IEOR242 Final Project Report

Group 7: Boya Shao (3039713094), Jialin Wang (3039679866),
Ketong Chen (3039669817), Xuanyang Wu (3034617548),
Yihan Liao(3039672430), Zilan Luo(3039670051)

December 12, 2023

Contents

1 Introduction

The observed disparity in life expectancy between the African region and the global average has drawn our attention. An analysis of life expectancy data reveals that the mean life expectancy in the African region is determined to be 57.7 years. This figure stands in contrast to the global average life expectancy provided by Data Commons, which is calculated at 71.3 years in 2021. The observed disparity underscores a noteworthy discrepancy between life expectancy in the African region and the global average, highlighting a substantial difference in health outcomes and demographic patterns between these two demographic entities. This stark contrast in life expectancy serves as a compelling research topic for the insurance company considering entering the African market.

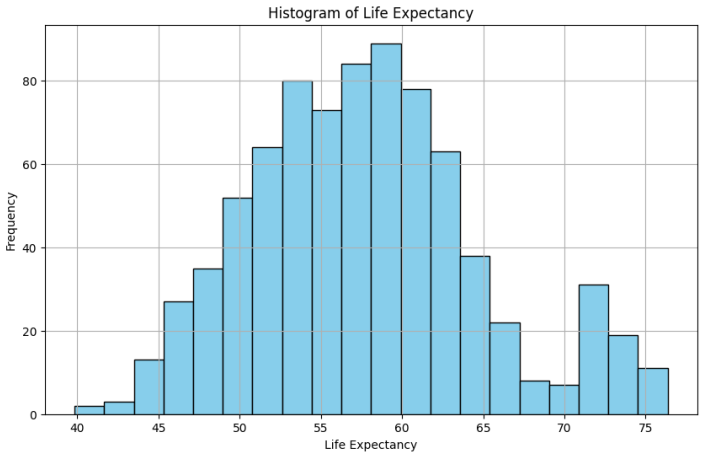


Figure 1: Histogram for Life Expectancy at Birth

Our study aims to dissect and analyze the various factors influencing life expectancy in the African region. This analysis is not only of academic interest but also holds practical significance for an international insurance company planning to expand into the African market. By identifying the critical determinants of life expectancy, we aim to provide the company with valuable insights that can inform their market entry strategy and policy planning. This research could serve as a precursor to more comprehensive market research, ultimately aiding the company in tailoring its products and services to meet the unique needs of the African demographic.

2 Objective

The report analyzes the life expectancy data integrated from GHO (Global Health Observatory) and UNESCO (United Nations Educational Scientific and Culture Organization). The main problem we would like to research is: **find the best parameters and model to predict the life expectancy in Africa**.

We would like to use the predicted life expectancy as the main tool to decide the potential market in Africa for a life insurance product. Given the expected life, the insurance company can design the premium amount and death benefit. With the selected significant parameters, the insurance company could decide which factors should be collected in Africa for the underwriting process.

3 Data and Techniques

3.1 Data Description

Our models will be built using the WHO National Life Expectancy dataset retrieved from Kaggle (https://www.kaggle.com/datasets/mmattson/who-national-life-expectancy), which covers information on 183 countries from the years 2000 to 2016. The dataset has 3111 observations in total and 799 observations for Africa; it has 32 columns. Aside from the original Kaggle dataset, we also add two columns "net migration" and "suicide mortality rate (per 100,000 population)" for these countries, which are collected by "The World Bank" website.

As our project emphasizes on African area, we will only use observations from the African region to train, validate, and test our model. We will apply predictor variables such as national GDP, BMI, year of schooling etc., to predict the life expectancy.

3.2 Proposed Models

The models we are going to experiment and compare include Ordinary Linear Regression (OLS), Categorical Linear Regression, Ridge Regression, LASSO, Random Forest Regressor and AdaBoost Regressor. For each model, we will conduct cross-validation to find the best hyperparameter that yields the best prediction outcome (high out-of-sample R^2 or low MSE). Then, we will select the best regression model to do the life expectancy prediction, evaluate the prediction results, and interpret important features. Our model selection is comprehensive since it contains traditional linear regression, bagging as well as boosting.

