LIFE EXPECTANCY MODEL

Contents

Introduction

Life Expectancy is a speculative measure for us to assume the age a person will live born in a specific year. It depends on certain factors like the place the person was born and the conditions he was born in. It also depends on the health conditions the person was brought up in.

Knowing the life expectancy helps us to take measures accordingly to improve the survival rate of the population. And specifically look for the factors which effect the most in decreasing the life expectancy.

Source of the data

Below mentioned is the source for the data for this model.

https://www.kaggle.com/datasets/kumarajarshi/life-expectancy-who

Background

This data was collected by Global Health Observatory (GHO) with World Health Organization (WHO). This data has been collected from 193 countries. Only the critical factors have been taken under account. There are only a few missing data for few countries.

Data has in total 22 columns which are the most critical factors which have been selected. And there are 2938 records. These are the factors:

- Country – Country name 193 countries in total
- Year – Year of the record (The records have been recorded from 2000-2015)
- Status - Tells about the status of the country (Developed or Developing)
- Life expectancy – Expectancy in age
- Adult Mortality – Mortality rate among the adults for both male and female per 1K population)
- Infant deaths – Infant death per 1K population
- Alcohol – Alcohol consumption in litres
- Percentage expenditure – Percentage spent on health care of GDP per catipa
- Hepatitis B - HepB immunized
- Measles – Cases per 1K population
- BMI – Average BMI for the population
- Under-five deaths – Deaths per 1K population
- Polio – Polio immunized
- Total expenditure – Percentage of expenditure spend by Government on healthcare
- Diphtheria - Diphtheria immunization
- HIV/AIDS – Death per 1K population
- GDP - Gross Domestic Product per capita
- Population - Population of the each country
- Thinness 1-19 years – Thinness around the age 1-19
- Thinness 5-9 years -Thinness around the age 5-9
- Income composition of resources – Composition of resources
- Schooling – Years of schooling

Steps Followed

- Explanatory Data Analysis
    - Univariate Analysis
    - Bivariate Analysis
- Data Preparation
    - Dealing with null values
    - Changing categorical variable to numeric values
    - Dropping the unnecessary features
- Choosing of dependent and independent variables
- Feature selection
- Model Creation
    - Linear Regression
    - Lasso
    - Ridge
    - Random Forest Regressor
    - K-Neighbors Regressor
    - XGB Regressor
- Model Comparison
- Model Selection
- Model Improvement

Explanatory Data Analysis

EDA helps us to explore the data to understand the data and treat it accordingly. It helps us to understand the distribution of the data.

Univariate Analysis

- Year - There is a very uneven distribution for the year.
- Status – We have more data for the developing countries compared to the developed countries .
- Life expectancy – Life expectancy is slightly skewed to the right.
- Adult Mortality - Adult mortality is skewed towards the left.
- Infant deaths - More than 99% of the data ranges from 0-180.
- Alcohol – Alcohol is skewed towards the left
- Percentage expenditure - More than 98% of data is in the range 0 – 1947.99
- Hepatitis B – Hep B is highly skewed to the right.
- Measles – More 99% of the data is in the range 0 - 21218
- BMI - BMI is distributed between 2 maximas
- Under-five deaths – Most of the data lies between 0 - 250
- Polio – Polio is skewed on the right

- Total expenditure – Total expenditure is normally distributed
- Diphtheria – Diphtheria is skewed on the right
- HIV/AIDS – Most of the data lies between 0.5 – 5.15
- GDP – GDP is heavily left skewed.
- Population – Most of the data is in the range 34 - 129385960
- Thinness 1-19 years – Thinness is left skewed
- Thinness 5-9 years - Thinness is left skewed
- Income composition of resources – Income is normally distributed
- Schooling – schooling is normally distributed.

