Capstone 2: Life Expectancy Predictions

This capstone will complete all of the steps of the Data Science Method.

1) Problem identification
2) Data wrangling
3) Exploratory data analysis
4) Pre-processing and training data
5) Modelling
6) Documentation

1) Problem identification

I will be using the Kaggle dataset "WHO methods and data sources for life tables 1990-2019". Information about this dataset can be found at https://cdn.who.int/media/docs/default-source/gho-documents/global-health-estimates/ghe2019_life-table-methods.pdf?sfvrsn=c433c229_5 and the dataset is at https://www.kaggle.com/code/vishesh1412/life-expectancy-prediction-feature-selection-ols.

I posed the question, how do certain elements of countries data affect life expectancy?

There are many recorded variables of human life around the world. The World Health Organization records these things annually, with variables like immunization factors, mortality factors, economic factors, social factors and health related factors. I wondered how knowing and adjusting these features impacts a country's life expectancy, and if this could be modeled through machine learning. I think it is useful to know correlations between factors like these and life expectancy when thinking about how to live your own life, and just how the world is at large. A model for this could answer questions like what is the strongest predicting factor contributing to a high, low, or medium life expectancy? It can also answer which kinds of countries are best positioned for a high life expectancy and what elements lower life expectancy countries can focus on to improve theirs.

2) Data wrangling

In this step, I needed to get the data in a format I could work with -- The data downloaded from Kaggle needed some changes before I could really look at it. The columns didn't have a consistent format, for example, in terms of capitalization and leading and trailing spaces.

In [1]:
```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import missingno as msno
```

In [2]:
```
df= pd.read_csv('Life_Expectancy_Data.csv')
pd.set_option('display.max_columns', 100)
pd.set_option('display.max_rows', 100)
df.head()
```

Out[2]:

	Country	Year	Status	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B
0	Afghanistan	2015	Developing	65.0	263.0	62	0.01	71.279624	65.0
1	Afghanistan	2014	Developing	59.9	271.0	64	0.01	73.523582	62.0
2	Afghanistan	2013	Developing	59.9	268.0	66	0.01	73.219243	64.0
3	Afghanistan	2012	Developing	59.5	272.0	69	0.01	78.184215	67.0
4	Afghanistan	2011	Developing	59.2	275.0	71	0.01	7.097109	68.0

In [3]:
```
df.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                  2928 non-null   float64
 4   Adult Mortality                  2928 non-null   float64
 5   infant deaths                    2938 non-null   int64  
 6   Alcohol                          2744 non-null   float64
 7   percentage expenditure           2938 non-null   float64
 8   Hepatitis B                      2385 non-null   float64
 9   Measles                          2938 non-null   int64  
 10   BMI                             2904 non-null   float64
 11  under-five deaths                2938 non-null   int64  
 12  Polio                            2919 non-null   float64
 13  Total expenditure                2712 non-null   float64
 14  Diphtheria                       2919 non-null   float64
 15   HIV/AIDS                        2938 non-null   float64
 16  GDP                              2490 non-null   float64
 17  Population                       2286 non-null   float64
 18   thinness  1-19 years            2904 non-null   float64
 19   thinness 5-9 years             2904 non-null   float64
 20  Income composition of resources  2771 non-null   float64
 21  Schooling                        2775 non-null   float64
dtypes: float64(16), int64(4), object(2)
memory usage: 505.1+ KB
```

In [4]:
```
#Looking at leading and trailing spaces on column names
[x for x in df.columns if x.endswith(' ') or x.startswith(' ')]
```

Out[4]:
```
['Life expectancy ',
 'Measles ',
 ' BMI ',
 'under-five deaths ',
 'Diphtheria ',
 ' HIV/AIDS',
 ' thinness  1-19 years',
 ' thinness 5-9 years']
```

In [5]:
```
df.columns = df.columns.str.strip()
df.columns = df.columns.str.lower()
df.columns
```

