LIFE EXPECTANCY ANALYSIS AND PREDICTION

- USING ML AND POWERBI

Problem Statement:

The data-set related to life expectancy, health factors for 193 countries has been collected from the same WHO data repository website and its corresponding economic data was collected from United Nation website. Among all categories of health-related factors only those critical factors were chosen which are more representative. It has been observed that in the past 15 years , there has been a huge development in health sector resulting in improvement of human mortality rates especially in the developing nations in comparison to the past 30 years. Therefore, in this project we have considered data from year 2000-2015 for 193 countries for further analysis.

Data Definition:

Input variables:

1. **Country** Country names
2. **Year** Year of data recorded
3. **Status** Developed or Developing status
4. **Adult Mortality** Adult Mortality Rates of both sexes (probability of dying between 15 and 60 years per 1000 population)
5. **infant deaths** Number of Infant Deaths per 1000 population
6. **Alcohol** Alcohol, recorded per capita (15+) consumption (in litres of pure alcohol)
7. **percentage expenditure** Expenditure on health as a percentage of Gross Domestic Product per capita(%)
8. **Hepatitis B** Hepatitis B (HepB) immunization coverage among 1-year-olds (%)
9. **Measles** number of reported cases per 1000 population
10. **BMI** Average Body Mass Index of entire population
11. **under-five deaths** Number of under-five deaths per 1000 population

12. **Polio** Polio (Pol3) immunization coverage among 1-year-olds (%)

13. **Total expenditure** General government expenditure on health as a percentage of total government expenditure (%)

14. **Diphtheria** Diphtheria tetanus toxoid and pertussis (DTP3) immunization coverage among 1-year-olds (%)
15. **HIV/AIDS**Deaths per 1 000 live births HIV/AIDS (0-4 years)
16. **GDP** Gross Domestic Product per capita (in USD)
17. **Population** Population of the country
18. **thinness 1-19 years** Prevalence of thinness among children and adolescents for Age 10 to 19 (%)
19. **thinness 5-9 years** Prevalence of thinness among children for Age 5 to 9(%)
20. **Income composition of resources** Human Development Index in terms of income composition of resources (index ranging from 0 to 1)
21. **Schooling** Number of years of Schooling(years)

Output variable (desired target):

1. **Life expectancy** (target) Life Expectancy in years

1. Import Data From MySQL Database

In [1]:
```
# Importing the Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import mysql.connector as my_sql
```

1.1 MySQL Connect

In [2]:
```
db=my_sql.connect(host="localhost",user="root",password="mytheesh@1626",database="
db
```

Out[2]:
```
<mysql.connector.connection_cext.CMySQLConnection at 0x23cd3ab4100>
```

In [3]:
```
#Set cursor Object_
mycursor=db.cursor()
mycursor=db.cursor( buffered=True , dictionary=True)
```

1.2 Read Data From MySQL Database

In [4]:
```
#Access table from yout Datbase
mycursor.execute('SELECT * FROM `led`')
```

In [5]:
```
ldf=pd.DataFrame(mycursor.fetchall())
data=ldf.copy()
```

In [6]:
```
data.head()
```

Out[6]:

	Country	Year	Status	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B
0	Afghanistan	2015	Developing	65.0	263	62	0.01	71.279624	65
1	Afghanistan	2014	Developing	59.9	271	64	0.01	73.523582	62
2	Afghanistan	2013	Developing	59.9	268	66	0.01	73.219243	64
3	Afghanistan	2012	Developing	59.5	272	69	0.01	78.184215	67
4	Afghanistan	2011	Developing	59.2	275	71	0.01	7.097109	68

5 rows × 22 columns

2. Exploratory Data Analysis

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2938 entries, 0 to 2937
Data columns (total 22 columns):
 #   Column                           Non-Null Count  Dtype  
---  ------                           --------------  -----  
 0   Country                          2938 non-null   object 
 1   Year                             2938 non-null   int64  
 2   Status                           2938 non-null   object 
 3   Life expectancy                  2938 non-null   float64
 4   Adult Mortality                  2938 non-null   int64  
 5   infant deaths                    2938 non-null   int64  
 6   Alcohol                          2938 non-null   float64
 7   percentage expenditure           2938 non-null   float64
 8   Hepatitis B                      2938 non-null   int64  
 9   Measles                          2938 non-null   int64  
 10  BMI                              2938 non-null   float64
 11  under-five deaths                2938 non-null   int64  
 12  Polio                            2938 non-null   int64  
 13  Total expenditure                2938 non-null   float64
 14  Diphtheria                       2938 non-null   int64  
 15  HIV/AIDS                         2938 non-null   float64
 16  GDP                              2938 non-null   float64
 17  Population                       2938 non-null   int64  
 18  thinness  1-19 years             2938 non-null   float64
 19  thinness 5-9 years               2938 non-null   float64
 20  Income composition of resources  2938 non-null   float64
 21  Schooling                        2938 non-null   float64
dtypes: float64(11), int64(9), object(2)
memory usage: 505.1+ KB
```

In [8]:
```
data.describe().T
```

Out[8]:

	count	mean	std	min	25%	50%	75
Year	2938.0	2.007519e+03	4.613841e+00	2000.0	2004.000000	2008.000000	2.012000e+
Life expectancy	2938.0	6.898931e+01	1.032744e+01	0.0	63.000000	72.000000	7.560000e+
Adult Mortality	2938.0	1.642355e+02	1.244511e+02	0.0	73.000000	144.000000	2.270000e+
infant deaths	2938.0	3.030395e+01	1.179265e+02	0.0	0.000000	3.000000	2.200000e+
Alcohol	2938.0	4.298928e+00	4.079748e+00	0.0	0.470000	3.130000	7.390000e+
percentage expenditure	2938.0	7.382513e+02	1.987915e+03	0.0	4.685343	64.912906	4.415341e+
Hepatitis B	2938.0	6.570558e+01	3.887832e+01	0.0	24.000000	87.000000	9.600000e+
Measles	2938.0	2.419592e+03	1.146727e+04	0.0	0.000000	17.000000	3.602500e+
BMI	2938.0	3.787777e+01	2.034492e+01	0.0	19.000000	43.000000	5.610000e+
under-five deaths	2938.0	4.203574e+01	1.604455e+02	0.0	0.000000	4.000000	2.800000e+
Polio	2938.0	8.201634e+01	2.427183e+01	0.0	77.000000	93.000000	9.700000e+
Total expenditure	2938.0	5.481406e+00	2.875063e+00	0.0	3.740000	5.540000	7.330000e+
Diphtheria	2938.0	8.179170e+01	2.454410e+01	0.0	78.000000	93.000000	9.700000e+
HIV/AIDS	2938.0	1.742103e+00	5.077785e+00	0.1	0.100000	0.100000	8.000000e-
GDP	2938.0	6.342091e+03	1.340950e+04	0.0	190.174435	1171.983435	4.779405e+
Population	2938.0	9.923150e+06	5.407586e+07	0.0	5874.250000	539357.500000	4.584371e+
thinness 1-19 years	2938.0	4.783696e+00	4.424924e+00	0.0	1.500000	3.300000	7.100000e+
thinness 5-9 years	2938.0	4.813955e+00	4.512880e+00	0.0	1.500000	3.300000	7.200000e+
Income composition of resources	2938.0	5.918802e-01	2.511398e-01	0.0	0.465000	0.662000	7.720000e-
Schooling	2938.0	1.132743e+01	4.265626e+00	0.0	9.500000	12.100000	1.410000e+

In [9]:
```
data.columns
```
Out[9]:
```
Index(['Country', 'Year', 'Status', 'Life expectancy', 'Adult Mortality',
       'infant deaths', 'Alcohol', 'percentage expenditure', 'Hepatitis B',
       'Measles', 'BMI', 'under-five deaths', 'Polio', 'Total expenditure',
       'Diphtheria', 'HIV/AIDS', 'GDP', 'Population', 'thinness  1-19 years',
       'thinness 5-9 years', 'Income composition of resources', 'Schooling'],
      dtype='object')
```
In [10]:
```
# 'shape' function gives the total number of rows and columns in the data
data.shape
```
Out[10]:
```
(2938, 22)
```

2.1 Checking Missing Values

In [11]:
```
# sort the variables on the basis of total null values in the variable
# 'isnull().sum()' returns the number of missing values in each variable
# 'ascending = False' sorts values in the descending order
# the variable with highest number of missing values will appear first
Total = data.isnull().sum().sort_values(ascending = False)

