Life Expectancy Prediction Using Machine Learning

Team Members: Brandon Ismalej, Jittapatana (Patrick) Prayoonpruk, Mishek Sambahangphe

Abstract

This study focuses on predicting life expectancy using machine learning models, specifically comparing a linear regression model with a Box-Cox transformation (selected through stepwise regression) and a Categorical Boosting (CatBoost) model. The dataset, sourced from the World Health Organization (WHO) and the United Nations, includes socio-economic, health, and environmental factors for 193 countries between 2000-2015. The study applies data preprocessing techniques such as imputation and feature selection. Feature selection was performed using stepwise regression, Principal Component Analysis (PCA), and all-subset selection methods. The models' performance is evaluated using the metrics Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared (R^2). The results indicate that the CatBoost model outperformed the linear regression with Box-Cox transformation, demonstrating better accuracy and predictive power for life expectancy prediction.

Introduction

Life expectancy is a crucial health indicator that reflects the overall well-being of a population and is influenced by various socio-economic, lifestyle, and environmental factors. This report explores the development and evaluation of machine learning models to predict life expectancy using a dataset from the World Health Organization (WHO) and the United Nations, which comprises variables such as healthcare access, income, education levels, and mortality rates. The study applies data preprocessing techniques, including handling missing values, feature scaling, and feature selection, to prepare the dataset for analysis. Multiple machine learning algorithms, including regression models and gradient boosting methods are evaluated for their predictive performance.

Dataset

The dataset is a comprehensive collection of health, social, and economic indicators for 193 countries, covering the years 2000-2015. The dataset contains 2938 rows and 22 columns featuring variables such as immunization, education, economic, social, and other health-related factors.

Objectives

The primary objective of this study is to explore and analyze the Kaggle dataset to uncover patterns, trends, and key variables influencing life expectancy. The models are compared using metrics like Root Mean Squared Error (RMSE) and R-squared (R^2) to determine their accuracy and reliability. These objectives collectively ensure a comprehensive and impactful approach to life expectancy prediction.

Challenges and Motivation

1. Data Quality and Missing Values
    a. Handling missing or incomplete data is a significant challenge, as gaps in critical information can lead to biased or inaccurate predictions. Proper imputation and preprocessing techniques are essential to mitigate these issues.
2. Scalability and Generalization
    a. Ensuring that the model performs well on unseen data from different populations or regions is a crucial challenge, particularly for global applications.
3. Dataset Imbalance
    a. If certain classes or ranges of life expectancy are underrepresented, it can bias the model's predictions, necessitating the use of balancing techniques or weighted metrics.

Methodology

Dataset Description:

1. Target Variable
    a. Life Expectancy: A continuous variable representing the average number of years a person is expected to live.
2. Features:
    a. Country: The name of the country (categorical).
    b. Year: The year of the observation (numerical).
    c. Status: Categorical variable indicating whether the country is "Developed" or "Developing".
    d. Health Indicators:
        i. Adult Mortality: Probability of dying between 15-60 years, per 1,000 population.

    ii. Infant Deaths: Number of infant deaths per 1,000 live births.
    iii. BMI (Body Mass Index): Average body mass index of the population
    iv. HIV/AIDS: Deaths per 1,000 population due to HIV/AIDS.
    v. Measles: Number of reported cases of measles per 1,000 population.
    vi. Hepatitis B, Polio, and Diphtheria: Immunization coverage percentage among 1-year-olds (numerical).
  e. Socioeconomic Indicators:
    i. GDP: Gross Domestic Product per capita in USD.
    ii. Income Composition of Resources: A composite index reflecting access to income-related resources (numerical).
    iii. Schooling: Average number of years of schooling (numerical).
  f. Health Expenditure:
    i. Percentage Expenditure: Expenditure on health as a percentage of GDP per capita.
    ii. Total Expenditure: Total Expenditure on health as a percentage of GDP.
  g. Demographic Features:
    i. Population: Total population size of the country (numerical).
    ii. Under-Five Deaths: Number of deaths of children under the age of five, per 1,000 live births.
    iii. Thinness 1-19 Years: Prevalence of thinness among individuals ages 1 to 19 years (numerical)
    iv. Thinness 5-9 Years: Prevalence of thinness among individuals aged 5 to 9 years (numerical.
3. Data Characteristics:
  a. Geographic Scope: Includes countries worldwide, allowing for cross-country comparisons of life expectancy rates and influential factors.
  b. Temporal Coverage: Observations span multiple years, enabling temporal trends analysis of life expectancy rate changes over time.
  c. Missing Data: Contains missing values across several features, requiring imputation or handling strategies.

Preprocessing Steps:

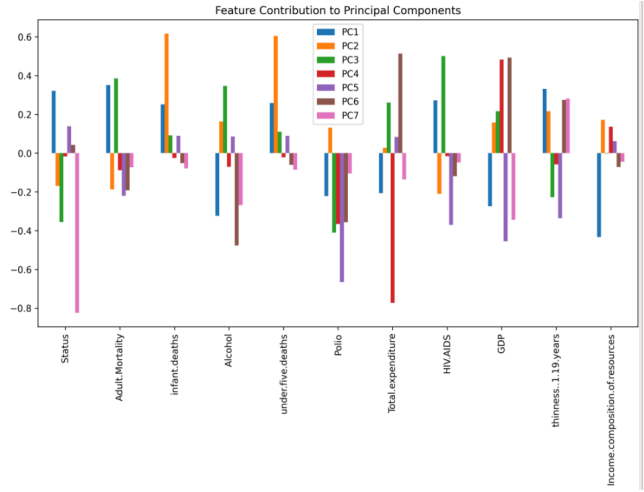
1. Choosing train/test set of data
  a. Use 2 years: Use one year as a training set and the next year as a test set.
    i. Year of 2015 is used in the test set
    ii. Year of 2014 is used in the train set
2. Imputation for missing data
  a. Missing values in the dataset were addressed using the missForest imputation technique, a non-parametric method based on Random Forests.
  b. The imputation process was performed with a maximum of 10 iterations and 100 trees per forest, ensuring robust and accurate predictions for the missing entries.
  c. The missForest approach was chosen due to its ability to handle mixed data types (categorical and continuous) and its robustness against overfitting or unrealistic imputations.

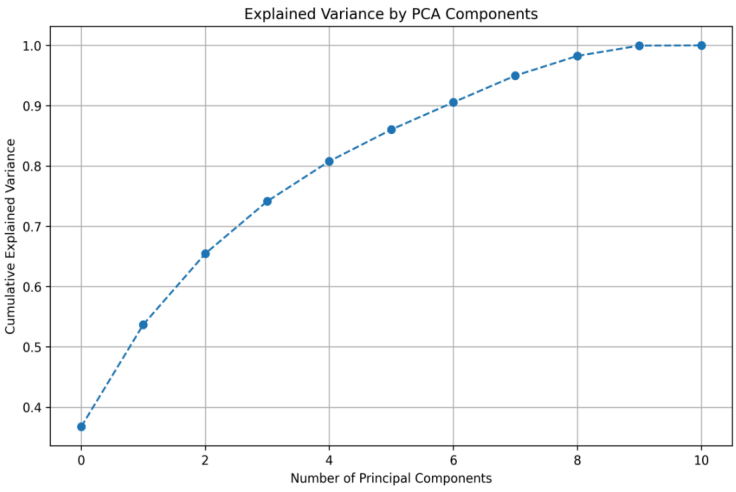
Feature Selection:

1. First Approach: Stepwise function (stepAIC)
    a. This method evaluates models iteratively by adding or removing predictors based on their contribution to the model's performance. The procedure was applied in both, forward, and backward directions, systematically exploring the variable space to optimize the Akaike Information Criterion (AIC).
2. Second Approach: All-Subset Regression
    a. all-subset regression was performed to thoroughly explore the variable space and identify the best combination of predictors for the model. This method evaluates all possible subsets of the predictors to determine which subset produces the optimal model based on predefined criteria, such as AIC and BIC
    b. Selected Features: Status, Adult mortality, Infant deaths, Alcohol, Under-five deaths, Polio, Total expenditure, HIV/AIDS, GDP, thinness 1-19 years, Income composition of resources

Dimension Reduction:

For select models (#5), Principal Component Analysis (PCA) was applied to reduce the dimensionality of the dataset, with only the features selected in the all-subset regression technique. This approach aimed to retain the majority of variance in the data while minimizing computational complexity. A cumulative explained variance threshold of 90% was used to determine the number of principal components to retain, which resulted in the selection of six principal components.





Model Selection:

1. Baseline Linear Regression: To establish a baseline, we performed an Ordinary Least Squares (OLS) regression using all available predictors in the dataset without any preprocessing or feature selection. The results demonstrate the initial capability of the model to explain variations in life expectancy, as summarized:
    a. Model Summary:

        i. R-Squared: 0.817 indicating that 81.7% of the variability in life expectancy can be explained by the predictors.
        ii. Adjusted R-Squared: 0.796, accounting for the number of predictors.
        iii. F-Statistic: 38.40, with a p-value of 2.85×10^{-50}, confirming the overall significance of the model.
    b. Key Predictors: Several key predictors were identified as having a significant p-value (<= 0.05)
        i. Adult.Mortality, HIV.AIDS, Income.composition.of.resources, Alcohol.consumption, Status

2. Multiple Regression: The optimal predictive model was selected through stepwise selection using the stepAIC method.
    a. Model Summary
        i. R-Squared: 0.8695 indicating that 87% of the variability in life expectancy can be explained by the predictors
        ii. Adjusted R-Squared: 0.8666, accounting for the number of predictors.
        iii. F-statistic: 296.5, with a p-value of 2.2e-16
    b. Key Predictors
        i. Income.composition.of.resources, HIV.AIDS, Adult.Mortality, Total.expenditure
3. Multiple Regression with Box-Cox transformation: the Box-Cox transformation was also applied to the response variable (life expectancy) with Optimal Lambda: 1.030303
    a. Model Summary
        i. R-Squared: 0.8693 indicating that 86.9% of the variability in life expectancy can be explained by the predictors
        ii. Adjusted R-Squared: 0.8664, accounting for the number of predictors.
        iii. F-statistic: 296.1, with a p-value of 2.2e-16
    b. Key Predictors
        i. Income.composition.of.resources, HIV.AIDS, Adult.Mortality, Total.expenditure
4. Generalized Least Squares (GLS): a Generalized Least Squares (GLS) model was employed to account for potential heteroscedasticity and correlations within the data. The GLS approach allows for a more flexible handling of non-constant variance in the residuals, which can violate standard regression assumptions.
    a. Model Summary
        i. Residual standard error: 3.127172
    b. Key Predictors
        i. Income.composition.of.resources, HIV.AIDS, Adult.Mortality, Total.expenditure
5. CatBoost Regressor: The CatBoost algorithm was applied directly to the set of features selected using all subset regression techniques.
    a. Model Summary:
        i. Mean Squared Error: 3.6907
        ii. Root Mean Squared Error: 1.92112

        iii. R-Squared: 0.9438
6. CatBoost (Categorical Boosting) with PCA: Six principal components were utilized as predictors in a CatBoost regression model, which is a gradient boosting algorithm designed to handle categorical and continuous variables efficient
    a. Model Summary:
        i. Mean Squared Error: 7.5519, indicating a strong predictive performance with low error.
        ii. Root Mean Squared Error: 2.74807
        iii. R-Squared: 0.8849
    b. Key Insights: The CatBoost model effectively captured nonlinear relationships in the PCS-transformed feature space.
7. Linear Regression with PCA:
    a. Model Summary:
        i. Mean Squared Error: 10.2248
        ii. Root Mean Squared Error: 2.74807
        iii. R-Squared: 0.8442

Evaluation Metrics:

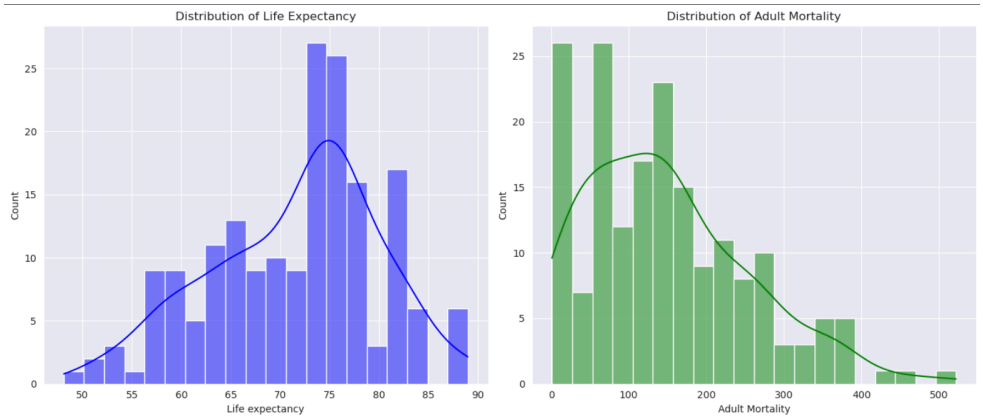
1. Root Mean Squared Error (RMSE)
    a. Multiple Regression with box-cox transformation shows the RMSE of 2.986278 or about 3 years off when doing the prediction
    b. GLS model shows the RMSE of 2.626593 or about 2.6 years off when doing the prediction
    c. CatBoost shows the RMSE of 1.92 or about 1.9 years off when doing the prediction

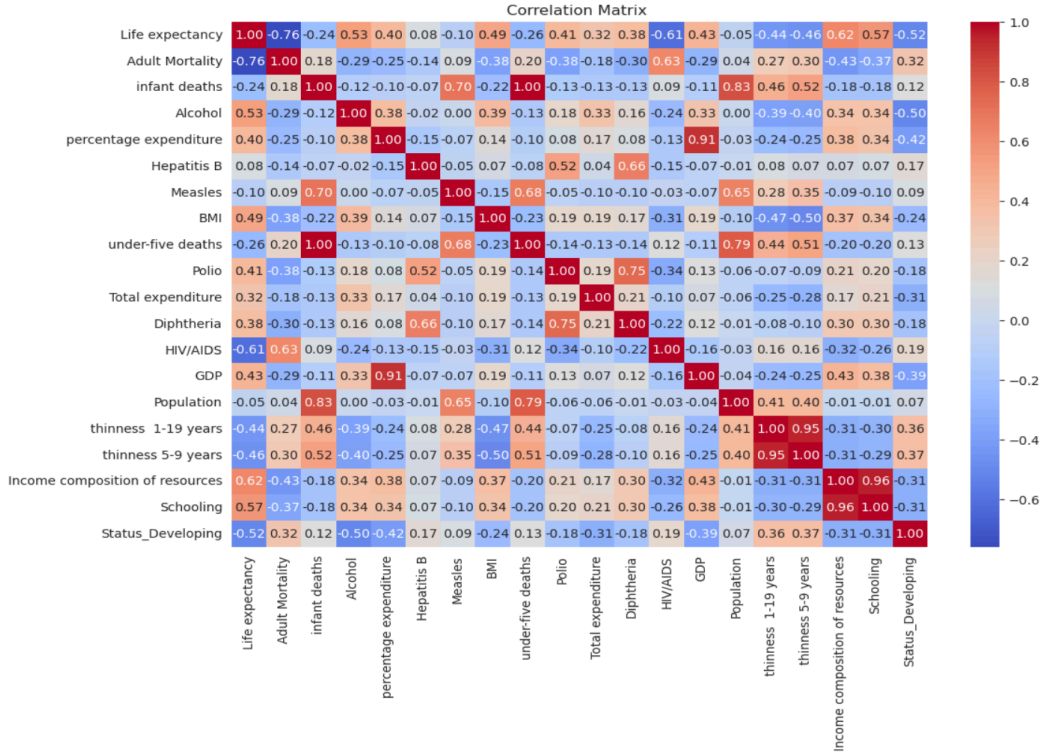
Data Analysis

Exploratory Data Analysis:

1. Summary Statistics: The dataset shows an average life expectancy of 71.54 years (range: 48.1 - 89.0). Key health and socioeconomic factors include:
    a. Adult Mortality: Mean of 148.69 per 1,000, negatively correlated with life expectancy (r = -0.76)
    b. Income Composition of Resources and Schooling: Positively correlated with life expectancy (r = 0.62 and 0.57, respectively)
    c. GDP and BMI: Show moderable variability and correlation
2. Correlation Analysis: A correlation matrix highlights:
    a. Strong positive correlations: Income Composition of Resources, Schooling.
    b. Strong negative correlations: Adult Mortality, HIV/AIDS.
3. Visual Analysis:

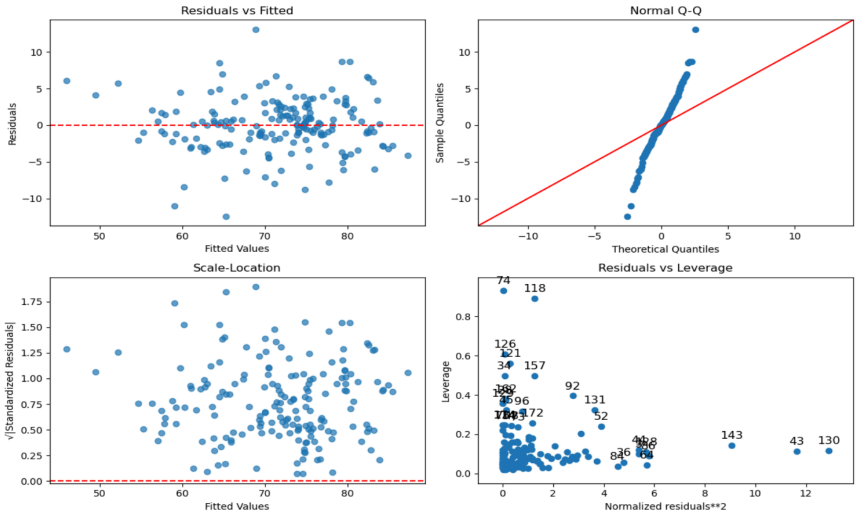
a. Life Expectancy: Unimodal Distribution with most values between 65 and 80 years.
b. Adult Mortality: Right-skewed distribution, with a small number of extreme outliers.





Diagnostics:

1. For the baseline regression model using all features:

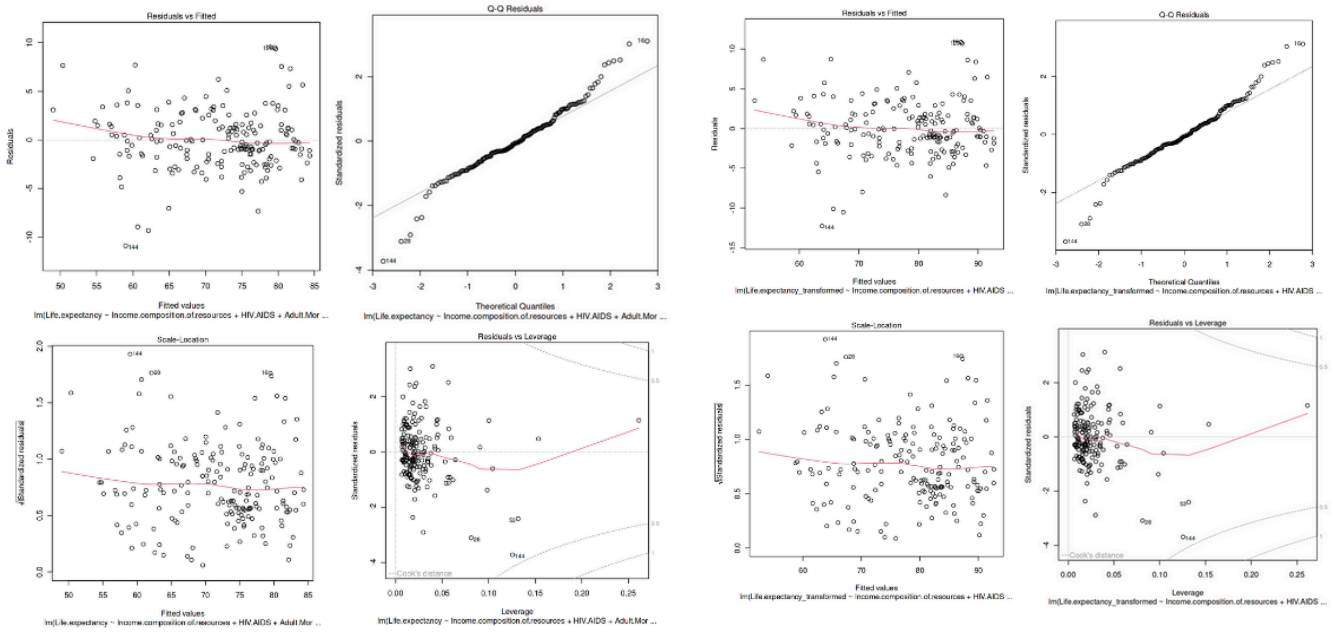


a.

b. Residual Analysis: The residuals versus fitted plot indicates a reasonable spread around zero, but some heteroscedasticity is evident. The Q-Q plots suggests deviations from normality at the tails.

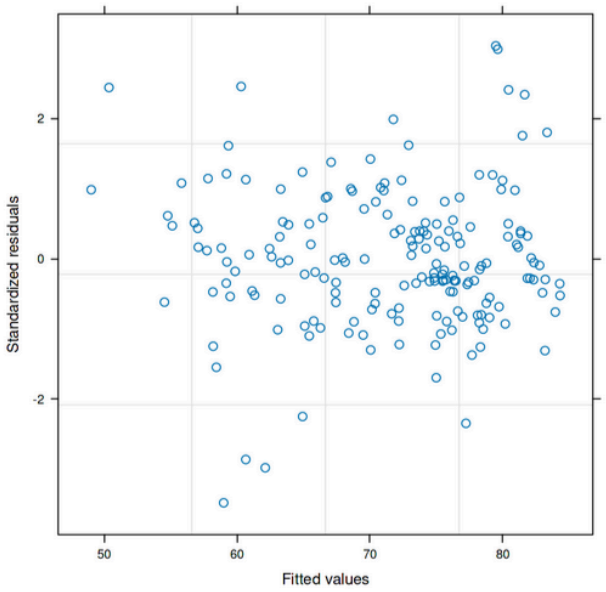
c. Leverage Points: Observations such as #74 and #118 shows high leverage and could influence the model disproportionately.

2. Comparing the diagnostic plots of multiple regression of untransformed model (left) with transformed model (right).



The transformation had minimal impact, as concerns about violations of linear assumptions—such as heteroscedasticity and non-normality of errors—persist.

3. The plot of the Generalized Least Squares (GLS)



Although the residuals show some scatter, there does not appear to be a clear systematic pattern. Heteroscedasticity could still be addressed depending on how significant we were to interpret.

Conclusion

Summary:

This study aimed to predict life expectancy using a comprehensive dataset of health, socioeconomic, and environmental factors for 193 countries spanning 2000-2015. A variety of machine learning models were implemented, including linear regression, Generalized Least Squares (GLS), and Catboost. Multiple preprocessing steps, including imputation, feature selection, and dimensionality reduction using PCA, were applied to optimize the predictive models. Among the models evaluated, the CatBoost algorithm emerged as the top-performing model.

Strengths and Limitations:

Strengths:

- The use of diverse feature selection techniques ensured that the most impactful predictors were included in the models.
- CatBoost demonstrated its robustness in handling nonlinear relationships and categorical data, even outperforming dimensionality-reduced models.
- Imputation using missForest ensured minimal data loss while maintaining the integrity of mixed data types.

Limitations:

- Data Limitations: The dataset exhibited missing values and class imbalance, potentially impacting model generalization.
- Overfitting Risk: Although CatBoost performed well, its complexity raises concerns about overfitting on unseen data from diverse populations or future timeframes.
- Residual Diagnostics: The baseline regression and GLS models indicated deviations from normality, suggesting that linear assumptions might not fully hold for this dataset.
- PCA Trade-Offs: While PCA simplified the feature set, it resulted in a slight decrease in performance compared to models using the original predictors.

Implications:

The findings presented in this work underscore the importance of using advanced machine learning-related techniques, like CatBoost, for health-related predictive modeling. Predicting life expectancy can aid policy makers and healthcare organizations in targeting interventions to improve health outcomes globally. This study highlights the relevance of socioeconomic and health-related variables, such as income composition, HIV/AIDS prevalence, and mortality rates, in influencing life expectancy.

Appendix

```
!pip install xgboost lightgbm catboost scikit-learn

Requirement already satisfied: xgboost in
/opt/conda/lib/python3.10/site-packages (2.0.3)
Requirement already satisfied: lightgbm in
/opt/conda/lib/python3.10/site-packages (4.2.0)
Requirement already satisfied: catboost in
/opt/conda/lib/python3.10/site-packages (1.2.7)
Requirement already satisfied: scikit-learn in
/opt/conda/lib/python3.10/site-packages (1.2.2)
Requirement already satisfied: numpy in
/opt/conda/lib/python3.10/site-packages (from xgboost) (1.26.4)
Requirement already satisfied: scipy in
/opt/conda/lib/python3.10/site-packages (from xgboost) (1.14.1)
Requirement already satisfied: graphviz in
/opt/conda/lib/python3.10/site-packages (from catboost) (0.20.3)
Requirement already satisfied: matplotlib in
/opt/conda/lib/python3.10/site-packages (from catboost) (3.7.5)
Requirement already satisfied: pandas>=0.24 in
/opt/conda/lib/python3.10/site-packages (from catboost) (2.2.3)
Requirement already satisfied: plotly in
/opt/conda/lib/python3.10/site-packages (from catboost) (5.22.0)
Requirement already satisfied: six in /opt/conda/lib/python3.10/site-
packages (from catboost) (1.16.0)
Requirement already satisfied: joblib>=1.1.1 in
/opt/conda/lib/python3.10/site-packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/opt/conda/lib/python3.10/site-packages (from scikit-learn) (3.5.0)
Requirement already satisfied: python-dateutil>=2.8.2 in
/opt/conda/lib/python3.10/site-packages (from pandas>=0.24->catboost)
(2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in
/opt/conda/lib/python3.10/site-packages (from pandas>=0.24->catboost)
(2024.1)
Requirement already satisfied: tzdata>=2022.7 in
/opt/conda/lib/python3.10/site-packages (from pandas>=0.24->catboost)
(2024.1)
Requirement already satisfied: contourpy>=1.0.1 in
/opt/conda/lib/python3.10/site-packages (from matplotlib->catboost)
(1.2.1)
Requirement already satisfied: cycler>=0.10 in
/opt/conda/lib/python3.10/site-packages (from matplotlib->catboost)
(0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/opt/conda/lib/python3.10/site-packages (from matplotlib->catboost)
(4.53.0)
Requirement already satisfied: kiwisolver>=1.0.1 in
/opt/conda/lib/python3.10/site-packages (from matplotlib->catboost)
(1.4.5)
Requirement already satisfied: packaging>=20.0 in
```

```
/opt/conda/lib/python3.10/site-packages (from matplotlib->catboost)
(21.3)
Requirement already satisfied: pillow>=6.2.0 in
/opt/conda/lib/python3.10/site-packages (from matplotlib->catboost)
(10.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/opt/conda/lib/python3.10/site-packages (from matplotlib->catboost)
(3.1.2)
Requirement already satisfied: tenacity>=6.2.0 in
/opt/conda/lib/python3.10/site-packages (from plotly->catboost)
(8.3.0)
```

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.tree import DecisionTreeRegressor
from xgboost import XGBRegressor
from lightgbm import LGBMRegressor
from catboost import CatBoostRegressor
from sklearn.svm import SVR
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
import seaborn as sns


train_data =
pd.read_csv("/kaggle/input/who-life/imputed_train_data_2014.csv")
test_data =
pd.read_csv("/kaggle/input/who-life/imputed_test_data_2015.csv")

# Drop missing values
train_data = train_data.dropna()
test_data = test_data.dropna()

# Convert categorical variable
train_data["Status"] = train_data["Status"].apply(lambda x: 1 if x ==
"Developing" else 0)
test_data["Status"] = test_data["Status"].apply(lambda x: 1 if x ==
"Developing" else 0)


features = [
    "Status", "Adult.Mortality", "infant.deaths", "Alcohol",
"under.five.deaths",
```

```
    "Polio", "Total.expenditure", "HIV.AIDS", "GDP",
"thinness..1.19.years",
    "Income.composition.of.resources"
]

X_train = train_data[features]
y_train = train_data["Life.expectancy"]

X_test = test_data[features]
y_test = test_data["Life.expectancy"]

def evaluate_model(model, X_train, X_test, y_train, y_test):
    # Train model
    model.fit(X_train, y_train)
    # Predict
    y_pred = model.predict(X_test)
    # Metrics
    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)
    print(f"{model.__class__.__name__}:")
    print(f"Mean Squared Error: {mse:.4f}")
    print(f"R^2 Score: {r2:.4f}")
    print("-" * 30)
    return mse, r2

dt_model = DecisionTreeRegressor(random_state=42, max_depth=5)
evaluate_model(dt_model, X_train, X_test, y_train, y_test)
```

```
DecisionTreeRegressor:
Mean Squared Error: 6.3309
R^2 Score: 0.9035
------------------------------

(6.330903415263037, 0.9035422830488038)
```

```
xgb_model = XGBRegressor(random_state=42, learning_rate=0.1,
n_estimators=100, max_depth=5)
evaluate_model(xgb_model, X_train, X_test, y_train, y_test)
```

```
XGBRegressor:
Mean Squared Error: 3.7733
R^2 Score: 0.9425
------------------------------

(3.7732715381596384, 0.9425103916243074)
```

```
lgbm_model = LGBMRegressor(random_state=42, learning_rate=0.1,
n_estimators=100, max_depth=5)
evaluate_model(lgbm_model, X_train, X_test, y_train, y_test)
```

```
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly
set num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly
set num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead
of testing was 0.002150 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 422
[LightGBM] [Info] Number of data points in the train set: 183, number
of used features: 11
[LightGBM] [Info] Start training from score 71.536612
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
```

```
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
```

```
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
```

```
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
```

```
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly
set num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
LGBMRegressor:
Mean Squared Error: 4.3978
R^2 Score: 0.9330
------------------------------

(4.397774085742927, 0.9329954636560709)

catboost_model = CatBoostRegressor(random_state=42, learning_rate=0.1,
n_estimators=100, max_depth=5, verbose=0)
evaluate_model(catboost_model, X_train, X_test, y_train, y_test)

CatBoostRegressor:
Mean Squared Error: 3.6907
R^2 Score: 0.9438
------------------------------

(3.6906995458145535, 0.9437684594454838)

from sklearn.preprocessing import StandardScaler
# Scale data for SVM
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

svm_model = SVR(kernel="rbf", C=1.0, epsilon=0.1)
evaluate_model(svm_model, X_train_scaled, X_test_scaled, y_train,
y_test)

SVR:
Mean Squared Error: 12.5083
R^2 Score: 0.8094
------------------------------

(12.508308738603809, 0.8094232647843529)

results = []

# Evaluate each model and append results
```

```
results.append(("Decision Tree", *evaluate_model(dt_model, X_train,
X_test, y_train, y_test)))
results.append(("XGBoost", *evaluate_model(xgb_model, X_train, X_test,
y_train, y_test)))
results.append(("LightGBM", *evaluate_model(lgbm_model, X_train,
X_test, y_train, y_test)))
results.append(("CatBoost", *evaluate_model(catboost_model, X_train,
X_test, y_train, y_test)))
results.append(("SVM", *evaluate_model(svm_model, X_train_scaled,
X_test_scaled, y_train, y_test)))

# Create a DataFrame for comparison
results_df = pd.DataFrame(results, columns=["Model", "MSE", "R^2"])
print(results_df)
```

```
DecisionTreeRegressor:
Mean Squared Error: 6.3309
R^2 Score: 0.9035
------------------------------
XGBRegressor:
Mean Squared Error: 3.7733
R^2 Score: 0.9425
------------------------------
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly
set num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly
set num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead
of testing was 0.000043 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 422
[LightGBM] [Info] Number of data points in the train set: 183, number
of used features: 11
[LightGBM] [Info] Start training from score 71.536612
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
```

```
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
```

```
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] No further splits with positive gain, best gain:
-inf
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly
set num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
LGBMRegressor:
Mean Squared Error: 4.3978
R^2 Score: 0.9330
------------------------------
CatBoostRegressor:
Mean Squared Error: 3.6907
R^2 Score: 0.9438
------------------------------
SVR:
Mean Squared Error: 12.5083
```

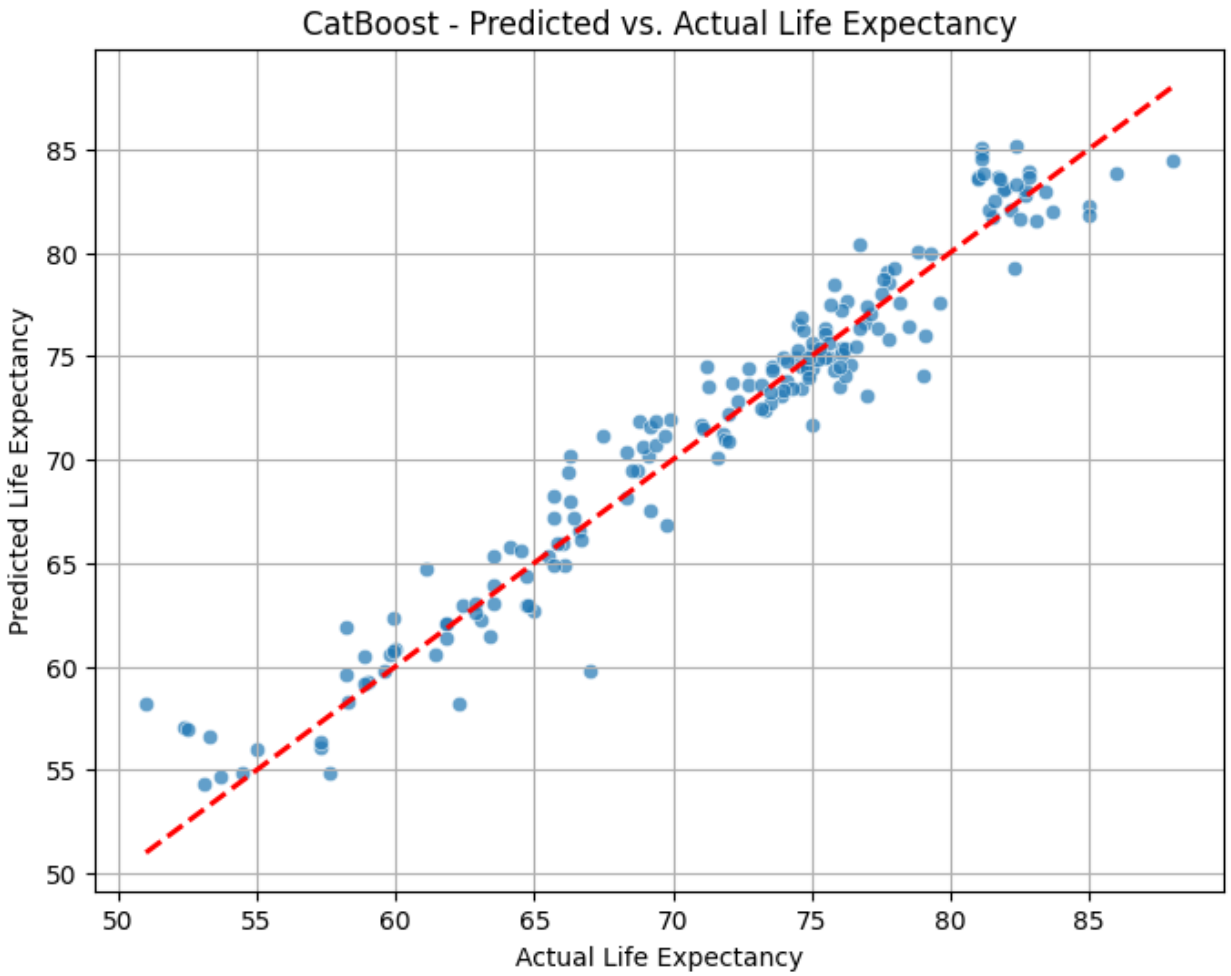
```
R^2 Score: 0.8094
------------------------------
           Model        MSE       R^2
0  Decision Tree   6.330903  0.903542
1        XGBoost   3.773272  0.942510
2       LightGBM   4.397774  0.932995
3       CatBoost   3.690700  0.943768
4            SVM  12.508309  0.809423
```

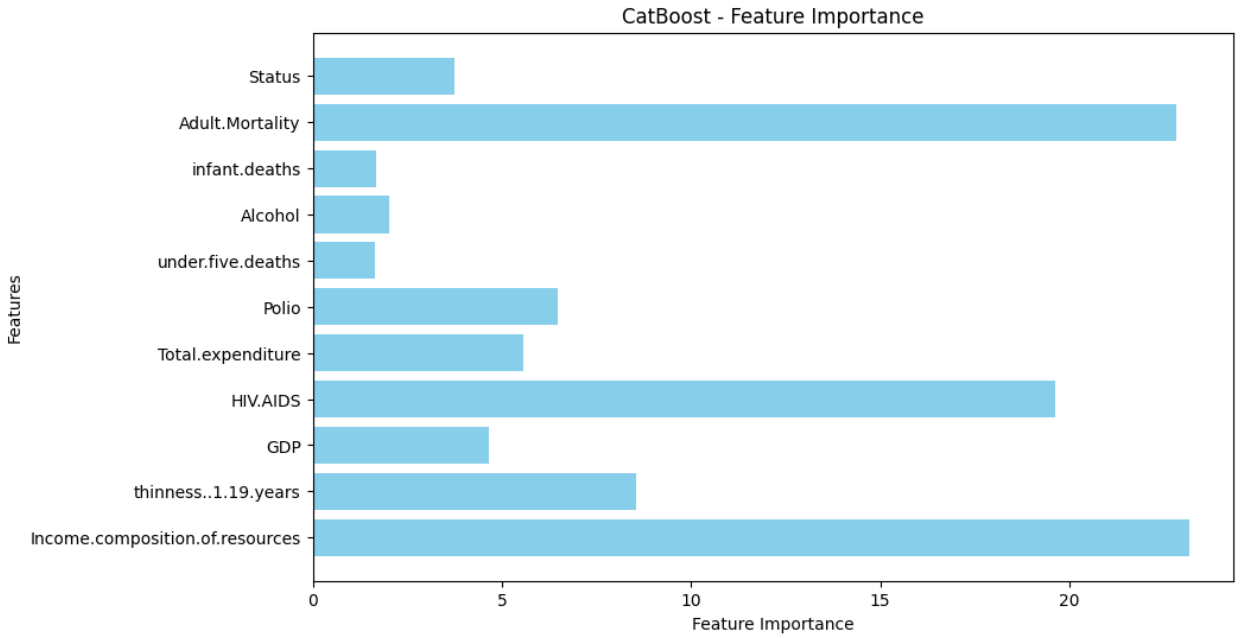
```python
# Predict on the test set
catboost_y_pred = catboost_model.predict(X_test)

# Scatterplot: Predicted vs Actual
plt.figure(figsize=(8, 6))
sns.scatterplot(x=y_test, y=catboost_y_pred, alpha=0.7)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()],
color='red', linestyle='--', linewidth=2)
plt.xlabel("Actual Life Expectancy")
plt.ylabel("Predicted Life Expectancy")
plt.title("CatBoost - Predicted vs. Actual Life Expectancy")
plt.grid(True)
plt.savefig("catboost_predicted_vs_actual.png", dpi=300,
bbox_inches='tight')
plt.show()
```

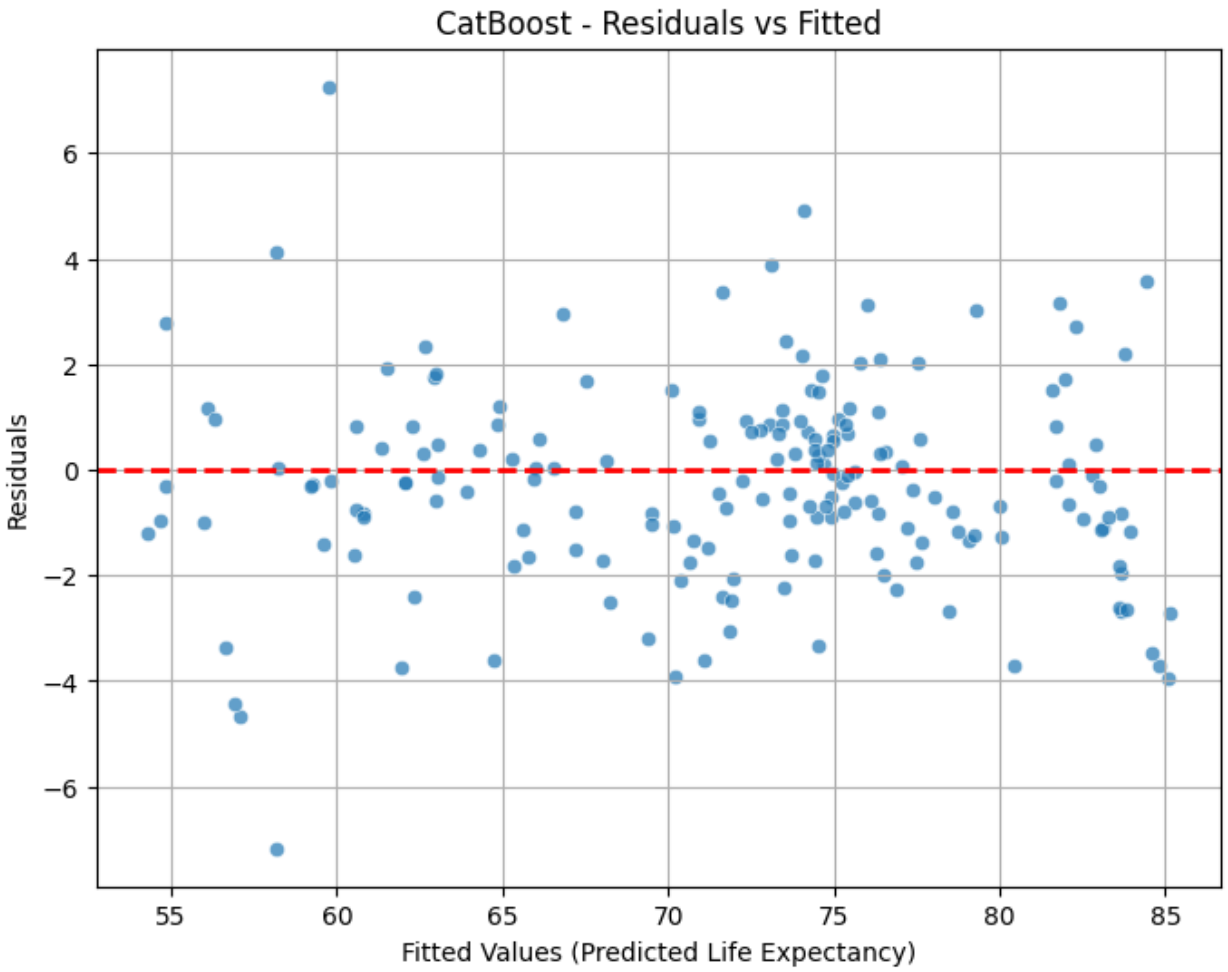
```python
# Train the CatBoost model
catboost_model.fit(X_train, y_train)

# Get feature importance
feature_importance = catboost_model.get_feature_importance()
features = X_train.columns

# Plot feature importance
plt.figure(figsize=(10, 6))
plt.barh(features, feature_importance, color="skyblue")
plt.xlabel("Feature Importance")
plt.ylabel("Features")
plt.title("CatBoost - Feature Importance")
plt.gca().invert_yaxis()
plt.savefig("catboost_feature_importance.png", dpi=300,
bbox_inches='tight')
plt.show()
```

```python
# Calculate residuals
residuals = y_test - catboost_y_pred
```

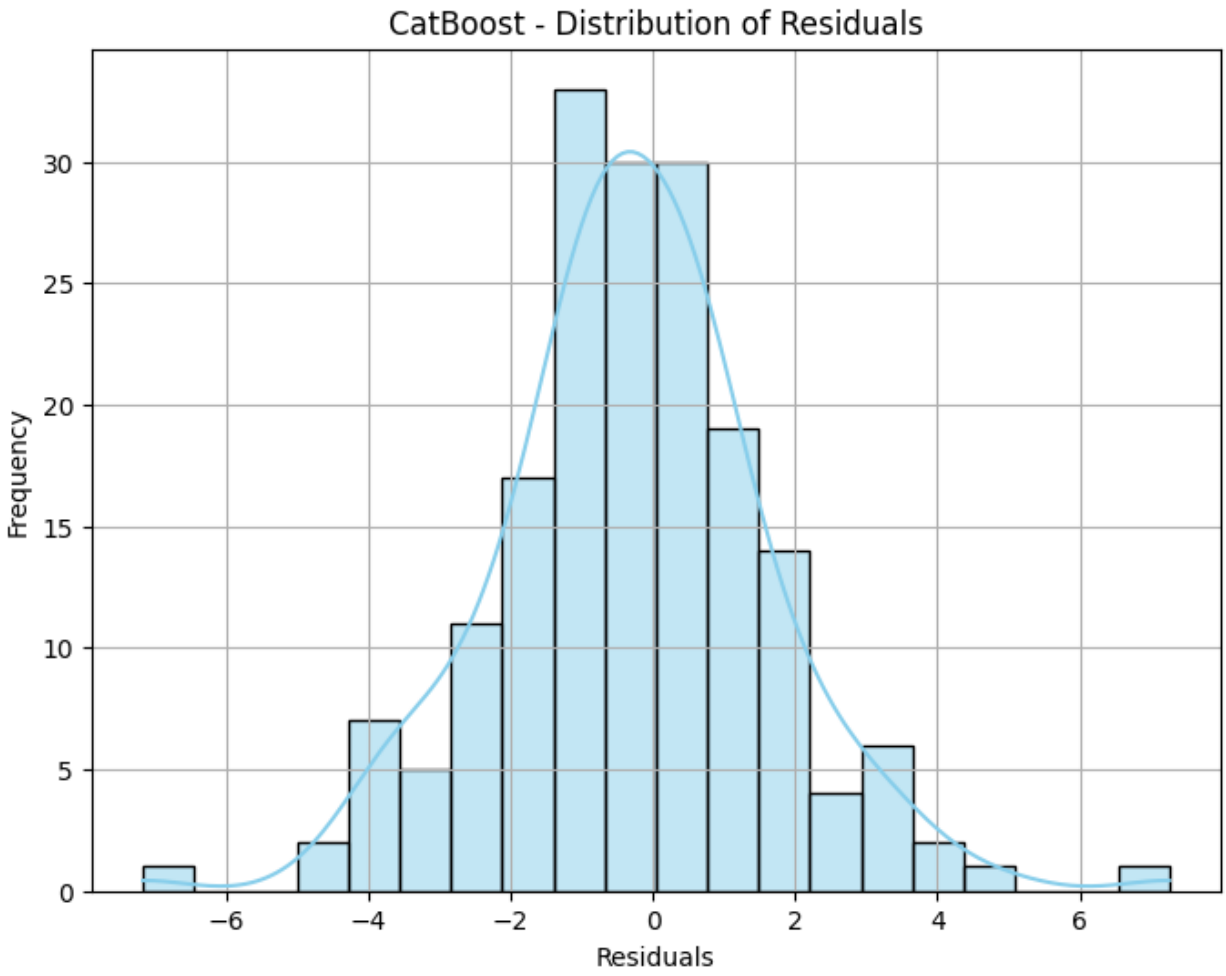
```
# Residuals vs Fitted
plt.figure(figsize=(8, 6))
sns.scatterplot(x=catboost_y_pred, y=residuals, alpha=0.7)
plt.axhline(0, color='red', linestyle='--', linewidth=2)
plt.xlabel("Fitted Values (Predicted Life Expectancy)")
plt.ylabel("Residuals")
plt.title("CatBoost - Residuals vs Fitted")
plt.grid(True)
plt.savefig("catboost_residuals_vs_fitted.png", dpi=300,
bbox_inches='tight')
plt.show()

# Histogram of Residuals
plt.figure(figsize=(8, 6))
sns.histplot(residuals, kde=True, bins=20, color="skyblue")
plt.xlabel("Residuals")
plt.ylabel("Frequency")
plt.title("CatBoost - Distribution of Residuals")
plt.grid(True)
plt.savefig("catboost_residuals_histogram.png", dpi=300,
bbox_inches='tight')
plt.show()
```


CatBoost - Predicted vs. Actual Life Expectancy
Predicted Life Expectancy
Actual Life Expectancy
50
55
60
65
70
75
80
85


CatBoost - Feature Importance
Features
Status
Adult.Mortality
infant.deaths
Alcohol
under.five.deaths
Polio
Total.expenditure
HIV.AIDS
GDP
thinness..1.19.years
Income.composition.of.resources
0
5
10
15
20
Feature Importance




```
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
```


CatBoost - Distribution of Residuals
Frequency
Residuals
0
5
10
15
20
25
30
−6
−4
−2
0
2
4
6

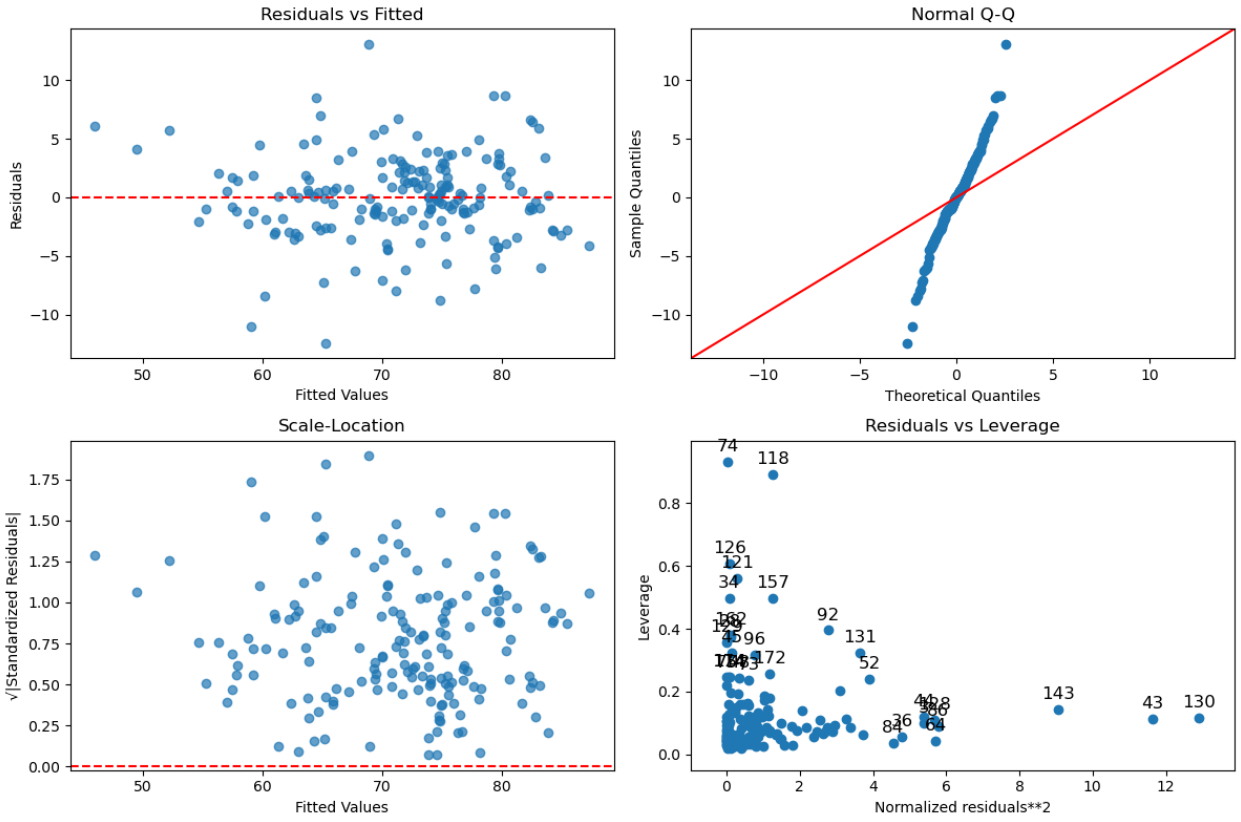
```python
import pandas as pd
import numpy as np
import statsmodels.api as sm
import matplotlib.pyplot as plt
from statsmodels.graphics.regressionplots import plot_leverage_resid2
from statsmodels.graphics.gofplots import qqplot

# Load the datasets
train_data =
pd.read_csv('/home/brandon-ism/Documents/data/m444/who_life/life-
expectancy-who/who_train.csv')
test_data =
pd.read_csv('/home/brandon-ism/Documents/data/m444/who_life/life-
expectancy-who/who_test.csv')

# Clean column names to remove whitespace
train_data.columns = train_data.columns.str.strip()
test_data.columns = test_data.columns.str.strip()

# Align train and test features
X_train, X_test = X_train.align(X_test, join='inner', axis=1)

# Ensure all data is numeric and handle the boolean columns
X_train = X_train.apply(pd.to_numeric,
errors='coerce').fillna(0).astype(float)
X_test = X_test.apply(pd.to_numeric,
errors='coerce').fillna(0).astype(float)
y_train = pd.to_numeric(y_train,
errors='coerce').fillna(0).astype(float)
y_test = pd.to_numeric(y_test,
errors='coerce').fillna(0).astype(float)

# Fit baseline linear regression model
X_train = sm.add_constant(X_train)
model = sm.OLS(y_train, X_train).fit()

# Diagnostics
print(model.summary())

# `const` = beta_0
```

```
                            OLS Regression Results

==============================================================================
========
Dep. Variable:        Life expectancy   R-squared:
0.817
Model:                            OLS   Adj. R-squared:
0.796
Method:                 Least Squares   F-statistic:
```

```
38.40
Date:                Tue, 26 Nov 2024   Prob (F-statistic):
2.85e-50
Time:                        19:04:21   Log-Likelihood:
-496.51
No. Observations:                 183   AIC:
1033.
Df Residuals:                     163   BIC:
1097.
Df Model:                          19

Covariance Type:            nonrobust

==============================================================================
==============================
                                   coef    std err          t
P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
------------------------------
const                           68.7256      2.294     29.958
0.000      64.196      73.256
Adult Mortality                 -0.0282      0.004     -7.187
0.000      -0.036      -0.020
infant deaths                    0.1253      0.064      1.972
0.050      -0.000       0.251
Alcohol                          0.2526      0.091      2.776
0.006       0.073       0.432
percentage expenditure       -9.047e-05      0.000     -0.303
0.763      -0.001       0.001
Hepatitis B                     -0.0209      0.015     -1.437
0.153      -0.050       0.008
Measles                      -2.475e-05   5.41e-05     -0.458
0.648      -0.000     8.2e-05
BMI                              0.0199      0.017      1.171
0.243      -0.014       0.054
under-five deaths               -0.0929      0.044     -2.124
0.035      -0.179      -0.007
Polio                            0.0189      0.023      0.840
0.402      -0.026       0.064
Total expenditure                0.1946      0.118      1.652
0.100      -0.038       0.427
Diphtheria                       0.0328      0.024      1.385
0.168      -0.014       0.080
HIV/AIDS                        -1.0545      0.279     -3.782
0.000      -1.605      -0.504
GDP                           4.165e-05   4.42e-05      0.942
0.347   -4.56e-05       0.000
Population                   -2.457e-09   7.43e-09     -0.330
0.741   -1.71e-08    1.22e-08
```

```
thinness  1-19 years                -0.2692      0.260     -1.036
0.302      -0.782       0.244
thinness 5-9 years                  0.0262      0.257      0.102
0.919      -0.481       0.533
Income composition of resources    10.2278      5.189      1.971
0.050      -0.019      20.474
Schooling                          -0.1231      0.263     -0.469
0.640      -0.642       0.396
Status_Developing                  -2.5297      1.003     -2.522
0.013      -4.510      -0.549
==============================================================================
Omnibus:                        7.671   Durbin-Watson:
2.209
Prob(Omnibus):                  0.022   Jarque-Bera (JB):
13.522
Skew:                          -0.091   Prob(JB):
0.00116
Kurtosis:                       4.319   Cond. No.
1.83e+09
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
[2] The condition number is large, 1.83e+09. This might indicate that
there are
strong multicollinearity or other numerical problems.


# Predictions and residuals
X_test = sm.add_constant(X_test, has_constant='add')
y_pred = model.predict(X_test)
residuals = model.resid


# Diagnostic Plots
plt.figure(figsize=(12, 8))


# Residuals vs Fitted
plt.subplot(2, 2, 1)
plt.scatter(model.fittedvalues, residuals, alpha=0.7)
plt.axhline(y=0, color='red', linestyle='--')
plt.title('Residuals vs Fitted')
plt.xlabel('Fitted Values')
plt.ylabel('Residuals')
```

```
# Normal Q-Q Plot
plt.subplot(2, 2, 2)
qqplot(residuals, line='45', ax=plt.gca())
plt.title('Normal Q-Q')


# Scale-Location Plot
plt.subplot(2, 2, 3)
standardized_residuals = residuals / np.std(residuals)
plt.scatter(model.fittedvalues,
np.sqrt(np.abs(standardized_residuals)), alpha=0.7)
plt.axhline(y=0, color='red', linestyle='--')
plt.title('Scale-Location')
plt.xlabel('Fitted Values')
plt.ylabel('√|Standardized Residuals|')


# Residuals vs Leverage
plt.subplot(2, 2, 4)
plot_leverage_resid2(model, ax=plt.gca())
plt.title('Residuals vs Leverage')

plt.tight_layout()
plt.show()
```



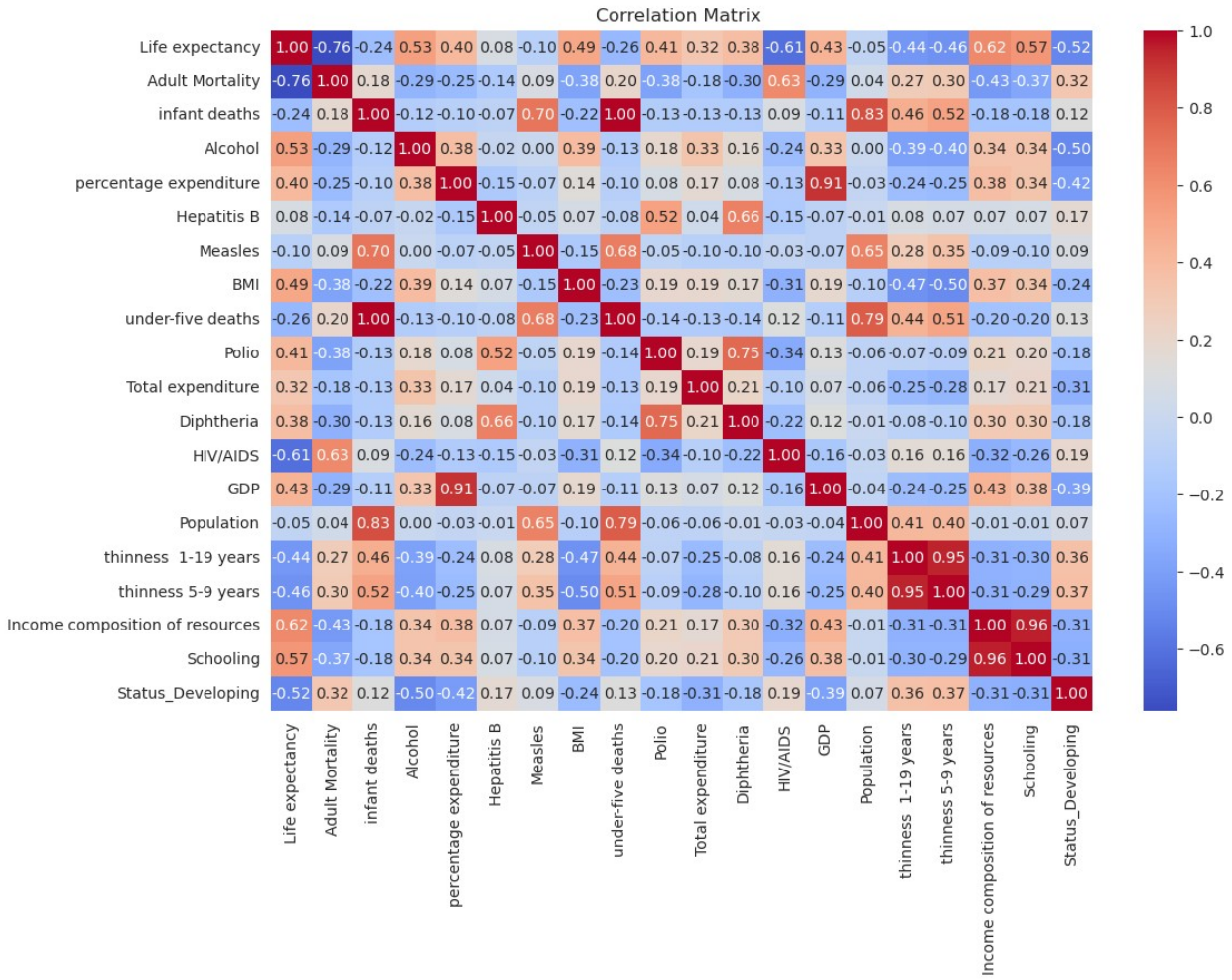
```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

# Load the dataset
data =
pd.read_csv('/home/brandon-ism/Documents/data/m444/who_life/life-
expectancy-who/who_train.csv')

# Clean column names
data.columns = data.columns.str.strip()

# Drop unnecessary columns for EDA
columns_to_drop = ["Country", "Year"]
eda_data = data.drop(columns=columns_to_drop)

# Convert 'Status' column to dummy variables
eda_data = pd.get_dummies(eda_data, columns=["Status"],
drop_first=True)

# Ensure all columns are strictly numeric
eda_data = eda_data.apply(pd.to_numeric, errors="coerce")

# Handle missing and infinite values
eda_data = eda_data.fillna(0).replace([np.inf, -np.inf], 0)
\
# Check column types
print("Column types:")
print(eda_data.dtypes)
```

```
Column types:
Life expectancy                    float64
Adult Mortality                    float64
infant deaths                        int64
Alcohol                            float64
percentage expenditure             float64
Hepatitis B                        float64
Measles                              int64
BMI                                float64
under-five deaths                    int64
Polio                              float64
Total expenditure                  float64
Diphtheria                         float64
HIV/AIDS                           float64
GDP                                float64
Population                         float64
thinness  1-19 years               float64
thinness 5-9 years                 float64
Income composition of resources    float64
```

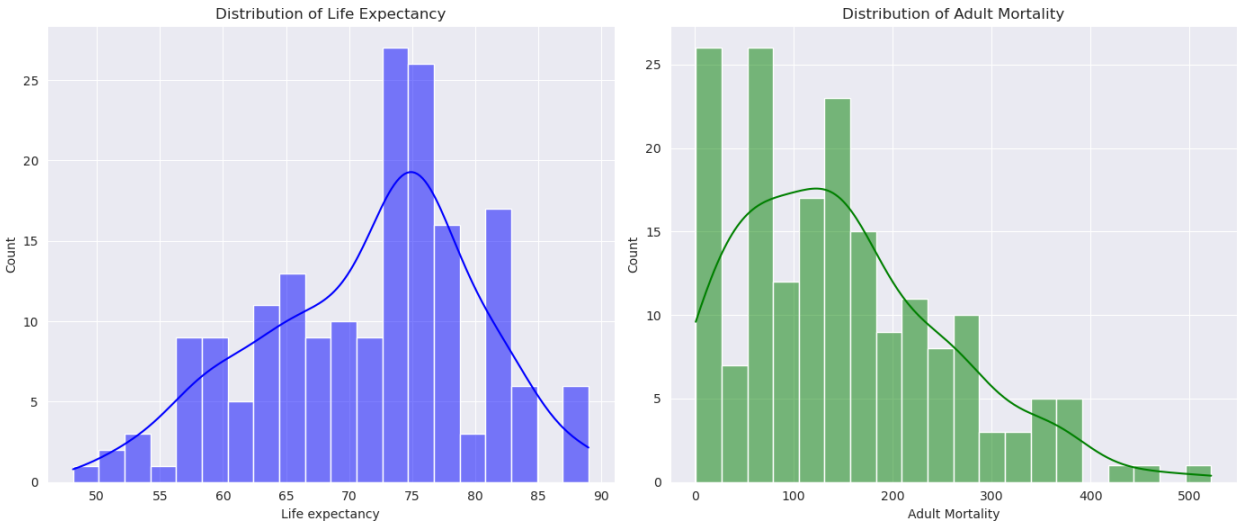
```
Schooling                            float64
Status_Developing                       bool
dtype: object

# 1. Summary stats
print("Summary Statistics:")
print(eda_data.describe())

Summary Statistics:
       Life expectancy  Adult Mortality  infant deaths     Alcohol  \
count       183.000000       183.000000     183.000000  183.000000
mean         71.536612       148.688525      24.557377    3.253443
std           8.560831       106.025532      87.045749    4.150709
min          48.100000         1.000000       0.000000    0.000000
25%          65.600000        66.000000       0.000000    0.010000
50%          73.600000       135.000000       2.000000    0.260000
75%          76.850000       216.500000      18.000000    6.660000
max          89.000000       522.000000     957.000000   15.190000

       percentage expenditure  Hepatitis B       Measles
BMI  \
count              183.000000   183.000000    183.000000  183.000000

mean              1001.912550    78.573770   1831.207650   40.582514

std               2553.290079    29.570065   8770.076631   21.424645

min                  0.000000     0.000000      0.000000    0.000000

25%                 11.062331    77.000000      0.000000   22.850000

50%                151.104555    92.000000     13.000000   47.000000

75%                703.207524    97.000000    316.000000   59.750000

max              19479.911610    99.000000  79563.000000   77.100000


       under-five deaths       Polio  Total expenditure  Diphtheria  \
count         183.000000  183.000000         183.000000  183.000000
mean           32.890710   84.726776           6.133224   84.081967
std           114.293045   20.868813           2.803438   23.032916
min             0.000000    8.000000           0.000000    2.000000
25%             0.000000   80.000000           4.370000   83.000000
50%             3.000000   94.000000           5.820000   94.000000
75%            22.000000   97.000000           7.720000   97.000000
max          1200.000000   99.000000          17.140000   99.000000

         HIV/AIDS            GDP    Population  thinness  1-19
years  \
count  183.000000     183.000000  1.830000e+02             183.00000
```

```
mean      0.681967    8483.131785  1.634394e+07                4.48306
std       1.388157   17383.161138  9.912282e+07                4.14449
min       0.100000       0.000000  0.000000e+00                0.00000
25%       0.100000     179.740270  1.281000e+04                1.50000
50%       0.100000    1684.542740  6.789140e+05                3.30000
75%       0.400000    7590.643088  4.703228e+06                6.55000
max       9.400000  119172.741800  1.293859e+09               26.80000

        thinness 5-9 years  Income composition of resources   Schooling
count           183.000000                       183.000000  183.000000
mean              4.625137                         0.650776   12.183060
std               4.252333                         0.216532    4.078985
min               0.000000                         0.000000    0.000000
25%               1.450000                         0.534000   10.600000
50%               3.400000                         0.712000   12.900000
75%               6.550000                         0.792000   14.800000
max              27.400000                         0.945000   20.400000
```

```python
# 2. Correlation matrix
plt.figure(figsize=(12, 8))
sns.heatmap(eda_data.corr(), annot=True, fmt=".2f", cmap="coolwarm",
cbar=True)
plt.title("Correlation Matrix ")
plt.show()
```

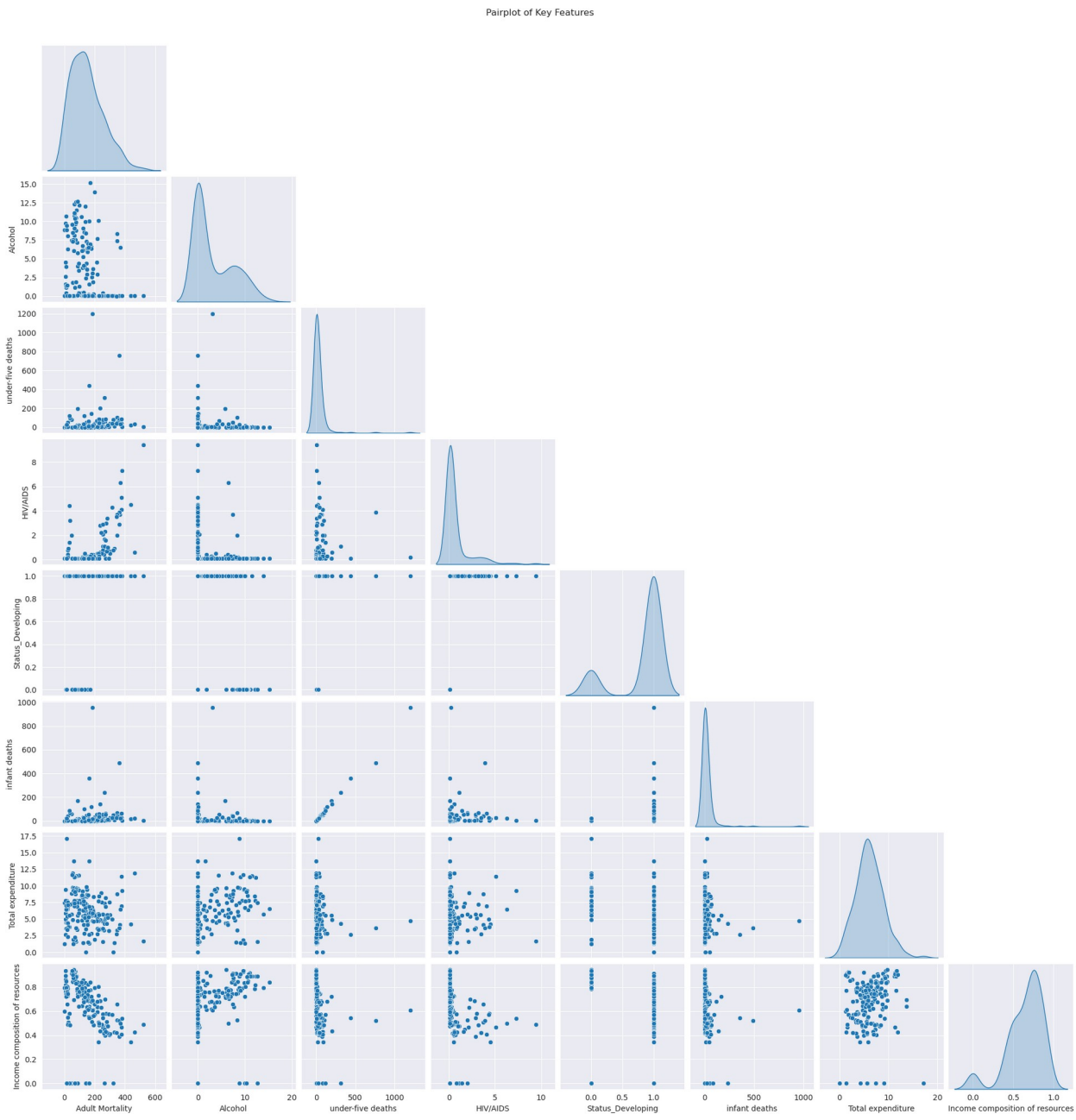



```
# 3. Distributions of the target variable and predictors
plt.figure(figsize=(14, 6))
plt.subplot(1, 2, 1)
sns.histplot(eda_data["Life expectancy"], kde=True, bins=20,
color='blue')
plt.title("Distribution of Life Expectancy")

plt.subplot(1, 2, 2)
sns.histplot(eda_data["Adult Mortality"], kde=True, bins=20,
color='green')
plt.title("Distribution of Adult Mortality")
plt.tight_layout()
plt.show()
```



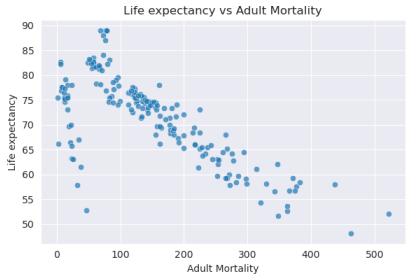
The features in #4 were chosen from the baseline linear model. They had significant or marginally significant features.

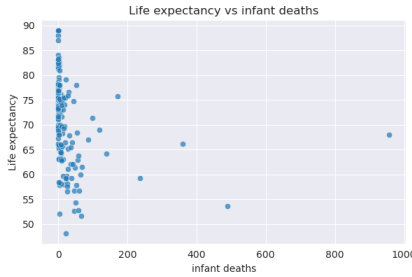
```
# 4. Pairplot for selected features
significant_features = [
    "Adult Mortality",
    "Alcohol",
    "under-five deaths",
    "HIV/AIDS",
    "Status_Developing",
    "infant deaths",
    "Total expenditure",
    "Income composition of resources"
]

sns.pairplot(eda_data[significant_features], diag_kind="kde",
corner=True)
plt.suptitle("Pairplot of Key Features", y=1.02)
plt.show()
```

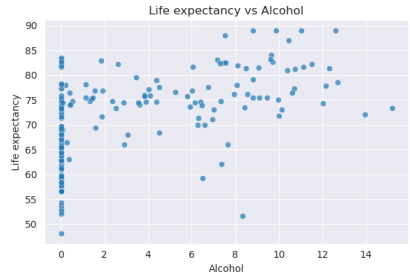


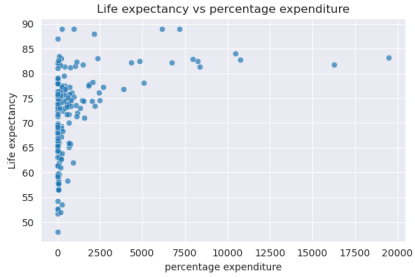

```
# 5. Plot the correlation of every feature with the target
target = "Life expectancy"
features = [col for col in eda_data.columns if col != target]


num_features = len(features)
cols = 3
rows = (num_features + cols - 1) // cols

plt.figure(figsize=(cols * 6, rows * 4))
for i, feature in enumerate(features, start=1):
```

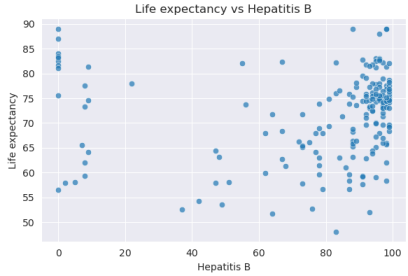
```
    plt.subplot(rows, cols, i)
    sns.scatterplot(data=eda_data, x=feature, y=target, alpha=0.7)
    plt.title(f"{target} vs {feature}")
    plt.xlabel(feature)
    plt.ylabel(target)

plt.tight_layout()
plt.show()
```

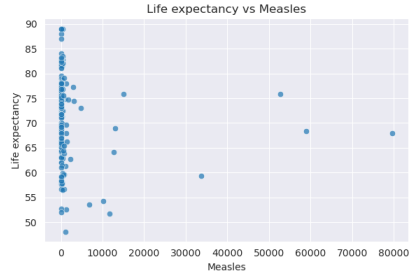

Life expectancy vs Adult Mortality
Life expectancy
Adult Mortality

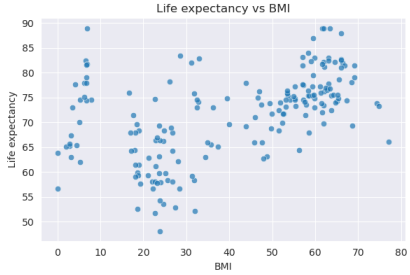

Life expectancy vs infant deaths
Life expectancy
infant deaths

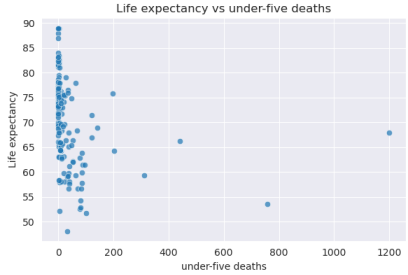

Life expectancy vs Alcohol
Life expectancy
Alcohol

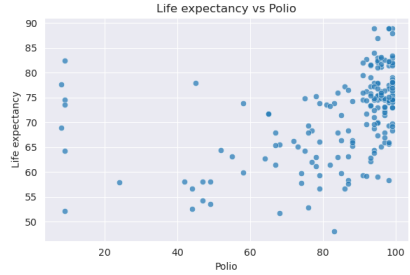

Life expectancy vs percentage expenditure
Life expectancy
percentage expenditure

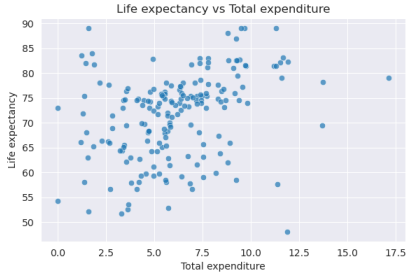

Life expectancy vs Hepatitis B
Life expectancy
Hepatitis B

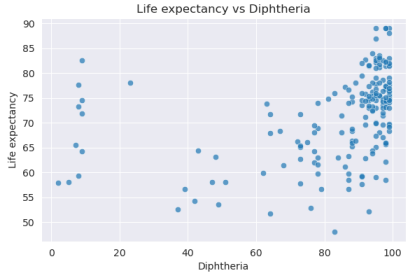

Life expectancy vs Measles
Life expectancy
Measles

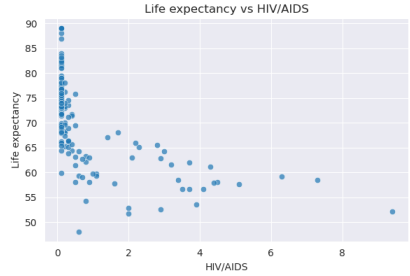

Life expectancy vs BMI
Life expectancy
BMI


Life expectancy vs under-five deaths
Life expectancy
under-five deaths

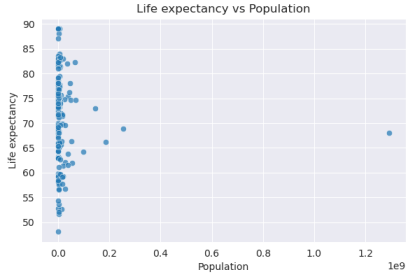

Life expectancy vs Polio
Life expectancy
Polio


Life expectancy vs Total expenditure
Life expectancy
Total expenditure


Life expectancy vs Diphtheria
Life expectancy
Diphtheria


Life expectancy vs HIV/AIDS
Life expectancy
HIV/AIDS

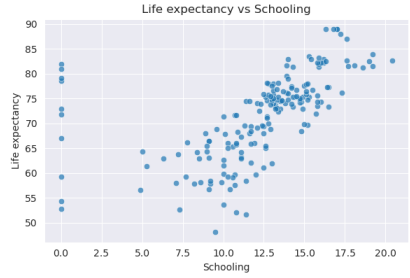

Life expectancy vs GDP
Life expectancy
GDP

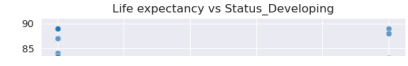

Life expectancy vs Population
Life expectancy
Population
1e9


Life expectancy vs thinness 1-19 years
Life expectancy
thinness 1-19 years


Life expectancy vs thinness 5-9 years
Life expectancy
thinness 5-9 years


Life expectancy vs Income composition of resources
Life expectancy
Income composition of resources


Life expectancy vs Schooling
Life expectancy
Schooling


Life expectancy vs Status_Developing

```
!pip install scikit-learn matplotlib seaborn
```

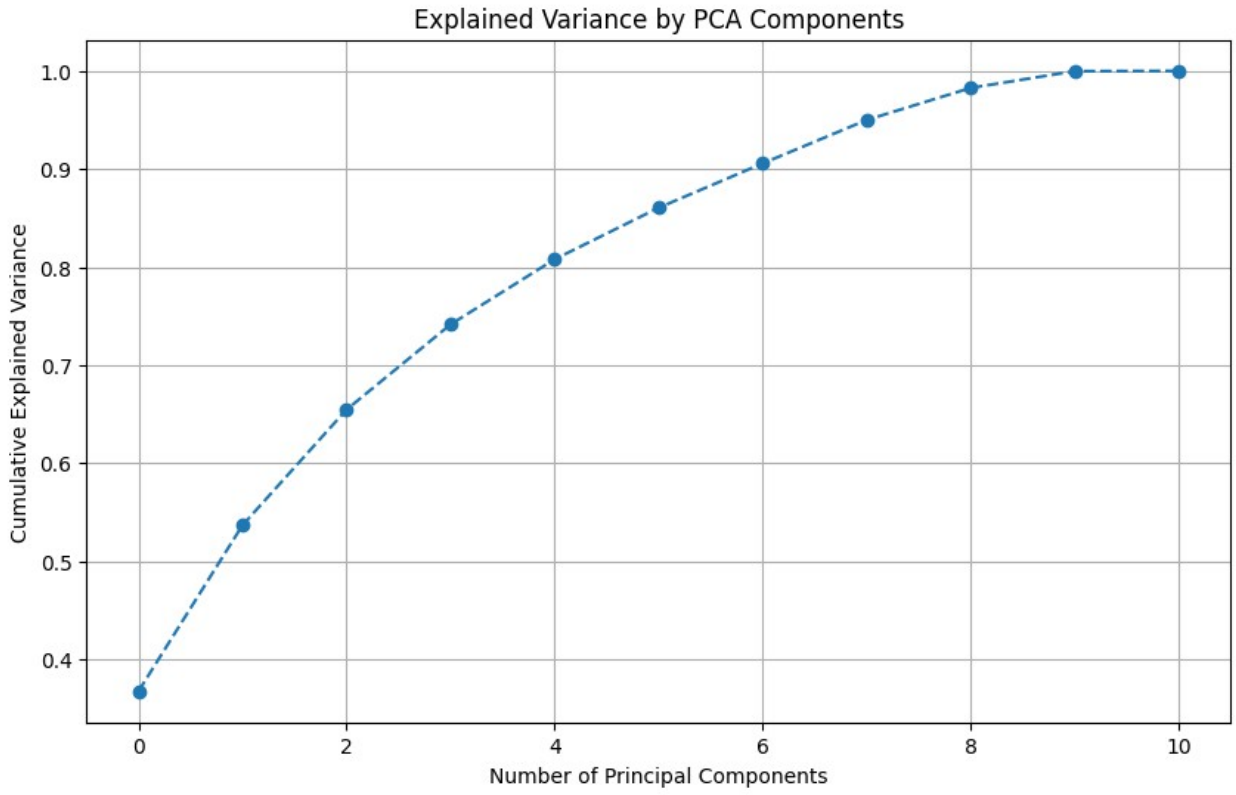
```
Requirement already satisfied: scikit-learn in
/opt/conda/lib/python3.10/site-packages (1.2.2)
Requirement already satisfied: matplotlib in
/opt/conda/lib/python3.10/site-packages (3.7.5)
Requirement already satisfied: seaborn in
/opt/conda/lib/python3.10/site-packages (0.12.2)
Requirement already satisfied: numpy>=1.17.3 in
/opt/conda/lib/python3.10/site-packages (from scikit-learn) (1.26.4)
Requirement already satisfied: scipy>=1.3.2 in
/opt/conda/lib/python3.10/site-packages (from scikit-learn) (1.14.1)
Requirement already satisfied: joblib>=1.1.1 in
/opt/conda/lib/python3.10/site-packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/opt/conda/lib/python3.10/site-packages (from scikit-learn) (3.5.0)
Requirement already satisfied: contourpy>=1.0.1 in
/opt/conda/lib/python3.10/site-packages (from matplotlib) (1.2.1)
Requirement already satisfied: cycler>=0.10 in
/opt/conda/lib/python3.10/site-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/opt/conda/lib/python3.10/site-packages (from matplotlib) (4.53.0)
Requirement already satisfied: kiwisolver>=1.0.1 in
/opt/conda/lib/python3.10/site-packages (from matplotlib) (1.4.5)
Requirement already satisfied: packaging>=20.0 in
/opt/conda/lib/python3.10/site-packages (from matplotlib) (21.3)
Requirement already satisfied: pillow>=6.2.0 in
/opt/conda/lib/python3.10/site-packages (from matplotlib) (10.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/opt/conda/lib/python3.10/site-packages (from matplotlib) (3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in
/opt/conda/lib/python3.10/site-packages (from matplotlib)
(2.9.0.post0)
Requirement already satisfied: pandas>=0.25 in
/opt/conda/lib/python3.10/site-packages (from seaborn) (2.2.3)
Requirement already satisfied: pytz>=2020.1 in
/opt/conda/lib/python3.10/site-packages (from pandas>=0.25->seaborn)
(2024.1)
Requirement already satisfied: tzdata>=2022.7 in
/opt/conda/lib/python3.10/site-packages (from pandas>=0.25->seaborn)
(2024.1)
Requirement already satisfied: six>=1.5 in
/opt/conda/lib/python3.10/site-packages (from python-dateutil>=2.7-
>matplotlib) (1.16.0)
```

```
import pandas as pd
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
import seaborn as sns
```

```
train_data =
pd.read_csv("/kaggle/input/who-life/imputed_train_data_2014.csv")
test_data =
pd.read_csv("/kaggle/input/who-life/imputed_test_data_2015.csv")

# Drop missing values
train_data = train_data.dropna()
test_data = test_data.dropna()

# Convert categorical variable
train_data["Status"] = train_data["Status"].apply(lambda x: 1 if x ==
"Developing" else 0)
test_data["Status"] = test_data["Status"].apply(lambda x: 1 if x ==
"Developing" else 0)


features = [
    "Status", "Adult.Mortality", "infant.deaths", "Alcohol",
"under.five.deaths",
    "Polio", "Total.expenditure", "HIV.AIDS", "GDP",
"thinness..1.19.years",
    "Income.composition.of.resources"
]

X_train = train_data[features]
y_train = train_data["Life.expectancy"]

X_test = test_data[features]
y_test = test_data["Life.expectancy"]


# Standardize the data
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

train_data.columns
```

```
Index(['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',
       'Country'],
      dtype='object')
```

```
import numpy as np
import matplotlib.pyplot as plt


pca = PCA()
X_train_pca = pca.fit_transform(X_train_scaled)



# Plot explained variance ratio
plt.figure(figsize=(10, 6))
plt.plot(np.cumsum(pca.explained_variance_ratio_), marker='o',
linestyle='--')
plt.xlabel("Number of Principal Components")
plt.ylabel("Cumulative Explained Variance")
plt.title("Explained Variance by PCA Components")
plt.grid(True)
plt.savefig("all_pca_explained_variance.png", dpi=300,
bbox_inches='tight')
plt.show()
```




```
# Retain enough components to explain 90% of variance
pca = PCA(n_components=0.90)
X_train_pca = pca.fit_transform(X_train_scaled)
X_test_pca = pca.transform(X_test_scaled)
```

```
print("Number of components selected:", pca.n_components_)
```

```
Number of components selected: 7
```

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

lr_model = LinearRegression()
lr_model.fit(X_train_pca, y_train)


y_pred = lr_model.predict(X_test_pca)
print("Linear Regression on PCA Components:")
print(f"Mean Squared Error: {mean_squared_error(y_test, y_pred):.4f}")
print(f"R^2 Score: {r2_score(y_test, y_pred):.4f}")
```

```
Linear Regression on PCA Components:
Mean Squared Error: 10.2248
R^2 Score: 0.8442
```

```
from xgboost import XGBRegressor


xgb_model = XGBRegressor(random_state=42)
xgb_model.fit(X_train_pca, y_train)


y_pred = xgb_model.predict(X_test_pca)
print("XGBoost on PCA Components:")
print(f"Mean Squared Error: {mean_squared_error(y_test, y_pred):.4f}")
print(f"R^2 Score: {r2_score(y_test, y_pred):.4f}")
```

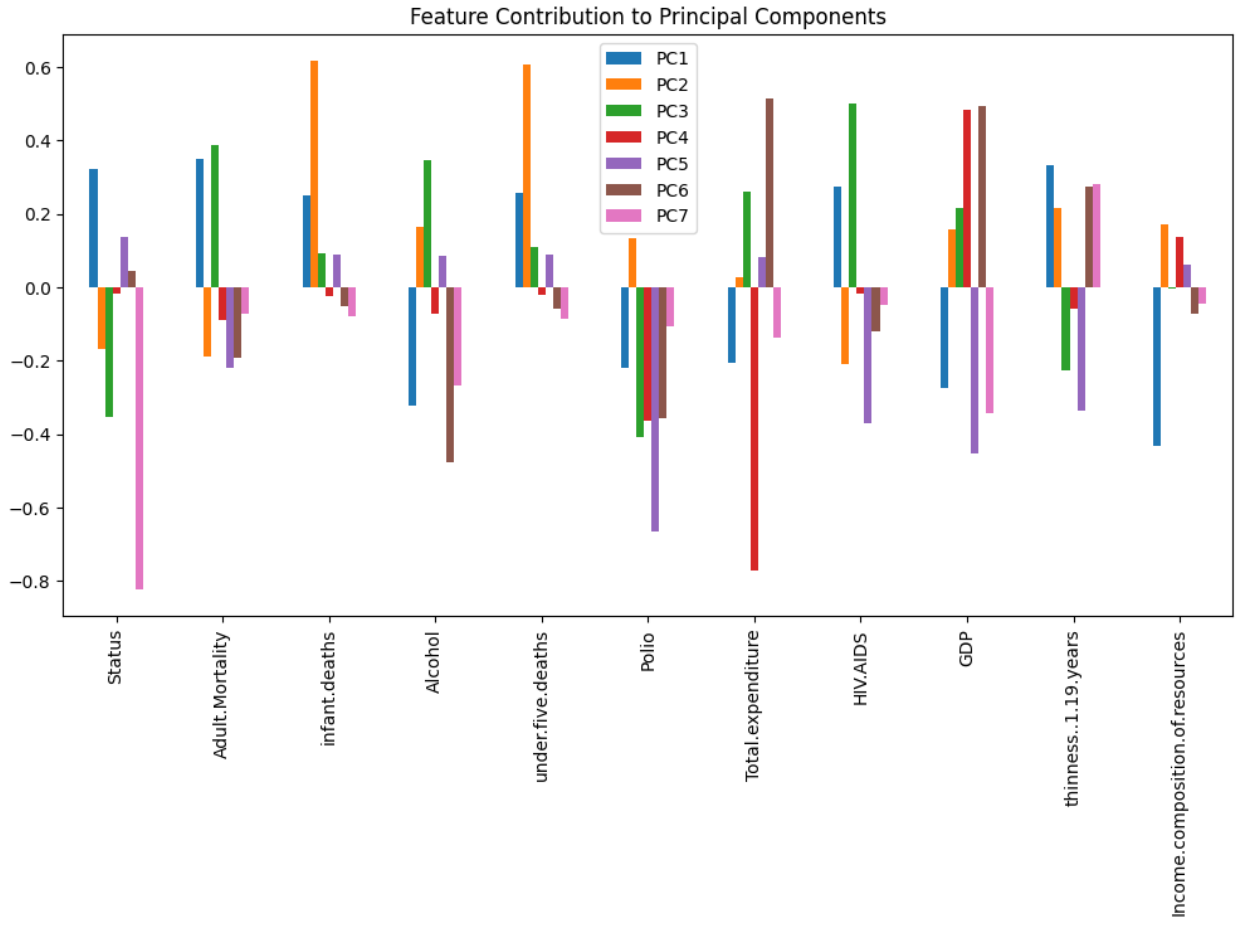
```
XGBoost on PCA Components:
Mean Squared Error: 9.4754
R^2 Score: 0.8556
```

```
from catboost import CatBoostRegressor
from sklearn.metrics import mean_squared_error, r2_score


catboost_pca_model = CatBoostRegressor(random_seed=42, verbose=0)


catboost_pca_model.fit(X_train_pca, y_train)

y_pred_pca_catboost = catboost_pca_model.predict(X_test_pca)

print("CatBoost on PCA Components:")
print(f"Mean Squared Error: {mean_squared_error(y_test,
y_pred_pca_catboost):.4f}")
print(f"R^2 Score: {r2_score(y_test, y_pred_pca_catboost):.4f}")
```

```
CatBoost on PCA Components:
Mean Squared Error: 7.5519
R^2 Score: 0.8849
```

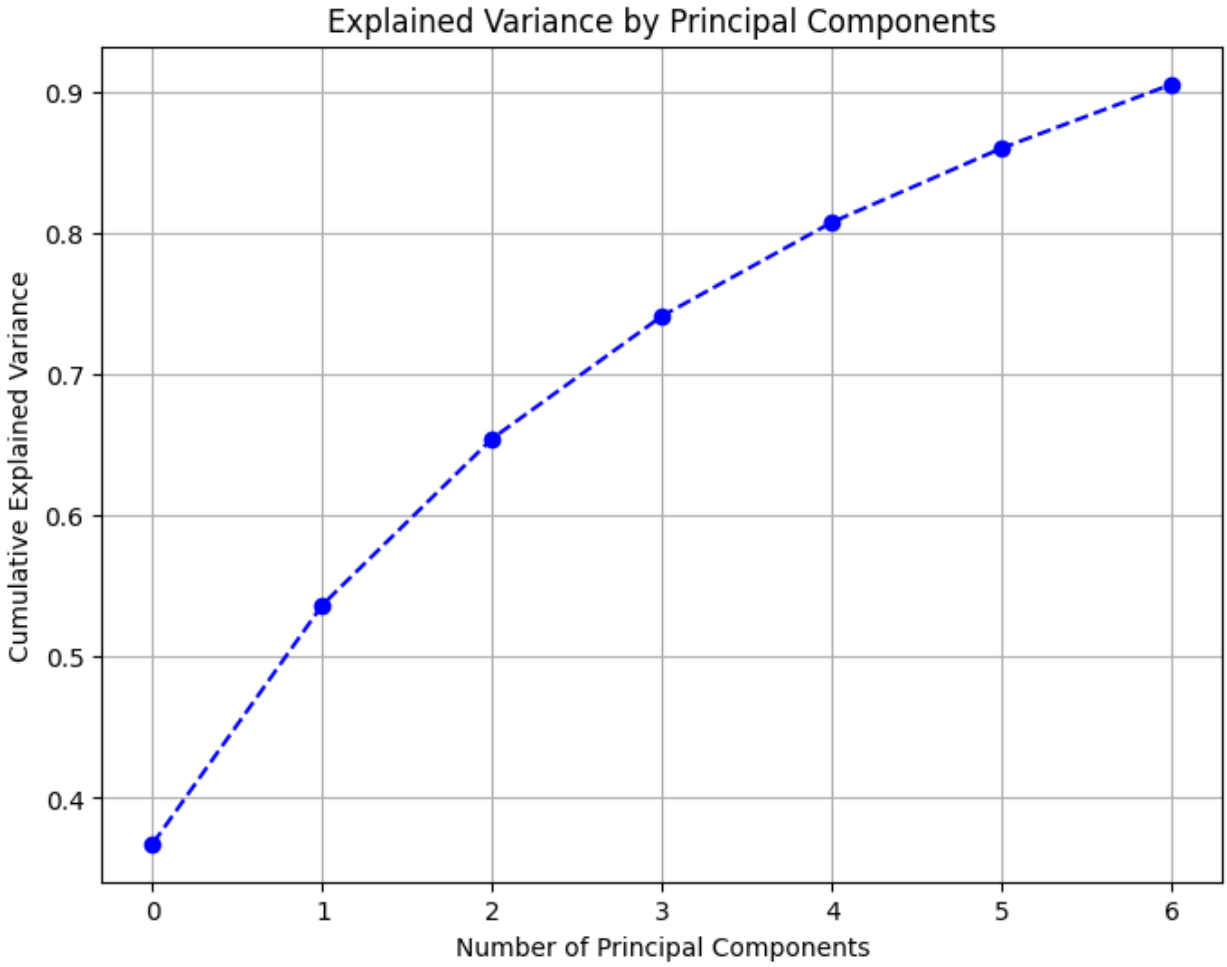
```
loadings = pd.DataFrame(pca.components_.T, columns=[f"PC{i+1}" for i
in range(pca.n_components_)], index=features)


plt.figure(figsize=(10, 8))
loadings.plot(kind='bar', figsize=(12, 6), legend=True)
plt.title("Feature Contribution to Principal Components")
plt.savefig("feature_contributions_to_pcs.png", dpi=300,
bbox_inches='tight')
plt.show()
```

```
<Figure size 1000x800 with 0 Axes>
```

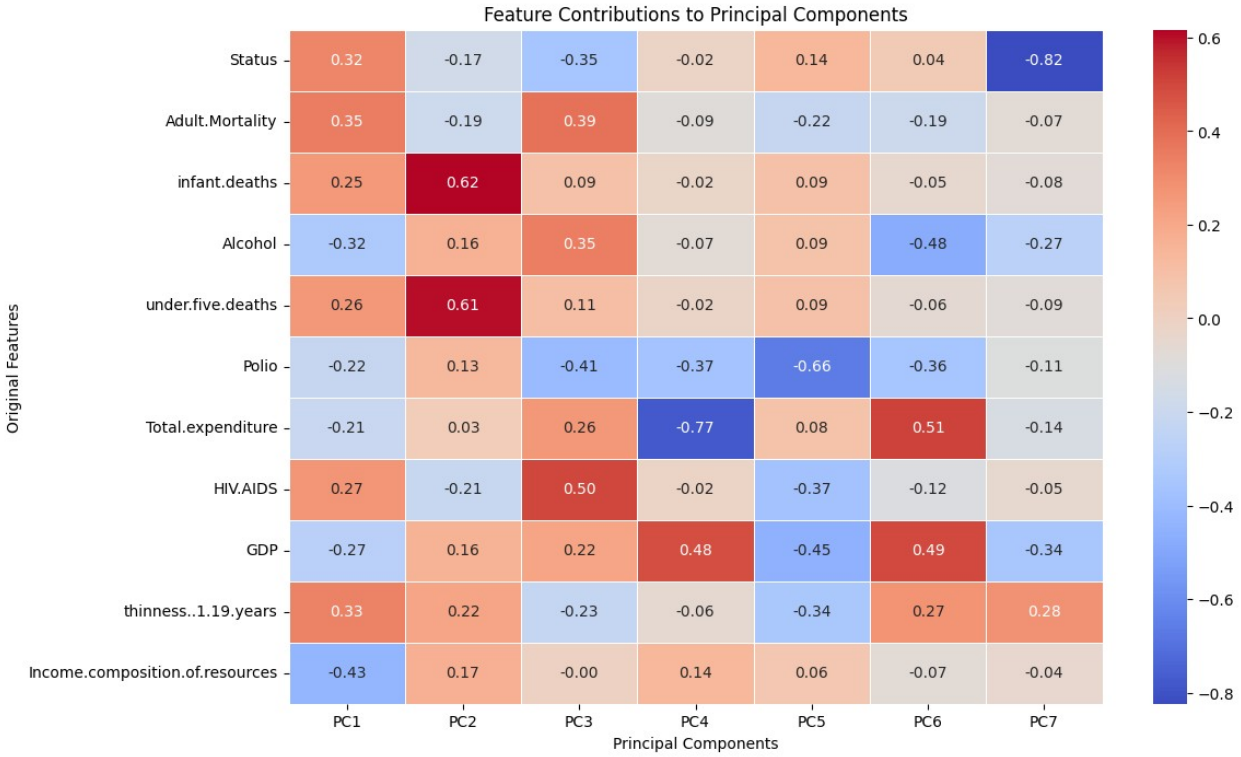



```
# Cumulative explained variance
plt.figure(figsize=(8, 6))
plt.plot(np.cumsum(pca.explained_variance_ratio_), marker='o',
```

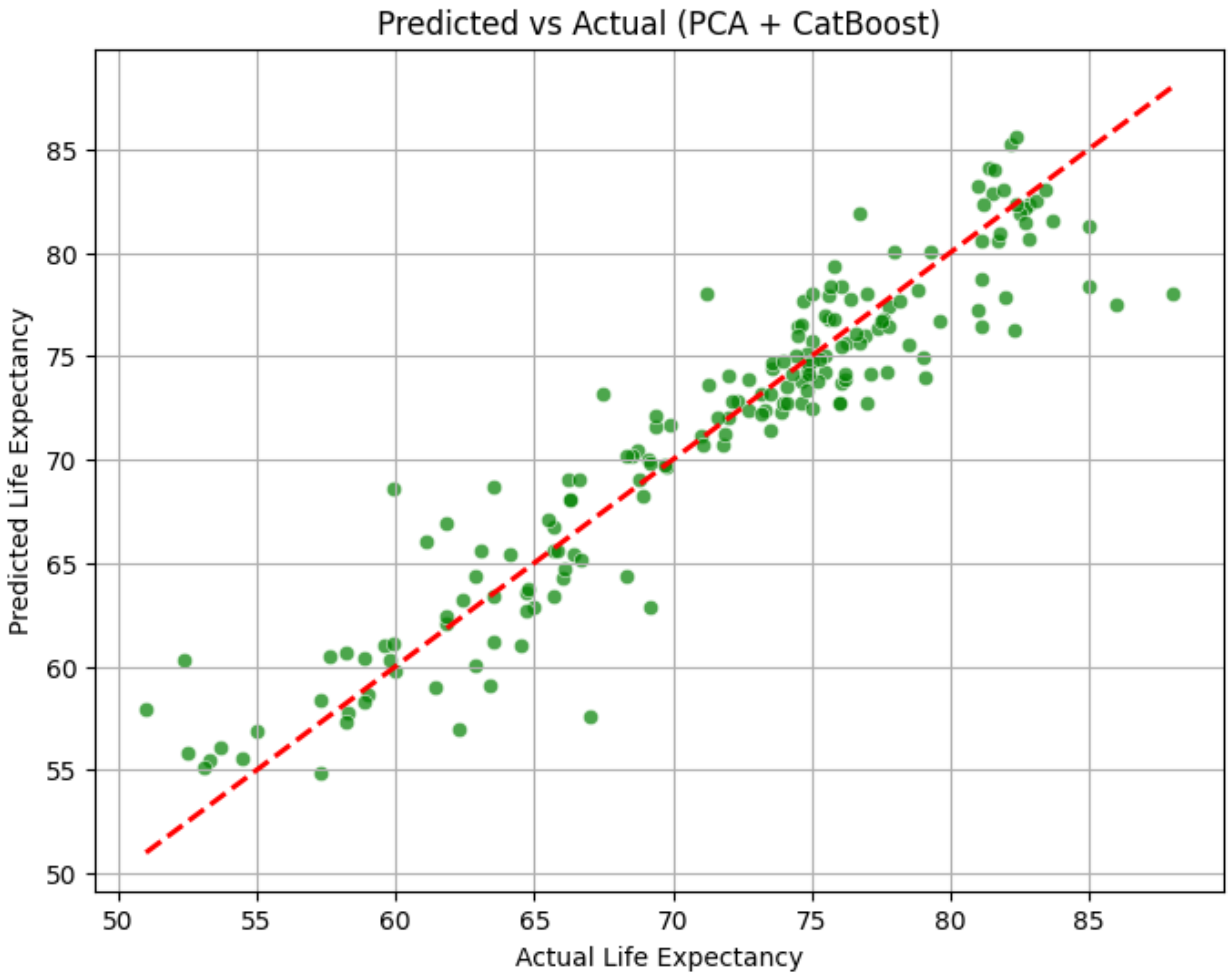
```
linestyle='--', color='b')
plt.xlabel("Number of Principal Components")
plt.ylabel("Cumulative Explained Variance")
plt.title("Explained Variance by Principal Components")
plt.grid(True)
plt.savefig("6_pca_explained_variance.png", dpi=300,
bbox_inches='tight')
plt.show()
```




```
#Feature contributions heatmapo
plt.figure(figsize=(12, 8))
sns.heatmap(loadings, annot=True, cmap="coolwarm", fmt=".2f",
linewidths=0.5)
plt.title("Feature Contributions to Principal Components")
plt.xlabel("Principal Components")
plt.ylabel("Original Features")
plt.savefig("pca_feature_contribution_heatmap.png", dpi=300,
bbox_inches='tight')
plt.show()
```

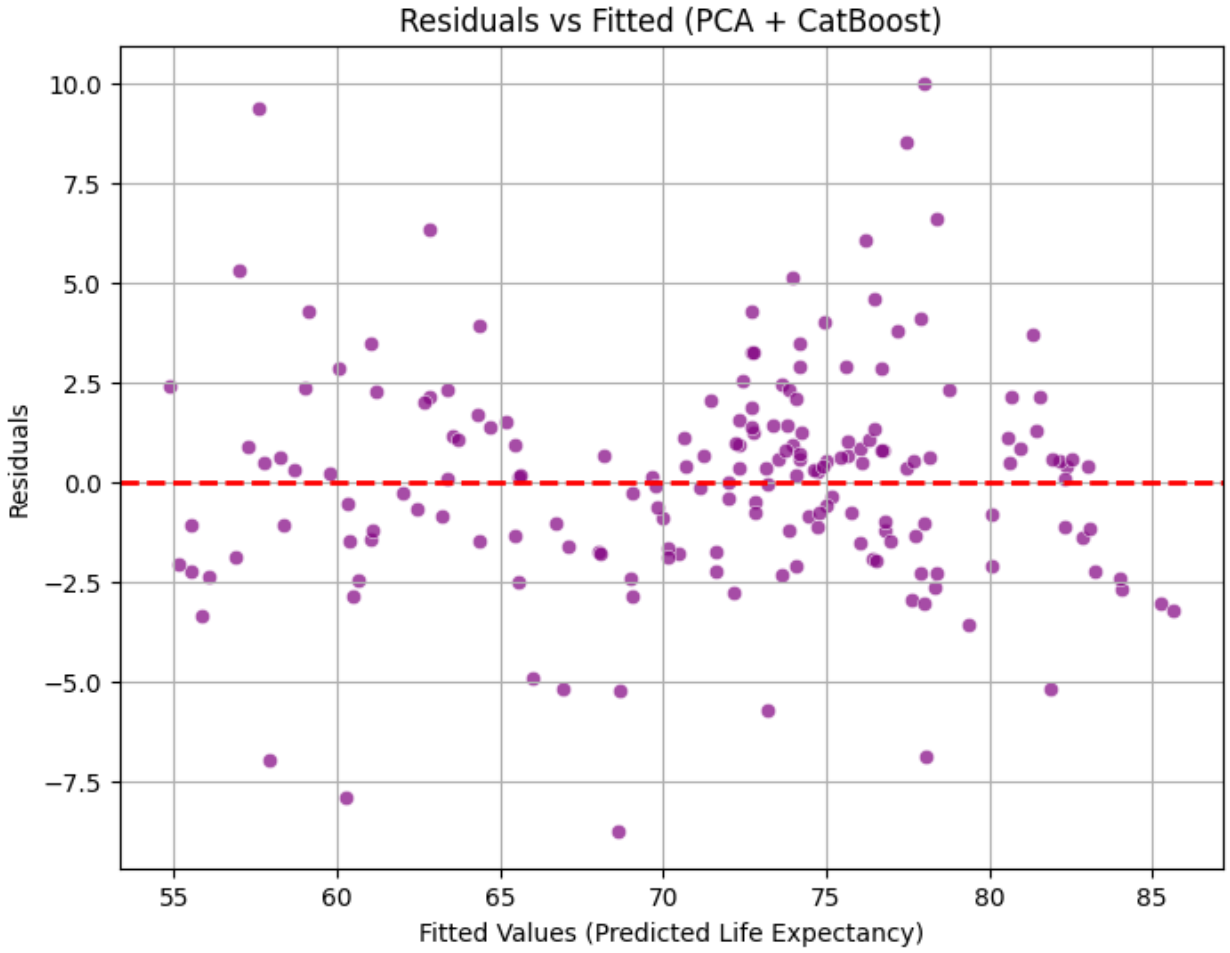



```
plt.figure(figsize=(8, 6))
sns.scatterplot(x=y_test, y=y_pred_pca_catboost, alpha=0.7,
color='green')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()],
color='red', linestyle='--', linewidth=2)
plt.xlabel("Actual Life Expectancy")
plt.ylabel("Predicted Life Expectancy")
plt.title("Predicted vs Actual (PCA + CatBoost)")
plt.grid(True)
plt.savefig("pca_catboost_predicted_vs_actual.png", dpi=300,
bbox_inches='tight')
plt.show()
```

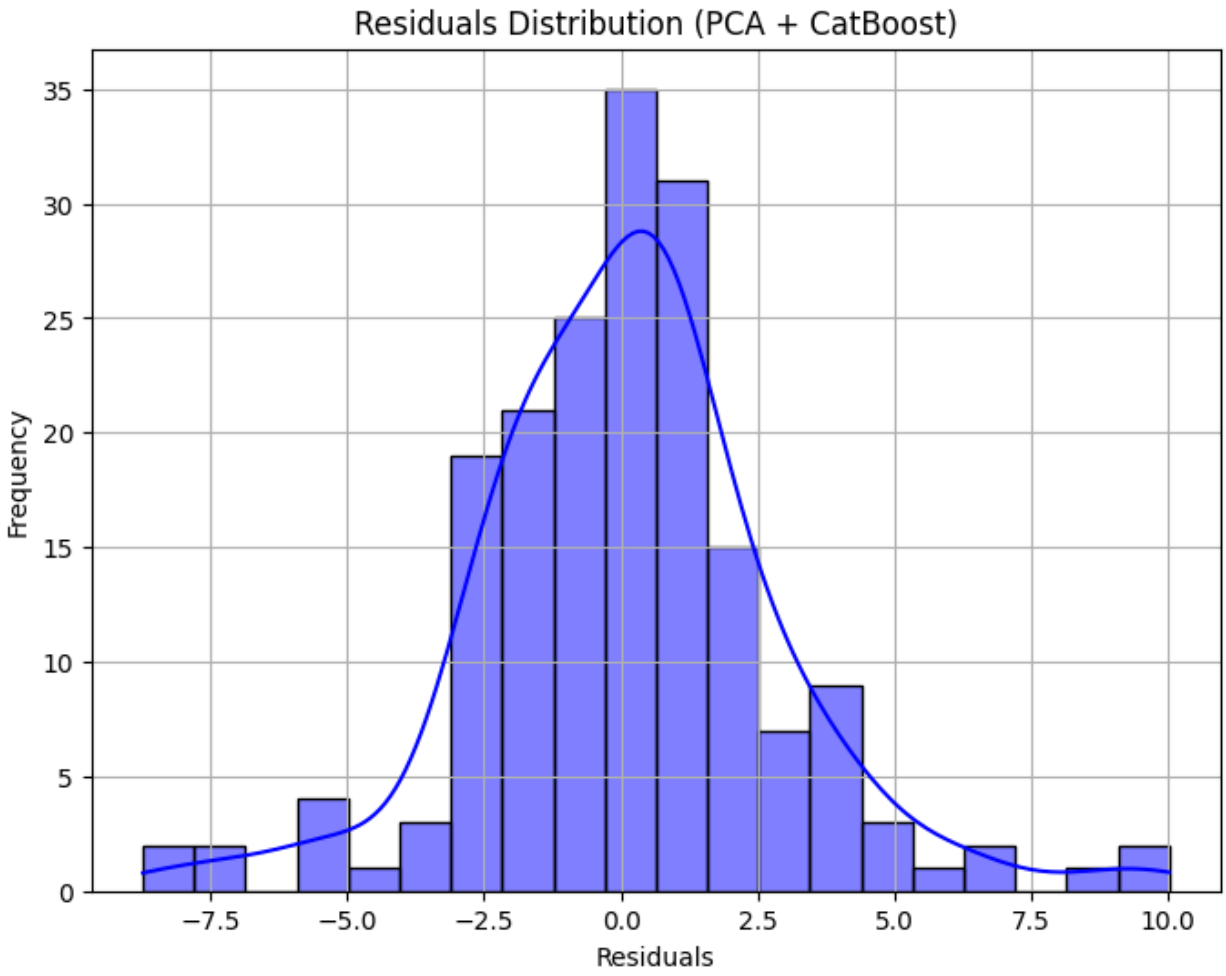



```
residuals_pca = y_test - y_pred_pca_catboost

plt.figure(figsize=(8, 6))
sns.scatterplot(x=y_pred_pca_catboost, y=residuals_pca, alpha=0.7,
color='purple')
plt.axhline(0, color='red', linestyle='--', linewidth=2)
plt.xlabel("Fitted Values (Predicted Life Expectancy)")
plt.ylabel("Residuals")
plt.title("Residuals vs Fitted (PCA + CatBoost)")
plt.grid(True)
plt.savefig("pca_catboost_residuals_vs_fitted.png", dpi=300,
bbox_inches='tight')
plt.show()

plt.figure(figsize=(8, 6))
sns.histplot(residuals_pca, kde=True, bins=20, color="blue")
plt.xlabel("Residuals")
plt.ylabel("Frequency")
plt.title("Residuals Distribution (PCA + CatBoost)")
plt.grid(True)
```

```
plt.savefig("pca_catboost_residual_histogram.png", dpi=300,
bbox_inches='tight')
plt.show()
```




```
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
```


Residuals Distribution (PCA + CatBoost)
Frequency
Residuals
0
5
10
15
20
25
30
35
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

In [1]:
```r
if (!require("MASS")) install.packages("MASS")   # For stepwise regression
if (!require("leaps")) install.packages("leaps") # For all-subset regression
if (!require("glmulti")) install.packages("glmulti") # For automated all-subse

library(MASS)
library(leaps)
library(glmulti)
```

```
Loading required package: MASS

Loading required package: leaps

Loading required package: glmulti

Warning message in library(package, lib.loc = lib.loc, character.only = TRUE,
logical.return = TRUE, :
“there is no package called ‘glmulti’”
Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)

also installing the dependency ‘rJava’


Loading required package: rJava
```

In [2]:
```r
# Load the dataset
data <- read.csv("/kaggle/input/who-life/who_train.csv")

# Drop unnecessary columns
data <- subset(data, select = -c(Country, Year))

# Convert categorical variables (if needed)
data$Status <- as.factor(data$Status)

# Handle missing values
data[is.na(data)] <- 0  # Replace NA values with 0

# Check the structure of the dataset
str(data)
```

```
'data.frame':   183 obs. of  20 variables:
 $ Status                         : Factor w/ 2 levels "Developed","Developin
g": 2 2 2 2 2 2 2 1 1 2 ...
 $ Life.expectancy                : num  59.9 77.5 75.4 51.7 76.2 76.2 74.6 8
2.7 81.4 72.5 ...
 $ Adult.Mortality                : num  271 8 11 348 131 118 12 6 66 119 ...
 $ infant.deaths                  : int  64 0 21 67 0 8 1 1 0 5 ...
 $ Alcohol                        : num  0.01 4.51 0.01 8.33 8.56 ...
 $ percentage.expenditure         : num  73.5 428.7 54.2 24 2423 ...
 $ Hepatitis.B                    : num  62 98 95 64 99 94 93 91 98 94 ...
 $ Measles                        : int  492 0 0 11699 0 1 13 340 117 0 ...
 $ BMI                            : num  18.6 57.2 58.4 22.7 47 62.2 54.1 66.
1 57.1 51.5 ...
 $ under.five.deaths              : int  86 1 24 101 0 9 1 1 0 6 ...
 $ Polio                          : num  58 98 95 68 96 92 95 92 98 97 ...
 $ Total.expenditure              : num  8.18 5.88 7.21 3.31 5.54 ...
 $ Diphtheria                     : num  62 98 95 64 99 94 93 92 98 94 ...
 $ HIV.AIDS                       : num  0.1 0.1 0.1 2 0.2 0.1 0.1 0.1 0.1 0.
1 ...
 $ GDP                            : num  613 4576 548 479 12888 ...
 $ Population                     : num  327582 288914 39113313 2692466 0 ...
 $ thinness..1.19.years           : num  17.5 1.2 6 8.5 3.3 1 2.1 0.6 1.8 2.8
...
 $ thinness.5.9.years             : num  17.5 1.3 5.8 8.3 3.3 0.9 2.1 0.6 2
2.9 ...
 $ Income.composition.of.resources: num  0.476 0.761 0.741 0.527 0.782 0.825
0.739 0.936 0.892 0.752 ...
 $ Schooling                      : num  10 14.2 14.4 11.4 13.9 17.3 12.7 20.
4 15.9 12.2 ...
```

In [3]:
```r
# Fit the full linear model
full_model <- lm(Life.expectancy ~ ., data = data)

# Display summary of the full model
summary(full_model)
```

```
Call:
lm(formula = Life.expectancy ~ ., data = data)

Residuals:
     Min       1Q   Median       3Q      Max 
-12.4441  -1.9236   0.0184   2.1429  13.1074 

Coefficients:
                                 Estimate Std. Error t value Pr(>|t|)    
(Intercept)                     6.873e+01  2.294e+00  29.958  < 2e-16 ***
StatusDeveloping               -2.530e+00  1.003e+00  -2.522 0.012622 *  
Adult.Mortality                -2.825e-02  3.931e-03  -7.187 2.26e-11 ***
infant.deaths                   1.253e-01  6.355e-02   1.972 0.050285 .  
Alcohol                         2.526e-01  9.098e-02   2.776 0.006146 ** 
percentage.expenditure         -9.047e-05  2.990e-04  -0.303 0.762614    
Hepatitis.B                    -2.087e-02  1.453e-02  -1.437 0.152738    
Measles                        -2.475e-05  5.408e-05  -0.458 0.647810    
BMI                             1.993e-02  1.702e-02   1.171 0.243253    
under.five.deaths              -9.288e-02  4.373e-02  -2.124 0.035184 *  
Polio                           1.895e-02  2.256e-02   0.840 0.402310    
Total.expenditure               1.946e-01  1.178e-01   1.652 0.100381    
Diphtheria                      3.280e-02  2.368e-02   1.385 0.167813    
HIV.AIDS                       -1.055e+00  2.788e-01  -3.782 0.000218 ***
GDP                             4.165e-05  4.420e-05   0.942 0.347377    
Population                     -2.457e-09  7.434e-09  -0.330 0.741482    
thinness..1.19.years           -2.692e-01  2.598e-01  -1.036 0.301669    
thinness.5.9.years              2.616e-02  2.569e-01   0.102 0.919007    
Income.composition.of.resources 1.023e+01  5.189e+00   1.971 0.050417 .  
Schooling                      -1.231e-01  2.627e-01  -0.469 0.639943    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.866 on 163 degrees of freedom
Multiple R-squared:  0.8174,    Adjusted R-squared:  0.7961 
F-statistic:  38.4 on 19 and 163 DF,  p-value: < 2.2e-16
```

In [4]:
```r
# Perform AIC-based stepwise regression
stepwise_model_aic <- step(full_model, direction = "both", trace = TRUE)

# Summary of the AIC-selected model
summary(stepwise_model_aic)
```

```
Start:  AIC=513.69
Life.expectancy ~ 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.year
s + 
    Income.composition.of.resources + Schooling

                                 Df Sum of Sq    RSS    AIC
- thinness.5.9.years              1      0.15 2435.8 511.70
- percentage.expenditure          1      1.37 2437.0 511.79
- Population                      1      1.63 2437.3 511.81
- Measles                         1      3.13 2438.8 511.93
- Schooling                       1      3.28 2438.9 511.94
- Polio                           1     10.54 2446.2 512.48
- GDP                             1     13.27 2448.9 512.68
- thinness..1.19.years            1     16.04 2451.7 512.89
- BMI                             1     20.49 2456.1 513.22
<none>                                       2435.6 513.69
```

In [5]:
```r
# Perform BIC-based stepwise regression
n <- nrow(data)  # Number of observations
stepwise_model_bic <- step(full_model, direction = "both", k = log(n), trace =

# Summary of the BIC-selected model
summary(stepwise_model_bic)
```

```
Start:  AIC=577.88
Life.expectancy ~ 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.year
s + 
    Income.composition.of.resources + Schooling

                                 Df Sum of Sq    RSS    AIC
- thinness.5.9.years              1      0.15 2435.8 572.68
- percentage.expenditure          1      1.37 2437.0 572.77
- Population                      1      1.63 2437.3 572.79
- Measles                         1      3.13 2438.8 572.91
- Schooling                       1      3.28 2438.9 572.92
- Polio                           1     10.54 2446.2 573.46
- GDP                             1     13.27 2448.9 573.67
- thinness..1.19.years            1     16.04 2451.7 573.87
- BMI                             1     20.49 2456.1 574.20
- Diphtheria                      1     28.68 2464.3 574.81
```

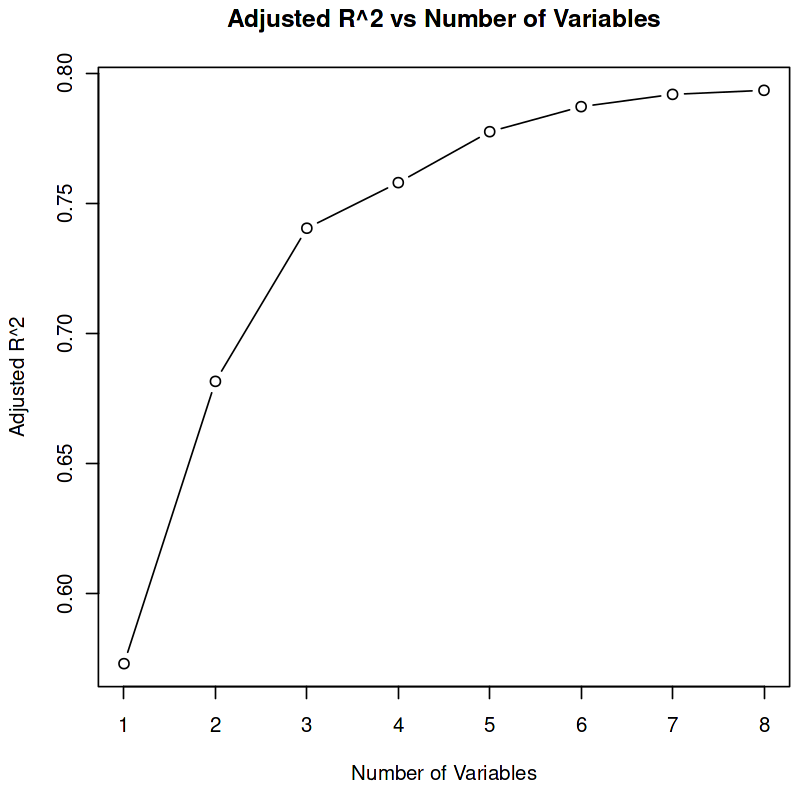
In [6]:

```r
# Adjusted R^2 for the AIC-selected model
adj_r2_aic <- summary(stepwise_model_aic)$adj.r.squared
cat("Adjusted R^2 for AIC-selected model:", adj_r2_aic, "\n")

# Adjusted R^2 for the BIC-selected model
adj_r2_bic <- summary(stepwise_model_bic)$adj.r.squared
cat("Adjusted R^2 for BIC-selected model:", adj_r2_bic, "\n")
```

```
Adjusted R^2 for AIC-selected model: 0.7996924
Adjusted R^2 for BIC-selected model: 0.7921614
```

In [7]:
```r
# Perform all-subset regression using leaps
all_subset <- regsubsets(Life.expectancy ~ ., data = data, nbest = 1)

# View summary of all-subset regression
subset_summary <- summary(all_subset)

# Plot Adjusted R^2 for each model size
plot(subset_summary$adjr2, xlab = "Number of Variables", ylab = "Adjusted R^2"
     main = "Adjusted R^2 vs Number of Variables")

# Identify the best model based on Adjusted R^2
best_model_size <- which.max(subset_summary$adjr2)
cat("Best model size:", best_model_size, "\n")

# Variables in the best model
best_model_vars <- names(coef(all_subset, best_model_size))
cat("Variables in the best model:", paste(best_model_vars, collapse = ", "), "
```

```
Best model size: 8
Variables in the best model: (Intercept), StatusDeveloping, Adult.Mortality,
Alcohol, Polio, Total.expenditure, HIV.AIDS, thinness.5.9.years, Income.compo
sition.of.resources
```



In [8]:
```r
# Perform all-subset regression using glmulti (AIC criterion)
glmulti_fit <- glmulti(Life.expectancy ~ ., data = data, level = 1, crit = "ai

# Summary of the best model
best_glmulti_model <- glmulti_fit@formulas[[1]]
best_model_fit <- lm(best_glmulti_model, data = data)
summary(best_model_fit)

# Display the formula for the best model
cat("Best model formula (glmulti):", as.character(best_glmulti_model), "\n")
```

```
Initialization...
TASK: Exhaustive screening of candidate set.
Fitting...

After 50 models:
Best model: Life.expectancy~1+Adult.Mortality+infant.deaths+Alcohol+percen
tage.expenditure
Crit= 1096.8789069601
Mean crit= 1196.69649114838

After 100 models:
Best model: Life.expectancy~1+Adult.Mortality+infant.deaths+Alcohol+percen
tage.expenditure+BMI
Crit= 1088.92843317719
Mean crit= 1187.17414391812

After 150 models:
Best model: Life.expectancy~1+Adult.Mortality+infant.deaths+Alcohol+percen
tage.expenditure+BMI
```

In [9]:
```r
# Compare Adjusted R^2 for all models
cat("Adjusted R^2 for full model:", summary(full_model)$adj.r.squared, "\n")
cat("Adjusted R^2 for AIC-selected model:", adj_r2_aic, "\n")
cat("Adjusted R^2 for BIC-selected model:", adj_r2_bic, "\n")
cat("Adjusted R^2 for best all-subset model:", max(subset_summary$adjr2), "\n"
```

```
Adjusted R^2 for full model: 0.7961125
Adjusted R^2 for AIC-selected model: 0.7996924
Adjusted R^2 for BIC-selected model: 0.7921614
Adjusted R^2 for best all-subset model: 0.7933889
```