HappinessScoreOfaCountry

March 8, 2021

[494]:

```
import numpy as np
import pandas as pd
from sklearn import datasets
from sklearn.datasets import load_boston
from sklearn import linear_model
from sklearn.linear_model import LinearRegression
from sklearn import preprocessing
from sklearn.preprocessing import PolynomialFeatures
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
import seaborn as sns
```

[495]:

```
np.random.seed(42)
```

[496]:

```
d = pd.read_csv('/home/hiraditya/Desktop/HomeWork/SJSU/cs156/jupiter/
 ↪JupyterBooks/homework3_input_data.csv')
```

[497]:

```
d.head()
```

[497]:

```
       Country          Region  Happiness Rank  Happiness Score  \
0  Switzerland  Western Europe               1            7.587
1      Iceland  Western Europe               2            7.561
2      Denmark  Western Europe               3            7.527
3       Norway  Western Europe               4            7.522
4       Canada   North America               5            7.427

   Standard Error  Economy (GDP per Capita)   Family  \
0         0.03411                   1.39651  1.34951
1         0.04884                   1.30232  1.40223
2         0.03328                   1.32548  1.36058
3         0.03880                   1.45900  1.33095
4         0.03553                   1.32629  1.32261

   Health (Life Expectancy)  Freedom  Trust (Government Corruption)  \
0                   0.94143  0.66557                        0.41978
```

```
1                   0.94784  0.62877                        0.14145
2                   0.87464  0.64938                        0.48357
3                   0.88521  0.66973                        0.36503
4                   0.90563  0.63297                        0.32957

   Generosity  Dystopia Residual
0     0.29678            2.51738
1     0.43630            2.70201
2     0.34139            2.49204
3     0.34699            2.46531
4     0.45811            2.45176
```

[498]:
```
d.head(1)
```

[498]:
```
       Country          Region  Happiness Rank  Happiness Score  \
0  Switzerland  Western Europe               1            7.587

   Standard Error  Economy (GDP per Capita)   Family  \
0         0.03411                   1.39651  1.34951

   Health (Life Expectancy)  Freedom  Trust (Government Corruption)  \
0                   0.94143  0.66557                        0.41978

   Generosity  Dystopia Residual
0     0.29678            2.51738
```

[499]:
```
d.info()

#print(d.Happiness_Rank)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 158 entries, 0 to 157
Data columns (total 12 columns):
 #   Column                         Non-Null Count  Dtype
---  ------                         --------------  -----
 0   Country                        158 non-null    object
 1   Region                         158 non-null    object
 2   Happiness Rank                 158 non-null    int64
 3   Happiness Score                158 non-null    float64
 4   Standard Error                 158 non-null    float64
 5   Economy (GDP per Capita)       158 non-null    float64
 6   Family                         158 non-null    float64
 7   Health (Life Expectancy)       158 non-null    float64
 8   Freedom                        158 non-null    float64
 9   Trust (Government Corruption)  158 non-null    float64
 10  Generosity                     158 non-null    float64
 11  Dystopia Residual              158 non-null    float64
```

```
dtypes: float64(9), int64(1), object(2)
memory usage: 14.9+ KB
```

[500]:

```
d.rename(columns = {'Happiness Rank' : 'Happiness_Rank',
                    'Happiness Score' : 'Happiness_Score',
                    'Standard Error' : 'Standard_Error',
                    'Economy (GDP per Capita)' : 'Economy',
                    'Health (Life Expectancy)' : 'Health',
                   'Trust (Government Corruption)' : 'Trust',
                   'Dystopia Residual' : 'Dystopia_Residual'}, inplace =␣
 ↪True)
```

[501]:

```
print("printing target = \n" , d.Happiness_Score)
```

```
printing target =
 0      7.587
1      7.561
2      7.527
3      7.522
4      7.427
       …
153    3.465
154    3.340
155    3.006
156    2.905
157    2.839
Name: Happiness_Score, Length: 158, dtype: float64
```

[502]:

```
print("Printing all features =  \n", d.columns)
```

```
Printing all features =
 Index(['Country', 'Region', 'Happiness_Rank', 'Happiness_Score',
       'Standard_Error', 'Economy', 'Family', 'Health', 'Freedom', 'Trust',
       'Generosity', 'Dystopia_Residual'],
      dtype='object')
```

[503]:

```
# target = Happiness_Score
#data = other attributes like country region

print("Printing selective features to calculate happiness score =  \n", d.
 ↪Family, d.Health, d.Freedom,
      d.Trust, d.Economy, d.Generosity, d.Dystopia_Residual)
```

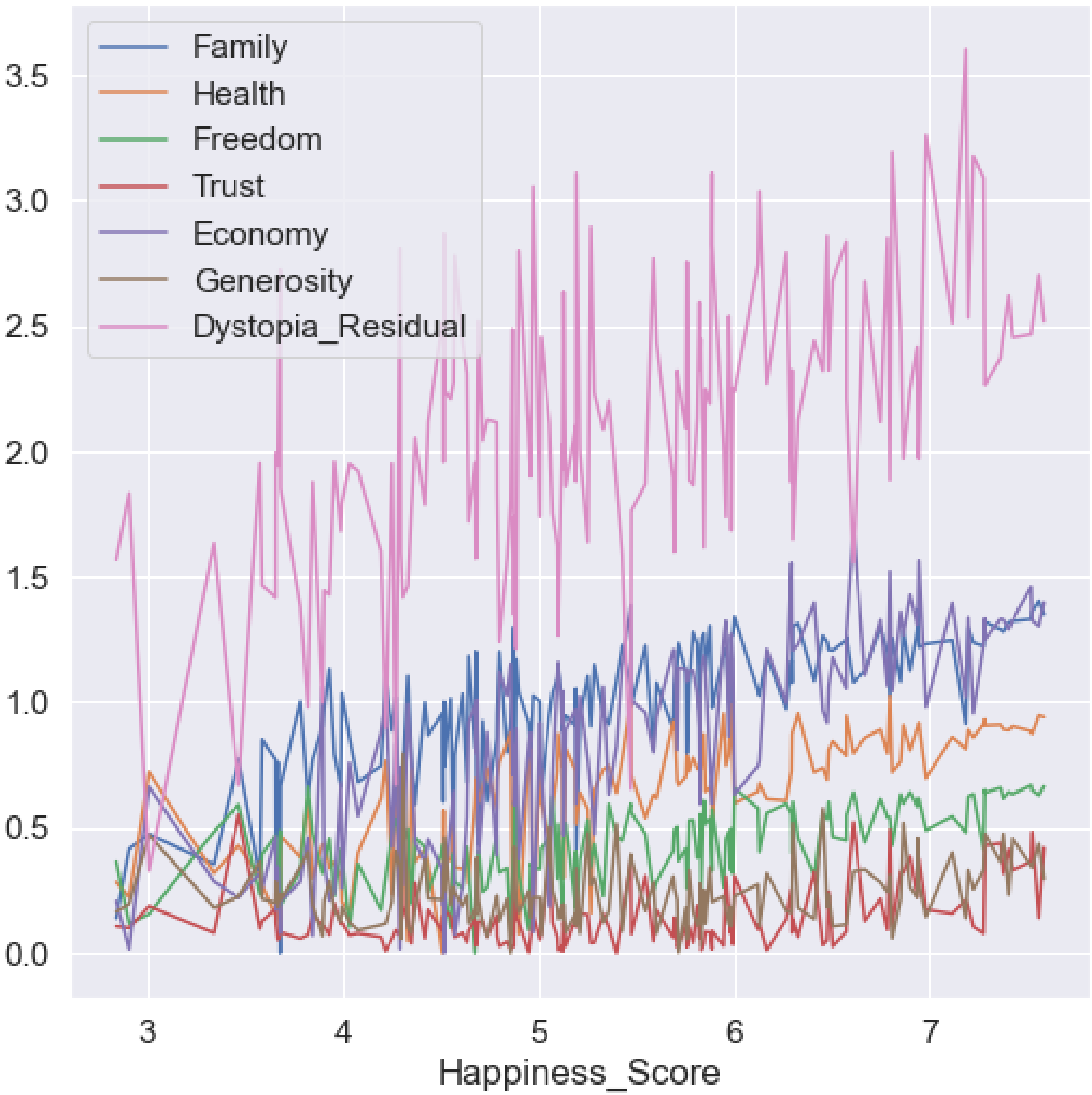
```
Printing selective features to calculate happiness score =
 0      1.34951
1      1.40223
2      1.36058
3      1.33095
```

```
4      1.32261
        …
153    0.77370
154    0.35386
155    0.47489
156    0.41587
157    0.13995
Name: Family, Length: 158, dtype: float64 0      0.94143
1      0.94784
2      0.87464
3      0.88521
4      0.90563
        …
153    0.42864
154    0.31910
155    0.72193
156    0.22396
157    0.28443
Name: Health, Length: 158, dtype: float64 0      0.66557
1      0.62877
2      0.64938
3      0.66973
4      0.63297
        …
153    0.59201
154    0.48450
155    0.15684
156    0.11850
157    0.36453
Name: Freedom, Length: 158, dtype: float64 0      0.41978
1      0.14145
2      0.48357
3      0.36503
4      0.32957
        …
153    0.55191
154    0.08010
155    0.18906
156    0.10062
157    0.10731
Name: Trust, Length: 158, dtype: float64 0      1.39651
1      1.30232
2      1.32548
3      1.45900
4      1.32629
        …
153    0.22208
154    0.28665
```

```
155    0.66320
156    0.01530
157    0.20868
Name: Economy, Length: 158, dtype: float64 0      0.29678
1      0.43630
2      0.34139
3      0.34699
4      0.45811
        …
153    0.22628
154    0.18260
155    0.47179
156    0.19727
157    0.16681
Name: Generosity, Length: 158, dtype: float64 0      2.51738
1      2.70201
2      2.49204
3      2.46531
4      2.45176
        …
153    0.67042
154    1.63328
155    0.32858
156    1.83302
157    1.56726
Name: Dystopia_Residual, Length: 158, dtype: float64
```