# calculate the percentage of missing values
# 'ascending = False' sorts values in the descending order
# the variable with highest percentage of missing values will appear first
Percent = (data.isnull().sum()*100/data.isnull().count()).sort_values(ascending = F

# concat the 'Total' and 'Percent' columns using 'concat' function
# 'keys' is the list of column names
# 'axis = 1' concats along the columns
missing_data = pd.concat([Total, Percent], axis = 1, keys = ['Total', 'Percentage o
missing_data
```

Out[11]:

	Total	Percentage of Missing Values
Country	0	0.0
Year	0	0.0
Income composition of resources	0	0.0
thinness 5-9 years	0	0.0
thinness 1-19 years	0	0.0
Population	0	0.0
GDP	0	0.0
HIV/AIDS	0	0.0
Diphtheria	0	0.0
Total expenditure	0	0.0
Polio	0	0.0
under-five deaths	0	0.0
BMI	0	0.0
Measles	0	0.0
Hepatitis B	0	0.0
percentage expenditure	0	0.0
Alcohol	0	0.0
infant deaths	0	0.0
Adult Mortality	0	0.0
Life expectancy	0	0.0
Status	0	0.0
Schooling	0	0.0

2.2 Visualization of the Data - PowerBI Report

Interface Jupyter and PowerBI to visualize the data.

- Get the report from powerBI by DeviceCodeLoginAuthentication from PowerBIClient Module.

In [12]:
```
#import library
from powerbiclient import Report, models
```

In [13]:
```
# Import the DeviceCodeLoginAuthentication class to authenticate against Power BI
from powerbiclient.authentication import DeviceCodeLoginAuthentication

# Initiate device authentication
device_auth = DeviceCodeLoginAuthentication()
```

```
Performing interactive authentication. Please follow the instructions on the termi
nal.
 To sign in, use a web browser to open the page https://microsoft.com/devicelogin
and enter the code HXXJN7MJ4 to authenticate.
You have logged in.
Interactive authentication successfully completed.
```

In [14]:
```
group_id="25ea950c-a927-44ab-a709-62db15b22de7"
report_id="77100a0b-f74a-40e5-ad28-7b3afa16ebe3"
report = Report(group_id=group_id, report_id=report_id, auth=device_auth)

report

#Use this Link to view if not interface
#https://app.powerbi.com/reportEmbed?reportId=77100a0b-f74a-40e5-ad28-7b3afa16ebe3&
```

```
Report()
```

3. Statistical Analysis on factors influencing Life Expectancy

3.1 The impact of Immunization coverage on life Expectancy

In [15]:
```
group_id="25ea950c-a927-44ab-a709-62db15b22de7"
report_id="1be3cf5a-bc0b-4aa3-bbf5-7515f4011320"
report2 = Report(group_id=group_id, report_id=report_id, auth=device_auth)

report2

#Use this Link to view if not interface
#https://app.powerbi.com/reportEmbed?reportId=1be3cf5a-bc0b-4aa3-bbf5-7515f4011320&
```

```
Report()
```

3.2 How Percentage Expenditure will affect lower life expectancy?

In [16]:
```
group_id="25ea950c-a927-44ab-a709-62db15b22de7"
report_id="d211d54d-e9c9-4f0c-8568-a765b67f5080"
```

```
report3 = Report(group_id=group_id, report_id=report_id, auth=device_auth)

report3
#Use this Link to view if not interface
#https://app.powerbi.com/reportEmbed?reportId=d211d54d-e9c9-4f0c-8568-a765b67f5080&
```

```
Report()
```

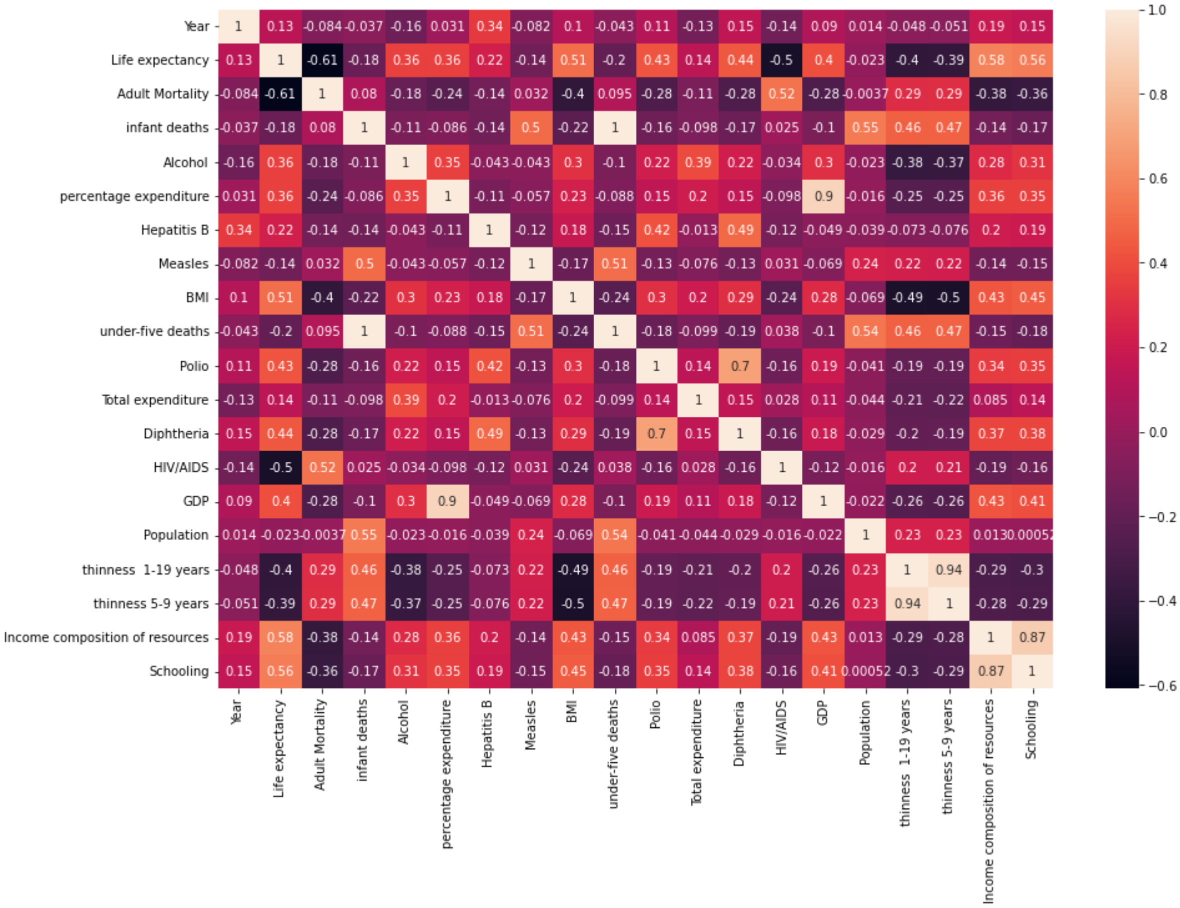
3.3 How does Infant and Adult mortality rates affect life expectancy?

In [17]:
```
group_id="25ea950c-a927-44ab-a709-62db15b22de7"
report_id="4a421673-aa6f-46b0-9819-a72f61e3c932"
report4 = Report(group_id=group_id, report_id=report_id, auth=device_auth)

