Mini Project

Life Expectancy Prediction Using ML

Aim: Write a program to implement a Machine Learning application (Predicting **Life Expectancy** using different Regression Algorithms and comparing their performance on the Life Expectancy dataset).

Dataset Used:

This dataset explores the factors influencing life expectancy across various countries and years, aiming to uncover patterns and disparities in health outcomes based on geographic locations. By examining key features such as adult mortality, alcohol consumption, healthcare expenditures, and socioeconomic indicators, this dataset provides insights into the complex interplay of factors shaping life expectancy worldwide.

Feature Description:

Feature	Description
Country	Name of the country
Year	Year of observation
Status	Urban or rural status
Life expectancy	Life expectancy at birth in years
Adult Mortality	Probability of dying between 15 and 60 years per 1000
Infant deaths	Number of infant deaths per 1000 population
Alcohol	Alcohol consumption, measured as liters per capita
Percentage expenditure	Expenditure on health as a percentage of GDP

Feature	Description
Hepatitis B	Hepatitis B immunization coverage among 1-year-olds (%)
Measles	Number of reported measles cases per 1000 population
BMI	Average Body Mass Index of the population
Under-five deaths	Number of deaths under age five per 1000 population
Polio	Polio immunization coverage among 1-year-olds (%)
Total expenditure	Total government health expenditure as a percentage of GDP
Diphtheria	Diphtheria tetanus toxoid and pertussis immunization coverage among 1-year-olds (%)
HIV/AIDS	Deaths per 1 000 live births due to HIV/AIDS (0-4 years)
GDP	Gross Domestic Product per capita (in USD)
Population	Population of the country
Thinness 1-19 years	Prevalence of thinness among children and adolescents aged 10–19 (%)
Thinness 5-9 years	Prevalence of thinness among children aged 5–9 (%)
Income composition of resources	Human Development Index in terms of income composition of resources (0 to 1)
Schooling	Number of years of schooling

Target Variable:

- Life expectancy

Algorithm:

1. Linear Regression

Linear Regression is a supervised learning regression algorithm that models the relationship between one dependent variable (Life Expectancy) and multiple independent variables (e.g., BMI, schooling, alcohol consumption).

It assumes a linear relationship:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n$$

Where:

- y = Predicted Life Expectancy
- x = Input features
- β_0 = Intercept
- $\beta_1 ... \beta_n$ = Weights learned by the model

Best values of β are found using the **Least Squares Method**:

$$\beta_1 = \frac{n\sum xy - (\sum x)(\sum y)}{n\sum x^2 - (\sum x)^2}$$

Linear Regression tries to minimize:

$$MSE = \frac{1}{n}\sum (y_i - \hat{y}_i)^2$$

2. Random Forest Regression

Random Forest is an ensemble learning method that builds multiple decision trees and averages their predictions to improve accuracy.

Working:

1. Select random subsets of data.
2. Build separate Decision Trees on each subset.
3. Each tree predicts Life Expectancy.
4. Final prediction = **Average of all tree predictions**.

$$\hat{y} = \frac{1}{N}\sum_{i=1}^{N} y_i$$

Advantages:

- Reduces overfitting
- Handles non-linear data
- More accurate than individual models