3.2.1 Ordinary Linear Regression

We define the estimators in the ordinary linear regression settings where we have a continuous response $\mathbf{Y} \in \mathbb{R}^p$, a design matrix X of size $n \times p$, and a parameter vector $\theta \in \mathbb{R}^p$. The ordinary linear regression is defined as:

$$\min_{\theta \in \mathbb{R}^n} \|X\theta - \mathbf{Y}\|^2 \tag{1}$$

Additionally, we could create dummy variables for the "country" to perform categorical linear regression, to investigate the difference between countries in Africa.

3.2.2 Ridge Regression

Ridge regression uses ℓ_2-norm penalty to do variable selection. It is easy to compute and has an explicit solution, which is also the unique global optimal solution. The coefficients of ridge regressions are relatively small, preventing overfitting and providing a more stable and reliable model prediction. The ridge regression is defined as:

$$\min_{\theta \in \mathbb{R}^n} \|X\theta - \mathbf{Y}\|^2 \text{ s.t. } \|\theta\|^2 \leq t$$

3.2.3 LASSO

LASSO, a widely used variable selection method in high-dimensional statistics, performs ℓ_1-norm penalty on the coefficients of OLS. And compared with ridge regression, it has no explicit solution, but better promotes sparsity in estimating the coefficient. In this way, it is easier to interpret the selected predictors. The LASSO is defined as:

$$\min_{\theta \in \mathbb{R}^n} \|X\theta - \mathbf{Y}\|^2 \text{ s.t. } \|\theta\|_1 \leq t \tag{2}$$

3.2.4 Random Forest Regressor

By building a regression tree, we recursively split the feature space to increase the purity. The decision tree $\hat{f}(\cdot)$ is defined as:

$$\hat{f}(\cdot) = \text{argmin}_{f(\cdot)\in\mathcal{F}}[\underbrace{\frac{1}{n}\sum_{i=1}^{n}\mathcal{L}(Y_i, f(X_i))}_{\text{Loss/impurity}} + \underbrace{R(f)}_{\text{Penalty/Complexity}}] \tag{3}$$

where $\mathcal{L}(\cdot,\cdot)$ is the loss function, $R(f)$ is the penalization. $R(f) = cp \cdot T$, where T is the number of leaves in the tree f. cp is the complexity parameter, which implies that each additional leaf should decrease the loss by at least cp. $\mathcal{F}$ is the set of all regression trees. The squared loss of a leaf node τ is given by:

$$\mathcal{L}(\tau) = \frac{1}{|\tau|}\sum_{(X,Y)\in\mathcal{D}(\tau)}(Y - \bar{Y}_\tau)^2 \tag{4}$$

where $\bar{Y}_\tau := \frac{1}{|\tau|}\sum_{(X,Y)\in\mathcal{D}(\tau)} Y$ is the mean outcome of the node τ. The squared loss of a non-leaf node τ is given recursively by:

$$\mathcal{L}(\tau) = \frac{|\tau_L|}{|\tau|}\mathcal{L}(\tau_L) + \frac{|\tau_R|}{|\tau|}\mathcal{L}(\tau_R) \tag{5}$$

The prediction of the regression tree is thus $\hat{Y} = \hat{f}(X) = \bar{Y}_\tau = \frac{1}{|\tau|}\sum_{(X,Y)\in\mathcal{D}(\tau)} Y$.

Random Forest belongs to the bagging strategy. The process can be summarized as:

- Bootstrapping: Selecting m random samples n data points from the training set $\mathcal{D}$ with replacement, denoted as $\{\mathcal{D}_1, \mathcal{D}_2, ..., \mathcal{D}_m\}$.
- Constructing a random forest: On each bootstrapped $\mathcal{D}_k$ $(k = 1, 2, ..., m)$, train a regression tree $\hat{T}_k$ using recursive partitioning, in which each split is only on a random subset of the features $\{1, 2, ..., p\}$. The collection of fitted decision trees $\hat{RF} = \{\hat{T}_1, \hat{T}_2, ..., \hat{T}_m\}$ forms a random forest.
- Aggregate the results of all the random trees. Take the average of the outcomes.