Out[5]:
```
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 [6]:
```
df.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                  2928 non-null   float64
 4   adult mortality                  2928 non-null   float64
 5   infant deaths                    2938 non-null   int64  
 6   alcohol                          2744 non-null   float64
 7   percentage expenditure           2938 non-null   float64
 8   hepatitis b                      2385 non-null   float64
 9   measles                          2938 non-null   int64  
 10  bmi                              2904 non-null   float64
 11  under-five deaths                2938 non-null   int64  
 12  polio                            2919 non-null   float64
 13  total expenditure                2712 non-null   float64
 14  diphtheria                       2919 non-null   float64
 15  hiv/aids                         2938 non-null   float64
 16  gdp                              2490 non-null   float64
 17  population                       2286 non-null   float64
 18  thinness  1-19 years             2904 non-null   float64
 19  thinness 5-9 years               2904 non-null   float64
 20  income composition of resources  2771 non-null   float64
 21  schooling                        2775 non-null   float64
dtypes: float64(16), int64(4), object(2)
memory usage: 505.1+ KB
```

Our column of interest is 3, life expectancy. Everything is numeric except for country and status (developing, etc). We have the column names normalized with no leading or trailing spaces or inconsistent capitalization.

In [7]:
```
df.sort_values(by='country')
missing = pd.concat([df.isnull().sum(), 100 * df.isnull().mean()], axis=1)
```

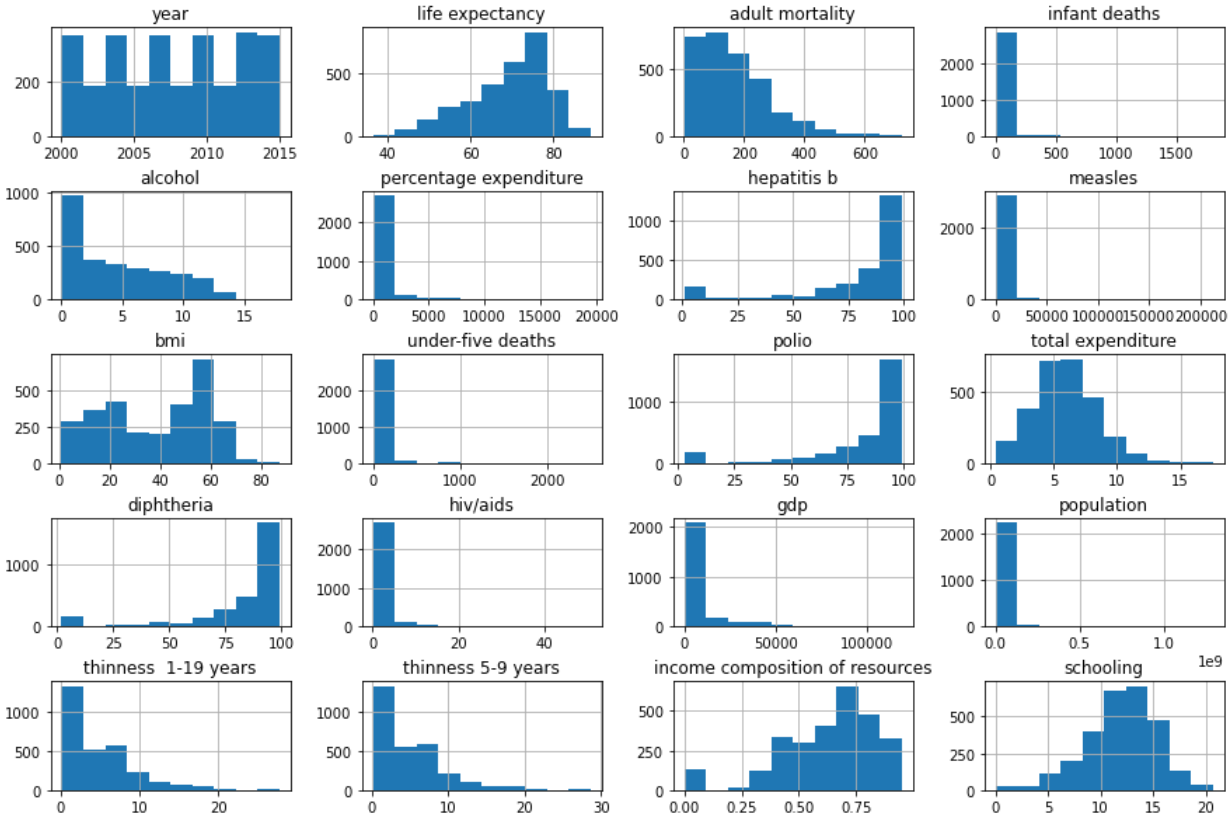
```
missing.columns=['count', '%']
missing.sort_values(by='count')
```

Out[7]:

	count	%
country	0	0.000000
year	0	0.000000
status	0	0.000000
infant deaths	0	0.000000
percentage expenditure	0	0.000000
measles	0	0.000000
hiv/aids	0	0.000000
under-five deaths	0	0.000000
life expectancy	10	0.340368
adult mortality	10	0.340368
polio	19	0.646698
diphtheria	19	0.646698
thinness 5-9 years	34	1.157250
thinness 1-19 years	34	1.157250
bmi	34	1.157250
schooling	163	5.547992
income composition of resources	167	5.684139
alcohol	194	6.603131
total expenditure	226	7.692308
gdp	448	15.248468
hepatitis b	553	18.822328
population	652	22.191967

There are obviously some missing values, but before dealing with them, let's look at some preliminary visualization of the variables.

In [8]:
```
df.hist(figsize=(15,10))
plt.subplots_adjust(hspace=0.5);
```



Some are clustered towards zero, but maybe that's because of the scale/max being too high or an outlier.

In [9]:
```
corr = df.corr()
fig, ax = plt.subplots(figsize=(10,10))
sns.heatmap(corr, linewidths=.5, ax=ax)
```

Out[9]:
```
<AxesSubplot:>
```

I see that infant deaths and under-five deaths are correlated, which makes sense. As well as thinness for overlapping age groups and GDP and percentage expenditure. BMI is inversely correlated with thinness. Mortality and AIDS are inversely correlated with Life Expectancy. All to be expected. Maybe doing PCA later will be helpful because there seems to be a lot of intervariable high correlation.

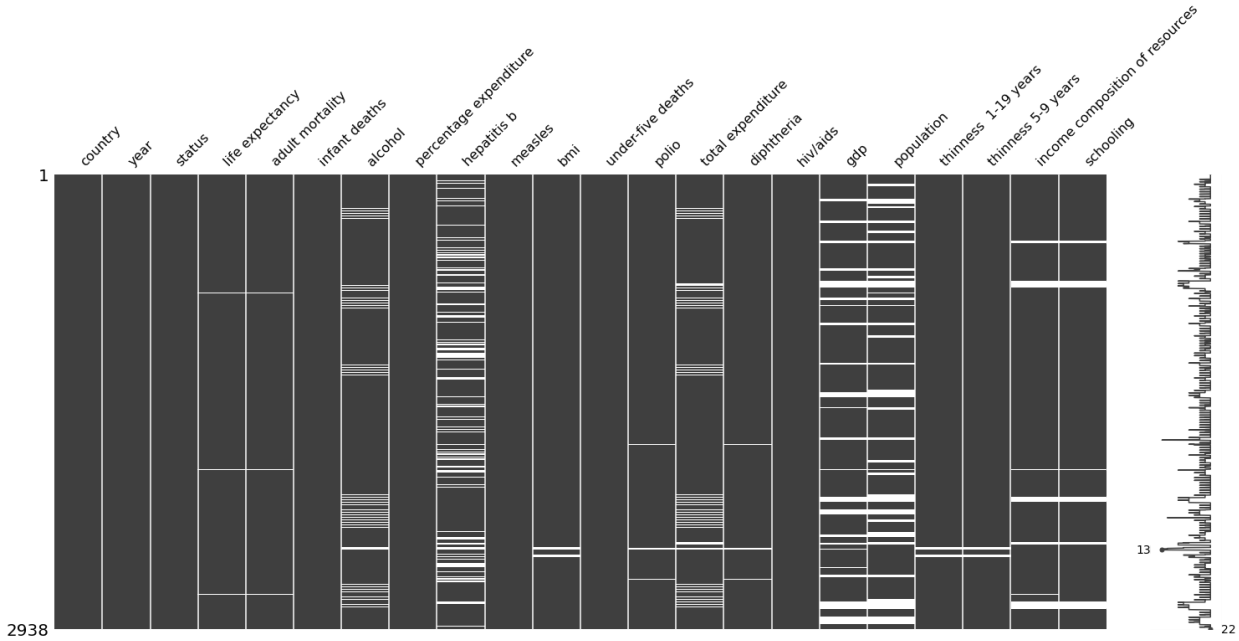
In [10]:
```
df.isna()
```

Out[10]:

	country	year	status	life expectancy	adult mortality	infant deaths	alcohol	percentage expenditure	hepatitis b	m
0	False	False	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	...	...	
2933	False	False	False	False	False	False	False	False	False	
2934	False	False	False	False	False	False	False	False	False	
2935	False	False	False	False	False	False	False	False	False	
2936	False	False	False	False	False	False	False	False	False	
2937	False	False	False	False	False	False	False	False	False	

2938 rows × 22 columns

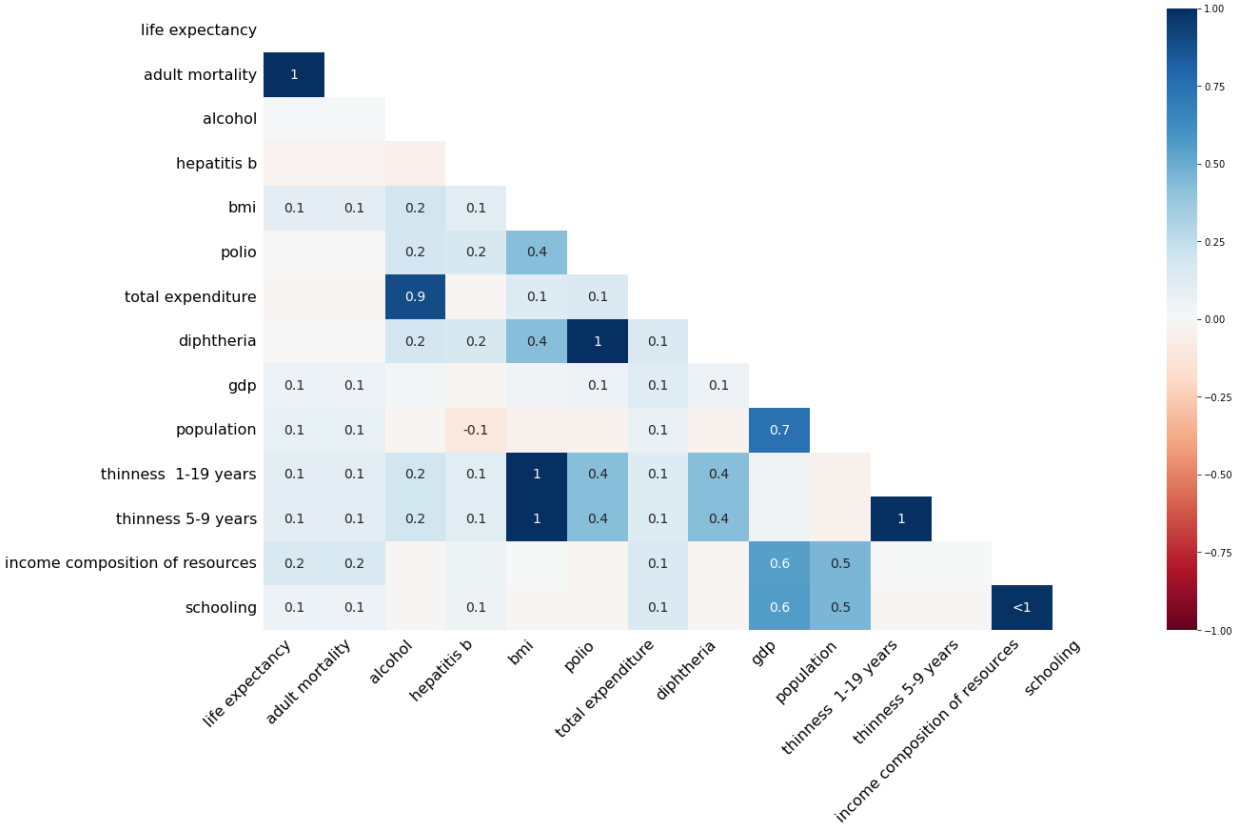
In [11]:
```
msno.matrix(df)
plt.show()
```



I took a look at the missing values, and a heatmap of the missing values, before dropping the rows with missing values, because there were relatively few.

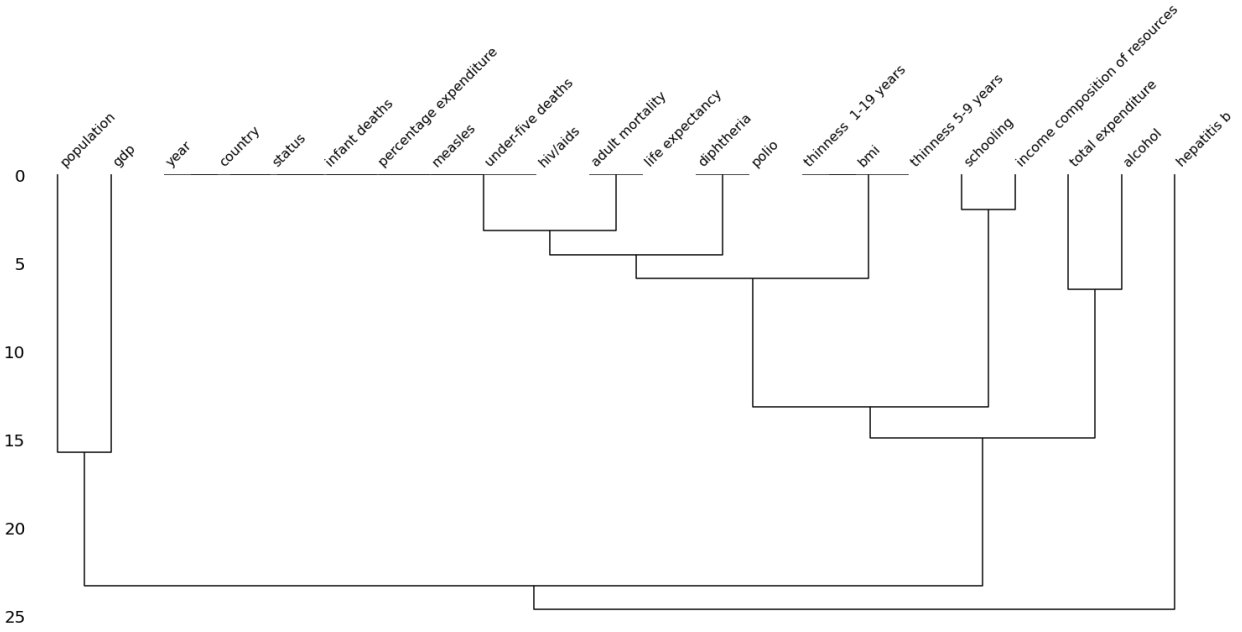
In [12]:
```
msno.heatmap(df)
```

Out[12]:
```
<AxesSubplot:>
```



In [13]:
```
msno.dendrogram(df)
```

Out[13]:
```
<AxesSubplot:>
```



Some of the missing values are correlated, but in an expected way, like thinness and BMI, if one is missing it makes sense for another to be missing, and probably had to do with data collection. You couldn't find GDP without population, similarly. Besides that, the missing values are distributed pretty normally, doesn't have a very clear pattern in the missingness.

With this dataset, it doesn't make sense to fill in population with an overall average because

countries are all different. Same with other features, inputting a average of all 'alcohol' for one country would not be the best; I think if I'm to fill in with a mean, it will be the mean of that feature only for that country.