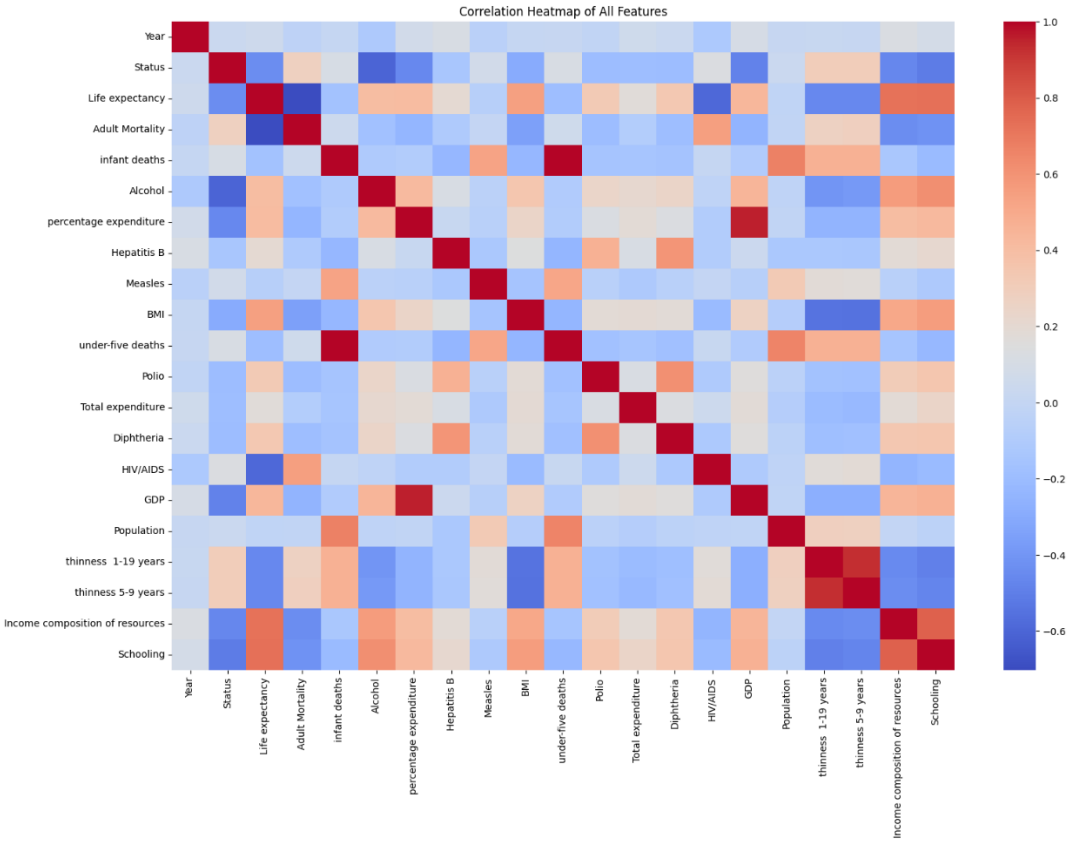
Program:

```
# 1. Import Libraries

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

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score, mean_squared_error


# 2. Load Dataset

df = pd.read_csv("/content/LifeExpectancy.csv")
df.columns = df.columns.str.strip()

# Remove missing values
df = df.dropna()
df = df.drop("Country", axis=1)
# Encode Status (Developed / Developing)
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df["Status"] = le.fit_transform(df["Status"])


# 3. HEATMAP of ALL FEATURES

plt.figure(figsize=(18,12))
sns.heatmap(df.corr(), cmap='coolwarm', annot=False)
plt.title("Correlation Heatmap of All Features")
plt.show()

# 4. SELECT HIGHLY CORRELATED FEATURES

target = "Life expectancy"

corr_with_target = df.corr()[target].abs().sort_values(ascending=False)

print("\nTop Features Correlated with Life Expectancy:\n")
print(corr_with_target)

# Select features with correlation > 0.4 (you can adjust threshold)
selected_features = corr_with_target[corr_with_target >
0.4].index.tolist()
```

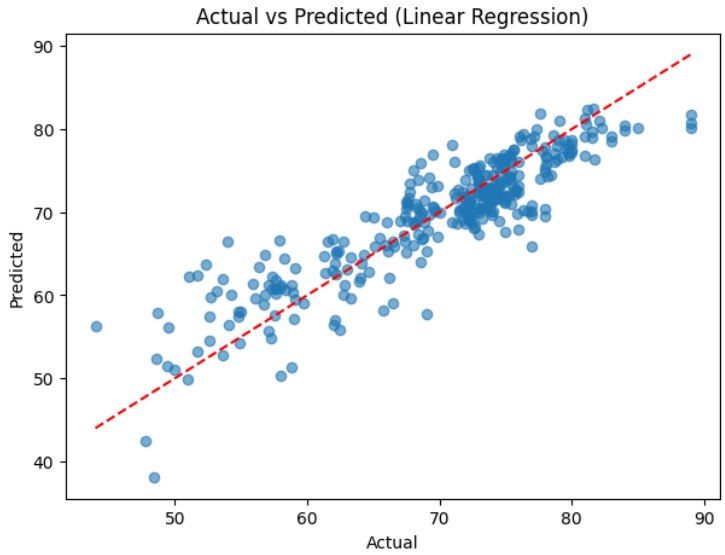
```
# Remove the target itself from the feature list
selected_features.remove(target)

print("\nSelected Features for Training:\n")
print(selected_features)

# 5. Prepare Data for Model

X = df[selected_features]
y = df[target]

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# 6. Train Linear Regression

lr = LinearRegression()
lr.fit(X_train, y_train)
lr_pred = lr.predict(X_test)


# 7. Train Random Forest

rf = RandomForestRegressor(random_state=42)
rf.fit(X_train, y_train)
rf_pred = rf.predict(X_test)

# 8. Model Evaluation
print("\n---- Linear Regression ----")
print("R2 Score:", r2_score(y_test, lr_pred))
print("MSE:", mean_squared_error(y_test, lr_pred))

print("\n---- Random Forest ----")
print("R2 Score:", r2_score(y_test, rf_pred))
print("MSE:", mean_squared_error(y_test, rf_pred))


# 9. ACTUAL vs PREDICTED PLOTS

plt.figure(figsize=(7,5))
plt.scatter(y_test, lr_pred, alpha=0.6)
plt.plot([y_test.min(), y_test.max()],
         [y_test.min(), y_test.max()],
         'r--')
plt.title("Actual vs Predicted (Linear Regression)")
plt.xlabel("Actual")
plt.ylabel("Predicted")
plt.show()
```