3.2.5 AdaBoost Regressor

The AdaBoost Regressor minimizes the weighted sum of squared errors, where N is the number of data points, D is the set of data points, M is the number of weak regressors, and α_m is the weight assigned to each weak regressor $f_m(x)$:

$$\text{AdaBoost Regressor}(x) = \sum_{m=1}^{M}\alpha_m f_m(x) \tag{6}$$

The objective is to find the α_m values and the weak regressors $f_m(x)$ that minimize the following weighted sum of squared errors:

$$\text{minimize} \quad \sum_{i=1}^{N}\left(y_i - \sum_{m=1}^{M}\alpha_m f_m(x_i)\right)^2 \tag{7}$$

$$\text{s.t.} \quad \alpha_m \geq 0 \quad \text{for}\, m = 1, 2, \ldots, M \tag{8}$$

$$\sum_{m=1}^{M}\alpha_m = 1 \tag{9}$$

In each iteration, AdaBoost Regressor iteratively updates the weights of the weak regressors and selects the best weak regressor to reduce the weighted sum of squared errors.

4 Data Analysis

4.1 Data Cleaning and Preparation

For the dataset, we first load the African data and drop columns that have more than thirty percent missing values. And then we drop columns that have the duplicate meaning as "life expectancy", and useless variables such as "country code". For the remaining missing values, we fill them with entries in the previous rows. For features like "hepatitis", it is difficult to fill in null entries since for some specific countries, there is no related data at all. So we just drop the column. Aside from the original Kaggle dataset, we also add two columns "net migration" and "suicide mortality rate (per 100,000 population)" for these countries, which are collected by "The World Bank" website.

Based on the correlation matrix, we observe that une_infant and infant_mort, une_gni and gni_capita are highly correlated (correlation are 0.99 and 0.93 respectively). We then drop one of the features in the pair.

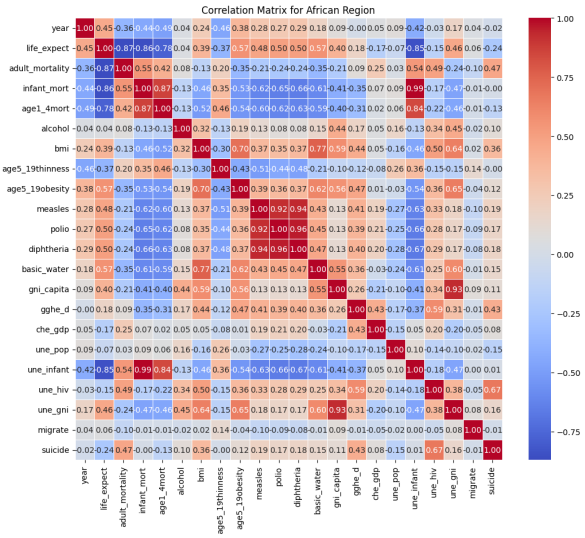


Figure 2: Correlation Matrix

Moreover, we separate data into the training set and testing set with regard to the "year" variable: years before 2012 are set to be training data, and years after 2012 are set to be testing data. This separation sets 75% of the data into training data and 25% into testing data, which is a reasonable percentage.

Lastly, we perform data normalization to both the training and testing dataset and create dummy variables with respect to the "country" for the categorical linear regression. By applying data normalization, we give an equal evaluation on the features with different scales, and this is very effective with respect to regularization methods like "Ridge Regression" and "LASSO".

We set out-of-sample R^2 as the measurement for the model performance. A larger out-of-sample R^2 usually represents a better regression model. Also, we take model sparsity into consideration. A larger sparsity usually represents better model interpretability.

4.2 Model Tuning

4.2.1 Ordinary Linear Regression

Since it is OLS, no hyperparameter tuning is needed. According to the result, the most significant predictors of life expectancy include adult mortality, infant mortality, age1-4 mortality and others. Each of these factors has a strong, statistically significant relationship with life expectancy.

4.2.2 Categorical Linear Regression

We introduce dummy variables for the "country" feature, which adds 46 "0-1" variables to the training data. Adding country dummies has given us insights into country-specific variations. Countries like Malawi and Uganda show a positive association with life expectancy, while others like Lesotho and Nigeria indicate a negative association, controlling for other factors. Also, it improves our out-of-sample R^2 value to 0.9881.

4.2.3 Ridge Regression

On the previous dataset with dummies, we first perform cross-validation to find the best penalization hyperparameter, which is 6, with the best mean cross-validated score. Using the best hyperparameter, we train the model with 64 variables and get the highest out-of-sample R^2: 0.9893.

4.2.4 LASSO

Similarly, we first perform cross-validation and test a range of alpha values to find the best penalization hyperparameter, which is 0.009. Using the best hyperparameter, we train the model and get the out-of-sample R^2 for model comparison, but the model only chooses 8 variables in the end.