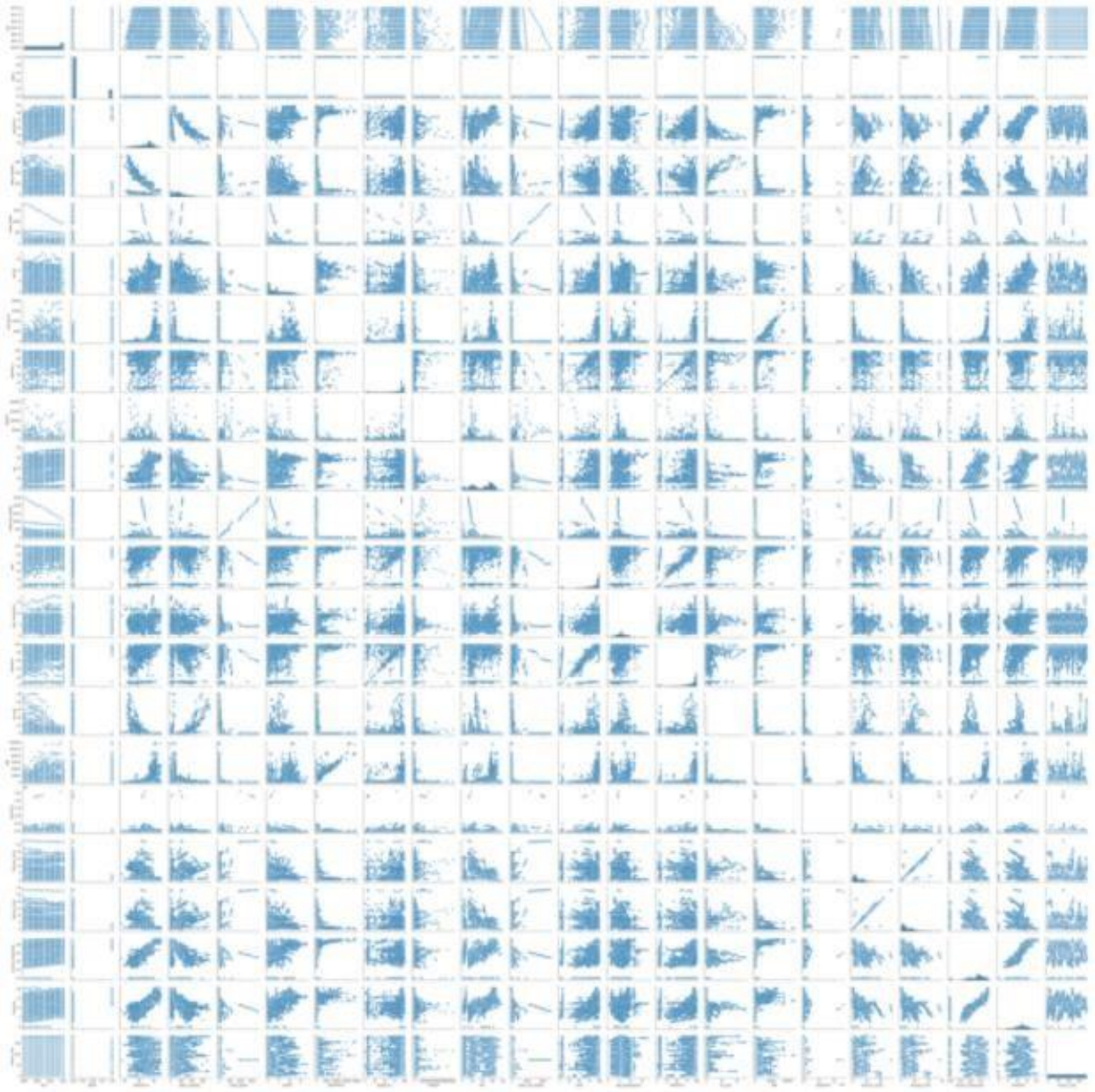
Bivariate Analysis

In the bivariate analysis we saw the relationship between:

- Expectancy and adult mortality.
- Expectancy and Schooling
- Expectancy and Income composition

Since Expectancy is our dependent variable. We can see these linear relations very clearly.

<Figure size 7200x7200 with 0 Axes>



Data Preparation

Before we feed the data to our model we will prepare the data so that we don't face any issue while training the model and reach the maximum potential of the model.

Dealing with null values

We removed the rows with the null values in the dependant feature. But for the independent variables we replaced the null values with the median.

