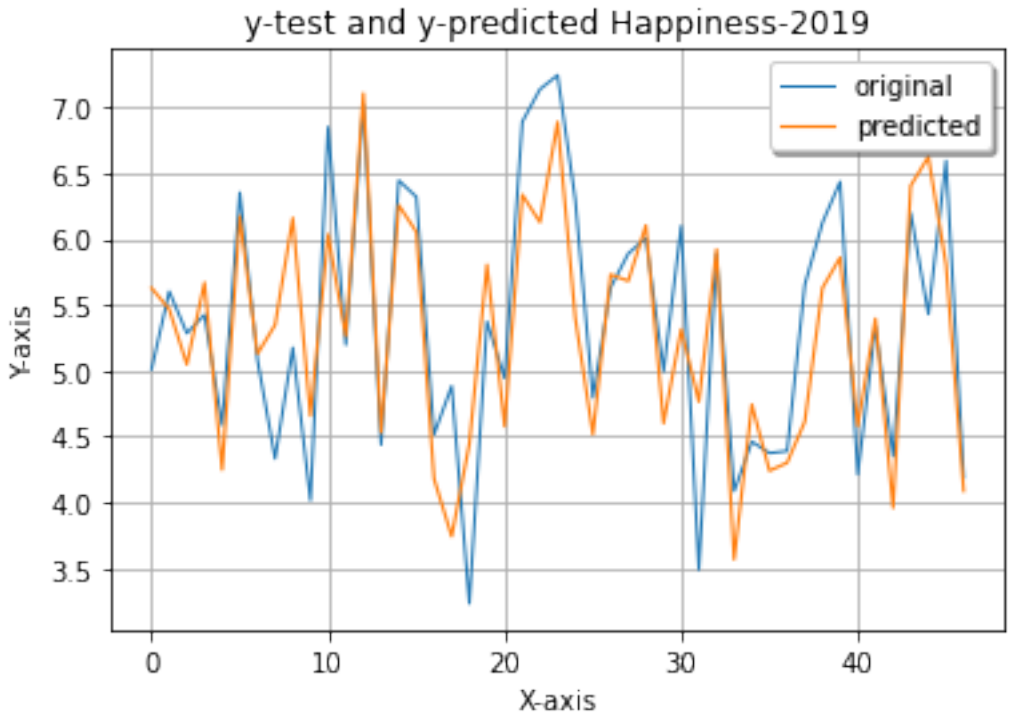
The mean squared error (MSE) or mean squared deviation (MSD) of an estimator measures the average of the squares of the errors—that is, the average squared difference between the estimated values and the actual value.

[116]:
```
ypred = rfr.predict(X_test)

mse = mean_squared_error(y_test, ypred)
print("MSE: ", mse)
print("RMSE: ", mse*(1/2.0))
```

```
MSE:  0.34732590460851087
RMSE:  0.17366295230425544
```

10.0.6 Finally, we'll visualize the original and predicted data (happyness_2019) in a plot.

[117]:
```
x_ax = range(len(y_test))
plt.plot(x_ax, y_test, linewidth=1, label="original")
plt.plot(x_ax, ypred, linewidth=1.1, label="predicted")
plt.title("y-test and y-predicted Happiness-2019")
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.legend(loc='best',fancybox=True, shadow=True)
plt.grid(True)
plt.show()
```



10.0.7 To calculate the root mean squared error in from a pandas data frame

[118]:
```
from sklearn.metrics import mean_squared_error
ypred=model.predict(X_test)
rms = np.sqrt(mean_squared_error(y_test,ypred))
rms
```

[118]:
```
0.5861206005175589
```

[119]:
```
params={"max_depth":list(range(5,10)),
       "max_features":[5,10],
       "n_estimators":[200,500,1000,]}
```

[120]:
```
from sklearn.model_selection import GridSearchCV
rf_model=RandomForestRegressor(random_state=42)
rf_model.fit(X_train,y_train)
cv_model=GridSearchCV(rf_model,params,cv=10,n_jobs=-1)
cv_model.fit(X_train,y_train)
```

[120]:
```
GridSearchCV(cv=10, error_score=nan,
             estimator=RandomForestRegressor(bootstrap=True, ccp_alpha=0.0,
                                             criterion='mse', max_depth=None,
                                             max_features='auto',
                                             max_leaf_nodes=None,
                                             max_samples=None,
                                             min_impurity_decrease=0.0,
                                             min_impurity_split=None,
                                             min_samples_leaf=1,
                                             min_samples_split=2,
                                             min_weight_fraction_leaf=0.0,
                                             n_estimators=100, n_jobs=None,
                                             oob_score=False, random_state=42,
                                             verbose=0, warm_start=False),
             iid='deprecated', n_jobs=-1,
             param_grid={'max_depth': [5, 6, 7, 8, 9], 'max_features': [5, 10],
                         'n_estimators': [200, 500, 1000]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring=None, verbose=0)
```

[121]:
```
cv_model.best_params_
```

[121]:
```
{'max_depth': 7, 'max_features': 5, 'n_estimators': 1000}
```

[137]:
```
son_model=RandomForestRegressor(max_depth=9,
                                max_features=5,
                                n_estimators=1000)
son_model.fit(X_train,y_train)
```

[137]:
```
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=9, max_features=5, max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      n_estimators=1000, n_jobs=None, oob_score=False,
                      random_state=None, verbose=0, warm_start=False)
```

[138]:
```
predicted=son_model.predict(X_test)
np.sqrt(mean_squared_error(y_test,predicted))
```

[138]:
```
0.5149659835784963
```

[146]:
```
model.score(X_train,y_train)
```

[146]:
```
0.8988680444456479
```

[152]:
```
cross_val_score(model,X_train,y_train,cv=10,scoring="r2").mean()
```

[152]:
```
0.7417583266139962
```