4.2.5 Random Forest Regressor

A range of *ccp_alpha* values is performed using cross-validation to maximize the R^2 score. The best *ccp_alpha* identified by the grid search is 0, which means no pruning was performed, and the full complexity of the model is retained. This suggests that for our specific dataset, the addition of pruning does not improve the model's performance on the cross-validated sets. The out-of-sample R^2 score for the model with the chosen *ccp_alpha* of 0 is approximately 0.9792.

4.2.6 AdaBoost Regressor

Through cross-validation, the model tuning with grid search identifies 200 trees (weak learners) as the optimal size for our ensemble, providing the best balance between performance and complexity. The AdaBoost model with 200 estimators achieves an out-of-sample R^2 score of approximately 0.9779. However, interestingly, the model chooses only 20 of the variables to make predictions.

4.3 Model Performance Evaluations

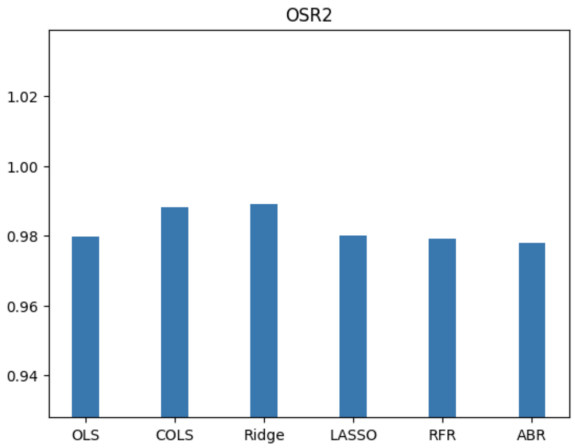
The out-of-sample R^2, sparsity for the Ordinary Linear Regression, Categorical Linear Regression, Ridge Regression, LASSO, Random Forest Regressor and AdaBoost Regressor are shown in the following Table 1 and Figure 3 for comparison:

Model Performance						
Model Name	OLS	Categorical OLS	Ridge	LASSO	Random Forest	AdaBoost Regressor
Out-of-sample R^2 (OSR2)	0.9798	0.9881	0.9893	0.9799	0.9792	0.9779
Number of features	18	64	64	8	64	20

Table 1: Model Overall Performance

4.4 Best Model Selection

From the view of the out-of-sample R^2, Ridge Regression is the best model in terms of prediction accuracy because it has the highest out-of-sample R^2. From the view of the sparsity, LASSO has the best performance since sparse features boost the model interpretability, and LASSO also has a relatively high out-of-sample R^2.



[0.979815145214685, 0.9880614819485806, 0.9891598969889291, 0.9799261580576188, 0.9791993059354297, 0.9779059649380244]

Figure 3: OSR2 for Overall Performance

4.5 Analysis and Discussion

4.5.1 Import Features

For all the models that we have run (except for the Categorical OLS regression), the top 3 most important features are the same:

- adult_mortality: Adult Mortality Rates of both sexes (probability of dying between 15 and 60 years per 1000 population)
- infant_mort: Death rate up to age 1
- age1_4mort: Death rate between ages 1 and 4

In Categorical OLS regression, the shared most important feature is adult_mortality, followed by country dummy variables.

The three most crucial features are highly relevant in real-life scenarios, as they pertain to the mortality of individuals across various age groups. In the field of actuarial science, mortality stands out as one of the paramount factors for calculating premiums and expenses. Consequently, all the models encompass the most vital statistics that must be collected for a prospective insurance company."

4.5.2 Best Model Application

We would like to use LASSO as our best model to predict the life expectancy. The following features are involved in LASSO:

- adult_mortality: Adult Mortality Rates of both sexes (probability of dying between 15 and 60 years per 1000 population)
- infant_mort: Death rate up to age 1
- age1_4mort: Death rate between ages 1 and 4
- age5_19thinness: Prevalence of thinness among children and adolescents, BMI $<$ (median - 2 s.d.) (crude estimate) (%)
- age5_19obesity: Prevalence of obesity among children and adolescents, BMI $>$ (median + 2 s.d.) (crude estimate) (%)
- gni_capita: Gross national income per capita (PPP int. $)
- gghe_d: Domestic general government health expenditure (GGHE-D) as percentage of gross domestic product (GDP)

- suicide: Suicide mortality rate (per 100,000 population)

In addition to accounting for top mortality factors, LASSO also takes into consideration health and economic factors, as well as suicide mortality. Health factors such as age5_19thinness and age5_19obesity appear to be distinctive elements influencing life expectancy, particularly tailored to the African region. Economic factors, including gni_capita and gghe_d, are also integral components contributing to vital factors influencing life expectancy. The results lead us to the conclusion that acquiring fundamental knowledge about factors such as the insured individual's degree of obesity, mental health condition, and the country's commitment to healthcare is imperative. These insights underscore the necessity of collecting comprehensive information to better understand and assess life expectancy dynamics.