However, overall, the missing data is not that significant compared with all of the data, and we can drop all of the missing data rows completely.

In [14]:
```
df = df.dropna()
```

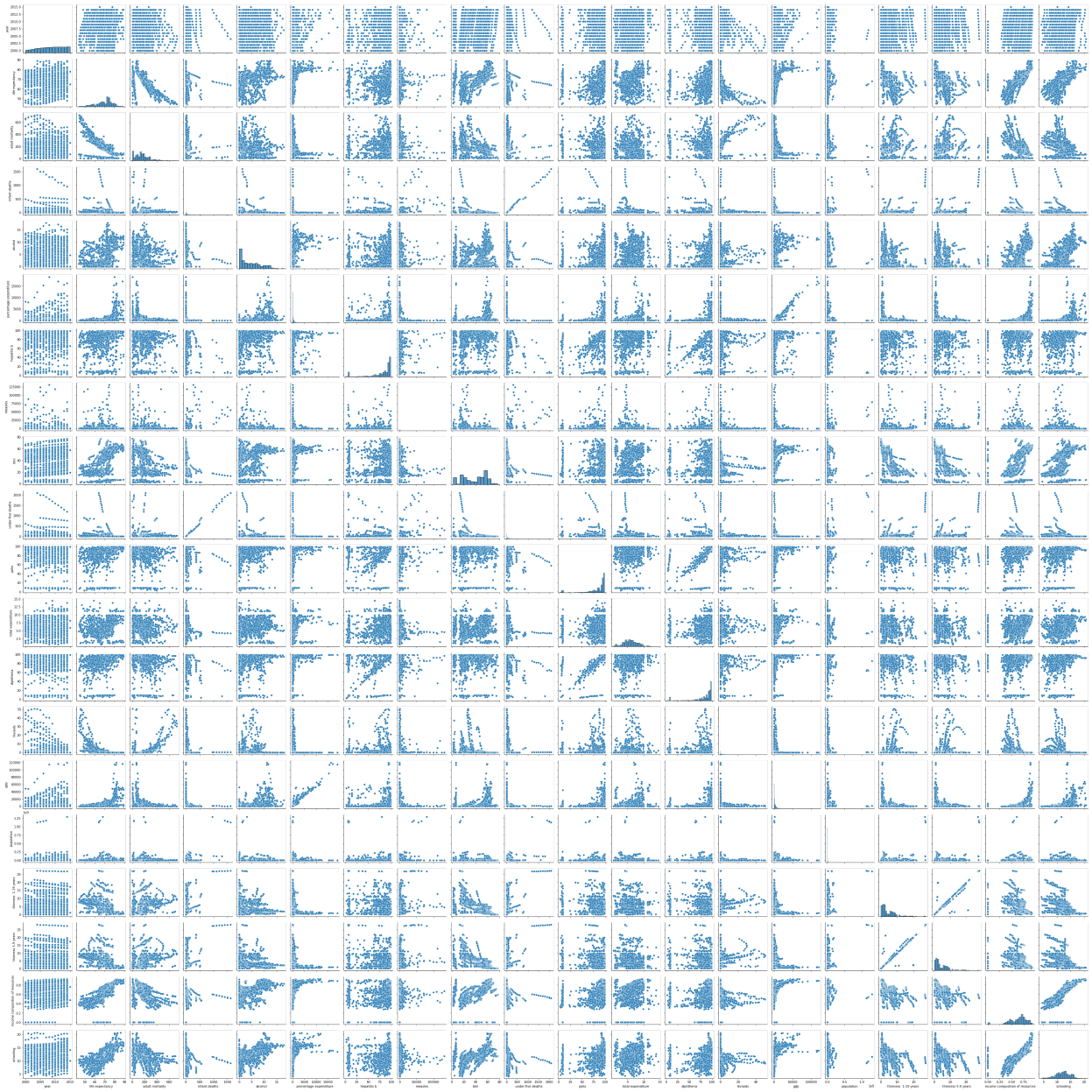
In [15]:
```
df.shape
```

Out[15]:
```
(1649, 22)
```

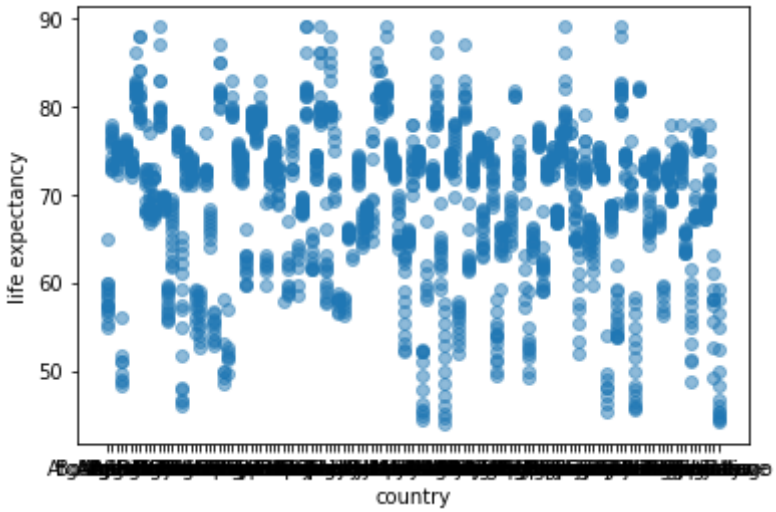
3) Exploratory data analysis

I asked, What relationships exist between my features? Let's take a look at each feature's relationship with each other in more detail than just correlation, and with our target feature. One useful visual in my EDA step was a pairplot, showing the relationship between every variable and every other variable.

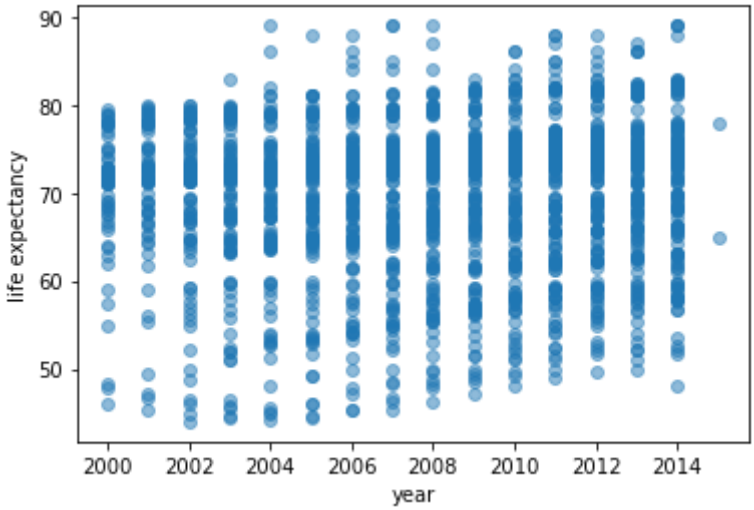
In [16]:
```
sns.pairplot(df,  palette='Set1')
plt.show()
```



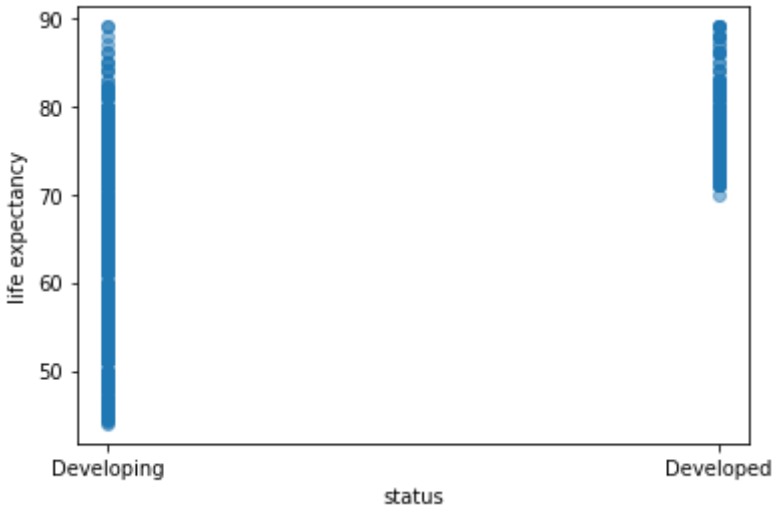
In [17]:
```
plt.scatter(df['country'], df['life expectancy'], alpha=0.5)
plt.xlabel('country')
plt.ylabel('life expectancy')
plt.show()
```



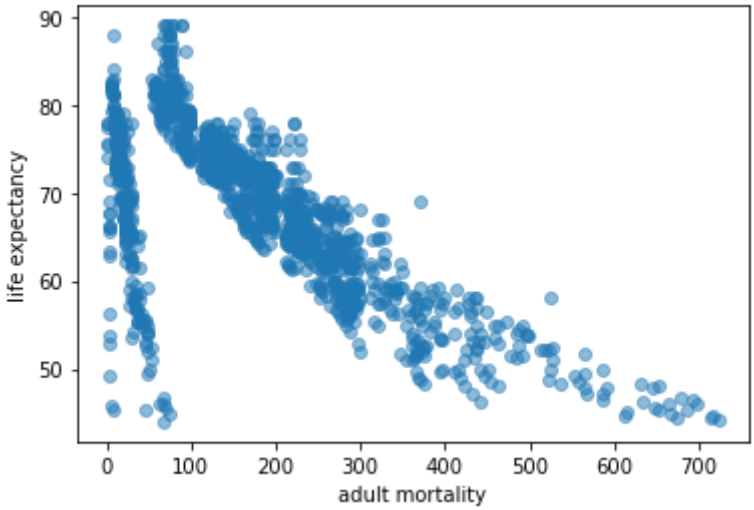
In [18]:
```
plt.scatter(df['year'], df['life expectancy'], alpha=0.5)
plt.xlabel('year')
plt.ylabel('life expectancy')
plt.show()
```



In [19]:
```
plt.scatter(df['status'], df['life expectancy'], alpha=0.5)
plt.xlabel('status')
plt.ylabel('life expectancy')
plt.show()
```



In [20]:
```
plt.scatter(df['adult mortality'], df['life expectancy'], alpha=0.5)
plt.xlabel('adult mortality')
plt.ylabel('life expectancy')
plt.show()
```



I want to check out the part with low adult mortality.

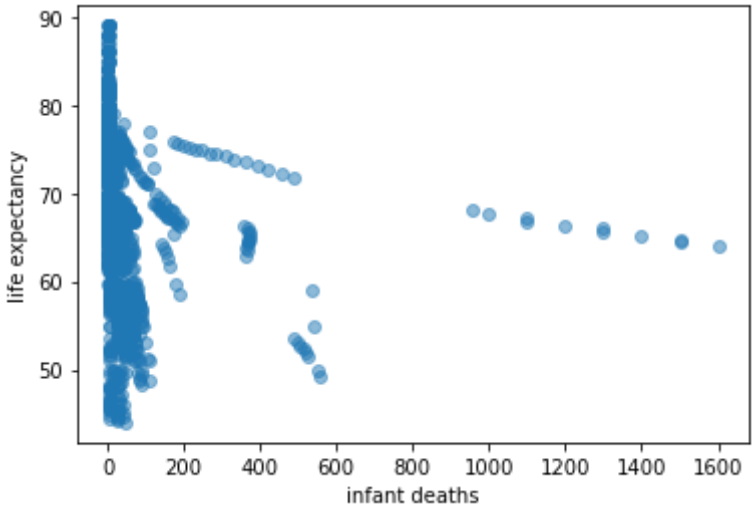
In [21]:
```
df[df['adult mortality']<10]
```

Out[21]:

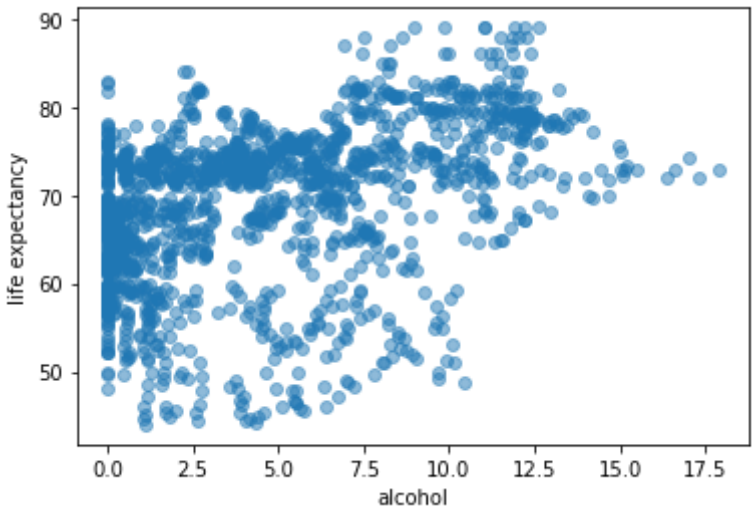
	country	year	status	life expectancy	adult mortality	infant deaths	alcohol	percentage expenditure	he
13	Afghanistan	2002	Developing	56.2	3.0	88	0.01	16.887351	
17	Albania	2014	Developing	77.5	8.0	0	4.51	428.749067	
23	Albania	2008	Developing	75.3	1.0	1	5.61	36.622068	
24	Albania	2007	Developing	75.9	9.0	1	5.58	32.246552	
113	Australia	2014	Developed	82.7	6.0	1	9.71	10769.363050	
131	Austria	2012	Developed	88.0	7.0	0	12.26	7878.372355	
136	Austria	2007	Developed	81.0	8.0	0	12.50	7453.864400	
141	Austria	2002	Developed	78.7	9.0	0	12.50	3979.057728	
244	Belgium	2011	Developed	83.0	8.0	0	10.11	713.529735	
254	Belgium	2001	Developed	78.0	1.0	1	11.01	2943.375396	
322	Bosnia and Herzegovina	2013	Developing	77.0	9.0	0	4.12	75.610629	
329	Bosnia and Herzegovina	2006	Developing	75.7	1.0	0	4.82	38.254141	
339	Botswana	2012	Developing	63.4	3.0	2	0.01	12.834474	
490	Cameroon	2005	Developing	52.8	4.0	58	5.03	73.032762	
501	Canada	2010	Developing	81.2	7.0	2	8.40	8649.674911	
552	Chile	2007	Developing	78.9	9.0	2	7.39	209.487587	
637	Costa Rica	2003	Developing	78.0	1.0	1	4.04	1070.268999	
639	Costa Rica	2001	Developing	77.5	1.0	1	4.29	868.371359	
679	Cyprus	2009	Developed	79.3	6.0	0	10.80	230.252418	
688	Cyprus	2000	Developed	78.1	7.0	0	9.56	950.802793	
903	Fiji	2010	Developing	69.1	2.0	0	2.25	62.083435	
1001	Germany	2008	Developed	79.9	8.0	2	11.36	8285.264655	
1006	Germany	2003	Developed	78.5	9.0	3	11.92	582.615703	
1034	Greece	2007	Developing	79.4	8.0	0	9.67	3632.243121	
1192	India	2009	Developing	66.0	2.0	1300	2.50	0.844186	
1254	Ireland	2011	Developed	84.0	7.0	0	11.72	6386.954370	
1267	Israel	2014	Developing	82.2	6.0	1	2.62	4348.335310	
1269	Israel	2012	Developing	81.8	6.0	1	2.78	3830.184587	
1285	Italy	2012	Developed	82.0	6.0	2	7.49	4793.904924	
1286	Italy	2011	Developed	82.0	6.0	2	6.98	5439.691769	
1287	Italy	2010	Developed	81.8	6.0	2	6.95	5219.668802	
1379	Kiribati	2014	Developing	66.1	2.0	0	0.01	97.871933	

	country	year	status	life expectancy	adult mortality	infant deaths	alcohol	percentage expenditure	he
1467	Lebanon	2006	Developing	74.1	1.0	1	1.82	534.579570	
1537	Lithuania	2000	Developed	71.6	2.0	0	9.87	373.260553	
1541	Luxembourg	2012	Developed	81.1	7.0	0	11.34	2284.582150	
1545	Luxembourg	2008	Developed	80.0	8.0	0	11.53	18961.348600	
1548	Luxembourg	2005	Developed	78.8	9.0	0	11.84	1346.246697	
1770	Mozambique	2009	Developing	53.8	4.0	70	1.18	39.752169	
1831	Netherlands	2013	Developed	81.4	6.0	1	8.68	1475.030397	
1903	Nigeria	2005	Developing	49.2	4.0	556	9.71	6.416253	
1952	Pakistan	2005	Developing	62.9	2.0	364	0.04	30.593208	
2061	Portugal	2009	Developed	79.3	9.0	0	12.03	337.102352	
2144	Russian Federation	2006	Developing	66.4	3.0	17	11.79	12.251841	
2252	Senegal	2012	Developing	65.6	2.0	20	0.28	10.206595	
2369	Solomon Islands	2007	Developing	67.6	2.0	0	0.85	28.901352	
2370	Solomon Islands	2006	Developing	67.6	2.0	0	0.99	240.485120	
2427	Spain	2013	Developed	82.4	6.0	1	9.25	423.680459	
2432	Spain	2008	Developed	81.3	7.0	2	10.24	5596.535203	
2501	Swaziland	2003	Developing	45.9	6.0	3	5.65	2.819124	
2931	Zimbabwe	2006	Developing	45.4	7.0	28	4.57	34.262169	

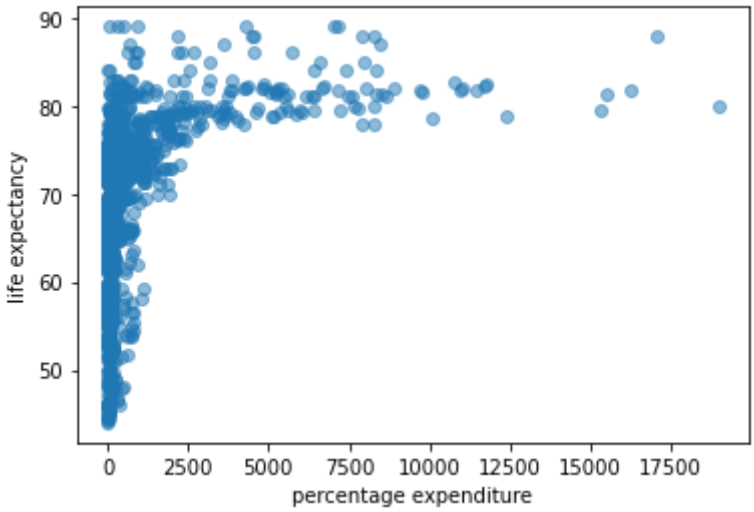
In [22]:
```