[504]:

```
#Economy (GDP per Capita), Family, Health (Life
#Expectancy), Freedom, Trust (Government Corruption), Generosity, Dystopia␣
 ↪Residual.

d.plot(x="Happiness_Score", y=["Family", "Health", "Freedom", "Trust",␣
 ↪"Economy", "Generosity", "Dystopia_Residual"])
plt.show()
```



```
[505]: d.head()
```

```
[505]:        Country          Region  Happiness_Rank  Happiness_Score  \
       0  Switzerland  Western Europe               1            7.587
       1      Iceland  Western Europe               2            7.561
       2      Denmark  Western Europe               3            7.527
       3       Norway  Western Europe               4            7.522
       4       Canada   North America               5            7.427

          Standard_Error  Economy   Family   Health  Freedom    Trust  Generosity  \
       0         0.03411  1.39651  1.34951  0.94143  0.66557  0.41978     0.29678
       1         0.04884  1.30232  1.40223  0.94784  0.62877  0.14145     0.43630
       2         0.03328  1.32548  1.36058  0.87464  0.64938  0.48357     0.34139
```

```
3          0.03880  1.45900  1.33095  0.88521  0.66973  0.36503      0.34699
4          0.03553  1.32629  1.32261  0.90563  0.63297  0.32957      0.45811

   Dystopia_Residual
0            2.51738
1            2.70201
2            2.49204
3            2.46531
4            2.45176
```

[506]:

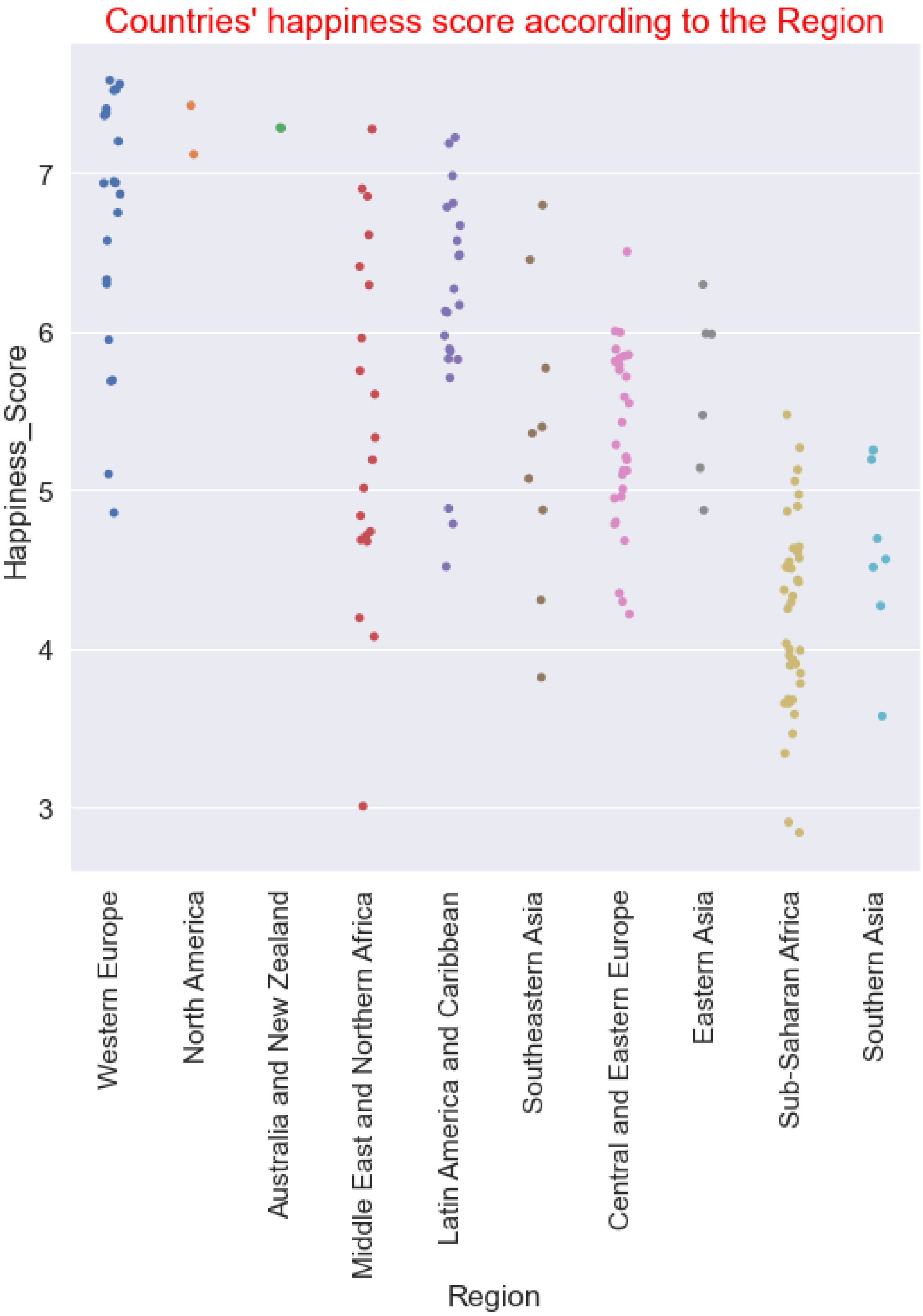
```
d.describe()
```

[506]:

```
       Happiness_Rank  Happiness_Score  Standard_Error     Economy  \
count      158.000000       158.000000      158.000000  158.000000
mean        79.493671         5.375734        0.047885    0.846137
std         45.754363         1.145010        0.017146    0.403121
min          1.000000         2.839000        0.018480    0.000000
25%         40.250000         4.526000        0.037268    0.545808
50%         79.500000         5.232500        0.043940    0.910245
75%        118.750000         6.243750        0.052300    1.158448
max        158.000000         7.587000        0.136930    1.690420

           Family      Health     Freedom       Trust  Generosity  \
count  158.000000  158.000000  158.000000  158.000000  158.000000
mean     0.991046    0.630259    0.428615    0.143422    0.237296
std      0.272369    0.247078    0.150693    0.120034    0.126685
min      0.000000    0.000000    0.000000    0.000000    0.000000
25%      0.856823    0.439185    0.328330    0.061675    0.150553
50%      1.029510    0.696705    0.435515    0.107220    0.216130
75%      1.214405    0.811013    0.549092    0.180255    0.309883
max      1.402230    1.025250    0.669730    0.551910    0.795880

       Dystopia_Residual
count         158.000000
mean            2.098977
std             0.553550
min             0.328580
25%             1.759410
50%             2.095415
75%             2.462415
max             3.602140
```