report4
#Use this Link to view if not interface
#https://app.powerbi.com/reportEmbed?reportId=4a421673-aa6f-46b0-9819-a72f61e3c932&
```

```
Report()
```

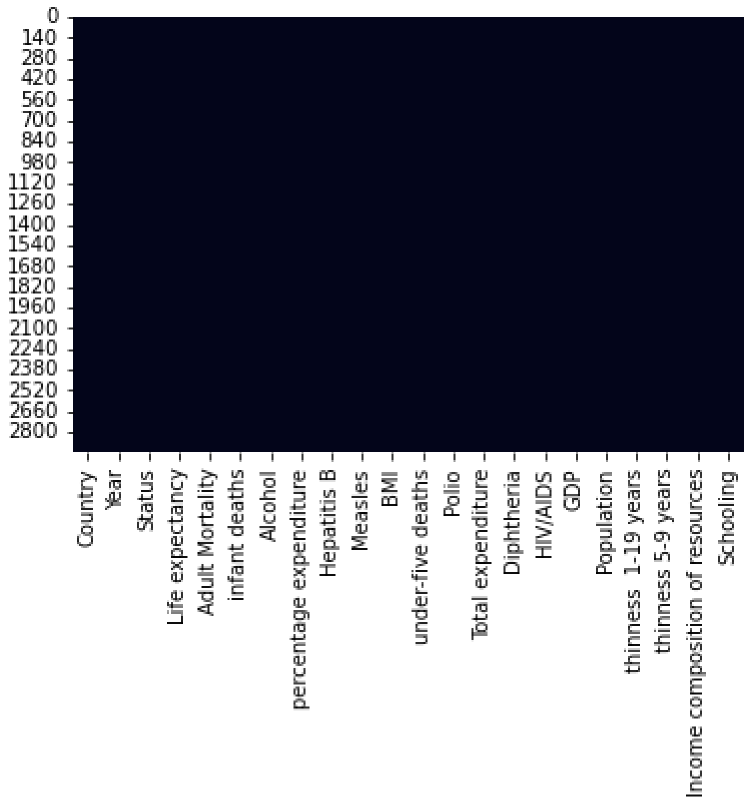
4. Data Preprocessing

In [18]:
```
plt.figure(figsize=(15,10))
sns.heatmap(data.corr(),annot=True)
plt.show()
```



4.1 Plot a heatmap - Visualization of missing values.

In [19]:
```python
# plot heatmap to check null values
# 'cbar = False' does not show the color axis
sns.heatmap(data.isnull(), cbar=False)

# display the plot
plt.show()
```



We determine:

- The horizontal lines in the heatmap correspond to the missing values. But there are no such line. This means there are no missing values.

In [20]:
```python
data.head().T
```

Out[20]:

	0	1	2	3	4
Country	Afghanistan	Afghanistan	Afghanistan	Afghanistan	Afghanistan
Year	2015	2014	2013	2012	2011
Status	Developing	Developing	Developing	Developing	Developing
Life expectancy	65.0	59.9	59.9	59.5	59.2
Adult Mortality	263	271	268	272	275
infant deaths	62	64	66	69	71
Alcohol	0.01	0.01	0.01	0.01	0.01
percentage expenditure	71.279624	73.523582	73.219243	78.184215	7.097109
Hepatitis B	65	62	64	67	68
Measles	1154	492	430	2787	3013
BMI	19.1	18.6	18.1	17.6	17.2
under-five deaths	83	86	89	93	97
Polio	6	58	62	67	68
Total expenditure	8.16	8.18	8.13	8.52	7.87
Diphtheria	65	62	64	67	68
HIV/AIDS	0.1	0.1	0.1	0.1	0.1
GDP	584.25921	612.696514	631.744976	669.959	63.537231
Population	33736494	327582	31731688	3696958	2978599
thinness 1-19 years	17.2	17.5	17.7	17.9	18.2
thinness 5-9 years	17.3	17.5	17.7	18.0	18.2
Income composition of resources	0.479	0.476	0.47	0.463	0.454
Schooling	10.1	10.0	9.9	9.8	9.5

4.2 Converting Categorial values into Numerical values

In [21]:

```
from sklearn.preprocessing import LabelEncoder,OneHotEncoder
labelencoder=LabelEncoder()
data['Country']=labelencoder.fit_transform(data['Country'])
data['Year']=labelencoder.fit_transform(data['Year'])
data['Status']=labelencoder.fit_transform(data['Status'])
data.head()
```

Out[21]:

	Country	Year	Status	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measle
0	0	15	1	65.0	263	62	0.01	71.279624	65	115
1	0	14	1	59.9	271	64	0.01	73.523582	62	49
2	0	13	1	59.9	268	66	0.01	73.219243	64	43
3	0	12	1	59.5	272	69	0.01	78.184215	67	278
4	0	11	1	59.2	275	71	0.01	7.097109	68	301

5 rows × 22 columns

In [22]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2938 entries, 0 to 2937
Data columns (total 22 columns):
 #   Column                           Non-Null Count  Dtype  
---  ------                           --------------  -----  
 0   Country                          2938 non-null   int32  
 1   Year                             2938 non-null   int64  
 2   Status                           2938 non-null   int32  
 3   Life expectancy                  2938 non-null   float64
 4   Adult Mortality                  2938 non-null   int64  
 5   infant deaths                    2938 non-null   int64  
 6   Alcohol                          2938 non-null   float64
 7   percentage expenditure           2938 non-null   float64
 8   Hepatitis B                      2938 non-null   int64  
 9   Measles                          2938 non-null   int64  
 10  BMI                              2938 non-null   float64
 11  under-five deaths                2938 non-null   int64  
 12  Polio                            2938 non-null   int64  
 13  Total expenditure                2938 non-null   float64
 14  Diphtheria                       2938 non-null   int64  
 15  HIV/AIDS                         2938 non-null   float64
 16  GDP                              2938 non-null   float64
 17  Population                       2938 non-null   int64  
 18  thinness  1-19 years             2938 non-null   float64
 19  thinness 5-9 years               2938 non-null   float64
 20  Income composition of resources  2938 non-null   float64
 21  Schooling                        2938 non-null   float64
dtypes: float64(11), int32(2), int64(9)
memory usage: 482.1 KB
```

4.3 Splitting the data into x and y

In [23]:
```
X=data.drop('Life expectancy',axis=1) #Features
y=data['Life expectancy']            #Target
```