Changing categorical variable to numeric values

Since we are using regression model we replaced the categorical data to numeric values using encoding. The only two categorical features are “Country ” and “Status”. We encoded the country with numeric values from 0 -182. And the “status” feature was replaced with 0 and 1 for Developing and Developed respectively.

```
cleanup = {"status":{"Developing": 0, "Developed": 1}}
```

```
data = data.replace(cleanup)
data.head()
```

```
ord_enc = OrdinalEncoder()
data["country_code"] = ord_enc.fit_transform(data[["country"]])
data["country_code"].unique()
```

```
array([  0.,   1.,   2.,   3.,   4.,   5.,   6.,   7.,   8.,   9.,  10.,
        11.,  12.,  13.,  14.,  15.,  16.,  17.,  18.,  19.,  20.,  21.,
        22.,  23.,  24.,  25.,  26.,  43.,  27.,  28.,  29.,  30.,  31.,
        32.,  33.,  34.,  35.,  36.,  37.,  38.,  39.,  40.,  41.,  42.,
        44.,  45.,  46.,  47.,  48.,  49.,  50.,  51.,  52.,  53.,  54.,
        55.,  56.,  57.,  58.,  59.,  60.,  61.,  62.,  63.,  64.,  65.,
        66.,  67.,  68.,  69.,  70.,  71.,  72.,  73.,  74.,  75.,  76.,
        77.,  78.,  79.,  80.,  81.,  82.,  83.,  84.,  85.,  86.,  87.,
        88.,  89.,  90.,  91.,  92.,  93.,  94.,  95.,  96.,  97.,  98.,
        99., 100., 101., 102., 103., 104., 105., 106., 107., 108., 109.,
       110., 111., 112., 113., 114., 115., 116., 117., 118., 119., 120.,
       121., 122., 123., 124., 125., 126., 127., 128., 129., 130., 131.,
       132., 133., 134., 135., 136., 137., 138., 139., 140., 141., 142.,
       143., 144., 145., 146., 147., 148., 149., 150., 151., 152., 153.,
       154., 155., 156., 157., 158., 159., 160., 161., 162., 163., 164.,
       165., 166., 167., 168., 169., 170., 171., 172., 173., 174., 175.,
       176., 177., 178., 179., 180., 181., 182.])
```

Dropping the unnecessary features

Using correlation with the dependent variable expectancy we removed the columns with low correlation. The columns removed are 'year', 'hepatitis_b', 'population' and 'measles'. We also removed country since we encoded and added a new feature called country code.

```
data_new.drop(columns = ['country', 'year', 'hepatitis_b', 'population', 'measles'] ,inplace = True)
```

Choosing of dependent and independent variables

Since we are building a model for predicting Life expectancy, our dependent variable is expectancy. And the rest of the features would be the independent variables.

Feature selection

We have done the feature selection using the correlation metric and removed the less correlated features with “expectancy”.

```
data.corr()
#year, hepatitis_b, population, measles has very poor correlation with expectancy
```