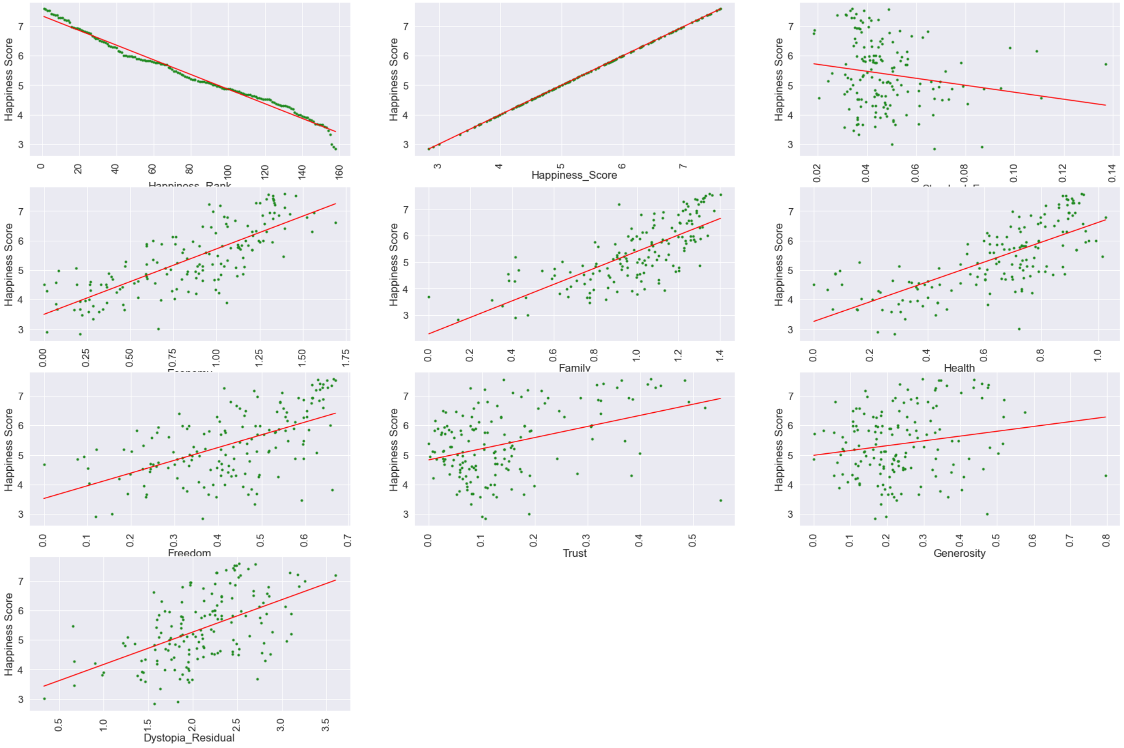
[507]:

```
x = sns.stripplot(x = "Region", y = "Happiness_Score", data = d, jitter = True)
plt.xticks(rotation = 90)
plt.title("Countries' happiness score according to the Region", color = 'red',␣
 ↪fontsize = 19)
plt.show()
```


Countries' happiness score according to the Region
Happiness_Score
3
4
5
6
7
Western Europe
North America
Australia and New Zealand
Middle East and Northern Africa
Latin America and Caribbean
Southeastern Asia
Central and Eastern Europe
Eastern Asia
Sub-Saharan Africa
Southern Asia
Region

[508]:

```
plt.figure(figsize=(30,20))
for i, col in enumerate(d.columns[2:13]):
    plt.subplot(4, 3, i+1)
    plt.xticks(rotation = 90)
    x = d[col]
    y =  d['Happiness_Score']
    plt.plot(x, y, '.', color="forestgreen")
    # create linear regression line:
    plt.plot(np.unique(x), np.poly1d(np.polyfit(x, y, 1))(np.
 ↪unique(x)),color="red")
    #truth value of a Series is ambiguous. Use a.empty, a.bool(), a.item(), a.
 ↪any() or a.all().

    plt.xlabel(col)
    plt.ylabel('Happiness Score')
```



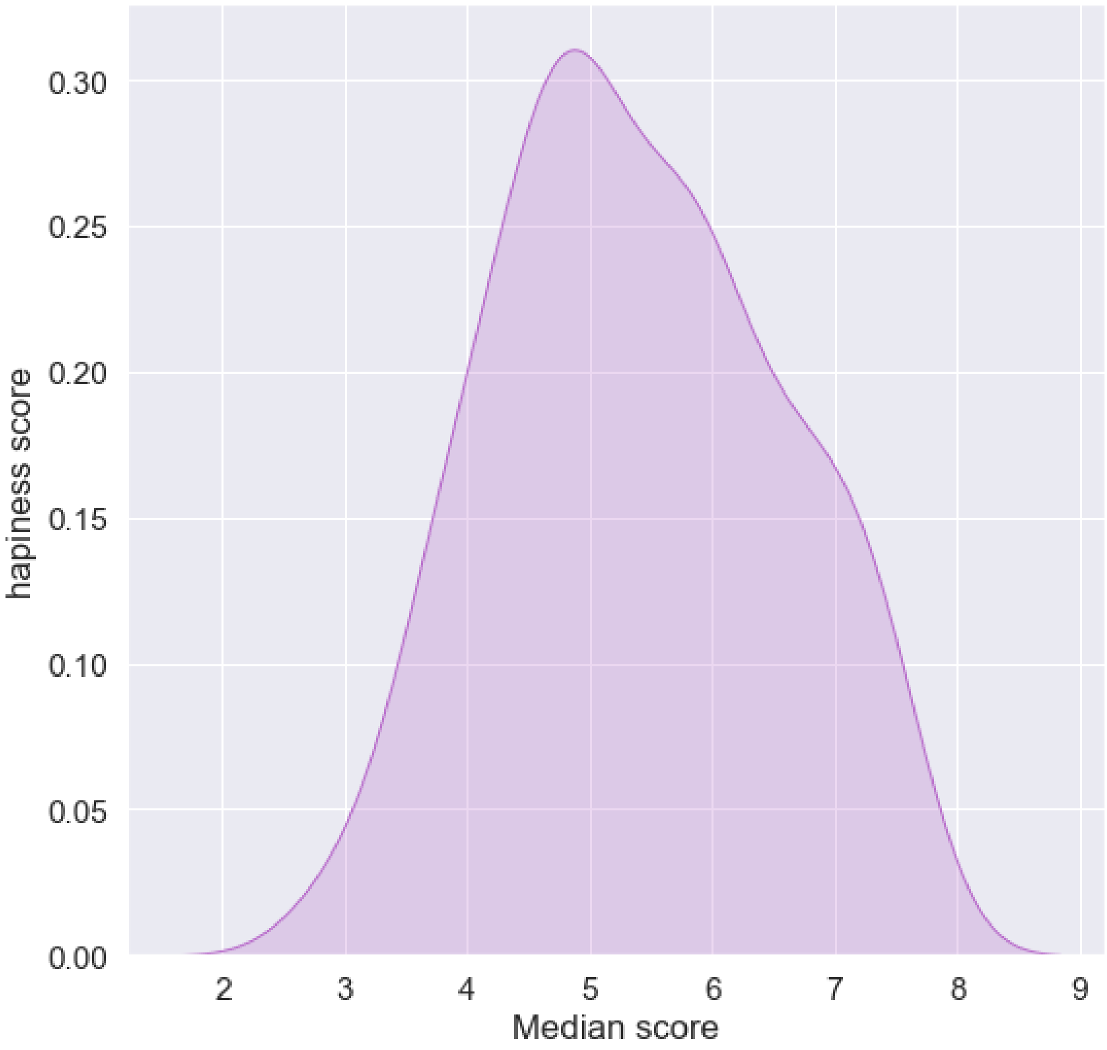
[509]:

```
sns.kdeplot(d['Happiness_Score'], color='#b667c9', shade=True, Label='Happiness␣
 ↪score')
plt.xlabel('Median score')
plt.ylabel('hapiness score')
```

```
/home/hiraditya/Desktop/HomeWork/SJSU/cs156/anaconda/lib/python3.8/site-
packages/seaborn/distributions.py:948: MatplotlibDeprecationWarning: Case-
```

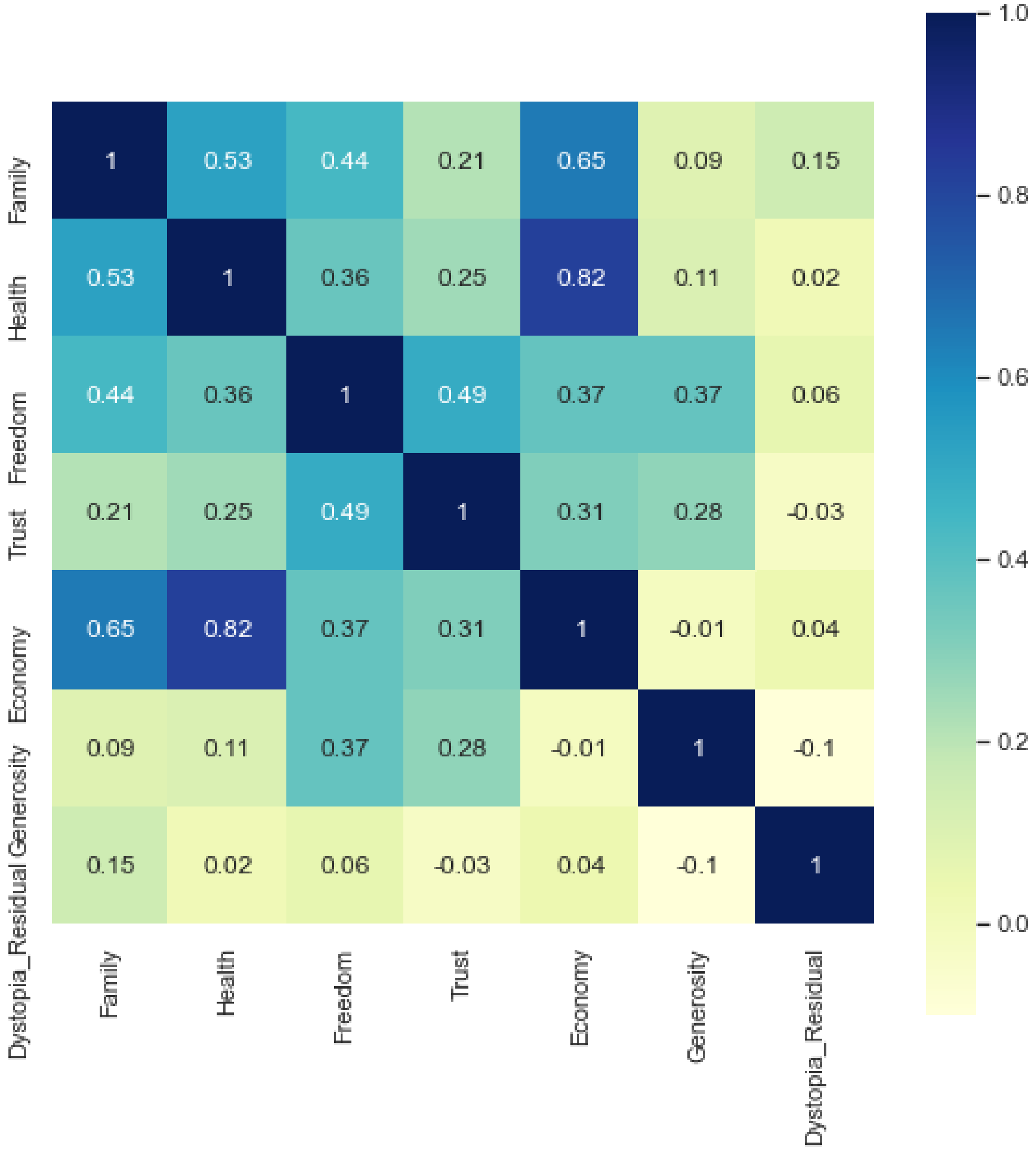
```
insensitive properties were deprecated in 3.3 and support will be removed two
minor releases later
  scout = self.ax.fill_between([], [], **plot_kws)
/home/hiraditya/Desktop/HomeWork/SJSU/cs156/anaconda/lib/python3.8/site-
packages/seaborn/distributions.py:991: MatplotlibDeprecationWarning: Case-
insensitive properties were deprecated in 3.3 and support will be removed two
minor releases later
  artist = ax.fill_between(
```