4.5.3 Life Expectancy of 60

For exploring the potential market of annuity and term life insurance for the elderly, we first try to predict the life expectancy of people of 60.

Model Name	OLS	Categorical OLS	Ridge	LASSO	Random Forest	AdaBoost Regressor
Out-of-sample R^2 (OSR2)	0.5599	0.6129	0.5594	0.5560	0.6227	0.6379

Table 2: Model Performance for life_exp60

As we can see from the table above, using the models that we implement life expectancy at birth as the dependent variable does not give a prediction of life expectancy of 60 as accurate as the life expectancy at birth. We can know that predicting the life expectancy of the elderly does not share the similar features as predicting the life expectancy at birth.

We then try to rerun the whole process, replacing the dependent variable life_expect by life_exp60. The following figure shows the average out-of-sample R^2 is better than the one of direct prediction from the original models. The original best model LASSO becomes the least accurate one. Categorical OLS regression, Ridge regression, and Random Forest share similar OSR^2; however, they may experience overfitting. AdaBoost Regressor becomes the most efficient one considering both high R^2 and relatively less flexibility.

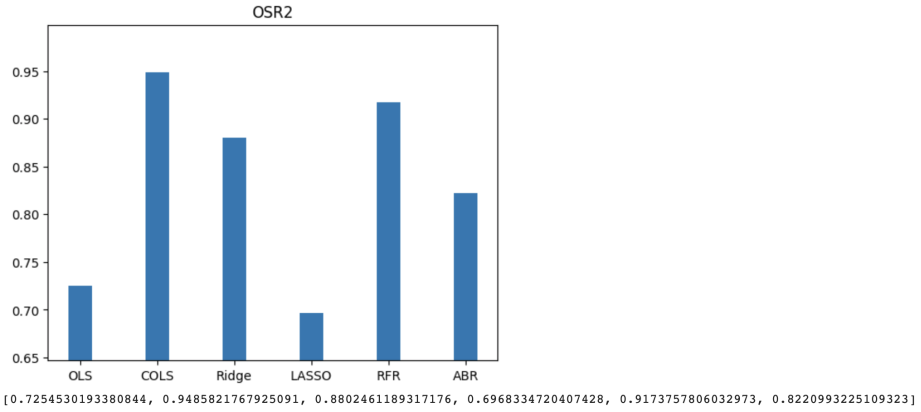


Figure 4: OSR2 for life_exp60 as dependent variable

We may consider finding better models to predict the life expectancy of 60 for insurance companies to develop insurance products with fewer costs in data collection.

5 Project Impact

With excellent prediction models and important features of life expectancy in hand, the project would have the following impacts:

Market Opportunity: Africa is a rapidly developing market with a growing middle class and increasing economic stability. By offering life insurance tailored to the needs of African populations, insurance companies can tap into a relatively untapped market and potentially see significant growth.

Data-Driven Insights: Collecting data to estimate life expectancy and understand the factors influencing it can provide valuable insights for both insurers and policymakers. This information can be used to develop more targeted insurance products and implement public health interventions.

Long-Term Business Sustainability: Expanding into new markets, especially emerging ones like Africa, can diversify an insurance company's portfolio and contribute to its long-term sustainability and growth.