plt.scatter(df['infant deaths'], df['life expectancy'], alpha=0.5)
plt.xlabel('infant deaths')
plt.ylabel('life expectancy')
plt.show()
```



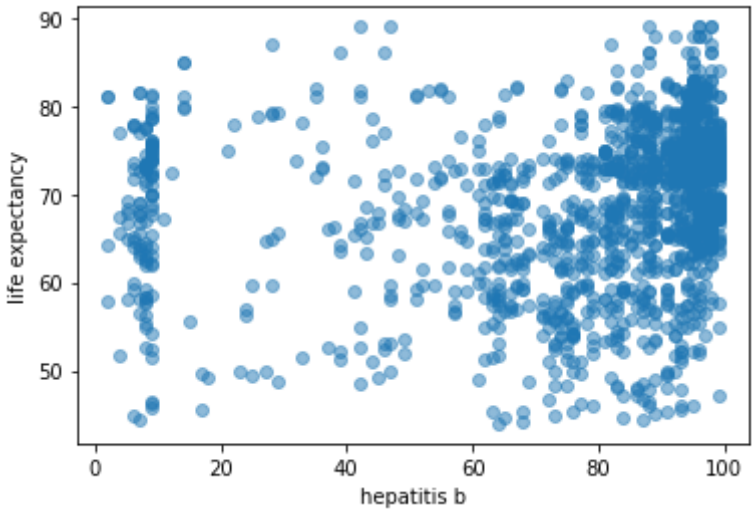
In [23]:
```
plt.scatter(df['alcohol'], df['life expectancy'], alpha=0.5)
plt.xlabel('alcohol')
plt.ylabel('life expectancy')
plt.show()
```



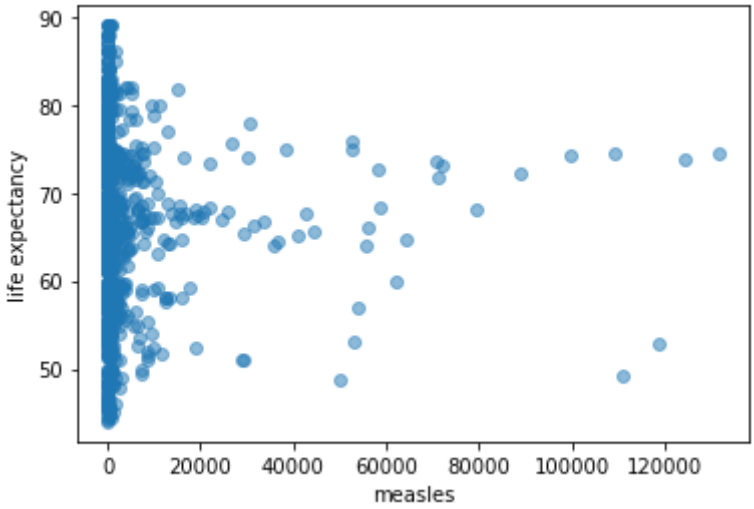
In [24]:
```
plt.scatter(df['percentage expenditure'], df['life expectancy'], alpha=0.5)
plt.xlabel('percentage expenditure')
plt.ylabel('life expectancy')
plt.show()
```



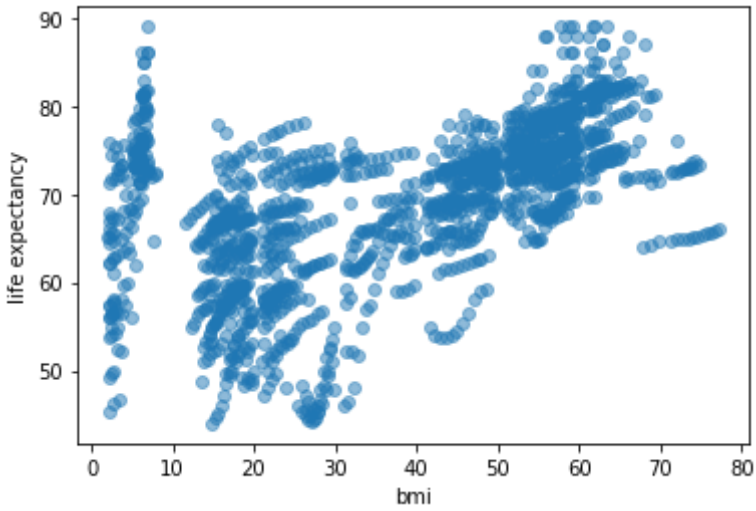
In [25]:
```
plt.scatter(df['hepatitis b'], df['life expectancy'], alpha=0.5)
plt.xlabel('hepatitis b')
plt.ylabel('life expectancy')
plt.show()
```



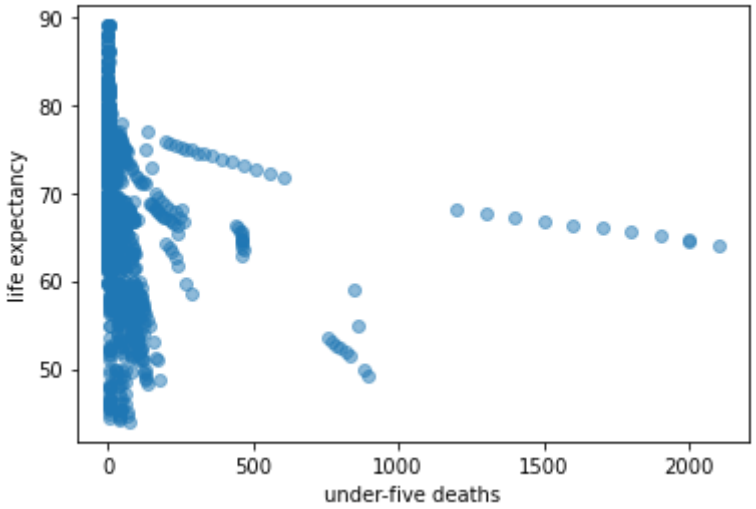
In [26]:
```
plt.scatter(df['measles'], df['life expectancy'], alpha=0.5)
plt.xlabel('measles')
plt.ylabel('life expectancy')
plt.show()
```



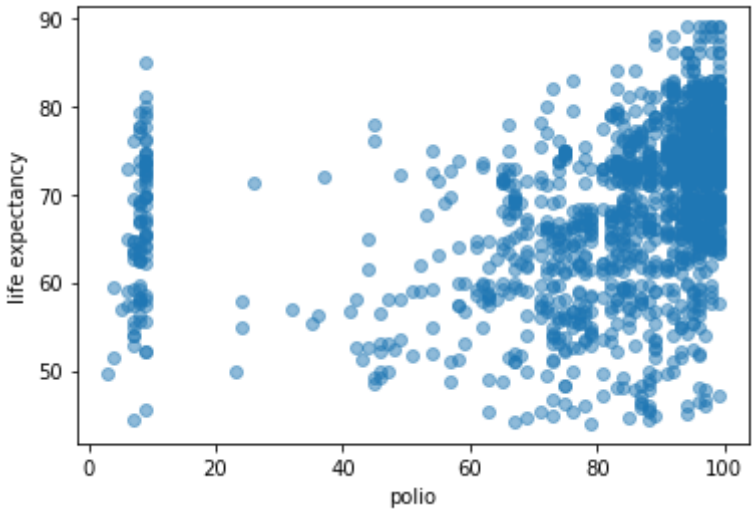
In [27]:
```
plt.scatter(df['bmi'], df['life expectancy'], alpha=0.5)
plt.xlabel('bmi')
plt.ylabel('life expectancy')
plt.show()
```



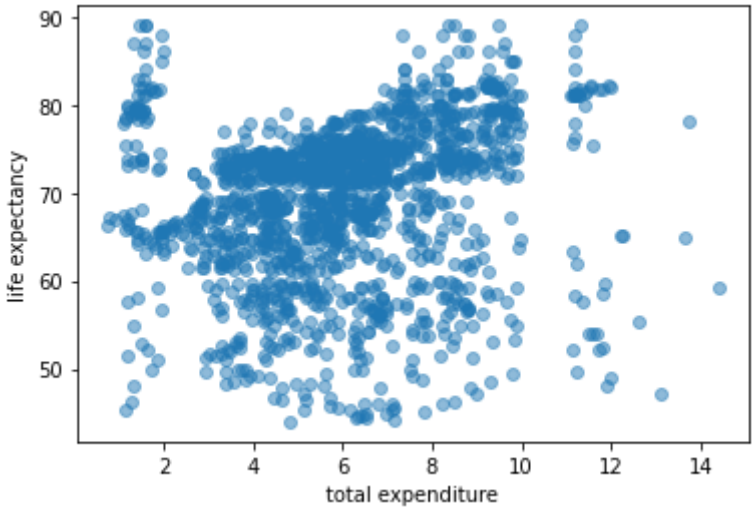
In [28]:
```
plt.scatter(df['under-five deaths'], df['life expectancy'], alpha=0.5)
plt.xlabel('under-five deaths')
plt.ylabel('life expectancy')
plt.show()
```



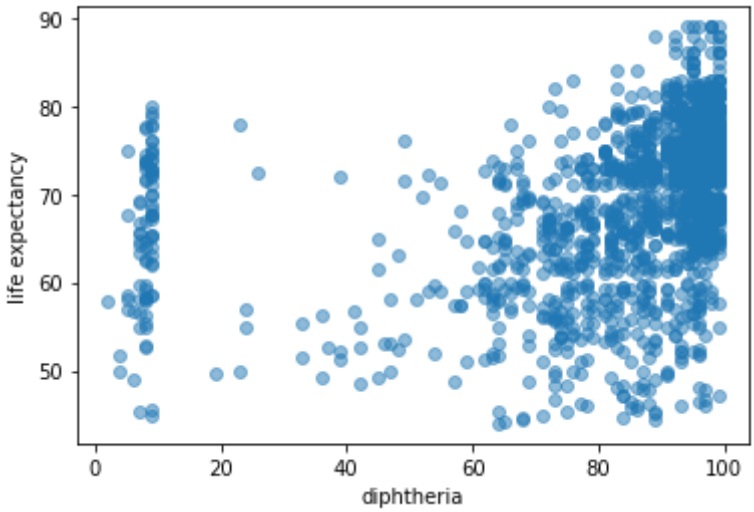
In [29]:
```
plt.scatter(df['polio'], df['life expectancy'], alpha=0.5)
plt.xlabel('polio')
plt.ylabel('life expectancy')
plt.show()
```



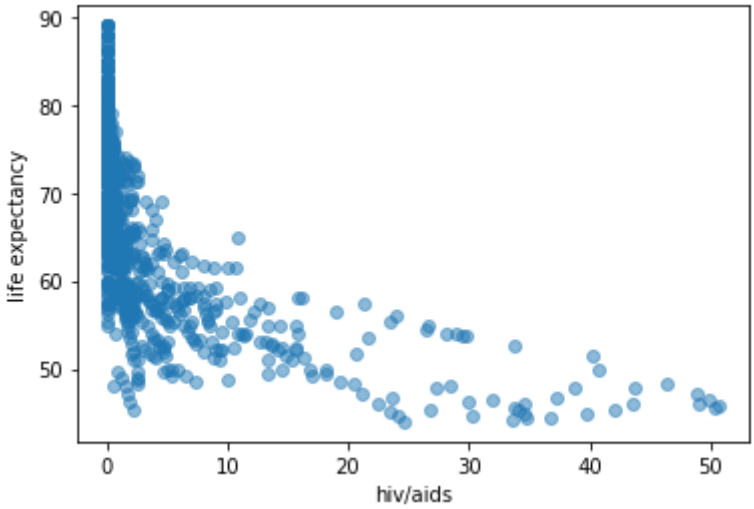
In [30]:
```
plt.scatter(df['total expenditure'], df['life expectancy'], alpha=0.5)
plt.xlabel('total expenditure')
plt.ylabel('life expectancy')
plt.show()
```



In [31]:
```
plt.scatter(df['diphtheria'], df['life expectancy'], alpha=0.5)
plt.xlabel('diphtheria')
plt.ylabel('life expectancy')
plt.show()
```

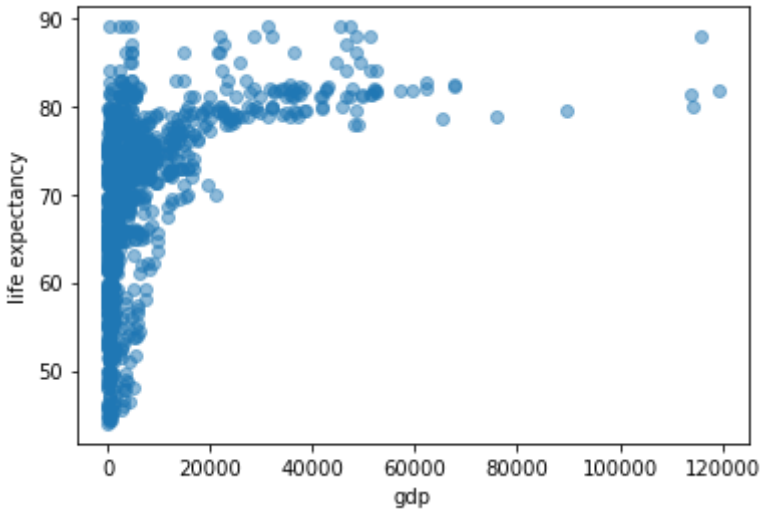


In [32]:
```
plt.scatter(df['hiv/aids'], df['life expectancy'], alpha=0.5)
plt.xlabel('hiv/aids')
plt.ylabel('life expectancy')
plt.show()
```

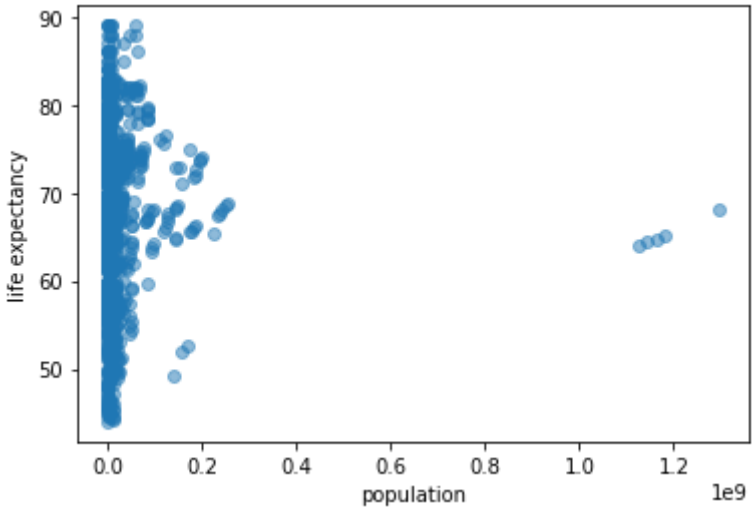


Here, hiv/aids has a clear negative correlation with life expectancy.

In [33]:
```
plt.scatter(df['gdp'], df['life expectancy'], alpha=0.5)
plt.xlabel('gdp')
plt.ylabel('life expectancy')
plt.show()
```



In [34]:
```
plt.scatter(df['population'], df['life expectancy'], alpha=0.5)
plt.xlabel('population')
plt.ylabel('life expectancy')
plt.show()
```



In [35]:
```
df[df['population']>1000000000]
```

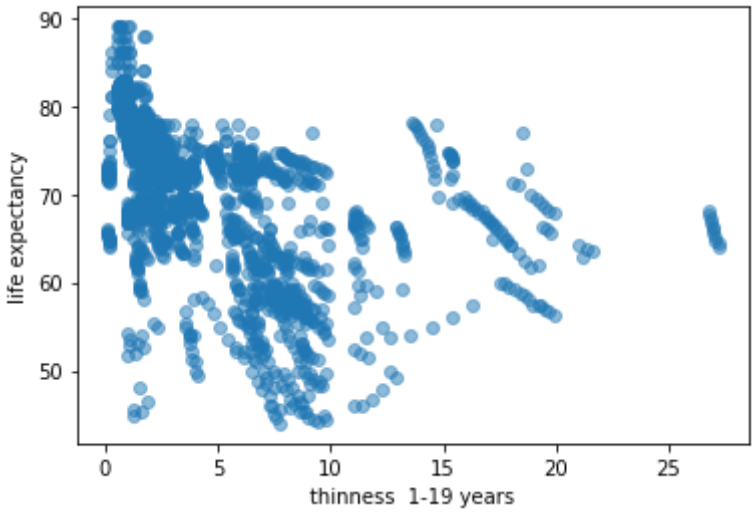
Out[35]:

	country	year	status	life expectancy	adult mortality	infant deaths	alcohol	percentage expenditure	hepatitis b
1187	India	2014	Developing	68.0	184.0	957	3.07	86.521539	79.0
1194	India	2007	Developing	65.2	26.0	1400	1.59	5.234770	6.0
1195	India	2006	Developing	64.8	28.0	1500	1.37	34.859427	6.0
1196	India	2005	Developing	64.4	211.0	1500	1.27	3.509637	8.0
1197	India	2004	Developing	64.0	214.0	1600	1.20	27.338009	6.0

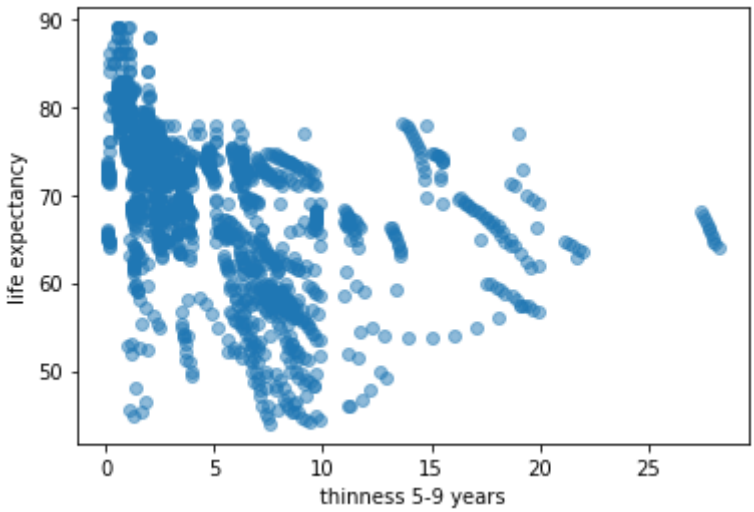
India is an outlier here because of their high population.

In [36]:
```
plt.scatter(df['thinness  1-19 years'], df['life expectancy'], alpha=0.5)
plt.xlabel('thinness  1-19 years')
```

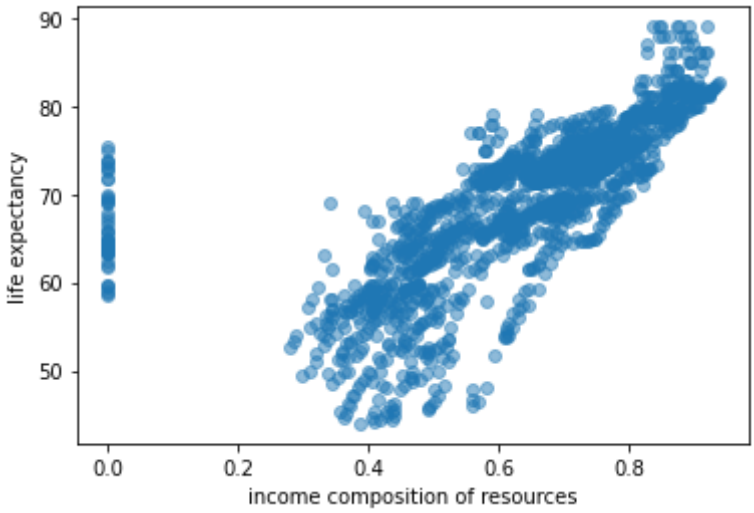
```
plt.ylabel('life expectancy')
plt.show()
```



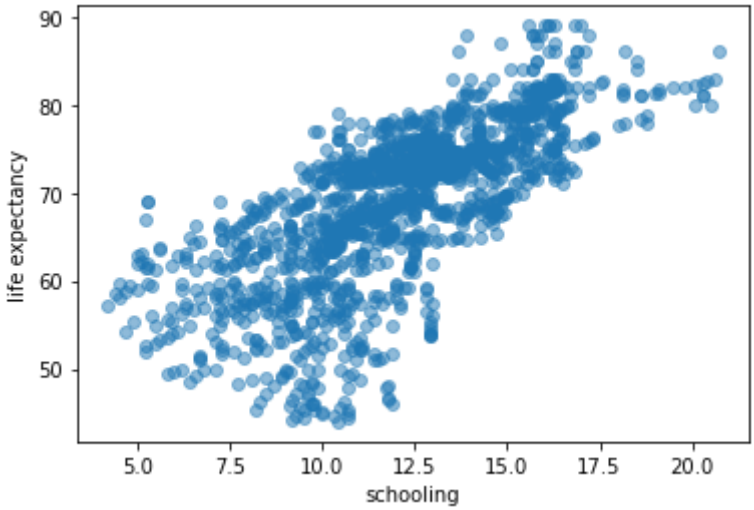
In [37]:
```
plt.scatter(df['thinness 5-9 years'], df['life expectancy'], alpha=0.5)
plt.xlabel('thinness 5-9 years')
plt.ylabel('life expectancy')
plt.show()
```



In [38]:
```
plt.scatter(df['income composition of resources'], df['life expectancy'], alpha
plt.xlabel('income composition of resources')
plt.ylabel('life expectancy')
plt.show()
```



In [39]:
```
plt.scatter(df['schooling'], df['life expectancy'], alpha=0.5)
plt.xlabel('schooling')
plt.ylabel('life expectancy')
plt.show()
```



We see that some variables are clearly correlated with life expectancy, which should help with prediction. From the correlation with each other visual from the last step, I saw that infant deaths and under-five deaths are correlated, which makes sense. As well as thinness for overlapping age groups and GDP and percentage expenditure. BMI is inversely correlated with thinness. Mortality and AIDS are inversely correlated with Life Expectancy. Not all of these variables will be necessary to predict life expectancy because of intervariable correlation but I don't want to throw out any features at this point.

Then, I split the DataFrame into X, the predictors, and y, the target, life expectancy.

In [40]:
```
#Time to split up the predictors and the target
X = df.loc[:, df.columns != 'life expectancy']
y = df[['life expectancy']]
```

4) Pre-processing and training data

I needed to get the data ready for modeling in this step.

In [41]:
```
X.columns
```

Out[41]:
```
Index(['country', 'year', 'status', '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 [42]:
```
y.columns
```