[509]:
```
Text(0, 0.5, 'hapiness score')
```

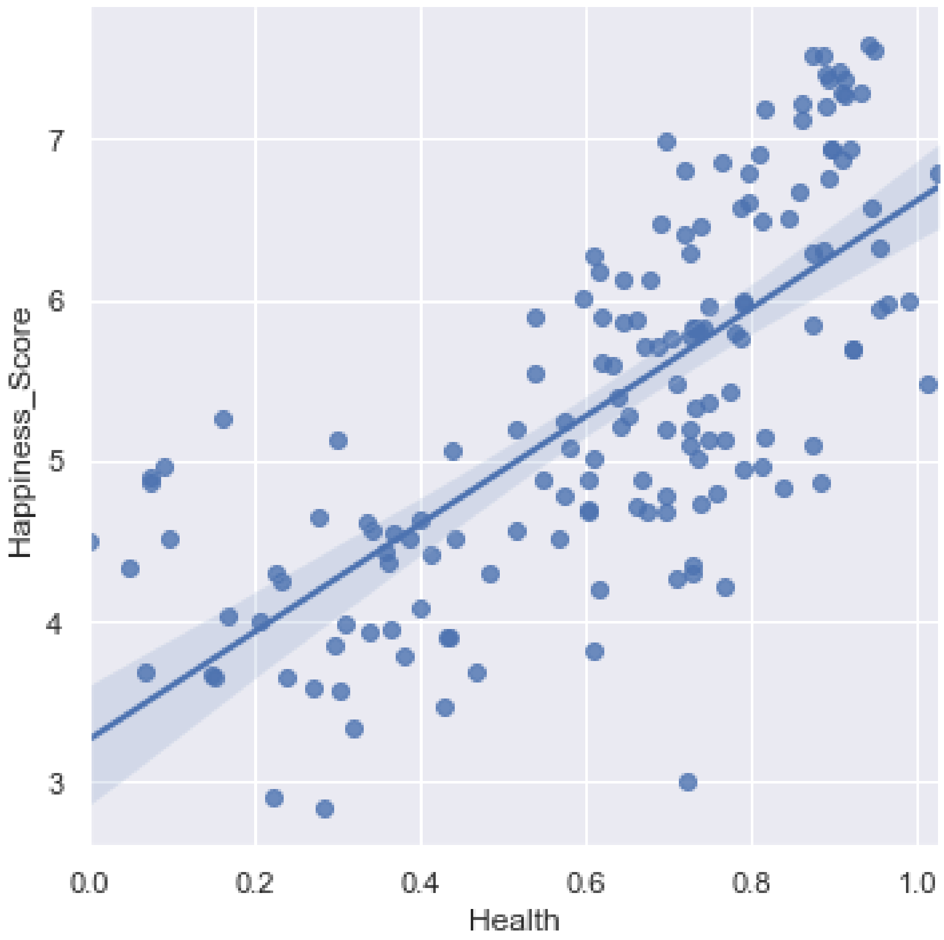


[510]:
```
features = d[["Family", "Health", "Freedom", "Trust", "Economy", "Generosity",␣
 ↪"Dystopia_Residual"]]
sns.set(rc={'figure.figsize': (8.5,8.5)})
sns.heatmap(features.corr().round(2), square=True, cmap='YlGnBu', annot=True)
```

```
[510]: <AxesSubplot:>
```

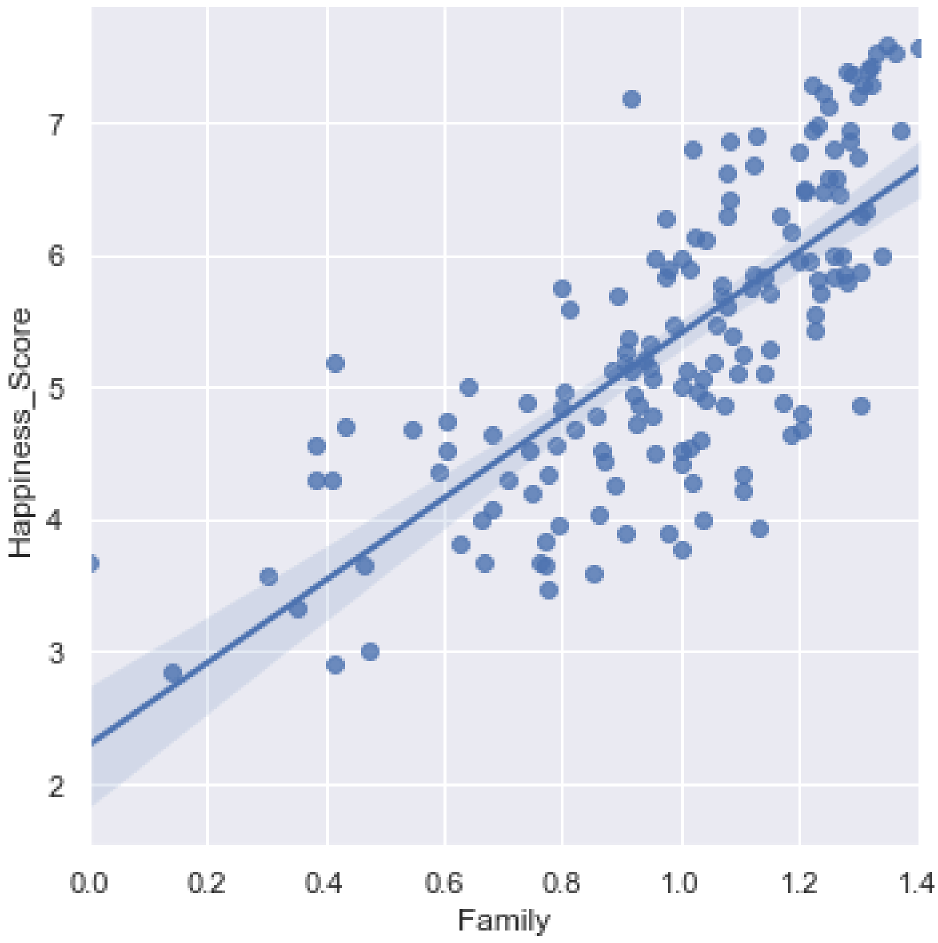


```
[511]: # Health vs. Hapiness score
       sns.lmplot(x = 'Health', y = 'Happiness_Score', data = d)

[511]: <seaborn.axisgrid.FacetGrid at 0x7f5d41dfb910>
```



