Using SVM to build an indicator of Economic Development

We start by importing libraries and reading the data:

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
data_hdi = pd.read_excel("HDI_data.xlsx")    #This file contains HDI data
```

The data present here contains HDI and its components: an education index, a health index, and an income index.

- *Education index* is an average of mean years of schooling (of adults) and expected years of schooling (of children). We here use the components of this index in a disaggregated way.
- The *life expectancy index* is given by life expectancy at birth expressed as an index using a minimum value of 20 years and a maximum value of 85 years.
- And the *income index* is given by GNI per capita (2017 PPP International USD, using natural logarithm) expressed as an index using a minimum value of USD 100 and a maximum value USD 75,000.

HDI data is available for the time period 1990-2019. However, we intend to expand the data to cover the 1900-2019 period instead. Therefore:

- For the *education index*:
  - Average of mean years of schooling: we used HDI data from 1990 to 2019; additionally, we used data from the V-dem database to obtain the values for the period 1900-1989 (in the v-dem database, the variable in question is "e_peaveduc")
  - Expected years of schooling: we used the HDI database for the period 1990-2019; we also used data from UNESCO Institute for Statistics for the period 1970-1989; the values for the period 1900-1969 were extrapolated (since it is a long period of time, the function to extrapolate the data was analyzed country by country, and we chose the one that allowed us to obtain a higher r-squared without reaching negative values and without showing a significant decrease in the values of expected education)
- For the life *expectancy index*: we use the HDI values for the period 1990-2019; we used data from the V-dem database for the period 1900-1989 (the variable in question is "e_pelifeex"; it was necessary to convert the values of this variable into an index, with a maximum of 80 years and a minimum of 20 years, so that the data from both sources would be compatible)
- For the *income index*: given the lack of data regarding GNI, we chose to use GDP instead. For this purpose, we used the Maddison Project Database, which gives us the real GDP per capita PPP 2011 USD. Similar to the HDI data, we decided to transform this data, using an index with a minimum of USD 100 and a maximum of USD 75,000 and a natural logarithm. This database only has values until 2018, so we extrapolated the data for 2019, using a linear trend according to to the values of the last 5 years of each country.

The compilation of this data is in the "databases" file.

Note: We will work with this file (databases) to analyze the SVM application with different kernels, but we will keep the file with the HDI data so that, in the end, we can compare the results (i.e. to understand if replacing the income index - GDP vs GNI - generates different results).

In [3]:
```
databases = pd.read_excel("databases.xlsx")  #This file contains data with different databases
```

In [4]:
```
# Correcting errors from the file:
databases['Av_education'] = pd.to_numeric(databases['Av_education'], errors='coerce')
databases['School_expectancy'] = pd.to_numeric(databases['School_expectancy'], errors='coerce')
databases['Life_expectancy_index'] = pd.to_numeric(databases['Life_expectancy_index'], errors='coerce')
databases['GDPpcLog'] = pd.to_numeric(databases['GDPpcLog'], errors='coerce')
databases['HDI_score'] = pd.to_numeric(databases['HDI_score'], errors='coerce')
```

In [5]:
```
databases
```

Out[5]:

	country	year	Av_education	School_expectancy	Life_expectancy_index	GDPpcLog	HDI_score