Out[42]:
```
Index(['life expectancy'], dtype='object')
```

One part of pre-processing is changing the non-numeric variables, country and development status in my case, into numeric using get_dummies. First, I will consider making dummy variables for categorical features. The only non-number features are the country and status columnmns. To make my models more effective later on, to ensure linear independence of the new dummy categories, I will drop the first column. For example, the status column has two options, Developed or Developing, and there will be a binary feature for Developing in which a 0 indicates the other option, Developed.

In [43]:
```
X = pd.get_dummies(X, columns=['country'], prefix='C', drop_first=True)
X=pd.get_dummies(X, columns=['status'], prefix='S', drop_first=True)
X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1649 entries, 0 to 2937
Columns: 152 entries, year to S_Developing
dtypes: float64(15), int64(4), uint8(133)
memory usage: 536.4 KB
```

In [44]:
```
X.head()
```

Out[44]:

	year	adult mortality	infant deaths	alcohol	percentage expenditure	hepatitis b	measles	bmi	under-five deaths	polio	expen
0	2015	263.0	62	0.01	71.279624	65.0	1154	19.1	83	6.0	
1	2014	271.0	64	0.01	73.523582	62.0	492	18.6	86	58.0	
2	2013	268.0	66	0.01	73.219243	64.0	430	18.1	89	62.0	
3	2012	272.0	69	0.01	78.184215	67.0	2787	17.6	93	67.0	
4	2011	275.0	71	0.01	7.097109	68.0	3013	17.2	97	68.0	

5 rows × 152 columns

Now everything is a number and can be used in machine learning models. I have decided to drop year, because it didn't make sense to standardize this time variable, numerically. Now we will standardize variables, so that features that have more extreme values (like Number of Infant Deaths per 1000 population which values are around 70, or population which is

much higher) versus Alcohol, recorded per capita (15+) consumption (in litres of pure alcohol) which is around 0.01, aren't given more importance in the model.