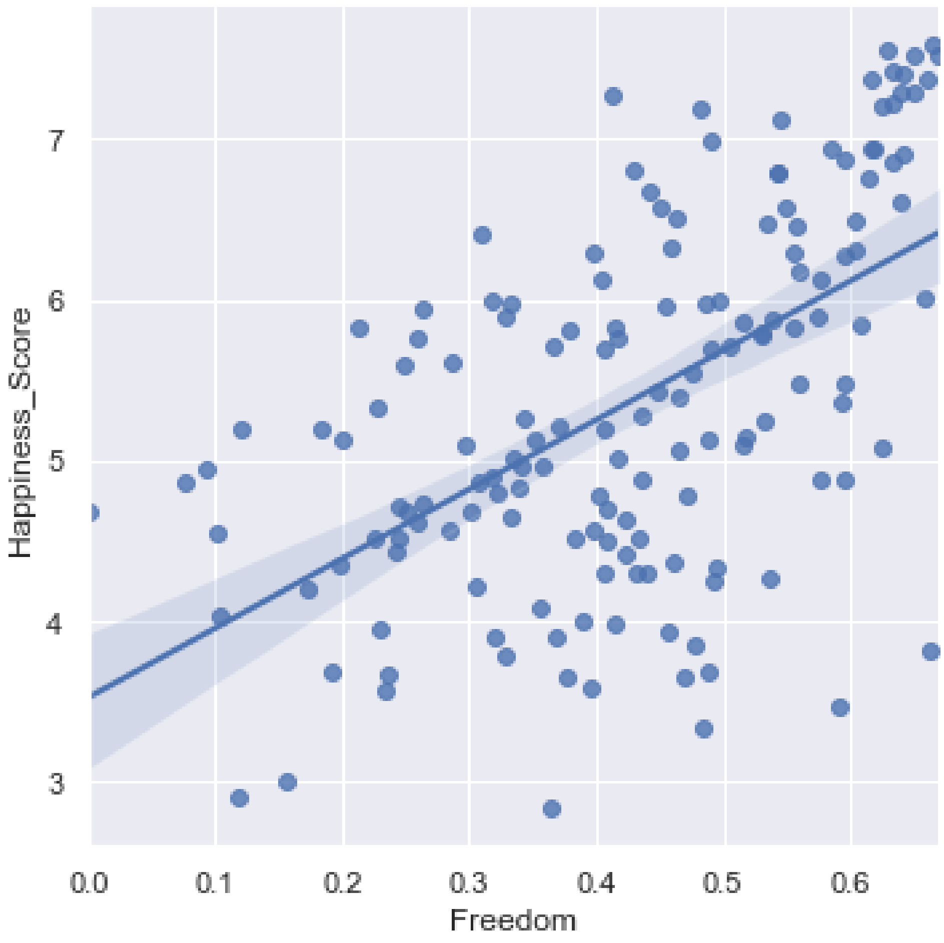
```
[512]: # Family vs. Hapiness score
       sns.lmplot(x = 'Family', y = 'Happiness_Score', data = d)
```

```
[512]: <seaborn.axisgrid.FacetGrid at 0x7f5d41dd9e80>
```



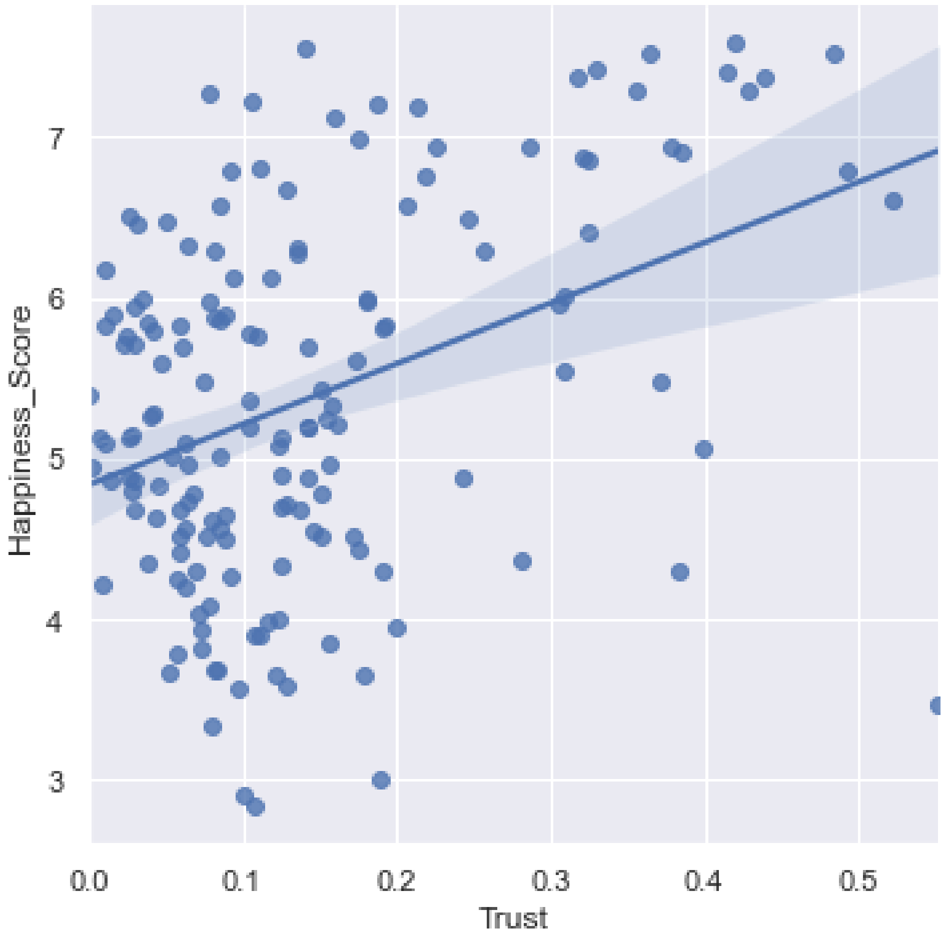
```
[513]: # Freedom vs. Hapiness score
       sns.lmplot(x = 'Freedom', y = 'Happiness_Score', data = d)
```

```
[513]: <seaborn.axisgrid.FacetGrid at 0x7f5d41c473d0>
```



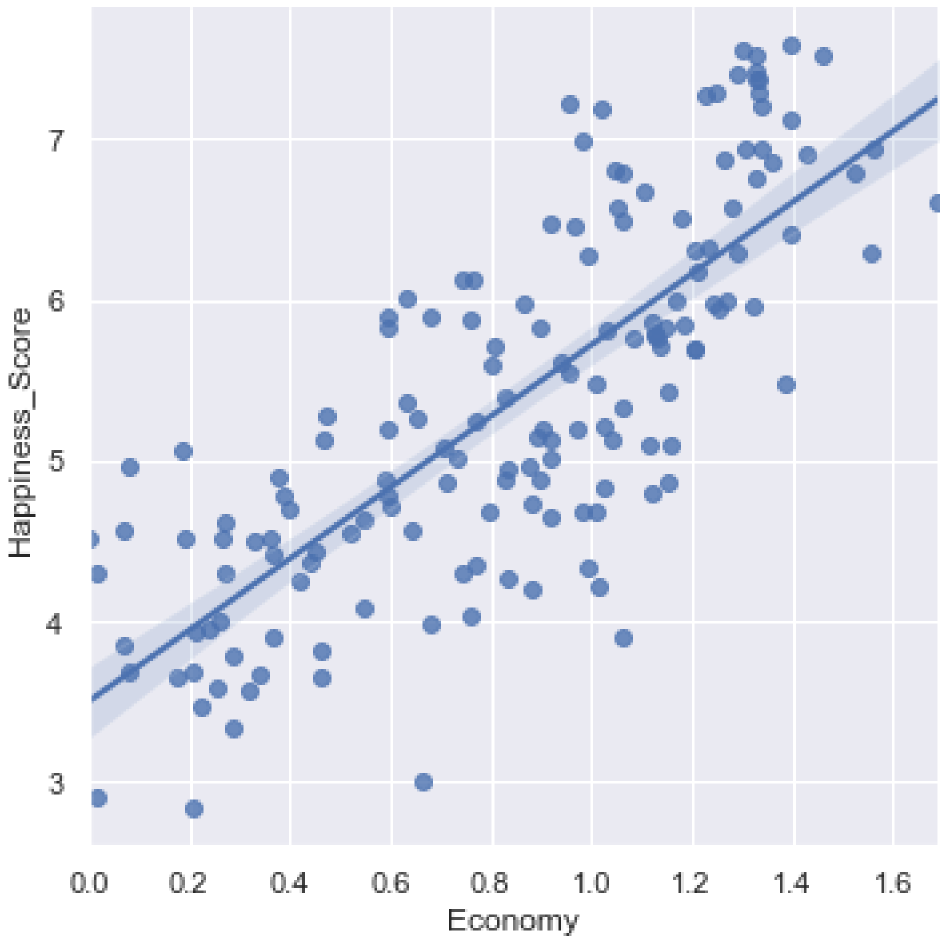
```
[514]: # Trust vs. Hapiness score
       sns.lmplot(x = 'Trust', y = 'Happiness_Score', data = d)
```

```
[514]: <seaborn.axisgrid.FacetGrid at 0x7f5d41ca7a00>
```



