Predicting Life Expectancy Using World Health Organization Information from Countries

Liz Atkin

Step 1, Problem Identification: I posed the question, how do certain elements of countries 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.

Step 2, Data Wrangling: 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.

```
df.info()
```

```
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
```

Data Wrangling cont.

I got rid of the leading and trailing spaces, and used str.lower() to make everything lowercase.

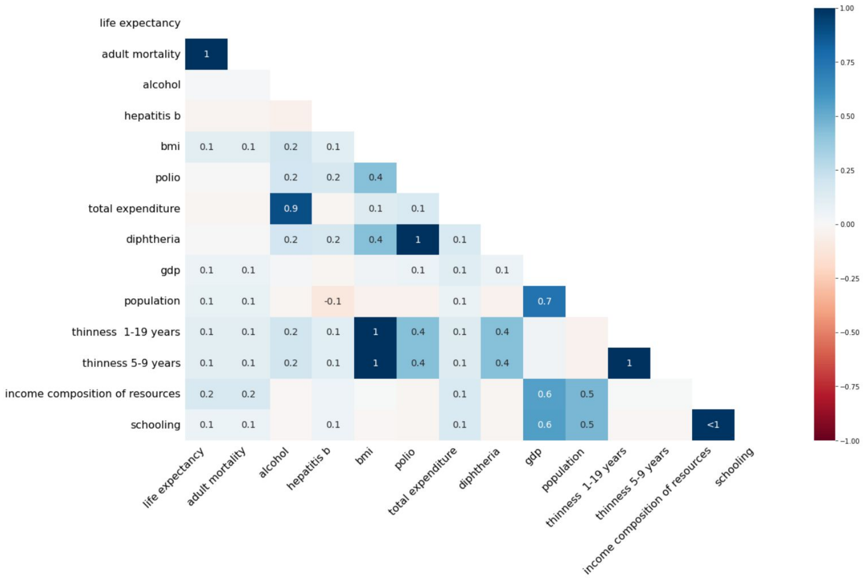
```
df.info()
```

```
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
```

Data Wrangling cont.

```
msno.heatmap(df)
```

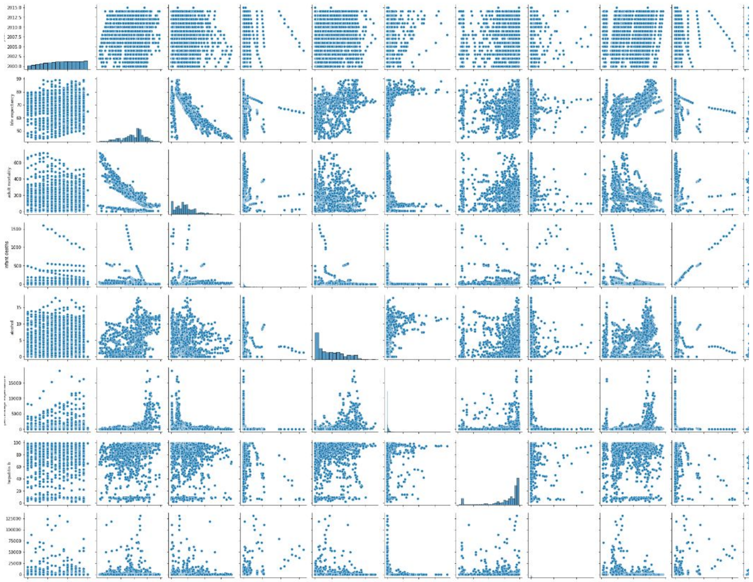
I took a look at the missing values, and a heatmap of the missing values, before dropping the rows with missing values, because there were relatively few.



Step 3, Exploratory Data Analysis: What relationships exist between my features?

One useful visual in my EDA step was a pairplot, showing the relationship between every variable and every other variable.

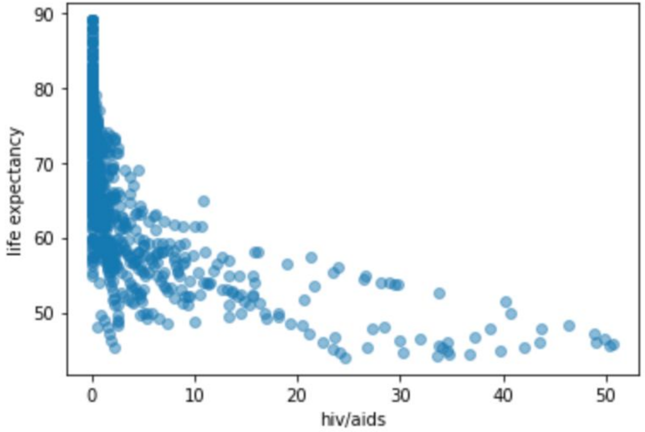
```
sns.pairplot(df,  palette='Set1')
plt.show()
```



EDA cont.