In [45]:
```
X = X.drop('year', axis=1)
```

In [46]:
```
X.describe()
```

Out[46]:

	adult mortality	infant deaths	alcohol	percentage expenditure	hepatitis b	measles	
count	1649.000000	1649.000000	1649.000000	1649.000000	1649.000000	1649.000000	164
mean	168.215282	32.553062	4.533196	698.973558	79.217708	2224.494239	3
std	125.310417	120.847190	4.029189	1759.229336	25.604664	10085.802019	1
min	1.000000	0.000000	0.010000	0.000000	2.000000	0.000000	
25%	77.000000	1.000000	0.810000	37.438577	74.000000	0.000000	1
50%	148.000000	3.000000	3.790000	145.102253	89.000000	15.000000	4
75%	227.000000	22.000000	7.340000	509.389994	96.000000	373.000000	5
max	723.000000	1600.000000	17.870000	18961.348600	99.000000	131441.000000	

8 rows × 151 columns

We will do standard scaling rather than min max because we want to be sure to catch outliers.

In general, what you must do is scale on the training set and transfer the scale over to the testing set. This is done by using the methods fit and transform separately.

First will split (note: the dataset X is still in tact, will be used for feature importance):

In [47]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0, test_
ss = StandardScaler()
X_train = ss.fit_transform(X_train)
X_test = ss.transform(X_test)
```

In [48]:
```
X_train=pd.DataFrame(X_train)
X_test=pd.DataFrame(X_test)
y_train=pd.DataFrame(y_train)
y_test=pd.DataFrame(y_test)
```

In [49]:
```
X_train.head()
```

Out[49]:

	0	1	2	3	4	5	6	7	
0	0.542453	-0.011323	-0.884300	-0.356731	-2.776135	-0.220375	-0.985985	-0.003912	-0.
1	0.029681	-0.240409	-0.577651	-0.382358	0.732037	-0.217324	-1.011618	-0.251169	0.(
2	-1.231900	-0.266842	0.030662	-0.258352	0.770588	-0.219883	0.546885	-0.270690	0.(
3	-1.272596	-0.249220	-0.782083	-0.353767	0.732037	-0.220474	0.936510	-0.257676	0.(
4	-0.076129	-0.240409	-0.318370	-0.252233	0.500729	-0.220572	0.352072	-0.244663	0.4

5 rows × 151 columns

In [50]:
```
X_test.head()
```

Out[50]:

	0	1	2	3	4	5	6	7	
0	2.105186	-0.258031	0.534264	-0.103354	0.577832	-0.220572	-0.206734	-0.257676	0.5
1	-0.694711	-0.249220	1.840640	0.022492	-0.193195	1.250903	-1.621690	-0.257676	0.6
2	-0.336584	0.032732	0.197698	0.234683	0.539280	-0.220572	-1.647323	-0.016926	0.5
3	2.121465	-0.046567	0.417090	-0.336903	0.693486	-0.220572	-1.790870	-0.023432	0.5
4	0.973833	-0.258031	0.895761	-0.368547	-1.041325	-0.211713	-0.350280	-0.257676	-1.7

5 rows × 151 columns

In [51]:
```
y_train.head()
```

Out[51]:

	life expectancy
1557	64.3
2448	72.3
27	73.0
2670	74.8
1145	73.2

In [52]:
```
y_test.head()
```

Out[52]:

	life expectancy
343	57.5
934	81.7
1688	75.6
2925	56.6
955	61.4

5) Modelling

I asked, Which models can give the best predictions, and how can we use multiple linear regression to find out feature importance?

I first used multiple linear regression (without the Ridge penalty) not for prediction, but for feature importance. The un-standardized model showed us information for each variable, like that HIV/AIDS had a comparatively large negative impact on Life Expectancy because of its coefficient of -0.3440. This is essentially the slope in actual units, that when the HIV/AIDS measurement increases by 1, life expectancy decreases by .344. Income composition of resources on the other hand has a positive relationship, 3.6470. The multiple linear regression with standardized features showed us that adult mortality, with a coefficient of 5.6682, has a higher importance than features like infant deaths, with the coefficient of -0.4981.

We want regression rather than classification, because we are trying to predict the continuous outcome of life expectancy. Our training data is labeled, so it is supervised. Some options can be decision trees (or, in the aggregate case, random forests), linear regression, and support vector machine (regression).

In [53]:
```
import statsmodels.api as sm
from statsmodels.graphics.api import abline_plot
from sklearn.metrics import mean_squared_error, r2_score, f1_score
from sklearn.model_selection import train_test_split
from sklearn import linear_model, preprocessing
import warnings
import numpy as np
from sklearn import tree, metrics
from sklearn.model_selection import train_test_split
import seaborn as sns
from sklearn.svm import SVR
import matplotlib.pyplot as plt
from io import StringIO
from IPython.display import Image
from sklearn.linear_model import Ridge
import pydotplus
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GridSearchCV
warnings.simplefilter(action="ignore", category=FutureWarning)
warnings.filterwarnings(action="ignore", module="scipy", message="^internal gel
```

In [54]:
```
print(X_train.shape, y_train.shape)
print(X_test.shape, y_test.shape)
```

```
(1319, 151) (1319, 1)
(330, 151) (330, 1)
```

Feature Importance

Ordinary Least Squares linear regression from stats models. We will use the unstandardized data first, and the summary of the features will show feature importances; more important have greater coefficients and the coefficients represent how much that feature impacts the literal years of life expectancy in the original units. the p value is also important. The

standardized coefficient is measured in units of standard deviation. We can rank independent variables with an absolute value of standardized coefficients. The most important variable will have the maximum absolute value of the standardized coefficient.

We will use the unstandardized data first, and the summary of the features will show feature importances; more important have greater coefficients and the coefficients represent how much that feature impacts the literal years of life expectancy in the original units. the p value is also important.

In [55]:
```
#with un-standardized
X_sm = sm.add_constant(X)

rModel = sm.OLS(y, X_sm)

rModel_results = rModel.fit()