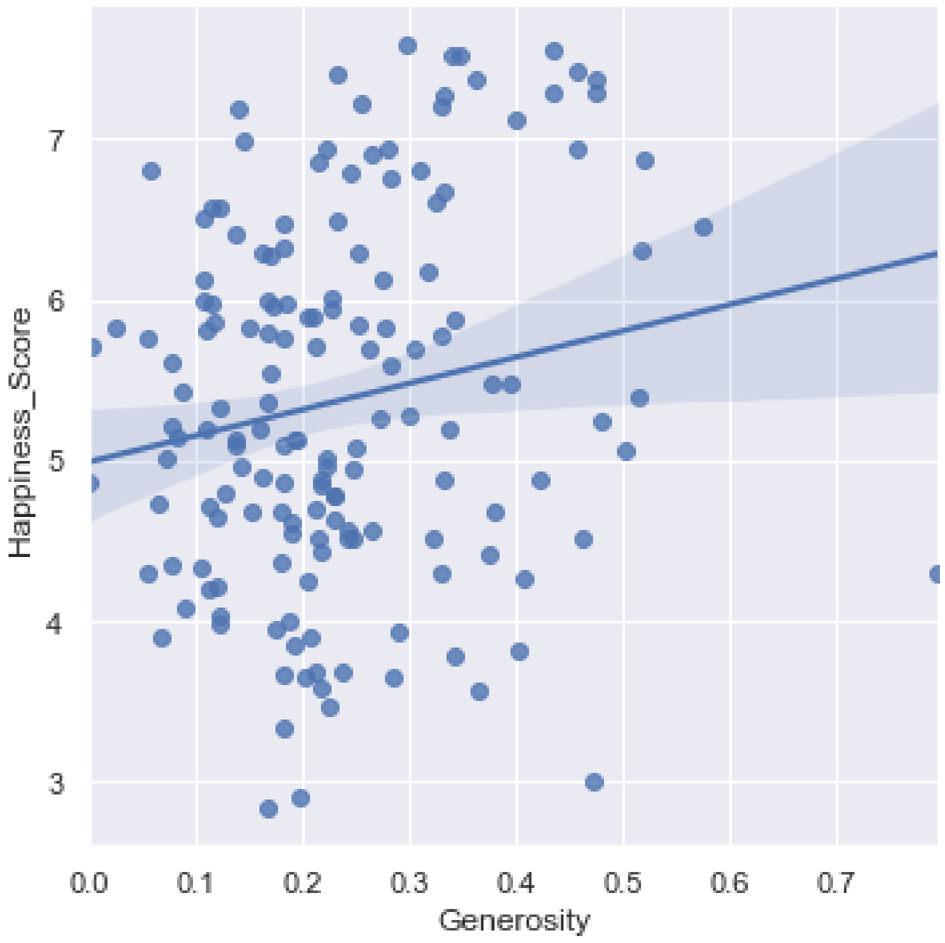
```
[515]: # Economy vs. Hapiness score
       sns.lmplot(x = 'Economy', y = 'Happiness_Score', data = d)
```

```
[515]: <seaborn.axisgrid.FacetGrid at 0x7f5d41bbce20>
```



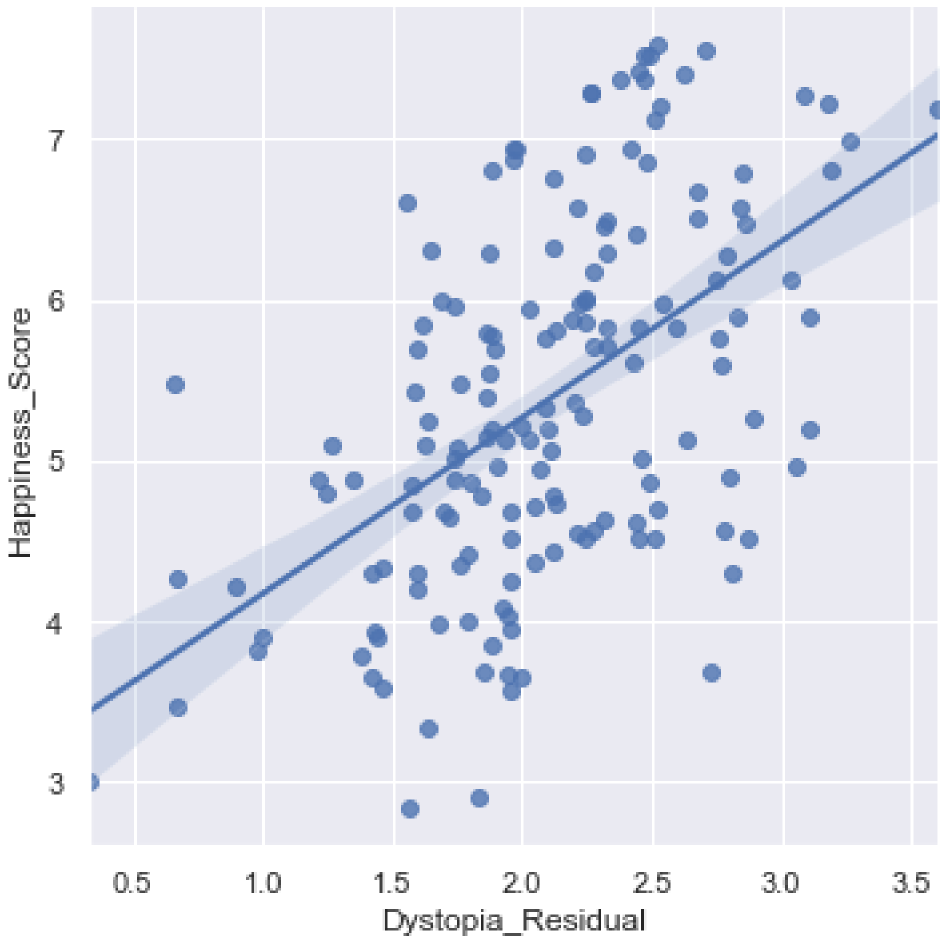
```
[516]: # Generosity vs. Hapiness score
       sns.lmplot(x = 'Generosity', y = 'Happiness_Score', data = d)
```

```
[516]: <seaborn.axisgrid.FacetGrid at 0x7f5d41b7bbe0>
```



```
[517]: #Dystopia_Residual vs. Hapiness score
       sns.lmplot(x = 'Dystopia_Residual', y = 'Happiness_Score', data = d)
```

```
[517]: <seaborn.axisgrid.FacetGrid at 0x7f5d41af5eb0>
```



[ ]:

[ ]:

[518]:
```
#Finding correlation
cor = d.corr()
cor
```

[518]:
```
                 Happiness_Rank  Happiness_Score  Standard_Error   Economy  \
Happiness_Rank         1.000000        -0.992105        0.158516 -0.785267
Happiness_Score       -0.992105         1.000000       -0.177254  0.780966
Standard_Error         0.158516        -0.177254        1.000000 -0.217651
Economy               -0.785267         0.780966       -0.217651  1.000000
Family                -0.733644         0.740605       -0.120728  0.645299
Health                -0.735613         0.724200       -0.310287  0.816478
Freedom               -0.556886         0.568211       -0.129773  0.370300
Trust                 -0.372315         0.395199       -0.178325  0.307885
```

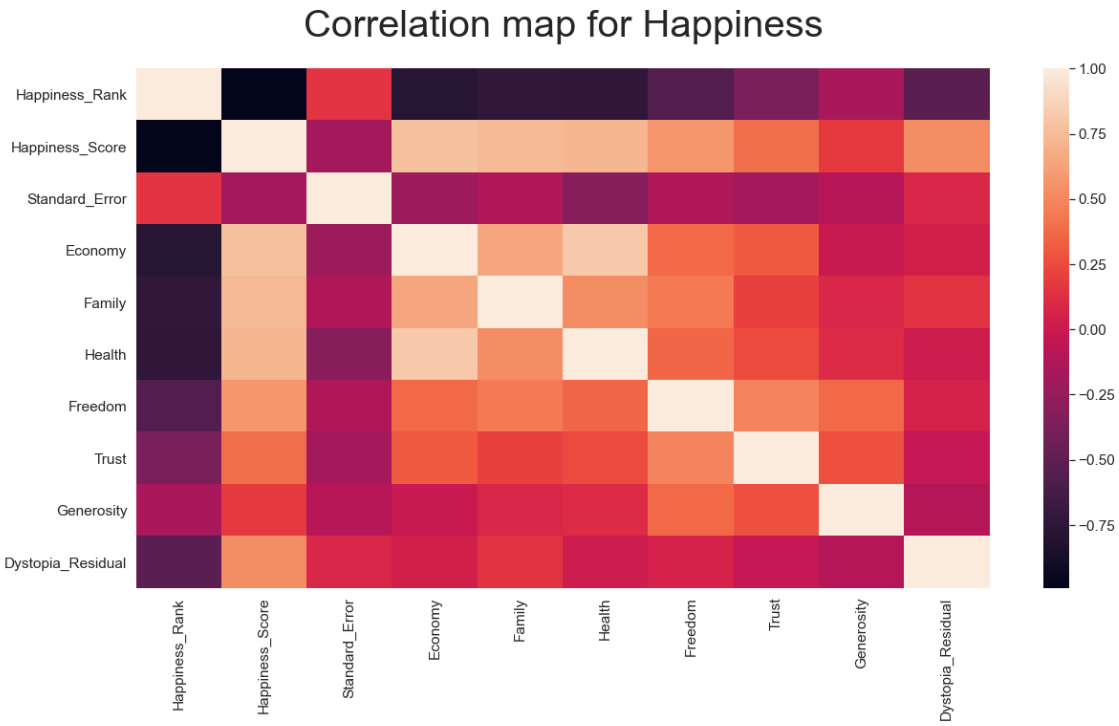
```
Generosity              -0.160142         0.180319       -0.088439 -0.010465
Dystopia_Residual       -0.521999         0.530474        0.083981  0.040059

                     Family    Health   Freedom     Trust  Generosity  \
Happiness_Rank    -0.733644 -0.735613 -0.556886 -0.372315   -0.160142
Happiness_Score    0.740605  0.724200  0.568211  0.395199    0.180319
Standard_Error    -0.120728 -0.310287 -0.129773 -0.178325   -0.088439
Economy            0.645299  0.816478  0.370300  0.307885   -0.010465
Family             1.000000  0.531104  0.441518  0.205605    0.087513
Health             0.531104  1.000000  0.360477  0.248335    0.108335
Freedom            0.441518  0.360477  1.000000  0.493524    0.373916
Trust              0.205605  0.248335  0.493524  1.000000    0.276123
Generosity         0.087513  0.108335  0.373916  0.276123    1.000000
Dystopia_Residual  0.148117  0.018979  0.062783 -0.033105   -0.101301

                   Dystopia_Residual
Happiness_Rank             -0.521999
Happiness_Score             0.530474
Standard_Error              0.083981
Economy                     0.040059
Family                      0.148117
Health                      0.018979
Freedom                     0.062783
Trust                      -0.033105
Generosity                 -0.101301
Dystopia_Residual           1.000000
```