4.4 Splitting the data into Train and Test Splits

In [24]:
```
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, random_st
```

In [25]:
```
X_train
```

Out[25]:

	Country	Year	Status	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles	BMI
1617	102	0	1	139	0	1.83	300.162103	96	20	15.2
1331	85	14	1	113	4	0.41	63.878452	98	20	64.8
1814	118	14	1	158	18	0.01	8.523486	92	1279	18.5
216	13	7	1	113	0	8.47	1641.309810	93	0	48.4
445	44	2	1	473	65	3.13	0.000000	48	5882	2.4
...	...	...	...	...	...	...	...	...	...	...
599	36	8	1	255	2	0.12	93.367890	81	0	2.6
1599	101	2	1	14	4	0.49	216.702948	95	408	27.9
1361	86	0	1	292	9	6.00	112.541157	99	245	43.9
1547	98	6	0	85	0	11.98	15345.490700	95	8	57.5
863	55	2	1	343	7	0.83	0.703132	86	460	13.1

2350 rows × 21 columns

In [26]:
```
y_train
```

Out[26]:
```
1617    69.6
1331    74.0
1814    69.6
216     74.2
445     47.7
        ...
599     61.0
1599    72.9
1361    63.9
1547    79.4
863     58.5
Name: Life expectancy, Length: 2350, dtype: float64
```

5. Modelling

FIND THE MOST SUITABLE ALGORITHM

In [27]:
```
# Importing the Models and Model Evaluators
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.model_selection import cross_val_score
```

5.1 Put models in a dictionary

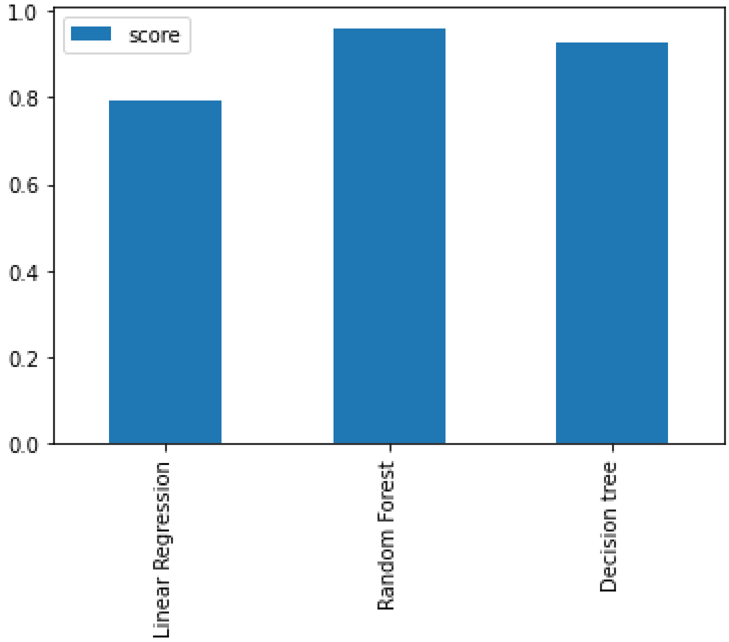
In [28]:
```
models = { "Linear Regression": LinearRegression(),
           "Random Forest": RandomForestRegressor(),
           "Decision tree": DecisionTreeRegressor()}

# Create function to fit and score models
def fit_and_score(models, X_train, X_test, y_train, y_test):
    """
    Fits and evaluates given machine learning models.
    models : a dict of different Scikit-Learn machine learning models
    X_train : training data
    X_test : testing data
    y_train : labels assosciated with training data
    y_test : labels assosciated with test data
    """
    # Random seed for reproducible results
    np.random.seed(101)
    # Make a list to keep model scores
    model_scores = {}
    # Loop through models
    for name, model in models.items():
        model.fit(X_train, y_train)
        model_scores[name] = model.score(X_test, y_test)
    return model_scores
```

5.2 Evaluate and Compare the Model Scores

In [29]:
```
model_scores = fit_and_score(models=models,
                             X_train=X_train,
                             X_test=X_test,
                             y_train=y_train,
                             y_test=y_test)
model_scores
```

Out[29]:
```
{'Linear Regression': 0.7946281268985743,
 'Random Forest': 0.9606835446908324,
 'Decision tree': 0.9295401567406675}
```

In [30]:
```
model_compare = pd.DataFrame(model_scores, index=['score'])
model_compare.T.plot.bar();
```



6. Hyerparameter tuning with RandomizedSearchCV

In [40]:

```
# Different RandomForestRegressor hyperparameters
rf_grid = {"n_estimators": np.arange(10, 100, 10),
           "max_depth": [None, 3, 5, 10],
           "min_samples_split": np.arange(2, 20, 2),
           "min_samples_leaf": np.arange(1, 20, 2),
           "max_features": [0.5, 1, "sqrt", "auto"],
           }

# Instantiate RandomizedSearchCV model
rs_model = RandomizedSearchCV(RandomForestRegressor(n_jobs=-1,
                                                    random_state=42),
                                                    param_distributions=rf_grid,
                                                    n_iter=20,
                                                    cv=5,
                                                    verbose=True)

# Fit the RandomizedSearchCV model
rs_model.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 20 candidates, totalling 100 fits
```

Out[40]:

```
RandomizedSearchCV(cv=5,
                   estimator=RandomForestRegressor(n_jobs=-1, random_state=42),
                   n_iter=20,
                   param_distributions={'max_depth': [None, 3, 5, 10],
                                        'max_features': [0.5, 1, 'sqrt',
                                                         'auto'],
                                        'min_samples_leaf': array([ 1,  3,  5,  7,
9, 11, 13, 15, 17, 19]),
                                        'min_samples_split': array([ 2,  4,  6,
8, 10, 12, 14, 16, 18]),
                                        'n_estimators': array([10, 20, 30, 40, 50,
60, 70, 80, 90])},
                   verbose=True)
```

In [41]:

```
# Find the best model hyperparameters
rs_model.best_params_
```

Out[41]:
```
{'n_estimators': 10,
 'min_samples_split': 14,
 'min_samples_leaf': 1,
 'max_features': 0.5,
 'max_depth': 10}
```

In [42]:
```
# Most ideal hyperparamters
ideal_model = RandomForestRegressor(n_estimators=30,
                                    min_samples_split=2,
                                    min_samples_leaf=1,
                                    max_features=0.5,
                                    max_depth=10,
                                    n_jobs=-1,
                                    random_state=101)

# Fit the ideal model
ideal_model.fit(X_train, y_train)
```

Out[42]:
```
RandomForestRegressor(max_depth=10, max_features=0.5, n_estimators=30,
                      n_jobs=-1, random_state=101)
```

In [43]:
```
ideal_model.score(X_test, y_test)
```

Out[43]:
```
0.9603446273350983
```

In [44]:
```
# Make predictions on test data
y_preds=ideal_model.predict(X_test)
```