0	Afghanistan	1900	0.025903	0.173368	0.153333	NaN	NaN
1	Afghanistan	1901	0.027107	0.179565	0.155000	NaN	NaN
2	Afghanistan	1902	0.028367	0.185984	0.155000	NaN	NaN
3	Afghanistan	1903	0.029685	0.192631	0.156667	NaN	NaN
4	Afghanistan	1904	0.031063	0.199517	0.156667	NaN	NaN
...	...	...	...	...	...	...	...
20230	Zimbabwe	2016	8.300000	10.400000	0.620000	0.649484	0.558
20231	Zimbabwe	2017	8.300000	10.500000	0.628000	0.682656	0.563
20232	Zimbabwe	2018	8.400000	10.500000	0.634000	0.702056	0.569
20233	Zimbabwe	2019	8.500000	11.000000	0.638000	0.689935	0.571
20234	Zimbabwe	2020	NaN	NaN	NaN	NaN	NaN

20235 rows × 7 columns

Building the priming dataset

For the priming dataset, we assume that observations with an HDI above or equal to 0.8 correspond to countries with high development, and observations with an HDI below 0.55 correspond to less developed countries. We give a label = 1 to developed countries and 0 to less developed countries:

In [6]:
```
is_developed = databases.HDI_score>=0.8
developedcountries=databases[is_developed]
developed=developedcountries[["country","year", "Av_education","School_expectancy",
                              "Life_expectancy_index","GDPpcLog"]]
developed['Label']=1
developed
```

Out[6]:

	country	year	Av_education	School_expectancy	Life_expectancy_index	GDPpcLog	Label
373	Andorra	2000	9.9	10.8	0.906	NaN	1
374	Andorra	2001	9.9	10.8	0.911	NaN	1
375	Andorra	2002	9.9	11.0	0.915	NaN	1
376	Andorra	2003	10.1	10.9	0.920	NaN	1
377	Andorra	2004	10.5	10.8	0.923	NaN	1
...	...	...	...	...	...	...	...
19560	Uruguay	2015	8.7	16.3	0.883	3.241006	1
19561	Uruguay	2016	8.7	16.6	0.885	3.252639	1
19562	Uruguay	2017	8.7	16.8	0.887	3.275614	1
19563	Uruguay	2018	8.8	16.8	0.889	3.289031	1
19564	Uruguay	2019	8.9	16.8	0.891	3.300816	1

1200 rows × 7 columns

In [7]:
```
is_not_developed = databases.HDI_score<0.55
leastdevelopedcountries=databases[is_not_developed]
leastdeveloped=leastdevelopedcountries[["country","year", "Av_education","School_expectancy",
                                        "Life_expectancy_index","GDPpcLog"]]
leastdeveloped['Label']=0
leastdeveloped
```

Out[7]:

	country	year	Av_education	School_expectancy	Life_expectancy_index	GDPpcLog	Label
90	Afghanistan	1990	1.5	2.6	0.467	0.141676	0
91	Afghanistan	1991	1.6	2.9	0.477	0.042054	0
92	Afghanistan	1992	1.6	3.2	0.487	-0.006866	0
93	Afghanistan	1993	1.7	3.6	0.496	-0.448286	0
94	Afghanistan	1994	1.8	3.9	0.505	-0.824432	0
...	...	...	...	...	...	...	...
20224	Zimbabwe	2010	7.3	10.1	0.471	0.552808	0
20225	Zimbabwe	2011	7.3	10.2	0.506	0.636146	0
20226	Zimbabwe	2012	7.9	10.3	0.539	0.697145	0
20227	Zimbabwe	2013	8.0	10.2	0.568	0.697145	0
20228	Zimbabwe	2014	8.2	10.3	0.591	0.690473	0

1371 rows × 7 columns

In [8]:
```
#joining developed and undeveloped countries in the same table
frames=[developed,leastdeveloped]
table=pd.concat(frames)
table.dropna(inplace=True)
```

In [9]:
```
table
```

Out[9]:

	country	year	Av_education	School_expectancy	Life_expectancy_index	GDPpcLog	Label
672	Argentina	2006	10.3	16.3	0.840	3.085717	1
673	Argentina	2007	10.3	16.3	0.843	3.167762	1
674	Argentina	2008	10.3	16.5	0.845	3.202470	1
675	Argentina	2009	10.3	16.8	0.848	3.135609	1
676	Argentina	2010	10.3	17.1	0.850	3.227119	1
...	...	...	...	...	...	...	...
20224	Zimbabwe	2010	7.3	10.1	0.471	0.552808	0

	country	year	Av_education	School_expectancy	Life_expectancy_index	GDPpcLog	Label
20225	Zimbabwe	2011	7.3	10.2	0.506	0.636146	0
20226	Zimbabwe	2012	7.9	10.3	0.539	0.697145	0
20227	Zimbabwe	2013	8.0	10.2	0.568	0.697145	0
20228	Zimbabwe	2014	8.2	10.3	0.591	0.690473	0

2394 rows × 7 columns

Applying the SVM

In [10]:
```
table1=table[["Av_education","School_expectancy",
              "Life_expectancy_index","GDPpcLog", "Label"]]