6 Reference

Bagus, U., Girancourt, F. J. de, Mahmood, R., & Manji, Q. (2020, December 16). Africa's insurance market is set for takeoff. McKinsey & Company. https://www.mckinsey.com/featured-insights/middle-east-and-africa/africas-insurance-market-is-set-for-takeoff

Data Commons. (n.d.). Life Expectancy. Data Commons Timelines. https://datacommons.org/tools/timeline#&place=Earth&statsVar=LifeExpectancy_Person

Digital, 3B. (2021, June 9). Expanding Africa's insurance sector with Mikir Shah, CEO, Africa Specialty Risks. Invest Africa. https://www.investafrica.com/insights-/expanding-africas-insurance-sector

Health Nutrition and population statistics. DataBank. (n.d.). https://databank.worldbank.org/reports.aspx?source=health-nutrition-and-population-statistics

MMattson. (2020, October 6). Who national life expectancy. Kaggle. https://www.kaggle.com/datasets/mmattson/who-national-life-expectancy

7 Appendix (Code Link)

Code for life_expect as dependent variable

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
from statsmodels.stats.outliers_influence import variance_inflation_factor

# Load Afirca Data
# data=pd.read_csv('who_life_exp.csv')
# data_whole=data.loc[data['region'] == 'Africa']
# # print(data_whole.info())

data_whole = pd.read_excel('new_data.xlsx')
# print(data_whole.head())

# Calculate the percentage of null values in each column
null_percentages = (data_whole.isnull().sum() / len(data_whole)) * 100

# Identify columns with more than 30% null values
columns_to_drop = null_percentages[null_percentages > 30].index

# Drop the identified columns from the DataFrame
print('Columns to drop: ', columns_to_drop)
data_new = data_whole.drop(columns=columns_to_drop)

# Print the modified DataFrame
# print(data_new.info())

# Drop more columns
data_new2 = data_new.drop(columns = ['country_code','region','life_exp60','une_life','
    hepatitis'])
# print(data_new2.info())

# Fill null values using previous values
df = data_new2.fillna(method='ffill')
# print(df.info())

# Rename columns
df.rename(columns={'age1-4mort':'age1_4mort', 'age5-19thinness':'age5_19thinness', '
    age5-19obesity':'age5_19obesity', 'gghe-d':'gghe_d'},inplace=True)

# Data multicollinearity
# Extract only numeric columns
dfX = df.copy().drop(columns = 'life_expect')
numeric_columns = dfX.select_dtypes(include=['float64', 'int64']).columns
X = dfX[numeric_columns]

# Calculate VIF for each independent variable
vif_data = pd.DataFrame()
vif_data["Variable"] = X.columns
vif_data["VIF"] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]

# Display VIF data
print("VIF Data for African Region:")
print(vif_data)

# Create a correlation matrix
correlation_matrix = df.corr()

# Plot a heatmap of the correlation matrix
plt.figure(figsize=(12, 10))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f", linewidths
    =0.5)
plt.title("Correlation Matrix for African Region")
plt.show()

# Drop highly correlated variates
df = df.drop(columns = ['une_infant', 'une_gni'])
```

```

# Check life_expect statistics
life0 = df['life_expect']

basic_stats = life0.describe()

medlife0 = life0.median()
varlife0 = life0.var()

print(basic_stats)
print(f"Median Life Expectancy: {medlife0}")
print(f"Variance of Life Expectancy: {varlife0}")

# Plot a histogram for Life_Expectancy
plt.figure(figsize=(10, 6))
plt.hist(life0, bins=20, color='skyblue', edgecolor='black')
plt.title('Histogram of Life Expectancy')
plt.xlabel('Life Expectancy')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()

# Separate into training and testing set
data_train = df[df['year'] <= 2012]
data_test = df[df['year'] >= 2013]

len(data_train), len(data_test)

# Form dataframe
x_train = data_train[['adult_mortality', 'infant_mort', 'age1_4mort', 'alcohol', 'bmi
    ', 'age5_19thinness', 'age5_19obesity', 'measles', 'polio', 'diphtheria', '
    basic_water', 'gni_capita', 'gghe_d', 'che_gdp', 'une_pop', 'une_hiv','migrate', '
    suicide']]
y_train = data_train['life_expect']

x_test = data_test[['adult_mortality', 'infant_mort', 'age1_4mort', 'alcohol', 'bmi',
    'age5_19thinness', 'age5_19obesity', 'measles', 'polio', 'diphtheria', 'basic_water
    ', 'gni_capita', 'gghe_d', 'che_gdp', 'une_pop', 'une_hiv', 'migrate', 'suicide']]
y_test = data_test['life_expect']