	year	status	expectancy	adult_mortality	infant_death	alcohol	expenditure	hepatitis_b	measles	bmi	...	total_e
year	1.000000e+00	5.545187e-15	0.170033	-0.079052	-0.036464	-0.065644	0.032723	0.022671	-0.081840	0.104057	...	
status	5.545187e-15	1.000000e+00	0.482136	-0.315284	-0.112708	0.579723	0.454032	0.127101	-0.077320	0.313314	...	
expectancy	1.700330e-01	4.821361e-01	1.000000	-0.696359	-0.196557	0.390674	0.381864	0.171255	-0.157586	0.558888	...	
adult_mortality	-7.905159e-02	-3.152844e-01	-0.696359	1.000000	0.078756	-0.191066	-0.242860	-0.123971	0.031176	-0.380592	...	
infant_death	-3.646405e-02	-1.127082e-01	-0.196557	0.078756	1.000000	-0.113919	-0.085906	-0.168393	0.501038	-0.226969	...	
alcohol	-6.564353e-02	5.797231e-01	0.390674	-0.191066	-0.113919	1.000000	0.344228	0.089540	-0.050490	0.321196	...	
expenditure	3.272257e-02	4.540322e-01	0.381864	-0.242860	-0.085906	0.344228	1.000000	0.039805	-0.056831	0.229920	...	
hepatitis_b	2.267079e-02	1.271006e-01	0.171255	-0.123971	-0.168393	0.089540	0.039805	1.000000	-0.073544	0.116743	...	
measles	-8.184033e-02	-7.731993e-02	-0.157586	0.031176	0.501038	-0.050490	-0.056831	-0.073544	1.000000	-0.176132	...	
bmi	1.040567e-01	3.133139e-01	0.558888	-0.380592	-0.226969	0.321196	0.229920	0.116743	-0.176132	1.000000	...	
under_five_deaths	-4.197985e-02	-1.156615e-01	-0.222529	0.094146	0.996628	-0.110801	-0.088152	-0.171539	0.507718	-0.237262	...	
polio	9.172655e-02	2.192209e-01	0.459458	-0.270597	-0.171273	0.212082	0.146546	0.362537	-0.136966	0.284278	...	
total_expenditure	7.139454e-02	2.942904e-01	0.209588	-0.112176	-0.126471	0.302242	0.177355	0.066908	-0.104294	0.227095	...	
diphtheria	1.318516e-01	2.158679e-01	0.473268	-0.270877	-0.175747	0.212766	0.142897	0.447373	-0.142680	0.283207	...	
hiv_aids	-1.387885e-01	-1.491780e-01	-0.556556	0.523821	0.024955	-0.047314	-0.098230	-0.086197	0.030673	-0.243364	...	
gdp	9.316964e-02	4.456821e-01	0.430991	-0.281715	-0.103175	0.312735	0.901803	0.076936	-0.069531	0.277118	...	
population	1.474886e-02	-3.788784e-02	-0.028842	-0.005392	0.551608	-0.027387	-0.017070	-0.125713	0.237096	-0.069296	...	
thinness_1to19	-4.480530e-02	-3.676434e-01	-0.467859	0.296076	0.464762	-0.416992	-0.250729	-0.099491	0.224606	-0.531240	...	
thinness_5to9	-4.786016e-02	-3.661381e-01	-0.462645	0.301855	0.470469	-0.406089	-0.252366	-0.102505	0.220869	-0.538194	...	
income_composition	2.358661e-01	4.614780e-01	0.688591	-0.436268	-0.141329	0.420009	0.375234	0.118166	-0.110884	0.473592	...	
schooling	2.073109e-01	4.947532e-01	0.717314	-0.435926	-0.192421	0.499675	0.387937	0.134263	-0.121817	0.510756	...	
country_code	3.339774e-14	3.267863e-02	-0.017901	0.040262	-0.028942	-0.063838	-0.032263	-0.013131	-0.023079	0.019452	...	

22 rows × 22 columns

We have also used the SelectKBest method to select the important features.

```
fs = SelectKBest(score_func=f_regression, k='all')
fs.fit(X_train, y_train)
```

```
SelectKBest(k='all', score_func=<function f_regress
```

```
X_train_fs = fs.transform(X_train)
X_test_fs = fs.transform(X_test)
```

```
for i in range(len(fs.scores_)):
    print('Feature %d: %f' % (i, fs.scores_[i]))
```

```
Feature 0: 680.386857
Feature 1: 2112.585349
Feature 2: 89.804639
Feature 3: 393.667963
Feature 4: 379.771026
Feature 5: 1114.638463
Feature 6: 115.679457
Feature 7: 568.498995
Feature 8: 90.565558
Feature 9: 618.655081
Feature 10: 1023.970120
Feature 11: 502.019052
Feature 12: 679.371560
Feature 13: 644.968818
Feature 14: 2016.846488
Feature 15: 2293.067779
Feature 16: 1.069000
```

Model Creation

We have selected the following 6 models for the regression model and created the basic version of it.