In [45]:
```
y_preds
```

Out[45]:
```
array([63.63871743, 54.61437355, 82.13750579, 65.30377368, 72.71244665,
       73.19095138, 68.32076229, 81.39145748, 60.50605556, 82.77565574,
       61.77695474, 71.49032985, 74.68715186, 74.78787866, 51.77066397,
       73.89830818, 68.39479661, 71.85149932, 70.66762802, 76.235606  ,
       56.79153877, 56.37379989, 74.95323766, 71.56775874, 68.9158556 ,
       53.51536707, 71.90182151, 73.47692476, 67.68670926, 73.90675876,
       55.27953202, 54.72088221, 74.48052633, 74.59705412, 82.0081258 ,
       71.21815827, 73.96010496, 63.09376225, 71.27051122, 58.72343685,
       59.46081599, 81.71413923, 64.90656457, 73.07743244, 68.08980783,
       69.2795689 , 73.77105125, 73.11682081, 74.52399945, 67.80539744,
       74.39713909, 80.71392478, 53.15812605, 82.03849881, 60.42375932,
       62.68041022, 81.5784296 , 68.72730062, 74.50394119, 52.45921872,
       67.75507964, 66.50342462, 72.00460163, 81.40728547, 73.69606161,
       62.3433981 , 80.72782033, 74.61329773, 82.43061338, 72.24585898,
       70.96581805, 58.17950963, 73.7166851 , 67.81980018, 81.2985423 ,
       48.94727951, 57.23667544, 58.05930987, 81.71012925, 53.65656299,
       51.94192585, 70.75572327, 58.6874403 , 72.54177067, 74.25808923,
       62.21675   , 82.69649973, 51.25479719, 79.36582974, 79.5125813 ,
       58.72532975, 68.30614213, 73.07849292, 53.48330766, 47.42130659,
       73.23219734, 74.27583412, 74.79671252, 72.54106585, 71.88704411,
       64.65219871, 62.55268801, 83.04621191, 72.6425394 , 73.45126176,
       71.96846829, 73.68540031, 80.32150465, 66.63326217, 81.9621259 ,
       49.55389726, 82.36307519, 72.7875512 , 74.88255596, 62.22616667,
       70.55991785, 69.37841776, 74.59844376, 74.21720906, 80.68575904,
       73.75100082, 73.49687647, 67.28234307, 59.89934781, 79.69699206,
       80.20733556, 73.53051274, 72.52915867, 51.19833333, 76.15775693,
       78.75394617, 72.93040895, 63.66435713, 75.43910794, 80.65677757,
       73.60359212, 76.41292286, 56.72290385, 73.71707137, 75.47156684,
       63.93641655, 72.68875952, 72.33933049, 62.81237809, 72.84659657,
       63.70649519, 82.00776189, 68.00451244, 82.66174711, 60.75427359,
       81.82997909, 56.30573911, 81.76641076, 81.88479316, 73.05195688,
       81.39022234, 73.31345319, 63.28557653, 75.83440934, 62.34498169,
       67.57367559, 81.92917414, 62.25247712, 63.61511903, 53.80792281,
       76.45633807, 81.92409613, 70.02515752, 81.99882333, 57.05513771,
       74.18952222, 71.87123635, 71.00461685, 59.48512326, 73.99723753,
       70.651865  , 72.32315807, 79.87905027, 73.10670241, 73.6536489 ,
       51.54661163, 80.65738679, 72.6985452 , 52.83853052, 73.98153348,
       75.51386233, 82.48055624, 81.41288187, 69.32496487, 62.59084127,
       73.50741618, 74.97764522, 48.23277746, 79.97657255, 72.1168998 ,
       62.55452529, 74.61001628, 73.98253771, 55.88477847, 77.54805568,
       71.2244915 , 53.94355597, 73.80302275, 73.10354395, 75.14224988,
       63.08149673, 74.39723723, 74.38608994, 53.13133765, 73.29934896,
       81.89793542, 83.2433543 , 73.91139736, 73.57563332, 75.70872102,
       72.97788016, 70.92982528, 58.27586018, 71.96053969, 77.08750699,
       79.21290158, 68.34547135, 56.07138736, 74.04469221, 73.05118869,
       78.34809269, 52.42898093, 68.44582106, 68.00867681, 79.81735391,
       72.56035429, 48.40486558, 73.80814493, 75.48889729, 76.69986599,
       58.88718771, 73.13527119, 72.76662019, 59.67134389, 79.20566839,
       80.1697424 , 75.69070105, 76.02297068, 58.43847212, 75.40091   ,
       74.15876766, 69.39340453, 64.46142473, 55.26198639, 57.96398917,
       72.88380873, 73.39619203, 60.7789881 , 80.04460329, 54.15676886,
       51.94796013, 53.71984805, 68.64469399, 57.77988935, 59.98388724,
       80.03237084, 81.90737619, 74.88852383, 72.81306448, 76.08612579,
       75.59280979, 81.20109882, 79.27296873, 73.36547134, 71.32810684,
       66.99934839, 76.55512158, 79.47747321, 49.18068696, 60.26510928,
       74.31202732, 75.60427241, 58.33982463, 72.74816515, 68.04334246,
       82.29642259, 78.91386073, 56.14159674, 76.36904078, 79.85891802,
       75.87796057, 73.38162132, 83.11022826, 79.27546255, 68.42467363,
       81.02985768, 61.32033221, 45.78906382, 72.20603917, 73.49765217,
       73.03552195, 73.83974693, 82.88721876, 80.05586354, 71.25516387,
       80.02405425, 74.34741692, 59.8645077 , 72.83132618, 75.77537952,
       82.65985891, 73.7662076 , 76.28931495, 62.83057672, 64.71577804,
       67.50375056, 52.04580396, 73.17624616, 54.26127385, 79.45053948,
       73.31445254, 74.86411866, 70.74316303, 65.03963268, 72.79445564,
```

```
54.7152156 , 58.89676831, 62.44371418, 70.71898806, 55.87015955,
77.29792511, 76.30933642, 62.37182314, 63.20027159, 79.56139179,
64.16454677, 58.80619227, 78.47507877, 83.052071  , 69.87545071,
81.63356572, 75.35810082, 72.6634504 , 67.02123808, 70.19868193,
78.63621971, 76.84746544, 54.75023122, 76.80802309, 74.07273237,
53.50585571, 59.96000271, 66.69475186, 72.47834255, 71.82216195,
71.22274556, 76.49731282, 76.75241084, 75.87965977, 72.62789949,
56.26317258, 80.65375874, 64.46387827, 82.33785296, 55.5951485 ,
83.78441081, 74.56141274, 54.82627229, 65.63914887, 73.8115937 ,
68.46902638, 73.47470837, 70.15515636, 53.86694728, 75.17260965,
69.1588019 , 69.26001424, 74.59844376, 74.97792914, 54.35645529,
48.11638939, 75.08530441, 72.12593092, 53.02978022, 83.06548069,
67.39678219, 79.90917659, 72.17209607, 83.85701256, 82.54428989,
65.10749448, 58.5671572 , 76.9725872 , 53.8905642 , 82.55676277,
73.79989896, 67.62462245, 75.72121617, 73.78204592, 79.70166696,
59.39301587, 63.499429  , 81.83156788, 73.0487879 , 46.40722222,
65.27966913, 74.85795132, 67.82286674, 62.4446506 , 64.60995214,
69.90822914, 79.93267601, 82.74218112, 73.91574973, 73.70306142,
63.64155052, 72.37191666, 73.44426312, 81.39667405, 68.22051405,
76.8698883 , 50.69975984, 68.58055759, 65.20807597, 74.60213728,
77.19506986, 74.96632293, 47.42656177, 68.4475116 , 54.5192464 ,
62.5884921 , 73.55829407, 52.44790805, 73.32499523, 69.90418036,
74.0510239 , 73.98470539, 66.96378176, 69.26326776, 73.47889454,
62.21520558, 73.69468419, 49.94561538, 63.44531524, 82.24373063,
82.71385965, 67.58409285, 81.47476012, 82.19039848, 84.02143992,
76.55886268, 73.07724788, 67.25682011, 57.02154304, 73.21403241,
82.45722819, 73.21579465, 58.57705306, 78.1072765 , 78.27303375,
66.69282158, 72.10306094, 73.39430854, 79.34077016, 83.53008325,
67.32415978, 74.66286157, 53.29561597, 57.13736673, 69.61440036,
70.00789967, 74.44331032, 73.96046363, 73.95535722, 73.90699687,
73.74574613, 55.83930313, 61.81073907, 68.23603943, 73.04809885,
59.891636  , 83.26710872, 68.80794906, 56.48554293, 71.84229257,
62.88144256, 52.20647181, 68.91540913, 52.62170683, 52.47109192,
83.77061318, 74.18436936, 76.62658002, 73.45927531, 55.75546421,
78.8638047 , 62.25787726, 75.89113465, 54.61323653, 74.47237667,
75.02940846, 75.63225438, 77.56717336, 58.64271914, 62.44547929,
56.51684426, 75.87048378, 58.47682385, 61.68770517, 61.02150161,
73.66777602, 74.04108258, 62.51733859, 55.28951693, 74.45731956,
82.49839066, 75.10146549, 59.62475616, 68.07524638, 73.80776655,
75.39993142, 84.975784  , 79.63739933, 58.56890922, 65.72148536,
59.97986595, 65.34699761, 55.42949979, 74.43729186, 64.52929867,
74.20112643, 76.40480755, 73.56884608, 66.4325756 , 75.63234437,
49.94402317, 78.35572455, 75.56635971, 53.68940066, 56.7727997 ,
70.29728013, 59.47144737, 70.7091457 , 73.77117778, 82.85449518,
70.33371067, 83.11276809, 77.59306734, 53.12017319, 73.98620186,
81.37033683, 56.8705842 , 78.73275471, 55.37351988, 77.70054233,
74.54115855, 73.48728966, 60.80583022, 73.1095988 , 73.21447867,
67.63041222, 72.77846677, 64.8617185 , 73.9088446 , 53.44537787,
65.08866854, 81.81456439, 73.64888641, 52.85019238, 83.93532585,
62.72832676, 55.68610423, 73.08149728, 69.15669485, 68.81852339,
73.91363397, 71.93353922, 52.33052107, 74.38667918, 78.53490027,
75.44395268, 52.53953168, 74.15431184, 62.3885393 , 53.0375126 ,
74.26489692, 74.60669744, 63.71087949, 83.00148141, 72.1822262 ,
56.13049499, 70.25289641, 74.36412438])
```

In [46]:
```
y_test
```

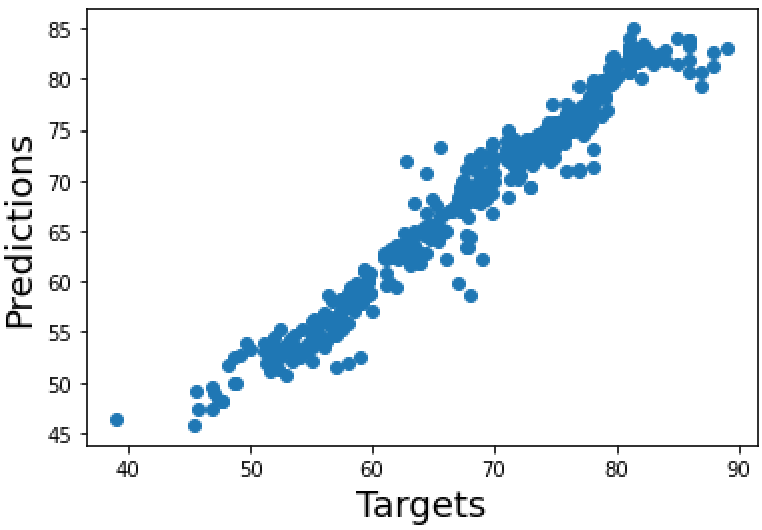
Out[46]:
```
1201    62.5
1628    53.6
1317    83.3
1392    64.3
1308    73.5
        ...
2431    81.6
819     73.3
2493    55.0
260     69.4
352     75.0
Name: Life expectancy, Length: 588, dtype: float64
```

7. Evaluating the Model

In [47]:
```
print("Mean squared error: %.2f"% mean_squared_error(y_test, y_preds))
print("Mean absolute error: %.2f"% mean_absolute_error(y_test, y_preds))
print('R_square score: %.2f' % r2_score(y_test, y_preds))
```

```
Mean squared error: 3.43
Mean absolute error: 1.24
R_square score: 0.96
```

In [48]:
```
plt.scatter(y_test,y_preds)
plt.xlabel('Targets' ,size = 18)
plt.ylabel('Predictions',size = 18)
plt.show()
```



8. Feature Importance

8.1 Find feature importance of our best model

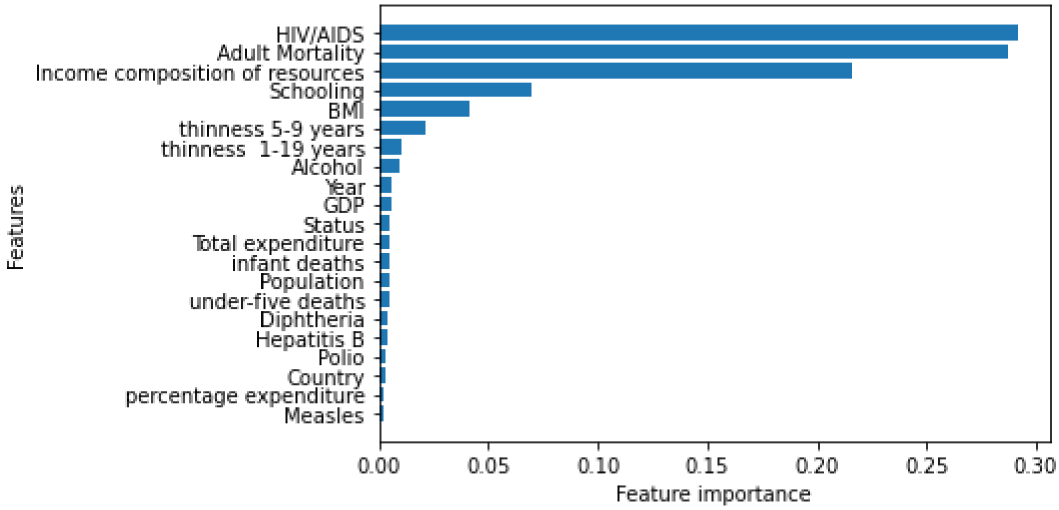
In [49]:
```
ideal_model.feature_importances_
```

Out[49]:
```
array([0.00278575, 0.00537193, 0.00521254, 0.28751652, 0.00494256,
       0.00924993, 0.00194836, 0.00378796, 0.00181467, 0.04146067,
       0.00473988, 0.00302173, 0.00498872, 0.0038887 , 0.29137285,
       0.00536718, 0.00482285, 0.0103422 , 0.02135234, 0.21623411,
       0.06977855])
```

In [50]:
```
# Function for plotting feature importance
def plot_features(columns, importances):
```

```
    df = (pd.DataFrame({"features": columns,
                        "feature_importances": importances})
          .sort_values("feature_importances", ascending=False)
          .reset_index(drop=True))

    # Plot the dataframe
    fig, ax = plt.subplots()
    ax.barh(df["features"], df["feature_importances"])
    ax.set_ylabel("Features")
    ax.set_xlabel("Feature importance")
    ax.invert_yaxis()
```

In [51]:
```
plot_features(X_train.columns, ideal_model.feature_importances_)
```



9. COMPARATIVE ANALYSIS OF LIFE EXPECTANCY

- BETWEEN DEVELOPED AND DEVELOPING COUNTRIES

In [52]:
```
le_df = ldf.copy()

#dropping unwanted columns
le_df.drop(['Year', 'Status'], axis=1, inplace=True)

#renaming columns
le_df.rename(columns={'Life expectancy':'Life Expectancy', 'infant deaths':'Infant
                      'percentage expenditure':'Percentage Expenditure',
                      'under-five deaths':'Under-Five Deaths',
                     'thinness  1-19 years':'Thinness 10-19 years',
                      'thinness 5-9 years':'Thinness 5-9 years'}, inplace=True)


numeric_data = le_df.select_dtypes(include=np.number)
numeric_col = numeric_data.columns
for i in numeric_col:
    mean = le_df[i].mean()
    le_df[i].fillna(mean,inplace = True)
le_df = le_df.groupby('Country').mean()
le_df
```

Out[52]:

Country	Life Expectancy	Adult Mortality	Infant Deaths	Alcohol	Percentage Expenditure	Hepatitis B	Measles	BM
Afghanistan	58.19375	269.0625	78.2500	0.014375	34.960110	64.5625	2362.2500	15.5187
Albania	75.15625	45.0625	0.6875	4.848750	193.259091	98.0000	53.3750	49.0687
Algeria	73.61875	108.1875	20.3125	0.381250	236.185241	58.5000	1943.8750	48.7437
Angola	49.01875	328.5625	83.7500	5.381875	102.100268	39.5000	3561.3125	18.0187
Antigua and Barbuda	75.05625	127.5000	0.0000	7.452500	1001.585226	92.1250	0.0000	38.4250
...	...	...	...	...	...	...	...	
Venezuela (Bolivarian Republic of)	73.38750	163.0000	9.3750	6.956250	0.000000	66.2500	165.0000	54.4875
Viet Nam	74.77500	126.5625	29.1875	2.894375	0.000000	71.1250	4232.9375	11.1875
Yemen	63.86250	211.8125	39.3750	0.044375	0.000000	55.6875	2761.1875	33.4875
Zambia	53.90625	354.3125	33.4375	2.099375	89.650407	48.0000	6563.8125	17.4500
Zimbabwe	50.48750	462.3750	26.5625	4.201875	20.364271	70.5625	923.0000	25.1375

193 rows × 19 columns

9.1 Splitting into dependant & independant variables

In [53]:

```
life = le_df['Life Expectancy']
features = le_df.drop(['Life Expectancy'], axis=1)
```

9.2 Plot Graph

In [54]:

```
countries = ['Brazil', 'Russian Federation', 'South Africa', 'China', 'India', 'Un
life_exp = []
hiv = []
adult_mortality = []
percent_expenditure = []
for i in countries:
    life_exp.append(life.loc[i])
    percent_expenditure.append(features.loc[i]['Percentage Expenditure'])
    hiv.append(features.loc[i]['HIV/AIDS'])
    adult_mortality.append(features.loc[i]['Adult Mortality'])
```

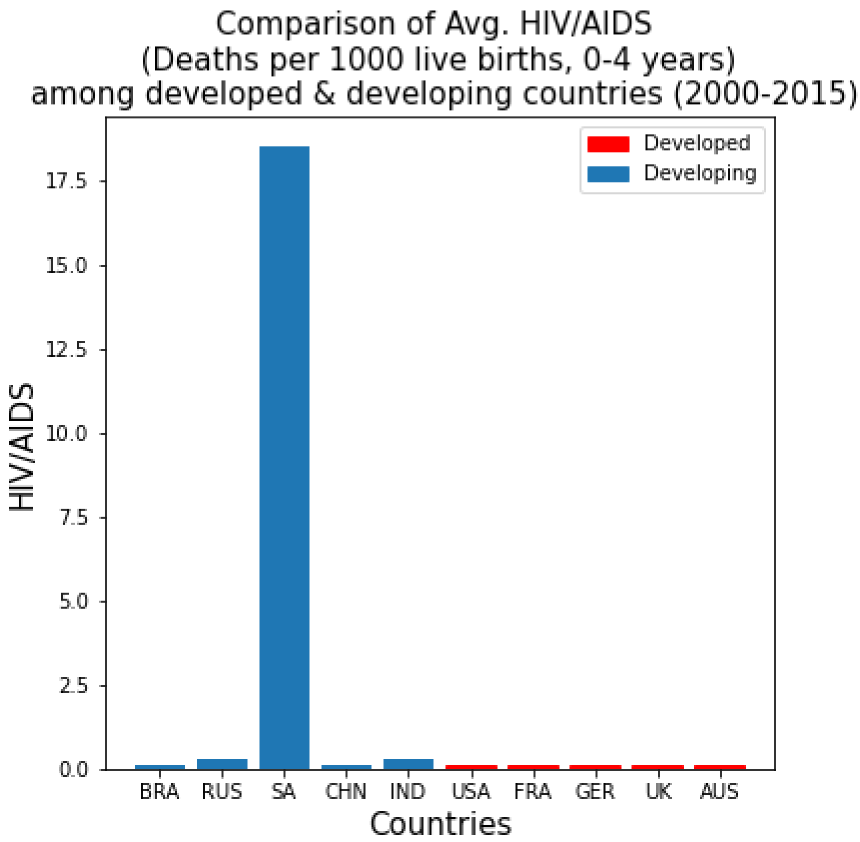
9.2.1 HIV Comparison

In [55]:

```
#HIV comparison plot
countries1 = ['BRA', 'RUS', 'SA', 'CHN', 'IND', 'USA', 'FRA', 'GER', 'UK', 'AUS']
fig = plt.figure(figsize = (6,6))
```

```
colors = {'Developed':'red', 'Developing':'#1f77b4'}
labels = list(colors.keys())
barlist = plt.bar(countries1, hiv)
barlist[5].set_color('r')
barlist[6].set_color('r')
barlist[7].set_color('r')
barlist[8].set_color('r')
barlist[9].set_color('r')
plt.title('Comparison of Avg. HIV/AIDS \n (Deaths per 1000 live births, 0-4 years)
plt.xlabel('Countries', fontsize = 15)
plt.ylabel('HIV/AIDS', fontsize = 15)
handles = [plt.Rectangle((0,0),1,1, color = colors[label]) for label in labels]
plt.legend(handles, labels)
```

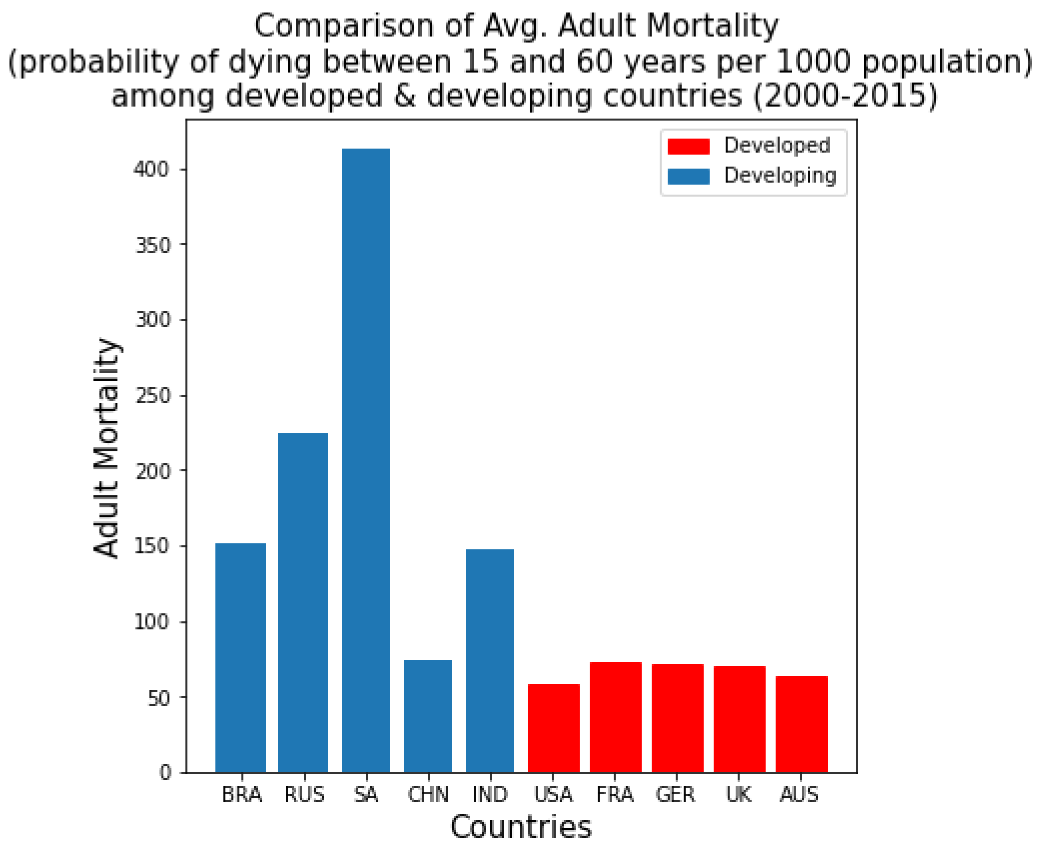
Out[55]: <matplotlib.legend.Legend at 0x23cdca88040>



9.2.2 Adult Mortality comparison

In [56]:
```
#Adult Mortality comparison plot
countries1 = ['BRA', 'RUS', 'SA', 'CHN', 'IND', 'USA', 'FRA', 'GER', 'UK', 'AUS']
fig = plt.figure(figsize = (6,6))
colors = {'Developed':'red', 'Developing':'#1f77b4'}
labels = list(colors.keys())
barlist = plt.bar(countries1, adult_mortality)
barlist[5].set_color('r')
barlist[6].set_color('r')
barlist[7].set_color('r')
barlist[8].set_color('r')
barlist[9].set_color('r')
plt.title('Comparison of Avg. Adult Mortality \n (probability of dying between 15 a
plt.xlabel('Countries', fontsize = 15)
plt.ylabel('Adult Mortality', fontsize = 15)
handles = [plt.Rectangle((0,0),1,1, color = colors[label]) for label in labels]
plt.legend(handles, labels)
```

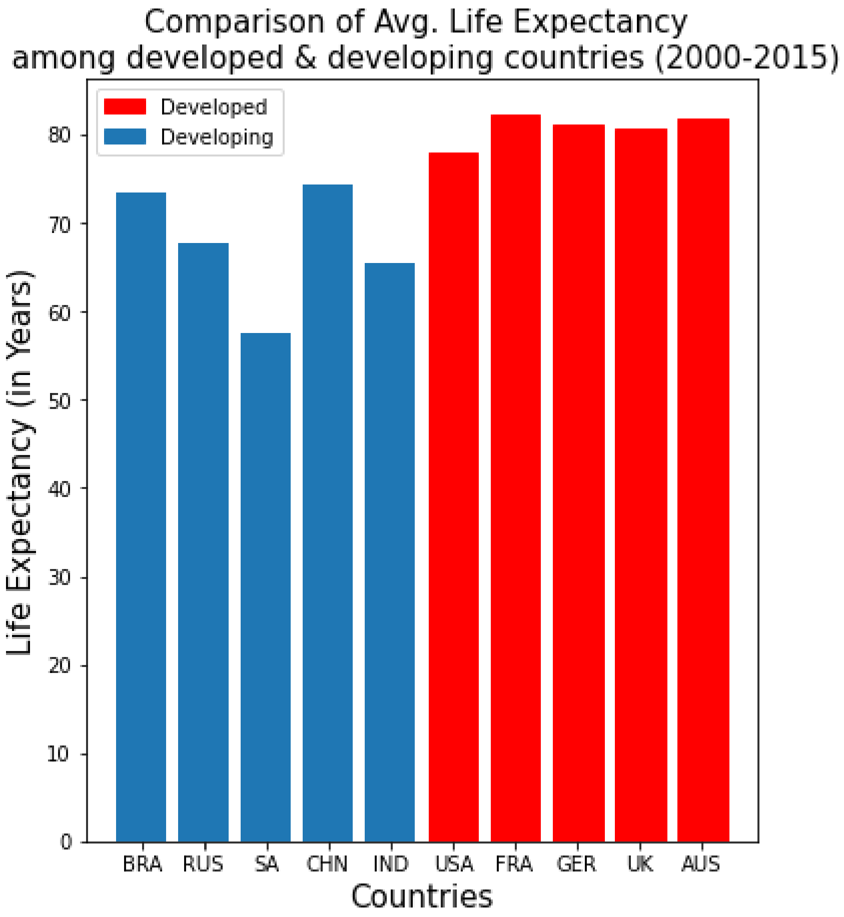
Out[56]: <matplotlib.legend.Legend at 0x23cdcac9b20>



9.2.3 Life Expectancy comparison

In [57]:
```
#Life Expectancy comparison plot
countries1 = ['BRA', 'RUS', 'SA', 'CHN', 'IND', 'USA', 'FRA', 'GER', 'UK', 'AUS']
fig = plt.figure(figsize = (6,7))
colors = {'Developed':'red', 'Developing':'#1f77b4'}
labels = list(colors.keys())
barlist = plt.bar(countries1, life_exp)
barlist[5].set_color('r')
barlist[6].set_color('r')
barlist[7].set_color('r')
barlist[8].set_color('r')
barlist[9].set_color('r')
plt.title('Comparison of Avg. Life Expectancy \n among developed & developing coun
plt.xlabel('Countries', fontsize = 15)
plt.ylabel('Life Expectancy (in Years)', fontsize = 15)
handles = [plt.Rectangle((0,0),1,1, color = colors[label]) for label in labels]
plt.legend(handles, labels)
```

Out[57]: <matplotlib.legend.Legend at 0x23cdc90c4c0>



CONLUSION

The prediction model is trained using three regression models, namely Linear Regression, Decision Tree Regressor and Random Forest Regressor. The selection of model is done on the basis of R 2 score, Mean Squared Error & Mean Absolute Error. Random Forest Regressor is selected for the development of the prediction model for life expectancy and the comparative analysis of life expectancy between developed and developing countries suggests that, developed countries have high life expectancy as compared to developing countries.