# Data normalization
mean_x = x_train.mean()
std_x = x_train.std()
normalized_train_x = (x_train - mean_x) / std_x

mean_y = y_train.mean()
std_y = y_train.std()
normalized_train_y = (y_train - mean_y) / std_y

mean_x_test = x_test.mean()
std_x_test = x_test.std()
normalized_test_x=(x_test - mean_x_test) / std_x_test

mean_y_test = y_test.mean()
std_y_test = y_test.std()
normalized_test_y = (y_test - mean_y_test) / std_y_test

# Add categorical column and dummies: country
normalized_train_x_cat = normalized_train_x.copy(deep = True)
normalized_train_x_cat['country'] = data_train['country']
normalized_train_x_cato = pd.get_dummies(normalized_train_x_cat, columns=['country'],
    drop_first = True)

normalized_test_x_cat = normalized_test_x.copy(deep = True)
normalized_test_x_cat['country'] = data_test['country']
normalized_test_x_cato = pd.get_dummies(normalized_test_x_cat, columns=['country'],
    drop_first = True)

# OSR2
def OSR2(model, train_y, test_x, test_y):
    y_test = test_y
```

```
    y_pred = model.predict(test_x)
    SSE = np.sum((y_test - y_pred)**2)
    SST = np.sum((y_test - np.mean(train_y))**2)
    return 1 - SSE/SST


# Basic Linear Model
import statsmodels.api as sm
model0 = sm.OLS(normalized_train_y, normalized_train_x).fit()
print(model0.summary())
score0 = OSR2(model0, normalized_train_y, normalized_test_x, normalized_test_y)
print(score0)

# Catogorical Linear Model
model1 = sm.OLS(normalized_train_y, normalized_train_x_cato).fit()
print(model1.summary())
score1 = OSR2(model1, normalized_train_y, normalized_test_x_cato, normalized_test_y)
print(score1)

from sklearn.linear_model import Ridge
from sklearn.model_selection import GridSearchCV

# Define Ridge Regression model
ridge_model = Ridge()

# Define a range of alpha values to test
alphas = np.array(range(1,10,1))

# Create a parameter grid for GridSearchCV
param_grid = {'alpha': alphas}

# Perform GridSearchCV with cross-validation
grid_search = GridSearchCV(ridge_model, param_grid, scoring='neg_mean_squared_error',
    cv=5)
grid_search.fit(normalized_train_x_cato, normalized_train_y)

# Print the best alpha and corresponding mean cross-validated score
best_alpha = grid_search.best_params_['alpha']
best_score = grid_search.best_score_

print(f'Best Alpha: {best_alpha}')
print(f'Best Mean Cross-Validated Score: {best_score:.4f}')

# OSR2
clf1 = Ridge(alpha=best_alpha)
model5 = clf1.fit(normalized_train_x_cato, normalized_train_y)
score5 = OSR2(model5, normalized_train_y, normalized_test_x_cato, normalized_test_y)
print(score5)

# Feature Importance
importance = model5.coef_
feat_importances = pd.Series(importance, index = normalized_train_x_cato.columns)
filtered_series = feat_importances[feat_importances != 0]
filtered_series.plot(kind='barh')
plt.show()
print(filtered_series)
print(len(filtered_series))

from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.linear_model import LassoCV
from sklearn import linear_model

# Initialize the LassoCV model with a list of alpha values to test
alphas = np.array([0.005,0.006,0.007,0.008,0.009,0.01,0.02,0.03,0.04,0.05])
lasso_cv_model = LassoCV(alphas=alphas, cv=5)

# Fit the LassoCV model
lasso_cv_model.fit(normalized_train_x_cato, normalized_train_y)

# Print the best alpha value
best_alpha = lasso_cv_model.alpha_
```

```
print(f'Best alpha: {best_alpha}')

# OSR2
clf2 = linear_model.Lasso(alpha=best_alpha)
model2 = clf2.fit(normalized_train_x_cato, normalized_train_y)
score2 = OSR2(model2, normalized_train_y, normalized_test_x_cato, normalized_test_y)
print(score2)

# Feature Importance
importance2 = model2.coef_
feat_importances2 = pd.Series(importance2, index = normalized_train_x_cato.columns)
filtered_series2 = feat_importances2[feat_importances2 != 0]
filtered_series2.plot(kind='barh')
plt.show()

# print the used features in LASSO
print(filtered_series2)

from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GridSearchCV

param = {'ccp_alpha':[0,0.001,0.002,0.003,0.004,0.005,0.01]}

RegRF = RandomForestRegressor(random_state=333)
grid_rf = GridSearchCV(RegRF, param,scoring='r2',cv=5).fit(normalized_train_x_cato,
    normalized_train_y)

print("The best complexity parameter is ",grid_rf.best_params_)