- Linear Regression
- Lasso
- Ridge
- Random Forest Regressor
- K-Neighbors Regressor
- XGB Regressor

We will select the best performing model using the score of test and train data.

```
lin_regression = LinearRegression()
lasso = Lasso()
ridge = Ridge()
randomForest = RandomForestRegressor()
knn = KNeighborsRegressor()
xgb = XGBRegressor()
reg_models = [lin_regression,lasso,ridge,randomForest,knn,xgb]
print(reg_models)
```

Model Comparison

We have compared the model scores out of which the best performing model is Random Forest. We will further optimize the model.

```
LinearRegression()
{'Train Accuracy': 82.11932546974231, 'Test Accuracy': 81.06079756779152}

Lasso()
{'Train Accuracy': 81.19700361216957, 'Test Accuracy': 79.63998560202758}

Ridge()
{'Train Accuracy': 82.11888922769258, 'Test Accuracy': 81.06013548865556}

RandomForestRegressor()
{'Train Accuracy': 99.500602754688, 'Test Accuracy': 95.96604804475997}

KNeighborsRegressor()
{'Train Accuracy': 87.53965052463826, 'Test Accuracy': 78.47109081736954}

XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
             gamma=0, gpu_id=-1, importance_type=None,
             interaction_constraints='', learning_rate=0.300000012,
             max_delta_step=0, max_depth=6, min_child_weight=1, missing=nan,
             monotone_constraints='()', n_estimators=100, n_jobs=8,
             num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0,
             reg_lambda=1, scale_pos_weight=1, subsample=1, tree_method='exact',
             validate_parameters=1, verbosity=None)
{'Train Accuracy': 99.90402414344662, 'Test Accuracy': 95.59719842068968}
```

Model Selection

We have selected the Random Forest regressor as it was giving the highest accuracy for the test data. But we can see that the model is being overfit.

Model Improvement

We have scaled the data before we fit the data to the model and we will improve the model using the Randomized Search CV.

```
RF_ht = RandomForestRegressor()

params_rf = {"n_estimators": np.arange(60,90,1),"max_depth": np.arange(5,20,1),
         "max_features":np.arange(5,15,1),'min_samples_leaf': range(4, 8, 1),
    'min_samples_split': range(10, 20, 1)}
```

```
rf_random = RandomizedSearchCV(estimator = RF_ht, param_distributions = params_rf,
              n_iter = 5, cv = 5, verbose=2, random_state=35, n_jobs = -1)
rf_random.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 5 candidates, totalling 25 fits
```

```
C:\Users\Naiomi\anaconda3\py\lib\site-packages\sklearn\model_selection\_search.py:922: UserWarning: One or more of the test sco
res are non-finite: [0.94725422 0.94257706        nan 0.94564258 0.94322984]
  warnings.warn(
```

```
RandomizedSearchCV(cv=5, estimator=RandomForestRegressor(), n_iter=5, n_jobs=-1,
                   param_distributions={'max_depth': array([ 5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]),
                                        'max_features': array([ 5,  6,  7,  8,  9, 10, 11, 12, 13, 14]),
                                        'min_samples_leaf': range(4, 8),
                                        'min_samples_split': range(10, 20),
                                        'n_estimators': array([60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75,
76,
       77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89])},
                   random_state=35, verbose=2)
```

Using crossvalidation we found out the best parameters for Random Forest Regressor.

```
print (rf_random.best_params_)
print (rf_random.best_score_)
```

```
{'n_estimators': 63, 'min_samples_split': 16, 'min_samples_leaf': 6, 'max_features': 11, 'max_depth': 14}
0.9472542226952712
```

But when we use the best parameters the model is still being overfit. So we reduced the max depth to 5.

```
: model.fit(X_train, y_train)
```

```
: RandomForestRegressor(max_depth=5, max_features=11, min_samples_leaf=6,
                        min_samples_split=16, n_estimators=63)
```

The final model which we got is with the accuracy of 92% for test data.

```
model.score(X_train, y_train) * 100
```

```
94.032380806992
```

```
model.score(X_test, y_test) * 100
```

```
92.08609567103056
```

Conclusion

We were able to create and train a model. And since the model was overfitting we optimized the model and trained it so that we got the maximum accuracy without overfitting the model too much.