# Distinction between attributes and labels
X = table1.drop('Label', axis=1)
y = table1['Label']

# Dividing our data in training set and test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20)
```

Testing different kernels, for classification:

In [11]:
```
# Linear SVM
from sklearn.svm import SVC
svclassifier = SVC(kernel='linear')
svclassifier.fit(X_train, y_train)

y_pred = svclassifier.predict(X_test)

from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
[[259   0]
 [  0 220]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       259
           1       1.00      1.00      1.00       220

    accuracy                           1.00       479
   macro avg       1.00      1.00      1.00       479
weighted avg       1.00      1.00      1.00       479
```

In [12]:
```
# Polynomial
from sklearn.svm import SVC
svclassifier = SVC(kernel='poly', degree=2)
svclassifier.fit(X_train, y_train)

y_pred = svclassifier.predict(X_test)

from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[259   0]
 [  0 220]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       259
           1       1.00      1.00      1.00       220

    accuracy                           1.00       479
   macro avg       1.00      1.00      1.00       479
weighted avg       1.00      1.00      1.00       479
```

In [13]:
```
# Gaussian
from sklearn.svm import SVC
svclassifier = SVC(kernel='rbf')
svclassifier.fit(X_train, y_train)

y_pred = svclassifier.predict(X_test)

from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[259   0]
 [  0 220]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       259
           1       1.00      1.00      1.00       220

    accuracy                           1.00       479
```

```
   macro avg       1.00      1.00      1.00       479
weighted avg       1.00      1.00      1.00       479
```

In [14]:
```
# Sigmoid
from sklearn.svm import SVC
svclassifier = SVC(kernel='sigmoid')
svclassifier.fit(X_train, y_train)

y_pred = svclassifier.predict(X_test)

from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[ 27 232]
 [220   0]]
              precision    recall  f1-score   support

           0       0.11      0.10      0.11       259
           1       0.00      0.00      0.00       220

    accuracy                           0.06       479
   macro avg       0.05      0.05      0.05       479
weighted avg       0.06      0.06      0.06       479
```

For classification, the sigmoid kernel performs very poorly, but the linear, polynomial and gaussian kernels work very well, with 100% prediction accuracy

Applying different kernels in classification and comparing results with HDI

Linear kernel:

In [15]:
```
# Linear kernel:

from sklearn.svm import SVC
svclassifier = SVC(kernel='linear')
svclassifier.fit(X_train, y_train)

y_pred = svclassifier.predict(X_test)



Atrib=databases[["Av_education","School_expectancy","Life_expectancy_index","GDPpcLog"]]  #defines attributes
Atrib.dropna(inplace=True)                                    #drops NaNs, or else the SVM cannot be applied

Atrib_l=Atrib
Label_HDI_SVM_linear = svclassifier.predict(Atrib_l)                                  # Applies the classifier


#Creating a table with the attributes, the respective country-year information,
    #the classification from the SVC and the HDI score:
Atrib_l['Label_HDI_SVM_linear'] = Label_HDI_SVM_linear
Atrib_l['country']= databases.country
Atrib_l['year']= databases.year
Atrib_l['HDI_score']= databases.HDI_score
```

Polynominal kernel:

In [16]:
```
# Polynominal kernel:

from sklearn.svm import SVC
svclassifier = SVC(kernel='poly', degree=2)
svclassifier.fit(X_train, y_train)

y_pred = svclassifier.predict(X_test)

Atrib=databases[["Av_education","School_expectancy","Life_expectancy_index","GDPpcLog"]]  #defines attributes
Atrib.dropna(inplace=True)                                    #drops NaNs, or else the SVM cannot be applied

Atrib_poly=Atrib                                                                                  #attributes

Label_HDI_SVM_poly = svclassifier.predict(Atrib_poly)                                 # Applies the classifier


#Creating a table with the attributes, the respective country-year information,
    #the classification from the SVC and the HDI score:
Atrib_poly['Label_HDI_SVM_poly'] = Label_HDI_SVM_poly
Atrib_poly['country']= databases.country
Atrib_poly['year']= databases.year
Atrib_poly['HDI_score']= databases.HDI_score
```

Gaussian kernel:

In [17]:
```
# Gaussian kernel:

from sklearn.svm import SVC
svclassifier = SVC(kernel='rbf')
svclassifier.fit(X_train, y_train)
```

```
y_pred = svclassifier.predict(X_test)


Atrib=databases[["Av_education","School_expectancy","Life_expectancy_index","GDPpcLog"]]  #defines attributes
Atrib.dropna(inplace=True)                                    #drops NaNs, or else the SVM cannot be applied
Atrib_rbf=Atrib

Label_HDI_SVM_rbf = svclassifier.predict(Atrib_rbf)                                    # Applies the classifier


#Creating a table with the attributes, the respective country-year information,
    #the classification from the SVC and the HDI score:
Atrib_rbf['Label_HDI_SVM_rbf'] = Label_HDI_SVM_rbf
Atrib_rbf['country']= databases.country
Atrib_rbf['year']= databases.year
Atrib_rbf['HDI_score']= databases.HDI_score
```

Note: The sigmoid kernel will not be applied, given the weak predictive capacity of this classifier.

Analyzing the correlation coefficients:

In [18]:
```
c=Atrib[["HDI_score"]]
c['Linear classification']=Atrib_l.Label_HDI_SVM_linear
c['Gaussian classification']=Atrib_rbf.Label_HDI_SVM_rbf
c['Polynomial classification']=Atrib_poly.Label_HDI_SVM_poly