[519]:

```
plt.subplots(figsize=(20,10))
sns.set(font_scale=1.4)
ax = plt.axes()
sns.heatmap(cor)
ax.set_title('Correlation map for Happiness', fontsize=40, y=1.05)
```

[519]:

```
Text(0.5, 1.05, 'Correlation map for Happiness')
```


Correlation map for Happiness

```
[520]: #Displaying positive correlations
       pos = cor[cor > 0.75]
       pos
```

```
[520]:                    Happiness_Rank  Happiness_Score  Standard_Error   Economy  \
       Happiness_Rank                1.0              NaN             NaN       NaN
       Happiness_Score               NaN         1.000000             NaN  0.780966
       Standard_Error                NaN              NaN             1.0       NaN
       Economy                       NaN         0.780966             NaN  1.000000
       Family                        NaN              NaN             NaN       NaN
       Health                        NaN              NaN             NaN  0.816478
       Freedom                       NaN              NaN             NaN       NaN
       Trust                         NaN              NaN             NaN       NaN
       Generosity                    NaN              NaN             NaN       NaN
       Dystopia_Residual             NaN              NaN             NaN       NaN

                          Family    Health  Freedom  Trust  Generosity  \
       Happiness_Rank        NaN       NaN      NaN    NaN         NaN
       Happiness_Score       NaN       NaN      NaN    NaN         NaN
       Standard_Error        NaN       NaN      NaN    NaN         NaN
       Economy               NaN  0.816478      NaN    NaN         NaN
       Family                1.0       NaN      NaN    NaN         NaN
       Health                NaN  1.000000      NaN    NaN         NaN
       Freedom               NaN       NaN      1.0    NaN         NaN
```

```
Trust                   NaN        NaN       NaN     1.0          NaN
Generosity              NaN        NaN       NaN     NaN          1.0
Dystopia_Residual       NaN        NaN       NaN     NaN          NaN

                   Dystopia_Residual
Happiness_Rank                   NaN
Happiness_Score                  NaN
Standard_Error                   NaN
Economy                          NaN
Family                           NaN
Health                           NaN
Freedom                          NaN
Trust                            NaN
Generosity                       NaN
Dystopia_Residual                1.0
```

[ ]:

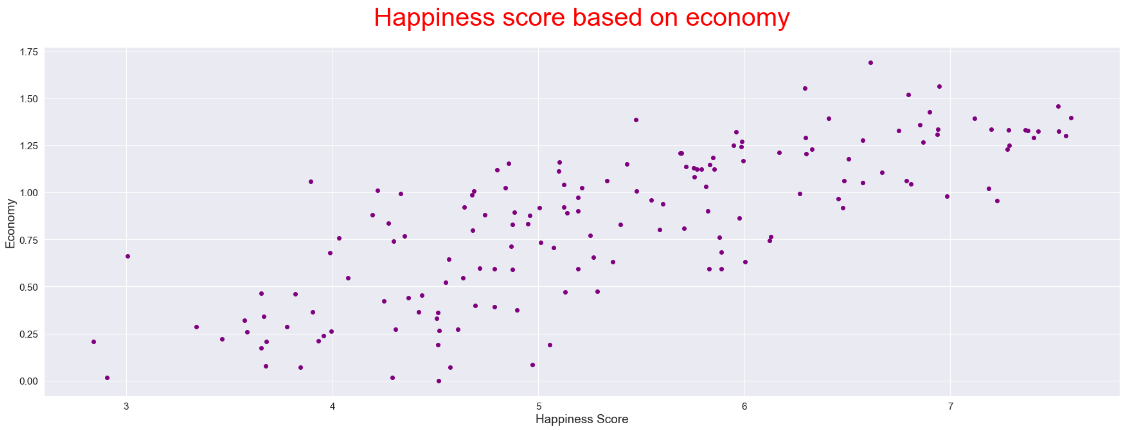
```

```

[521]:

```
plt.figure( figsize=(30,10))
plt.scatter(d['Happiness_Score'], d['Economy'], color='purple')
plt.title('Happiness score based on economy', color = "red", fontsize=40, y=1.
 ↪05)
plt.xlabel('Happiness Score', fontsize=19)
plt.ylabel('Economy', fontsize=19)
```

[521]:

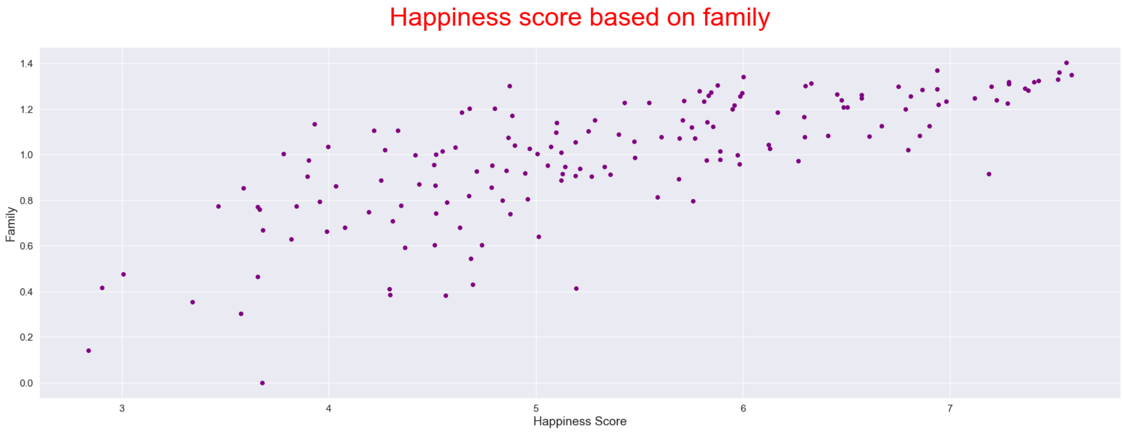
```
Text(0, 0.5, 'Economy')
```



[522]:

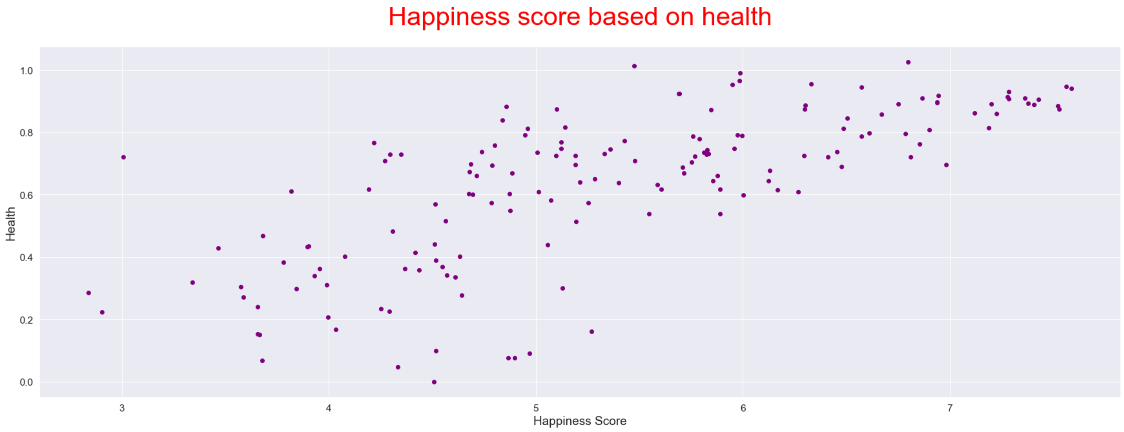
```
plt.figure( figsize=(30,10))
plt.scatter(d['Happiness_Score'], d['Family'], color='purple')
plt.title('Happiness score based on family', color = "red", fontsize=40, y=1.05)
plt.xlabel('Happiness Score', fontsize=19)
plt.ylabel('Family', fontsize=19)
```

[522]:
```
Text(0, 0.5, 'Family')
```



[523]:
```python
plt.figure( figsize=(30,10))
plt.scatter(d['Happiness_Score'], d['Health'], color='purple')
plt.title('Happiness score based on health', color = "red", fontsize=40, y=1.05)
plt.xlabel('Happiness Score', fontsize=19)
plt.ylabel('Health', fontsize=19)
```

[523]:
```
Text(0, 0.5, 'Health')
```



[524]:
```python
X = d.Health
Y = d.Happiness_Score
```

[525]:
```python
print("happiness  = ", Y)
```

```
happiness  =  0      7.587
1      7.561
2      7.527
```

```
3      7.522
4      7.427
       …
153    3.465
154    3.340
155    3.006
156    2.905
157    2.839
Name: Happiness_Score, Length: 158, dtype: float64
```

```
[526]: print("health  = ", X)
```

```
health  =  0      0.94143
1      0.94784
2      0.87464
3      0.88521
4      0.90563
        …
153    0.42864
154    0.31910
155    0.72193
156    0.22396
157    0.28443
Name: Health, Length: 158, dtype: float64
```

```
[527]: X.shape
```

```
[527]: (158,)
```

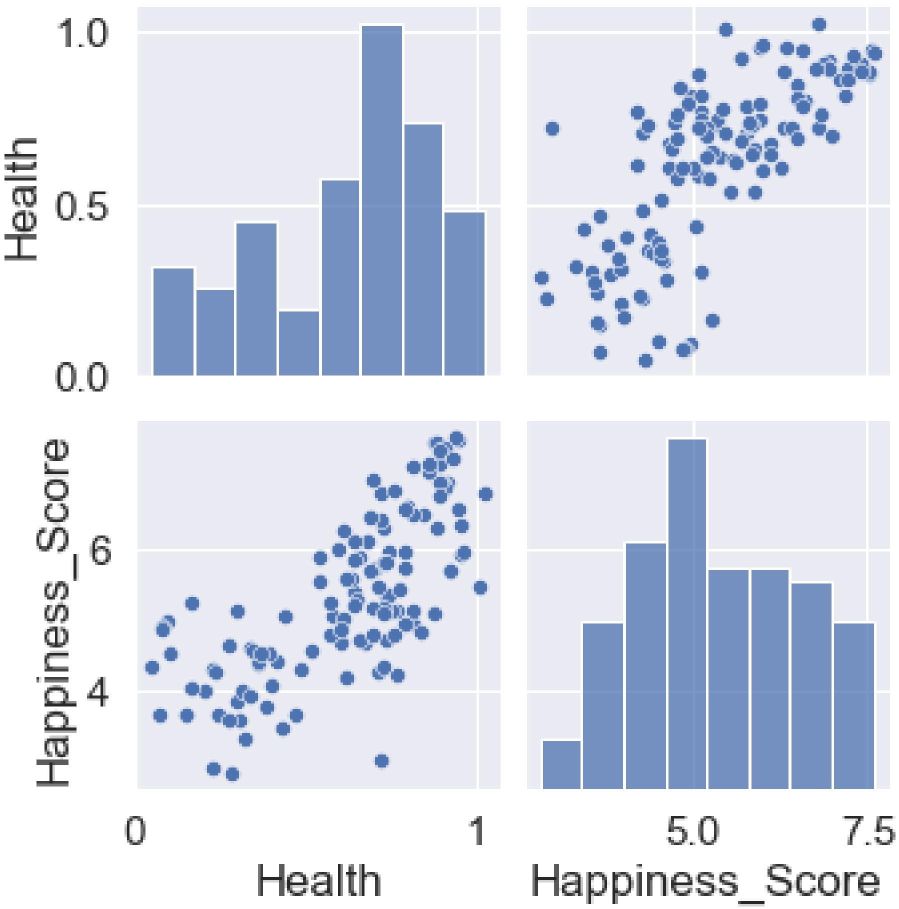
```
[528]: Y_test = np.array([d.Economy, d.Health, d.Trust, d.Freedom, d.Generosity, d.
        ↪Family, d.Dystopia_Residual]).T
```

```
[529]: X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2,␣
        ↪random_state=0)
       X_train.shape, Y_train.shape, X_test.shape, Y_test.shape
```

```
[529]: ((126,), (126,), (32,), (32,))
```

```
[530]: # check to see what our training data looks like
       df = pd.DataFrame(X_train)
       df['Happiness_Score'] = Y_train
       sns.pairplot(df)
```

```
[530]: <seaborn.axisgrid.PairGrid at 0x7f5d41c503a0>
```



[531]:
```
print("y train = ", Y_train)
```

```
y train =  16     6.946
130    4.292
134    4.194
22     6.810
93     4.971
       …
9      7.284
103    4.800
67     5.605
117    4.550
47     5.975
Name: Happiness_Score, Length: 126, dtype: float64
```

[532]:
```
print("x train = ", X_train)
```

```
x train =  16     0.91894
130    0.22562
```

```
134    0.61712
22     0.72052
93     0.09131
        …
9      0.93156
103    0.75905
67     0.61766
117    0.36878
47     0.79075
Name: Health, Length: 126, dtype: float64
```

[ ]:

```

```

[533]:

```
d.keys()
```

[533]:

```
Index(['Country', 'Region', 'Happiness_Rank', 'Happiness_Score',
       'Standard_Error', 'Economy', 'Family', 'Health', 'Freedom', 'Trust',
       'Generosity', 'Dystopia_Residual'],
      dtype='object')
```

[534]:

```
d.Happiness_Score.shape
```

[534]:

```
(158,)
```

[535]:

```
d.Health.shape
```

[535]:

```
(158,)
```

[536]:

```
lm = LinearRegression()
```

[537]:

```
#X = d.drop("Country", axis = 1)
Y_test = np.array([d.Economy, d.Health, d.Trust, d.Freedom, d.Generosity, d.
 ↪Family, d.Dystopia_Residual]).T
```

[538]:

```
model = lm.fit(Y_test, d.Happiness_Score)
```

[539]:

```
#LinearRegression(copy_X=True, fit_intercept=True, normalize=False)
```

[540]:

```
# The coefficients:
print('Coefficients: \n', model.coef_)
Y_test_pred = model.predict(Y_test)
```

```
Coefficients:
 [1.0001014  0.99988261 0.99991914 0.99969531 1.00006126 0.99997035
 1.00003038]
```

[541]:

```
# The mean squared error:
print('Mean squared error: %.2f' % mean_squared_error(d.Economy, Y_test_pred))
print('Mean squared error: %.2f' % mean_squared_error(d.Health, Y_test_pred))
print('Mean squared error: %.2f' % mean_squared_error(d.Family, Y_test_pred))
print('Mean squared error: %.2f' % mean_squared_error(d.Generosity, Y_test_pred))
print('Mean squared error: %.2f' % mean_squared_error(d.Trust, Y_test_pred))
print('Mean squared error: %.2f' % mean_squared_error(d.Freedom, Y_test_pred))
print('Mean squared error: %.2f' % mean_squared_error(d.Dystopia_Residual, Y_test_pred))

# The coefficient of determination (1 is perfect prediction):
print('Coefficient of determination: %.2f' % r2_score(d.Economy, Y_test_pred))
print('Coefficient of determination: %.2f' % r2_score(d.Health, Y_test_pred))
print('Coefficient of determination: %.2f' % r2_score(d.Family, Y_test_pred))
print('Coefficient of determination: %.2f' % r2_score(d.Generosity, Y_test_pred))
print('Coefficient of determination: %.2f' % r2_score(d.Trust, Y_test_pred))
print('Coefficient of determination: %.2f' % r2_score(d.Freedom, Y_test_pred))
print('Coefficient of determination: %.2f' % r2_score(d.Dystopia_Residual, Y_test_pred))
```

```
Mean squared error: 21.27
Mean squared error: 23.48
Mean squared error: 20.14
Mean squared error: 27.67
Mean squared error: 28.59
Mean squared error: 25.60
Mean squared error: 11.68
Coefficient of determination: -130.69
Coefficient of determination: -386.00
Coefficient of determination: -272.25
Coefficient of determination: -1734.09
Coefficient of determination: -1995.66
Coefficient of determination: -1133.72
Coefficient of determination: -37.35
```

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

```
[ ]:
```