# OSR2
RF = RandomForestRegressor(ccp_alpha = grid_rf.best_params_['ccp_alpha'])
model3 = RF.fit(normalized_train_x_cato, normalized_train_y)
score3 = OSR2(model3, normalized_train_y, normalized_test_x_cato, normalized_test_y)
print(score3)

# Feature Importance
importance3 = model3.feature_importances_
feat_importances3 = pd.Series(importance3, index = normalized_train_x_cato.columns)
filtered_series3 = feat_importances3[feat_importances3 != 0]
filtered_series3.plot(kind='barh')
plt.show()
print(filtered_series3)
print(len(filtered_series3))

from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import AdaBoostRegressor
from sklearn.model_selection import GridSearchCV

param_grid = {
    'n_estimators': [50, 100, 200]}

AdaBoost_reg = AdaBoostRegressor(random_state=42)

grid_search = GridSearchCV(AdaBoost_reg, param_grid, cv=5, scoring='r2')
grid_search.fit(normalized_train_x_cato, normalized_train_y)

best_params = grid_search.best_params_
print("Best Parameters:", best_params)

# best_model = grid_search.best_estimator_
# score = OSR2(best_model, normalized_train_y, normalized_test_x_cato,
    normalized_test_y)
# print("OSR2 Score on Test Data:", score)

from sklearn.ensemble import AdaBoostRegressor

AdaBoost_reg = AdaBoostRegressor(n_estimators=best_params['n_estimators'],
    random_state=42)
model4 = AdaBoost_reg.fit(normalized_train_x_cato, normalized_train_y)
score4 = OSR2(model4, normalized_train_y, normalized_test_x_cato, normalized_test_y)
print(score4)
```

```
267
268 # Feature Importance
269 importance4 = model4.feature_importances_
270 feat_importances4 = pd.Series(importance4, index = normalized_train_x_cato.columns)
271 filtered_series4 = feat_importances4[feat_importances4 != 0]
272 filtered_series4.plot(kind='barh')
273 plt.show()
274
275 scores = [score0, score1, score5, score2, score3, score4]
276 models = ['OLS','COLS','Ridge','LASSO','RFR','ABR']
277 plt.bar(models,scores,width=0.3)
278 plt.ylim(bottom=min(scores)-0.05,top=max(scores) + 0.05)
279 plt.title('OSR2')
280 plt.show()
281 print(scores)
```

Code for life_exp60 as dependent variable

```
1 # Keep life expectancy 60 as the data we want to predict
2 life60 = data_new['life_exp60']
3
4 # Append life_exp60 to the processed dataset
5 df2 = df.copy()
6 df2['life60'] = life60
7
8 # Take out the test set
9 y_life60_train = df2[df2['year'] <= 2012]['life60']
10 y_life60_test = df2[df2['year'] >= 2013]['life60']
11
12 # Normalize the life_exp60
13 mean_y60 = y_life60_train.mean()
14 std_y60 = y_life60_train.std()
15 normalized_y60 = (y_life60_train - mean_y60) / std_y60
16
17 mean_y60_test = y_life60_test.mean()
18 std_y60_test = y_life60_test.std()
19 normalized_y60_test = (y_life60_test - mean_y60_test) / std_y60_test
20
21 model_num = [model0, model1, model5, model2, model3, model4]
22 models = ['OLS','COLS','Ridge','LASSO','RFR','ABR']
23
24 # Create an empty list to store OSR2 scores for each model
25 osr2_scores = []
26
27 # Iterate over the models and calculate OSR2 for each
28 for model, model_name in zip(model_num, models):
29   if model == model0:
30     osr2 = OSR2(model, normalized_train_y, normalized_test_x, normalized_y60_test)
31   else:
32     osr2 = OSR2(model, normalized_train_y, normalized_test_x_cato, normalized_y60_test
     )
33   osr2_scores.append(osr2)
34   print(f"OSR2 for {model_name}: {osr2}")
35
36 # Print the list of OSR2 scores for all models
37 print("List of OSR2 scores for all models:", osr2_scores)
38
39 # so if we use the life_exp model to predict the life_exp60, all of the models
     proposed are not desirable as the OSR2 are not too high.
```