I then took a closer look at each variable with life expectancy. Here, hiv/aids has a clear negative correlation with life expectancy. Then, I split the DataFrame into X, the predictors, and y, the target, life expectancy.

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



Step 4, Pre-Processing and Training Data: I needed to get the data ready for modeling

One part of pre-processing is changing the non-numeric variables, country and development status in my case, into numeric using get_dummies

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

Pre-processing and training cont.

I split the data into train and test, and scaled the data according to a scaler trained on the training data.

```
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_size=0.2)
ss = StandardScaler()
X_train = ss.fit_transform(X_train)
X_test = ss.transform(X_test)
```

Step 5, Modeling: 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.

Step 5, Feature Importance with Un-Standardized Data

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.

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:	Sun, 08 Jan 2023	**Prob (F-statistic):**	0.00
Time:	18:55:19	**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

Step 5, Feature Importance with Standardized Data

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.

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:	Sun, 08 Jan 2023	**Prob (F-statistic):**	0.00
Time:	18:55:20	**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

Step 5, Random Forest

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

I used grid search CV for choosing the number of estimators and max depth of the trees in the forest.

Step 5, Random Forest

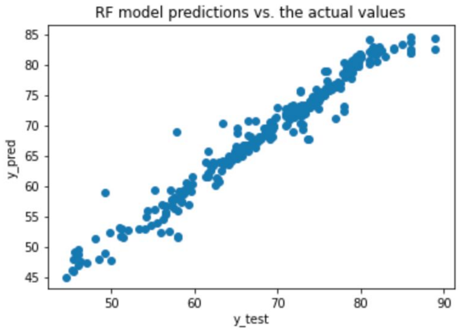
We ended up with an R2 score of 0.96. The chart of predicted versus actual values looks good.

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': None, 'n_estimators': 1000}
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')`



Step 5, Ridge Regression

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

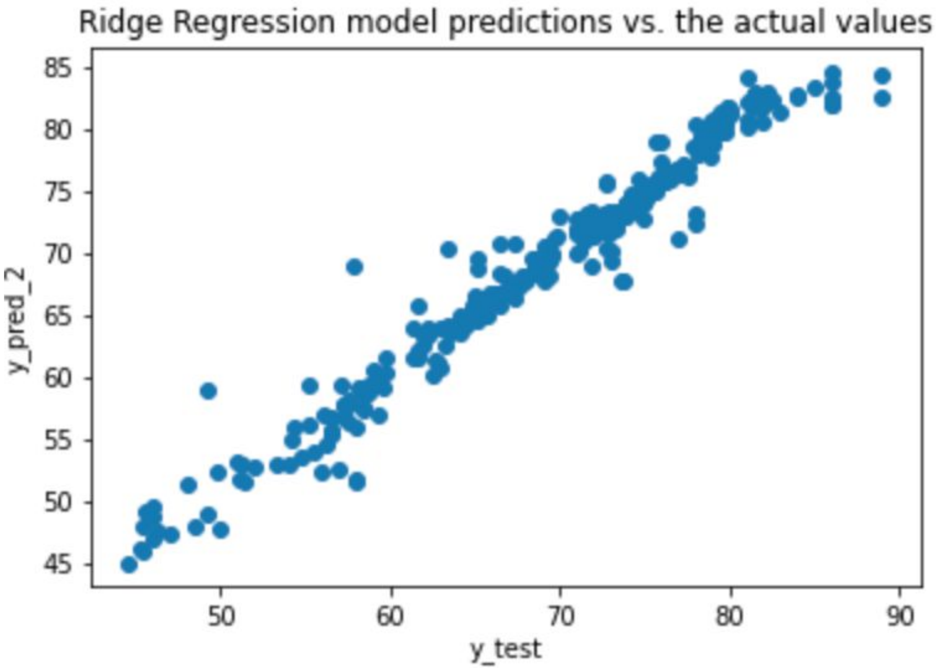
In [60]:

```
#Ridge regression. adds some bias to the least squares to have a better prediction

# 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
```

Step 5, Ridge Regression

Pretty good predictions.



Step 6, Documentation and Conclusion

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 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.

The models, Ridge Multiple Linear Regression and Random Forest, when trained using cross validation to tune the hyperparameters 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.