Correlation=c.corr()
print('Correlation between our binary classifiers and the original HDI scores:')
Correlation
```

```
Correlation between our binary classifiers and the original HDI scores:
```

Out[18]:

	HDI_score	Linear classification	Gaussian classification	Polynomial classification
HDI_score	1.000000	0.820553	0.822313	0.815555
Linear classification	0.820553	1.000000	0.985479	0.976869
Gaussian classification	0.822313	0.985479	1.000000	0.968210
Polynomial classification	0.815555	0.976869	0.968210	1.000000

Our classifiers have a strong positive correlation (above 80%) with the HDI scores, with the Gaussian classifier having the highest correlation coefficient.

Testing different kernels in regressions:

In [19]:
```
#Linear Regression
from sklearn.svm import SVR
svregression_l = SVR(kernel='linear')
svregression_l.fit(X_train, y_train)

y_pred_l = svregression_l.predict(X_test)

accuracy_linear = svregression_l.score(X_test, y_test)
print("Accuracy of predictions with the linear regression: ", accuracy_linear)
```

```
Accuracy of predictions with the linear regression:  0.9239053012369964
```

In [21]:
```
#Polynomial Regression
from sklearn.svm import SVR
svregression_p = SVR(kernel='poly', degree=2)
svregression_p.fit(X_train, y_train)

y_pred_p = svregression_p.predict(X_test)

accuracy_poly = svregression_p.score(X_test, y_test)
print("Accuracy of predictions with the polynomial regression (degree=2): ", accuracy_poly)
```

```
Accuracy of predictions with the polynomial regression (degree=2):  0.9371871645219617
```

In [22]:
```
#Gaussian Regression
from sklearn.svm import SVR
svregression_g = SVR(kernel='rbf')
svregression_g.fit(X_train, y_train)

y_pred_g = svregression_g.predict(X_test)

accuracy_gauss = svregression_g.score(X_test, y_test)
print("Accuracy of predictions with the gaussian regression: ", accuracy_gauss)
```

```
Accuracy of predictions with the gaussian regression:  0.9833787842462866
```

The accuracy of predictions of the three kernels is very satisfactory (above 90%). The regressor that shows the best results is the one that uses the Gaussian kernel.

Applying different kernels in regressions:

Linear regression

In [23]:
```
Atrib=databases[["Av_education","School_expectancy","Life_expectancy_index","GDPpcLog"]]  #defines attributes
Atrib.dropna(inplace=True)                                    #drops NaNs, or else the SVM cannot be applied

Atrib_linear_r=Atrib

Score_HDI_SVM_linear_r = svregression_l.predict(Atrib_linear_r)               # Applies the SVM (regression)


#Creating a table with the attributes, the respective country-year information,
    #the classification from the SVC and the HDI score:
Atrib_linear_r['Score_HDI_SVM_linear_r'] = Score_HDI_SVM_linear_r
Atrib_linear_r['country']= databases.country
Atrib_linear_r['year']= databases.year
Atrib_linear_r['HDI_score']= databases.HDI_score

table_final_linear_r=Atrib[["country","year","Av_education","School_expectancy","Life_expectancy_index",
                            "GDPpcLog","HDI_score","Score_HDI_SVM_linear_r"]]
```

Polynomial Regression

In [24]:
```
Atrib=databases[["Av_education","School_expectancy","Life_expectancy_index","GDPpcLog"]]  #defines attributes
Atrib.dropna(inplace=True)                                    #drops NaNs, or else the SVM cannot be applied

Atrib_poly_r=Atrib

Score_HDI_SVM_poly_r = svregression_p.predict(Atrib_poly_r)                    # Applies the SVM (regression)


#Creating a table with the attributes, the respective country-year information,
    #the classification from the SVC and the HDI score:
Atrib_poly_r['Score_HDI_SVM_poly_r'] = Score_HDI_SVM_poly_r
Atrib_poly_r['country']= databases.country
Atrib_poly_r['year']= databases.year
Atrib_poly_r['HDI_score']= databases.HDI_score

table_final_poly_r=Atrib[["country","year","Av_education","School_expectancy","Life_expectancy_index",
                          "GDPpcLog","HDI_score","Score_HDI_SVM_poly_r"]]
```

Gaussian Regression

In [25]:
```
Atrib=databases[["Av_education","School_expectancy","Life_expectancy_index","GDPpcLog"]]  #defines attributes
Atrib.dropna(inplace=True)                                    #drops NaNs, or else the SVM cannot be applied
Atrib_rbf_r=Atrib

Score_HDI_SVM_rbf_r = svregression_g.predict(Atrib_rbf_r)                      # Applies the SVM (regression)


#Creating a table with the attributes, the respective country-year information,
    #the classification from the SVC and the HDI score:
Atrib_rbf_r['Score_HDI_SVM_rbf_r'] = Score_HDI_SVM_rbf_r
Atrib_rbf_r['country']= databases.country
Atrib_rbf_r['year']= databases.year
Atrib_rbf_r['HDI_score']= databases.HDI_score

table_final_rbf_r=Atrib[["country","year","Av_education","School_expectancy","Life_expectancy_index",
                         "GDPpcLog","HDI_score","Score_HDI_SVM_rbf_r"]]
```

Analyzing the correlation coefficients:

In [26]:
```
correlation=Atrib[["HDI_score"]]
correlation['Results from Linear regression']=table_final_linear_r.Score_HDI_SVM_linear_r
correlation['Results from Polynomial regression']=table_final_poly_r.Score_HDI_SVM_poly_r
correlation['Results from Gaussian regression']=table_final_rbf_r.Score_HDI_SVM_rbf_r

Correlation=correlation.corr()
print('Correlation between our countinuous indicators and the original HDI scores:')
Correlation
```

```
Correlation between our countinuous indicators and the original HDI scores:
```

Out[26]:

	HDI_score	Results from Linear regression	Results from Polynomial regression	Results from Gaussian regression
HDI_score	1.000000	0.987949	0.971399	0.938678
Results from Linear regression	0.987949	1.000000	0.976499	0.874830
Results from Polynomial regression	0.971399	0.976499	1.000000	0.933962
Results from Gaussian regression	0.938678	0.874830	0.933962	1.000000

Our regressors have a strong positive correlation (above 90%) with the HDI scores, with the linear regressor having the highest correlation coefficient.

Results

The indicator with the best results (in terms of accuracy) is the Gaussian:

In [27]:
```
table_final_rbf_r
```

Out[27]:

	country	year	Av_education	School_expectancy	Life_expectancy_index	GDPpcLog	HDI_score	Score_HDI_SVM_rbf_r
50	Afghanistan	1950	0.220	1.003679	0.200000	0.343504	NaN	0.070749
51	Afghanistan	1951	0.229	1.039555	0.206667	0.356675	NaN	0.070080
52	Afghanistan	1952	0.238	1.076713	0.216667	0.374276	NaN	0.069195
53	Afghanistan	1953	0.247	1.115200	0.228333	0.420045	NaN	0.066706
54	Afghanistan	1954	0.256	1.155061	0.240000	0.424421	NaN	0.066590
...	...	...	...	...	...	...	...	...
20229	Zimbabwe	2015	8.200	10.300000	0.608000	0.667453	0.553	0.078423
20230	Zimbabwe	2016	8.300	10.400000	0.620000	0.649484	0.558	0.088206
20231	Zimbabwe	2017	8.300	10.500000	0.628000	0.682656	0.563	0.100472
20232	Zimbabwe	2018	8.400	10.500000	0.634000	0.702056	0.569	0.108277
20233	Zimbabwe	2019	8.500	11.000000	0.638000	0.689935	0.571	0.148028

11405 rows × 8 columns

We will now apply an SVM to HDI data only (which contains data for the period 1990-2019 and, for the income index, GNI is used instead of GDP). The purpose is to then compare these results with those of our indicator and see if the replacement of the income index and the different priming dataset (due to the existence of certain incompatibilities between the data from different sources) produce different results.

In [28]:
```
#adjusting a few errors...
data_hdi['HDI_score'] = pd.to_numeric(data_hdi['HDI_score'], errors='coerce')

is_developed = data_hdi.HDI_score>=0.8
developedcountries=data_hdi[is_developed]
developed=developedcountries[["Country","Year", "Expected_years_of_schooling", "Mean_years_of_schooling",
                              "Life_expectancy_index", "Income_index"]]
developed['Label']=1

is_not_developed = data_hdi.HDI_score<0.55
leastdevelopedcountries=data_hdi[is_not_developed]
leastdeveloped=leastdevelopedcountries[["Country","Year", "Expected_years_of_schooling", "Mean_years_of_schooling",
                                        "Life_expectancy_index", "Income_index"]]
leastdeveloped['Label']=0

#joining developed and undeveloped countries in the same table
frames=[developed,leastdeveloped]
table=pd.concat(frames)

table.dropna(inplace=True)
table1=table[["Expected_years_of_schooling", "Mean_years_of_schooling", "Life_expectancy_index",
              "Income_index", "Label"]]

# Distinction between attributes and labels
X = table1.drop('Label', axis=1)
y = table1['Label']

# Dividing our data in training set and test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20)

#Gaussian Regression
from sklearn.svm import SVR
svregression = SVR(kernel='rbf')
svregression.fit(X_train, y_train)

y_pred = svregression.predict(X_test)

data_hdi.dropna(inplace=True)
Atribu=data_hdi[["Expected_years_of_schooling", "Mean_years_of_schooling",
                 "Life_expectancy_index", "Income_index"]]

Atribu['Expected_years_of_schooling'] = pd.to_numeric(Atribu['Expected_years_of_schooling'], errors='coerce')
Atribu['Mean_years_of_schooling'] = pd.to_numeric(Atribu['Mean_years_of_schooling'], errors='coerce')
Atribu['Life_expectancy_index'] = pd.to_numeric(Atribu['Life_expectancy_index'], errors='coerce')
Atribu['Income_index'] = pd.to_numeric(Atribu['Income_index'], errors='coerce')
Atribu.dropna(inplace=True)

Score = svregression.predict(Atribu)

final_gaussian=data_hdi [["Country","Year", "Expected_years_of_schooling", "Mean_years_of_schooling",
                          "Life_expectancy_index", "Income_index","HDI_score"]]
final_gaussian['Score_2'] = Score
```

Let us see the evolution of the indicators (our indicator, the indicator using only HDI data, and the HDI scores) for some countries:

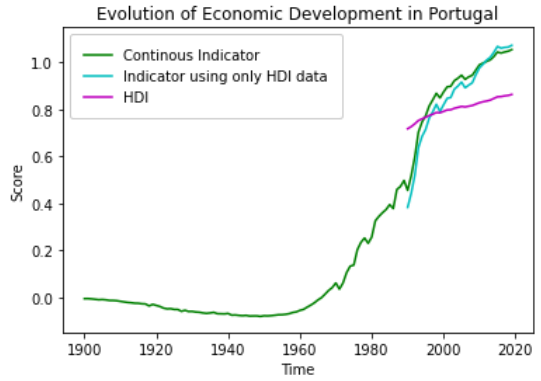
Portugal

In [29]:
```
is_Portugal= table_final_rbf_r.country=="Portugal"
Portugal=table_final_rbf_r[is_Portugal]

is_Portugal2= final_gaussian.Country==" Portugal"
```

```
Portugal2=final_gaussian[is_Portugal2]


plt.plot(Portugal.year, Portugal.Score_HDI_SVM_rbf_r, 'g', label="Continous Indicator")
plt.plot(Portugal2.Year, Portugal2.Score_2, 'c', label="Indicator using only HDI data")
plt.plot(Portugal.year, Portugal.HDI_score, 'm', label="HDI")
plt.legend(fancybox=True, framealpha=1, borderpad=1)
plt.title('Evolution of Economic Development in Portugal')
plt.xlabel('Time')
plt.ylabel('Score')
plt.ylim((-0.15,1.15))
```

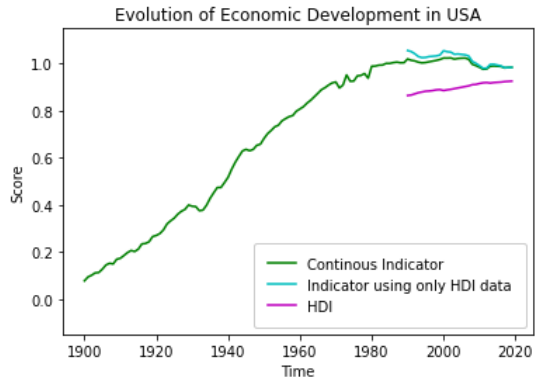


United States of America

In [30]:
```
is_USA= table_final_rbf_r.country=="United States of America"
USA=table_final_rbf_r[is_USA]

is_USA2= final_gaussian.Country==" United States"
USA2=final_gaussian[is_USA2]


plt.plot(USA.year, USA.Score_HDI_SVM_rbf_r, 'g', label="Continous Indicator")
plt.plot(USA2.Year, USA2.Score_2, 'c', label="Indicator using only HDI data")
plt.plot(USA.year, USA.HDI_score, 'm', label="HDI")
plt.legend(fancybox=True, framealpha=1, borderpad=1)
plt.title('Evolution of Economic Development in USA')
plt.xlabel('Time')
plt.ylabel('Score')
plt.ylim((-0.15,1.15))
```

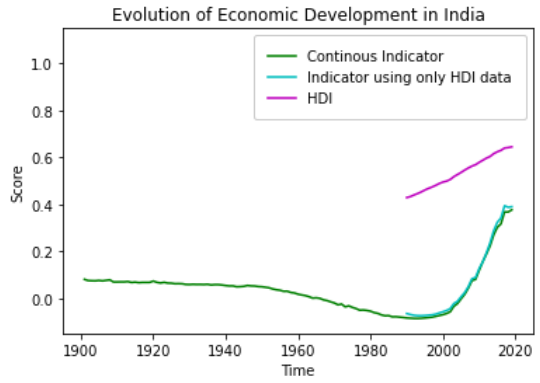


India

In [31]:
```
is_India= table_final_rbf_r.country=="India"
India=table_final_rbf_r[is_India]

is_India2= final_gaussian.Country==" India"
India2=final_gaussian[is_India2]


plt.plot(India.year, India.Score_HDI_SVM_rbf_r, 'g', label="Continous Indicator")
plt.plot(India2.Year, India2.Score_2, 'c', label="Indicator using only HDI data")
plt.plot(India.year, India.HDI_score, 'm', label="HDI")
plt.legend(fancybox=True, framealpha=1, borderpad=1)
plt.title('Evolution of Economic Development in India')
plt.xlabel('Time')
plt.ylabel('Score')
plt.ylim((-0.15,1.15))
```

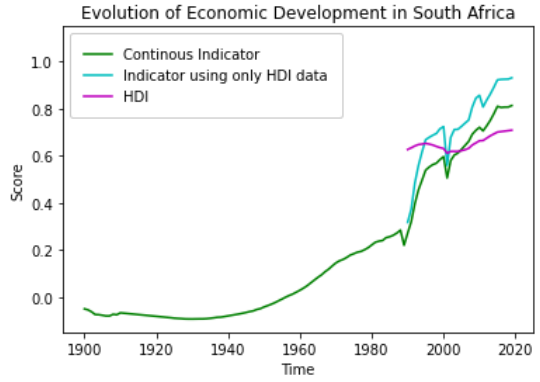


South Africa

In [32]:
```
is_South_Africa= table_final_rbf_r.country=="South Africa"
South_Africa=table_final_rbf_r[is_South_Africa]

is_South_Africa2= final_gaussian.Country==" South Africa"
South_Africa2=final_gaussian[is_South_Africa2]


plt.plot(South_Africa.year, South_Africa.Score_HDI_SVM_rbf_r, 'g', label="Continous Indicator")
plt.plot(South_Africa2.Year, South_Africa2.Score_2, 'c', label="Indicator using only HDI data")
plt.plot(South_Africa.year, South_Africa.HDI_score, 'm', label="HDI")
plt.legend(fancybox=True, framealpha=1, borderpad=1)
plt.title('Evolution of Economic Development in South Africa')
plt.xlabel('Time')
plt.ylabel('Score')
plt.ylim((-0.15,1.15))
```

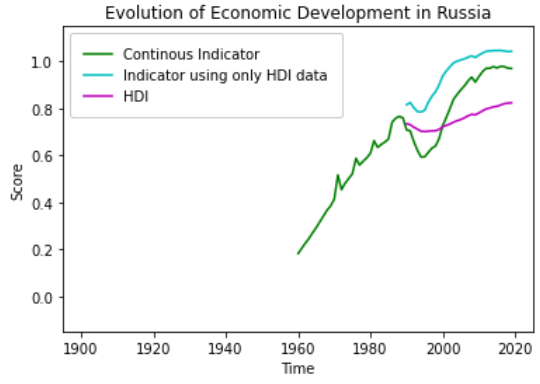


Russia

In [39]:
```
is_Russia= table_final_rbf_r.country=="Russia"
Russia=table_final_rbf_r[is_Russia]

is_Russia2= final_gaussian.Country==" Russian Federation"
Russia2=final_gaussian[is_Russia2]


plt.plot(Russia.year, Russia.Score_HDI_SVM_rbf_r, 'g', label="Continous Indicator")
plt.plot(Russia2.Year, Russia2.Score_2, 'c', label="Indicator using only HDI data")
plt.plot(Russia.year, Russia.HDI_score, 'm', label="HDI")
plt.legend(fancybox=True, framealpha=1, borderpad=1)
plt.title('Evolution of Economic Development in Russia')
plt.xlabel('Time')
plt.ylabel('Score')
plt.ylim((-0.15,1.15))
plt.xlim((1895,2025))
```

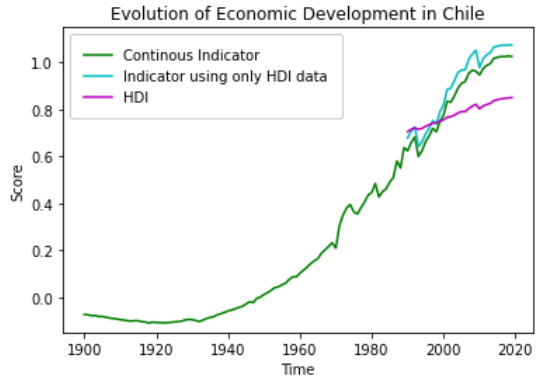


Chile

In [46]:
```
is_Chile = table_final_rbf_r.country=="Chile"
Chile=table_final_rbf_r[is_Chile]

is_Chile2= final_gaussian.Country==" Chile"
Chile2=final_gaussian[is_Chile2]


plt.plot(Chile.year, Chile.Score_HDI_SVM_rbf_r, 'g', label="Continous Indicator")
plt.plot(Chile2.Year, Chile2.Score_2, 'c', label="Indicator using only HDI data")
plt.plot(Chile.year, Chile.HDI_score, 'm', label="HDI")
plt.legend(fancybox=True, framealpha=1, borderpad=1)
plt.title('Evolution of Economic Development in Chile')
plt.xlabel('Time')
plt.ylabel('Score')
plt.ylim((-0.15,1.15))
```

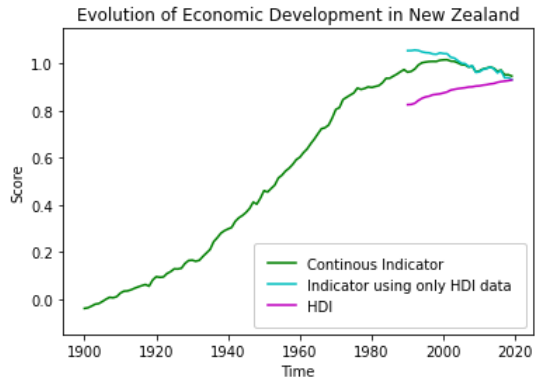


New Zealand

In [45]:
```
is_NZ = table_final_rbf_r.country=="New Zealand"
NZ=table_final_rbf_r[is_NZ]

is_NZ2= final_gaussian.Country==" New Zealand"
NZ2=final_gaussian[is_NZ2]


plt.plot(NZ.year, NZ.Score_HDI_SVM_rbf_r, 'g', label="Continous Indicator")
plt.plot(NZ2.Year, NZ2.Score_2, 'c', label="Indicator using only HDI data")
plt.plot(NZ.year, NZ.HDI_score, 'm', label="HDI")
plt.legend(fancybox=True, framealpha=1, borderpad=1)
plt.title('Evolution of Economic Development in New Zealand')
plt.xlabel('Time')
plt.ylabel('Score')
plt.ylim((-0.15,1.15))
```

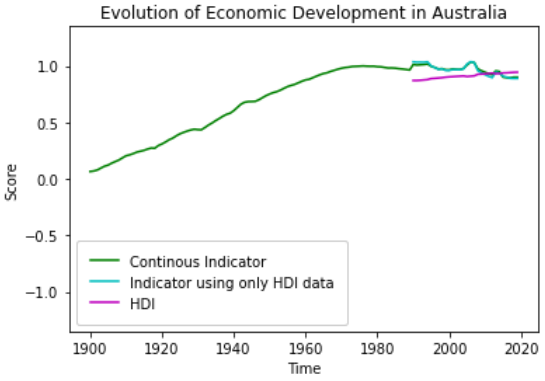


Australia

In [44]:
```
is_Australia= table_final_rbf_r.country=="Australia"
Australia=table_final_rbf_r[is_Australia]

is_Australia2= final_gaussian.Country==" Australia"
Australia2=final_gaussian[is_Australia2]


plt.plot(Australia.year, Australia.Score_HDI_SVM_rbf_r, 'g', label="Continous Indicator")
plt.plot(Australia2.Year, Australia2.Score_2, 'c', label="Indicator using only HDI data")
plt.plot(Australia.year, Australia.HDI_score, 'm', label="HDI")
plt.legend(fancybox=True, framealpha=1, borderpad=1)
plt.title('Evolution of Economic Development in Australia')
plt.xlabel('Time')
plt.ylabel('Score')
plt.ylim((-1.35,1.35))
```


Evolution of Economic Development in Australia
1.0
0.5
0.0
−0.5
−1.0
Score
1900
1920
1940
1960
1980
2000
2020
Time
Continous Indicator
Indicator using only HDI data
HDI