rModel_results.summary()
```

Out[55]:

OLS Regression Results

Dep. Variable:	life expectancy	**R-squared:**	0.964
Model:	OLS	**Adj. R-squared:**	0.960
Method:	Least Squares	**F-statistic:**	266.6
Date:	Tue, 07 Feb 2023	**Prob (F-statistic):**	0.00
Time:	17:40:16	**Log-Likelihood:**	-3186.4
No. Observations:	1649	**AIC:**	6675.
Df Residuals:	1498	**BIC:**	7491.
Df Model:	150		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	58.6439	1.033	56.749	0.000	56.617	60.671
adult mortality	-0.0010	0.001	-1.841	0.066	-0.002	6.86e-05
infant deaths	0.0469	0.016	2.874	0.004	0.015	0.079
alcohol	-0.1237	0.032	-3.916	0.000	-0.186	-0.062
percentage expenditure	-0.0002	0.000	-1.210	0.226	-0.000	9.67e-05
hepatitis b	0.0076	0.003	3.003	0.003	0.003	0.013
measles	-7.81e-06	6.81e-06	-1.147	0.251	-2.12e-05	5.54e-06
bmi	0.0018	0.004	0.507	0.612	-0.005	0.009
under-five deaths	-0.0367	0.012	-3.185	0.001	-0.059	-0.014
polio	-0.0007	0.003	-0.264	0.792	-0.006	0.005
total expenditure	-0.0002	0.028	-0.005	0.996	-0.055	0.054
diphtheria	-0.0009	0.003	-0.297	0.767	-0.007	0.005
hiv/aids	-0.3440	0.016	-21.091	0.000	-0.376	-0.312
gdp	3.865e-05	1.93e-05	2.000	0.046	7.39e-07	7.66e-05
population	9.821e-11	9.77e-10	0.100	0.920	-1.82e-09	2.02e-09
thinness 1-19 years	-0.0052	0.034	-0.150	0.881	-0.073	0.062
thinness 5-9 years	0.0493	0.033	1.499	0.134	-0.015	0.114
income composition of resources	3.6470	0.587	6.216	0.000	2.496	4.798
schooling	0.8966	0.065	13.789	0.000	0.769	1.024
C_Albania	12.7647	0.848	15.054	0.000	11.101	14.428
C_Algeria	10.2748	0.832	12.356	0.000	8.644	11.906
C_Angola	-6.4041	0.861	-7.439	0.000	-8.093	-4.715
C_Argentina	8.9588	0.990	9.048	0.000	7.016	10.901
C_Armenia	11.3074	0.831	13.612	0.000	9.678	12.937
C_Australia	2.2501	0.600	3.752	0.000	1.074	3.427

C_Austria	6.4476	0.481	13.402	0.000	5.504	7.391
C_Azerbaijan	9.1588	0.823	11.132	0.000	7.545	10.773
C_Bangladesh	10.6261	0.740	14.363	0.000	9.175	12.077
C_Belarus	6.1985	0.972	6.375	0.000	4.291	8.106
C_Belgium	4.4403	0.476	9.338	0.000	3.508	5.373
C_Belize	6.8832	0.833	8.263	0.000	5.249	8.517
C_Benin	-0.3442	0.727	-0.473	0.636	-1.771	1.083
C_Bhutan	6.4942	0.706	9.195	0.000	5.109	7.880
C_Bosnia and Herzegovina	13.3669	0.885	15.097	0.000	11.630	15.104
C_Botswana	-0.4606	0.829	-0.556	0.578	-2.086	1.165
C_Brazil	9.0040	0.885	10.175	0.000	7.268	10.740
C_Bulgaria	-0.2465	0.463	-0.533	0.594	-1.154	0.661
C_Burkina Faso	2.4838	0.831	2.990	0.003	0.854	4.113
C_Burundi	-0.6254	0.770	-0.812	0.417	-2.137	0.886
C_Cabo Verde	10.4248	0.779	13.387	0.000	8.897	11.952
C_Cambodia	5.7071	0.785	7.268	0.000	4.167	7.247
C_Cameroon	-1.5478	0.813	-1.903	0.057	-3.143	0.047
C_Canada	16.5499	1.040	15.913	0.000	14.510	18.590
C_Central African Republic	-2.8551	0.909	-3.141	0.002	-4.638	-1.072
C_Chad	-2.1040	0.875	-2.406	0.016	-3.819	-0.388
C_Chile	14.7231	1.007	14.618	0.000	12.747	16.699
C_China	12.2206	1.256	9.733	0.000	9.758	14.684
C_Colombia	10.9856	0.828	13.261	0.000	9.361	12.611
C_Comoros	2.0787	0.759	2.738	0.006	0.590	3.568
C_Costa Rica	15.5484	0.845	18.411	0.000	13.892	17.205
C_Croatia	2.8583	0.612	4.672	0.000	1.658	4.058
C_Cyprus	5.8344	0.478	12.208	0.000	4.897	6.772
C_Djibouti	7.7896	0.888	8.776	0.000	6.048	9.531
C_Dominican Republic	10.2807	0.837	12.286	0.000	8.639	11.922
C_Ecuador	11.6787	0.849	13.753	0.000	10.013	13.344
C_El Salvador	9.0157	0.830	10.861	0.000	7.387	10.644
C_Equatorial Guinea	-0.1400	1.853	-0.076	0.940	-3.775	3.495
C_Eritrea	6.8225	0.816	8.360	0.000	5.222	8.423
C_Estonia	8.9354	1.011	8.839	0.000	6.952	10.918
C_Ethiopia	6.9841	0.900	7.763	0.000	5.219	8.749
C_Fiji	4.0303	0.836	4.823	0.000	2.391	5.669

C_France	16.9462	1.038	16.325	0.000	14.910	18.982
C_Gabon	3.4506	0.878	3.929	0.000	1.728	5.173
C_Georgia	11.0290	0.834	13.224	0.000	9.393	12.665
C_Germany	5.1703	0.500	10.332	0.000	4.189	6.152
C_Ghana	2.5958	0.738	3.519	0.000	1.149	4.043
C_Greece	15.3159	0.967	15.837	0.000	13.419	17.213
C_Guatemala	12.7202	0.876	14.525	0.000	11.002	14.438
C_Guinea	0.4732	0.822	0.576	0.565	-1.138	2.085
C_Guinea-Bissau	0.4558	0.911	0.500	0.617	-1.332	2.243
C_Guyana	5.5207	0.800	6.903	0.000	3.952	7.089
C_Haiti	4.3870	1.384	3.169	0.002	1.672	7.102
C_Honduras	12.2448	0.807	15.168	0.000	10.661	13.828
C_India	7.4202	3.004	2.470	0.014	1.527	13.313
C_Indonesia	5.4102	0.837	6.465	0.000	3.769	7.052
C_Iraq	9.4007	0.799	11.769	0.000	7.834	10.967
C_Ireland	5.1746	0.799	6.476	0.000	3.607	6.742
C_Israel	14.3831	0.915	15.712	0.000	12.588	16.179
C_Italy	6.1634	0.459	13.428	0.000	5.263	7.064
C_Jamaica	12.1595	0.873	13.928	0.000	10.447	13.872
C_Jordan	8.8841	0.811	10.957	0.000	7.294	10.475
C_Kazakhstan	2.5504	0.878	2.906	0.004	0.829	4.272
C_Kenya	1.1167	0.745	1.500	0.134	-0.344	2.577
C_Kiribati	4.1386	0.868	4.766	0.000	2.435	5.842
C_Latvia	-1.4021	0.460	-3.051	0.002	-2.304	-0.501
C_Lebanon	10.8133	0.818	13.216	0.000	9.208	12.418
C_Lesotho	-3.9182	0.862	-4.544	0.000	-5.610	-2.227
C_Liberia	2.2984	0.866	2.654	0.008	0.600	3.997
C_Lithuania	-2.4691	0.475	-5.194	0.000	-3.402	-1.537
C_Luxembourg	6.7315	0.528	12.753	0.000	5.696	7.767
C_Madagascar	4.2250	0.728	5.806	0.000	2.798	5.652
C_Malawi	-3.4271	0.809	-4.237	0.000	-5.014	-1.840
C_Malaysia	10.0075	0.756	13.241	0.000	8.525	11.490
C_Maldives	12.5223	0.705	17.755	0.000	11.139	13.906
C_Mali	0.1945	0.776	0.251	0.802	-1.327	1.716
C_Malta	6.5856	0.511	12.896	0.000	5.584	7.587
C_Mauritania	5.6726	0.784	7.235	0.000	4.135	7.210

C_Mauritius	8.6256	0.781	11.046	0.000	7.094	10.157
C_Mexico	12.9872	0.847	15.341	0.000	11.327	14.648
C_Mongolia	3.0163	0.826	3.652	0.000	1.396	4.636
C_Montenegro	10.4282	0.961	10.854	0.000	8.544	12.313
C_Morocco	11.1812	0.745	15.011	0.000	9.720	12.642
C_Mozambique	0.3624	0.800	0.453	0.650	-1.206	1.931
C_Myanmar	5.7420	0.711	8.080	0.000	4.348	7.136
C_Namibia	4.6205	0.979	4.719	0.000	2.700	6.541
C_Nepal	6.0378	0.707	8.542	0.000	4.651	7.424
C_Netherlands	2.6333	1.030	2.557	0.011	0.613	4.653
C_Nicaragua	12.4098	0.817	15.196	0.000	10.808	14.012
C_Niger	7.9883	1.004	7.957	0.000	6.019	9.957
C_Nigeria	2.0082	1.745	1.151	0.250	-1.415	5.432
C_Pakistan	6.6190	1.144	5.785	0.000	4.375	8.863
C_Panama	13.6378	0.865	15.757	0.000	11.940	15.335
C_Papua New Guinea	3.8555	0.806	4.783	0.000	2.274	5.437
C_Paraguay	10.8034	0.866	12.469	0.000	9.104	12.503
C_Peru	10.6662	0.893	11.947	0.000	8.915	12.417
C_Philippines	5.9963	0.832	7.204	0.000	4.363	7.629
C_Poland	0.7246	0.459	1.579	0.114	-0.175	1.625
C_Portugal	4.9074	0.470	10.451	0.000	3.986	5.828
C_Romania	0.5191	0.464	1.118	0.264	-0.391	1.430
C_Russian Federation	4.2093	0.931	4.522	0.000	2.384	6.035
C_Rwanda	3.1615	0.775	4.079	0.000	1.641	4.682
C_Samoa	11.0359	0.868	12.721	0.000	9.334	12.738
C_Sao Tome and Principe	5.6489	0.786	7.188	0.000	4.107	7.190
C_Senegal	6.4458	0.789	8.165	0.000	4.897	7.994
C_Serbia	10.9271	0.943	11.593	0.000	9.078	12.776
C_Seychelles	9.3813	0.810	11.584	0.000	7.793	10.970
C_Sierra Leone	-9.2761	0.820	-11.307	0.000	-10.885	-7.667
C_Solomon Islands	9.0584	0.820	11.048	0.000	7.450	10.667
C_South Africa	1.8312	0.837	2.189	0.029	0.190	3.472
C_Spain	6.1174	0.465	13.162	0.000	5.206	7.029
C_Sri Lanka	9.2502	0.778	11.886	0.000	7.724	10.777
C_Suriname	8.9374	0.883	10.127	0.000	7.206	10.668
C_Swaziland	2.9205	0.952	3.066	0.002	1.052	4.789

C_Sweden	6.0387	0.894	6.752	0.000	4.284	7.793
C_Syrian Arab Republic	12.1001	0.860	14.076	0.000	10.414	13.786
C_Tajikistan	5.7715	0.793	7.274	0.000	4.215	7.328
C_Thailand	10.4295	0.778	13.402	0.000	8.903	11.956
C_Timor-Leste	4.3024	0.888	4.846	0.000	2.561	6.044
C_Togo	-1.8514	0.881	-2.102	0.036	-3.579	-0.124
C_Tonga	8.0950	0.892	9.072	0.000	6.345	9.845
C_Trinidad and Tobago	8.9665	0.819	10.948	0.000	7.360	10.573
C_Tunisia	9.6077	0.804	11.950	0.000	8.031	11.185
C_Turkey	10.4974	0.782	13.418	0.000	8.963	12.032
C_Turkmenistan	5.3532	0.841	6.364	0.000	3.703	7.003
C_Uganda	0.5026	0.818	0.615	0.539	-1.101	2.106
C_Ukraine	6.1788	0.894	6.915	0.000	4.426	7.931
C_Uruguay	10.8256	0.908	11.924	0.000	9.045	12.606
C_Uzbekistan	5.9575	0.815	7.313	0.000	4.359	7.555
C_Vanuatu	11.5812	0.824	14.052	0.000	9.965	13.198
C_Zambia	-0.4936	0.831	-0.594	0.553	-2.125	1.137
C_Zimbabwe	-0.8515	0.839	-1.015	0.310	-2.497	0.794
S_Developing	-9.8348	0.802	-12.259	0.000	-11.408	-8.261

Omnibus:	580.978	**Durbin-Watson:**	1.179
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	2700.103
Skew:	1.615	**Prob(JB):**	0.00
Kurtosis:	8.373	**Cond. No.**	2.67e+21

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 1.19e-24. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

We see that some variables like "income composition of resources" have a large effect on life expectancy; when that variable is increased by 1, life expectancy increases by 3.6. The country variables coefficients are notably large too.

The standardized coefficient is measured in units of standard deviation. We can rank independent variables with an absolute value of standardized coefficients. The most important variable will have the maximum absolute value of the standardized coefficient.

In [56]:
```
#with standardized
ss2 = StandardScaler()
X_all_ss = ss.fit_transform(X)
X_sm_2 = sm.add_constant(X_all_ss)

rModel2 = sm.OLS(y, X_sm_2)

rModel2_results = rModel2.fit()