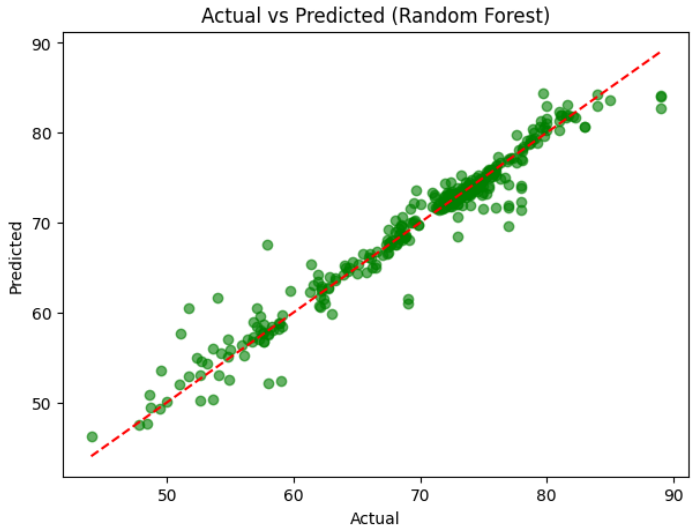
```
plt.figure(figsize=(7,5))
plt.scatter(y_test, rf_pred, alpha=0.6, color='green')
plt.plot([y_test.min(), y_test.max()],
         [y_test.min(), y_test.max()],
         'r--')
plt.title("Actual vs Predicted (Random Forest)")
plt.xlabel("Actual")
plt.ylabel("Predicted")
plt.show()


# 10. MODEL COMPARISON

scores = [r2_score(y_test, lr_pred), r2_score(y_test, rf_pred)]
models = ["Linear Regression", "Random Forest"]

plt.figure(figsize=(7,5))
plt.bar(models, scores, color=["blue", "green"])
plt.ylabel("R2 Score")
plt.title("Model Comparison")
plt.ylim(0,1)
plt.show()
```

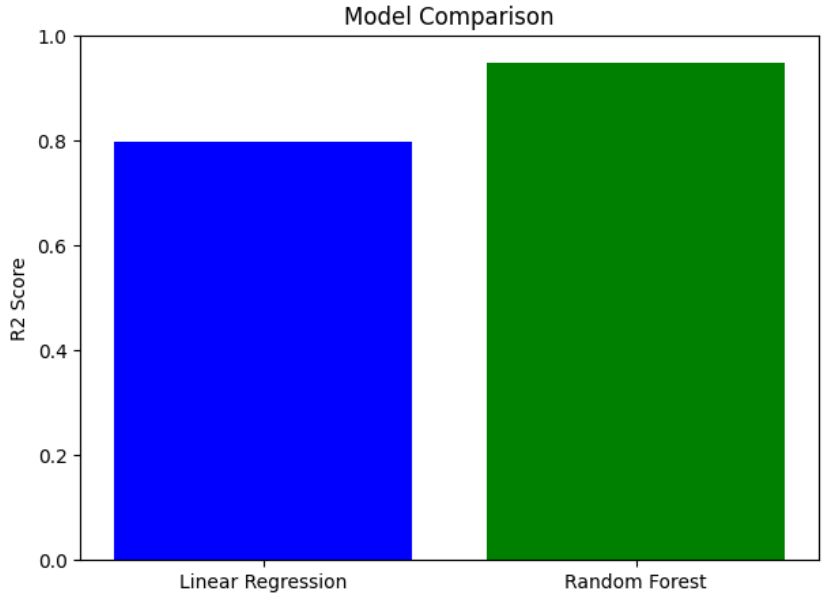
Output:



```
Top Features Correlated with Life Expectancy:

Life expectancy                    1.000000
Schooling                          0.727630
Income composition of resources    0.721083
Adult Mortality                    0.702523
HIV/AIDS                           0.592236
BMI                                0.542042
thinness  1-19 years               0.457838
thinness 5-9 years                 0.457508
Status                             0.442798
GDP                                0.441322
percentage expenditure             0.409631
Alcohol                            0.402718
Diphtheria                         0.341331
Polio                              0.327294
Hepatitis B                        0.199935
under-five deaths                  0.192265
Total expenditure                  0.174718
infant deaths                      0.169074
Measles                            0.068881
Year                               0.050771
Population                         0.022305
Name: Life expectancy, dtype: float64

Selected Features for Training:

['Schooling', 'Income composition of resources', 'Adult Mortality',
'HIV/AIDS', 'BMI', 'thinness  1-19 years', 'thinness 5-9 years',
'Status', 'GDP', 'percentage expenditure', 'Alcohol']

---- Linear Regression ----
R2 Score: 0.7980060121411877
MSE: 14.346114867993228

---- Random Forest ----
R2 Score: 0.9478852543788122
MSE: 3.701318712121211
```


Actual vs Predicted (Linear Regression)
Predicted
Actual
90
80
70
60
50
40
50
60
70
80
90


Actual vs Predicted (Random Forest)
Predicted
Actual
90
80
70
60
50
50
60
70
80
90


Model Comparison
R2 Score
1.0
0.8
0.6
0.4
0.2
0.0
Linear Regression
Random Forest

S