rModel2_results.summary()
```

Out[56]:

OLS Regression Results

Dep. Variable:	life expectancy	**R-squared:**	0.964
Model:	OLS	**Adj. R-squared:**	0.960
Method:	Least Squares	**F-statistic:**	266.6
Date:	Tue, 07 Feb 2023	**Prob (F-statistic):**	0.00
Time:	17:40:17	**Log-Likelihood:**	-3186.4
No. Observations:	1649	**AIC:**	6675.
Df Residuals:	1498	**BIC:**	7491.
Df Model:	150		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	69.3023	0.043	1605.238	0.000	69.218	69.387
x1	-0.1312	0.071	-1.841	0.066	-0.271	0.009
x2	5.6682	1.972	2.874	0.004	1.800	9.537
x3	-0.4981	0.127	-3.916	0.000	-0.748	-0.249
x4	-0.2738	0.226	-1.210	0.226	-0.718	0.170
x5	0.1938	0.065	3.003	0.003	0.067	0.320
x6	-0.0787	0.069	-1.147	0.251	-0.213	0.056
x7	0.0358	0.071	0.507	0.612	-0.103	0.174
x8	-5.9821	1.878	-3.185	0.001	-9.666	-2.298
x9	-0.0162	0.061	-0.264	0.792	-0.137	0.104
x10	-0.0003	0.064	-0.005	0.996	-0.125	0.125
x11	-0.0202	0.068	-0.297	0.767	-0.153	0.113
x12	-2.0746	0.098	-21.091	0.000	-2.268	-1.882
x13	0.4434	0.222	2.000	0.046	0.008	0.878
x14	0.0069	0.069	0.100	0.920	-0.128	0.142
x15	-0.0238	0.158	-0.150	0.881	-0.334	0.287
x16	0.2293	0.153	1.499	0.134	-0.071	0.529
x17	0.6675	0.107	6.216	0.000	0.457	0.878
x18	2.5056	0.182	13.789	0.000	2.149	2.862
x19	1.2512	0.083	15.054	0.000	1.088	1.414
x20	0.8364	0.068	12.356	0.000	0.704	0.969
x21	-0.4450	0.060	-7.439	0.000	-0.562	-0.328
x22	0.7923	0.088	9.048	0.000	0.621	0.964
x23	1.0735	0.079	13.612	0.000	0.919	1.228
x24	0.4530	0.070	6.463	0.000	0.316	0.590

x25	0.8673	0.060	14.530	0.000	0.750	0.984
x26	0.8100	0.073	11.132	0.000	0.667	0.953
x27	0.9032	0.063	14.363	0.000	0.780	1.027
x28	0.5885	0.092	6.375	0.000	0.407	0.770
x29	0.6767	0.061	11.144	0.000	0.558	0.796
x30	0.6535	0.079	8.263	0.000	0.498	0.809
x31	-0.0304	0.064	-0.473	0.636	-0.157	0.096
x32	0.6166	0.067	9.195	0.000	0.485	0.748
x33	1.0881	0.072	15.097	0.000	0.947	1.229
x34	-0.0437	0.079	-0.556	0.578	-0.198	0.111
x35	0.8548	0.084	10.175	0.000	0.690	1.020
x36	0.2317	0.054	4.315	0.000	0.126	0.337
x37	0.1830	0.061	2.990	0.003	0.063	0.303
x38	-0.0509	0.063	-0.812	0.417	-0.174	0.072
x39	0.9220	0.069	13.387	0.000	0.787	1.057
x40	0.4205	0.058	7.268	0.000	0.307	0.534
x41	-0.1202	0.063	-1.903	0.057	-0.244	0.004
x42	1.4067	0.088	15.913	0.000	1.233	1.580
x43	-0.1719	0.055	-3.141	0.002	-0.279	-0.065
x44	-0.1368	0.057	-2.406	0.016	-0.248	-0.025
x45	1.0847	0.074	14.618	0.000	0.939	1.230
x46	1.1602	0.119	9.733	0.000	0.926	1.394
x47	1.0430	0.079	13.261	0.000	0.889	1.197
x48	0.1767	0.065	2.738	0.006	0.050	0.303
x49	1.4762	0.080	18.411	0.000	1.319	1.633
x50	0.3853	0.051	7.605	0.000	0.286	0.485
x51	0.8091	0.056	14.367	0.000	0.699	0.920
x52	0.5412	0.062	8.776	0.000	0.420	0.662
x53	0.9761	0.079	12.286	0.000	0.820	1.132
x54	1.1088	0.081	13.753	0.000	0.951	1.267
x55	0.8560	0.079	10.861	0.000	0.701	1.011
x56	-0.0034	0.046	-0.076	0.940	-0.093	0.086
x57	0.5297	0.063	8.360	0.000	0.405	0.654
x58	0.7274	0.082	8.839	0.000	0.566	0.889
x59	0.4853	0.063	7.763	0.000	0.363	0.608
x60	0.3826	0.079	4.823	0.000	0.227	0.538

x61	1.6089	0.099	16.325	0.000	1.416	1.802
x62	0.2679	0.068	3.929	0.000	0.134	0.402
x63	1.0471	0.079	13.224	0.000	0.892	1.202
x64	0.7460	0.063	11.900	0.000	0.623	0.869
x65	0.2296	0.065	3.519	0.000	0.102	0.358
x66	1.4541	0.092	15.837	0.000	1.274	1.634
x67	0.9876	0.068	14.525	0.000	0.854	1.121
x68	0.0329	0.057	0.576	0.565	-0.079	0.145
x69	0.0274	0.055	0.500	0.617	-0.080	0.135
x70	0.5065	0.073	6.903	0.000	0.363	0.650
x71	0.1527	0.048	3.169	0.002	0.058	0.247
x72	1.1625	0.077	15.168	0.000	1.012	1.313
x73	0.6040	0.245	2.470	0.014	0.124	1.084
x74	0.5137	0.079	6.465	0.000	0.358	0.669
x75	0.7652	0.065	11.769	0.000	0.638	0.893
x76	0.4323	0.052	8.326	0.000	0.330	0.534
x77	1.3655	0.087	15.712	0.000	1.195	1.536
x78	0.8403	0.057	14.794	0.000	0.729	0.952
x79	1.0335	0.074	13.928	0.000	0.888	1.179
x80	0.8435	0.077	10.957	0.000	0.692	0.994
x81	0.2421	0.083	2.906	0.004	0.079	0.406
x82	0.0988	0.066	1.500	0.134	-0.030	0.228
x83	0.3929	0.082	4.766	0.000	0.231	0.555
x84	0.1220	0.054	2.250	0.025	0.016	0.228
x85	1.0266	0.078	13.216	0.000	0.874	1.179
x86	-0.3330	0.073	-4.544	0.000	-0.477	-0.189
x87	0.1494	0.056	2.654	0.008	0.039	0.260
x88	0.0207	0.058	0.358	0.721	-0.093	0.134
x89	0.8942	0.062	14.312	0.000	0.772	1.017
x90	0.3737	0.064	5.806	0.000	0.247	0.500
x91	-0.3031	0.072	-4.237	0.000	-0.443	-0.163
x92	0.9501	0.072	13.241	0.000	0.809	1.091
x93	1.1889	0.067	17.755	0.000	1.058	1.320
x94	0.0165	0.066	0.251	0.802	-0.113	0.146
x95	0.7881	0.052	15.054	0.000	0.685	0.891
x96	0.4404	0.061	7.235	0.000	0.321	0.560

x97	0.8189	0.074	11.046	0.000	0.674	0.964
x98	1.2330	0.080	15.341	0.000	1.075	1.391
x99	0.2864	0.078	3.652	0.000	0.133	0.440
x100	0.7683	0.071	10.854	0.000	0.629	0.907
x101	1.0615	0.071	15.011	0.000	0.923	1.200
x102	0.0333	0.073	0.453	0.650	-0.111	0.177
x103	0.4880	0.060	8.080	0.000	0.370	0.607
x104	0.2540	0.054	4.719	0.000	0.148	0.360
x105	0.5132	0.060	8.542	0.000	0.395	0.631
x106	0.2617	0.056	4.693	0.000	0.152	0.371
x107	1.1782	0.078	15.196	0.000	1.026	1.330
x108	0.4810	0.060	7.957	0.000	0.362	0.600
x109	0.1559	0.135	1.151	0.250	-0.110	0.422
x110	0.5626	0.097	5.785	0.000	0.372	0.753
x111	1.2513	0.079	15.757	0.000	1.095	1.407
x112	0.3660	0.077	4.783	0.000	0.216	0.516
x113	0.9182	0.074	12.469	0.000	0.774	1.063
x114	0.9066	0.076	11.947	0.000	0.758	1.055
x115	0.5693	0.079	7.204	0.000	0.414	0.724
x116	0.3239	0.055	5.911	0.000	0.216	0.431
x117	0.7210	0.059	12.131	0.000	0.604	0.838
x118	0.3044	0.052	5.884	0.000	0.203	0.406
x119	0.3862	0.085	4.522	0.000	0.219	0.554
x120	0.2796	0.069	4.079	0.000	0.145	0.414
x121	1.0478	0.082	12.721	0.000	0.886	1.209
x122	0.4801	0.067	7.188	0.000	0.349	0.611
x123	0.5247	0.064	8.165	0.000	0.399	0.651
x124	0.8895	0.077	11.593	0.000	0.739	1.040
x125	0.8907	0.077	11.584	0.000	0.740	1.041
x126	-0.6445	0.057	-11.307	0.000	-0.756	-0.533
x127	0.8600	0.078	11.048	0.000	0.707	1.013
x128	0.1739	0.079	2.189	0.029	0.018	0.330
x129	0.8359	0.059	14.276	0.000	0.721	0.951
x130	0.7862	0.066	11.886	0.000	0.656	0.916
x131	0.6939	0.069	10.127	0.000	0.559	0.828
x132	0.2773	0.090	3.066	0.002	0.100	0.455

x133	0.4292	0.049	8.753	0.000	0.333	0.525
x134	0.8408	0.060	14.076	0.000	0.724	0.958
x135	0.5104	0.070	7.274	0.000	0.373	0.648
x136	0.9902	0.074	13.402	0.000	0.845	1.135
x137	0.2797	0.058	4.846	0.000	0.167	0.393
x138	-0.1204	0.057	-2.102	0.036	-0.233	-0.008
x139	0.7685	0.085	9.072	0.000	0.602	0.935
x140	0.7930	0.072	10.948	0.000	0.651	0.935
x141	0.9122	0.076	11.950	0.000	0.762	1.062
x142	0.9966	0.074	13.418	0.000	0.851	1.142
x143	0.4734	0.074	6.364	0.000	0.327	0.619
x144	0.0445	0.072	0.615	0.539	-0.097	0.186
x145	0.5866	0.085	6.915	0.000	0.420	0.753
x146	1.0278	0.086	11.924	0.000	0.859	1.197
x147	0.5269	0.072	7.313	0.000	0.386	0.668
x148	1.0995	0.078	14.052	0.000	0.946	1.253
x149	-0.0383	0.065	-0.594	0.553	-0.165	0.088
x150	-0.0808	0.080	-1.015	0.310	-0.237	0.075
x151	-2.5292	0.165	-15.344	0.000	-2.853	-2.206

Omnibus:	580.978	**Durbin-Watson:**	1.179
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	2700.102
Skew:	1.615	**Prob(JB):**	0.00
Kurtosis:	8.373	**Cond. No.**	6.72e+15

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 2.48e-28. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

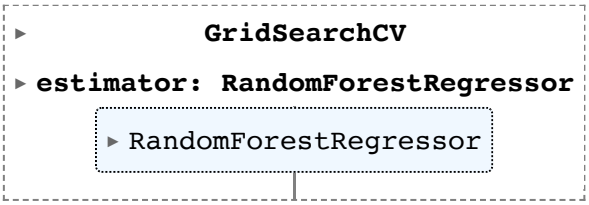
The coefficients show the order of importance when the variables are standardized.

Now, for modeling: I used grid search CV for choosing the number of estimators and max depth of the trees in the random forest.

In [57]:
```
rf = RandomForestRegressor()
parameters = {'n_estimators': [10, 20, 50, 100, 1000],
              'max_depth': [None, 1, 2, 3, 10, 20]}
grid_search = GridSearchCV(rf, parameters, cv=5, scoring='neg_mean_squared_erro
```

```
error_score='raise'
grid_search.fit(X_train, y_train.values.ravel())
```

Out[57]:

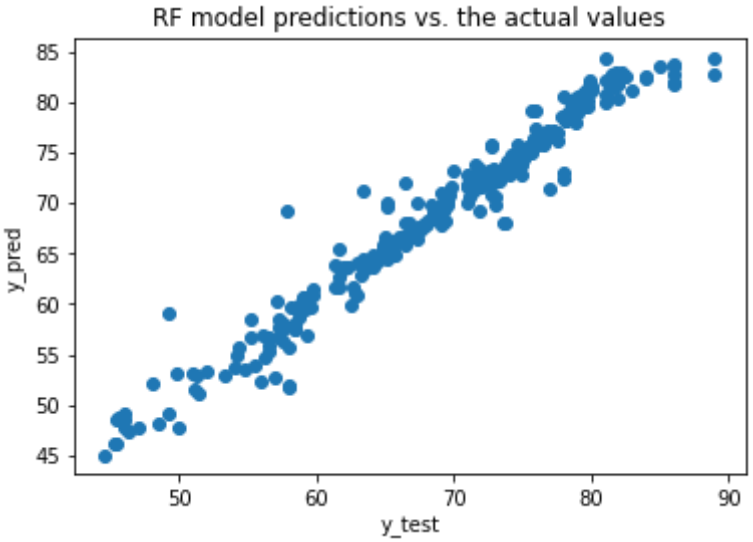


In [58]:
```
# Make predictions using the trained model
y_pred = grid_search.predict(X_test)

# Print the best hyperparameters found
print(grid_search.best_params_)

from sklearn.metrics import r2_score
r2 = r2_score(y_test, y_pred)

print(f'R2 score: {r2:.2f}')
```

```
{'max_depth': 20, 'n_estimators': 100}
R2 score: 0.96
```

In [59]:
```
plt.scatter(y_test, y_pred)
plt.title("RF model predictions vs. the actual values")
plt.xlabel("y_test")
plt.ylabel("y_pred")
```

Out[59]:
```
Text(0, 0.5, 'y_pred')
```



We ended up with an R2 score of 0.96. The chart of predicted versus actual values looks good.

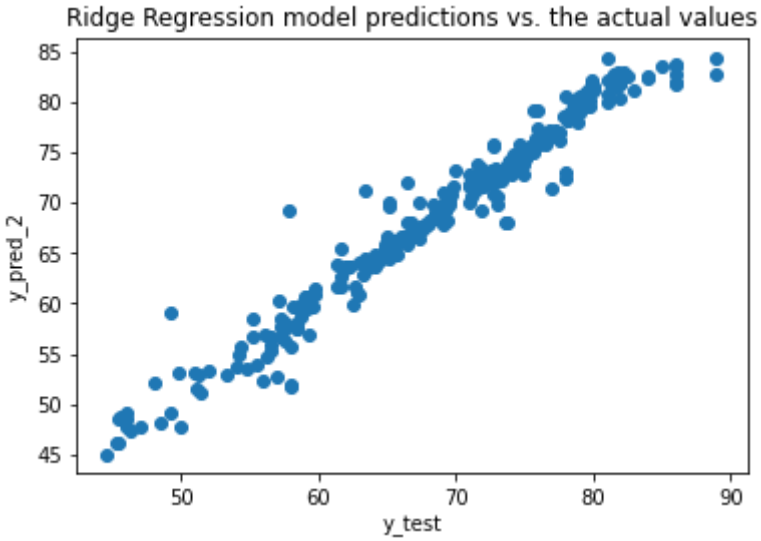
In [60]:
```
#Ridge regression. adds some bias to the least squares to have a better predict

# Set up the grid search parameters
parameters_2 = {'alpha': [0.001, 0.01, 0.1, 1, 10, 100]}

# Create the grid search object
grid_search_2 = GridSearchCV(Ridge(), parameters_2, cv=5)
```

```
# Fit the grid search to the training data
grid_search_2.fit(X_train, y_train)

# Print the optimal value of alpha
print(f'Optimal value of alpha: {grid_search_2.best_params_["alpha"]}')

# Predict on the test data using the optimal value of alpha
y_pred_2 = grid_search.predict(X_test)

r2_2 = r2_score(y_test, y_pred_2)
print(f'R2 score: {r2_2:.2f}')

plt.scatter(y_test, y_pred)
plt.title("Ridge Regression model predictions vs. the actual values")
plt.xlabel("y_test")
plt.ylabel("y_pred_2")
```

```
Optimal value of alpha: 0.1
R2 score: 0.96
```

Out[60]:

```
Text(0, 0.5, 'y_pred_2')
```



Setting up CV grid search and getting our R2: also 0.96

6) Documentation

In conclusion, I first used multiple linear regression (without the Ridge penalty) not for prediction, but for feature importance. The un-standardized model showed us information for each variable, like that HIV/AIDS had a comparitively large negative impact on Life Expectancy because of its coefficient of -0.3440. This is essentially the slope in actual units, that when the HIV/AIDS measurement increases by 1, life expectancy decreases by .344. Income composition of resources on the other hand has a positive relationship, 3.6470. The multiple linear regression with standardized features showed us that adult mortality, with a coefficient of 5.6682, has a higher importance than features like infant deaths, with the coefficient of -0.4981.

The models, Ridge Multiple Linear Regression and Random Forest, when trained using cross validation to tune the hyperparametes of Alpha for Ridge, and the n_parameters and max_depth for Random Forest, actually gave the same R2 score of .96. This is pretty good, and both models give good predictions.

In [ ]: