```
# basic libraries
import pandas as pd

df = pd.read_csv("Life Expectancy Data.csv")

df.columns = df.columns.str.strip()

# quick checks so we know it loaded correctly
print("Shape (rows, columns):", df.shape)
display(df.head(3))
print("Columns:", df.columns.tolist())
```

```
Shape (rows, columns): (2938, 22)
```

	Country	Year	Status	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles	...	Polio	Total expenditure	Diphtheria	HIV/AIDS	GDP	Population	thinness 1-19 years	thinness 5-9 years	Income composition of resources	Schooling
0	Afghanistan	2015	Developing	65.0	263.0	62	0.01	71.279624	65.0	1154	...	6.0	8.16	65.0	0.1	584.259210	33736494.0	17.2	17.3	0.479	10.1
1	Afghanistan	2014	Developing	59.9	271.0	64	0.01	73.523582	62.0	492	...	58.0	8.18	62.0	0.1	612.696514	327582.0	17.5	17.5	0.476	10.0
2	Afghanistan	2013	Developing	59.9	268.0	66	0.01	73.219243	64.0	430	...	62.0	8.13	64.0	0.1	631.744976	31731688.0	17.7	17.7	0.470	9.9

3 rows × 22 columns

```
Columns: ['Country', 'Year', 'Status', 'Life expectancy', 'Adult Mortality', 'infant deaths', 'Alcohol', 'percentage expenditure', 'Hepatitis B', 'Measles', 'BMI', 'under-five deaths', 'Polio', 'Total expenditure', 'Di
```

```
# summary statistics for all numeric columns
print("Summary of numeric columns:")
display(df.select_dtypes(include=["number"]).describe().T)

# check how many records exist for each year
print("\nRecord count by Year:")
print(df["Year"].value_counts().sort_index())
```

```
Summary of numeric columns:
```

	count	mean	std	min	25%	50%	75%	max
Year	2938.0	2.007519e+03	4.613841e+00	2000.00000	2004.000000	2.008000e+03	2.012000e+03	2.015000e+03
Life expectancy	2928.0	6.922493e+01	9.523867e+00	36.30000	63.100000	7.210000e+01	7.570000e+01	8.900000e+01
Adult Mortality	2928.0	1.647964e+02	1.242921e+02	1.00000	74.000000	1.440000e+02	2.280000e+02	7.230000e+02
infant deaths	2938.0	3.030395e+01	1.179265e+02	0.00000	0.000000	3.000000e+00	2.200000e+01	1.800000e+03
Alcohol	2744.0	4.602861e+00	4.052413e+00	0.01000	0.877500	3.755000e+00	7.702500e+00	1.787000e+01
percentage expenditure	2938.0	7.382513e+02	1.987915e+03	0.00000	4.685343	6.491291e+01	4.415341e+02	1.947991e+04
Hepatitis B	2385.0	8.094046e+01	2.507002e+01	1.00000	77.000000	9.200000e+01	9.700000e+01	9.900000e+01
Measles	2938.0	2.419592e+03	1.146727e+04	0.00000	0.000000	1.700000e+01	3.602500e+02	2.121830e+05
BMI	2904.0	3.832125e+01	2.004403e+01	1.00000	19.300000	4.350000e+01	5.620000e+01	8.730000e+01
under-five deaths	2938.0	4.203574e+01	1.604455e+02	0.00000	0.000000	4.000000e+00	2.800000e+01	2.500000e+03
Polio	2919.0	8.255019e+01	2.342805e+01	3.00000	78.000000	9.300000e+01	9.700000e+01	9.900000e+01
Total expenditure	2712.0	5.938190e+00	2.498320e+00	0.37000	4.260000	5.755000e+00	7.492500e+00	1.760000e+01
Diphtheria	2919.0	8.232408e+01	2.371691e+01	2.00000	78.000000	9.300000e+01	9.700000e+01	9.900000e+01
HIV/AIDS	2938.0	1.742103e+00	5.077785e+00	0.10000	0.100000	1.000000e-01	8.000000e-01	5.060000e+01
GDP	2490.0	7.483158e+03	1.427017e+04	1.68135	463.935626	1.766948e+03	5.910806e+03	1.191727e+05
Population	2286.0	1.275338e+07	6.101210e+07	34.00000	195793.250000	1.386542e+06	7.420359e+06	1.293859e+09
thinness 1-19 years	2904.0	4.839704e+00	4.420195e+00	0.10000	1.600000	3.300000e+00	7.200000e+00	2.770000e+01
thinness 5-9 years	2904.0	4.870317e+00	4.508882e+00	0.10000	1.500000	3.300000e+00	7.200000e+00	2.860000e+01
Income composition of resources	2771.0	6.275511e-01	2.109036e-01	0.00000	0.493000	6.770000e-01	7.790000e-01	9.480000e-01
Schooling	2775.0	1.199279e+01	3.358920e+00	0.00000	10.100000	1.230000e+01	1.430000e+01	2.070000e+01

```
Record count by Year:
Year
2000    183
2001    183
2002    183
2003    183
2004    183
2005    183
2006    183
2007    183
2008    183
2009    183
2010    183
2011    183
2012    183
2013    193
2014    183
2015    183
Name: count, dtype: int64
```

```
# Handling Missing Values

print("Missing values per column (top 10):")
display(df.isna().sum().sort_values(ascending=False).head(10).to_frame("Missing Count"))

# Fill numeric columns with their median value
num_cols = df.select_dtypes(include="number").columns
df[num_cols] = df[num_cols].fillna(df[num_cols].median())

cat_cols = df.select_dtypes(exclude="number").columns
for c in cat_cols:
    df[c] = df[c].fillna(df[c].mode()[0])

print("\nTotal missing values left:", df.isna().sum().sum())
```

```
Missing values per column (top 10):
```

	Missing Count
Population	652
Hepatitis B	553
GDP	448
Total expenditure	226
Alcohol	194
Income composition of resources	167
Schooling	163
thinness 1-19 years	34
thinness 5-9 years	34
BMI	34

```
Total missing values left: 0
```

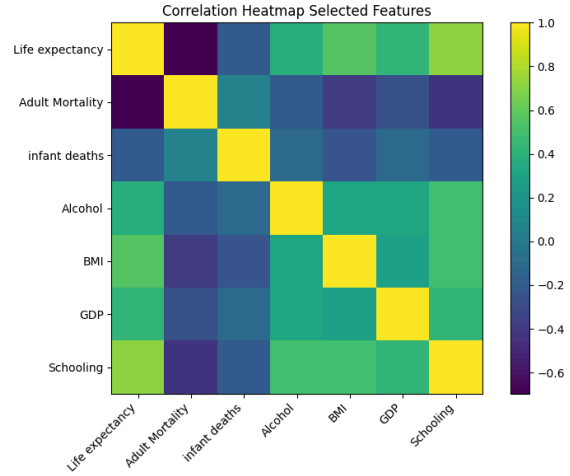
4. Correlation Heatmap (EDA)

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

# pick a subset of columns for correlation (you can add more later)
corr_cols = ["Life expectancy", "Adult Mortality", "infant deaths",
             "Alcohol", "BMI", "GDP", "Schooling"]

corr_df = df[corr_cols].corr()

plt.figure(figsize=(8,6))
im = plt.imshow(corr_df, interpolation="nearest")
plt.colorbar(im)
plt.xticks(range(len(corr_cols)), corr_cols, rotation=45, ha="right")
plt.yticks(range(len(corr_cols)), corr_cols)
plt.title("Correlation Heatmap Selected Features")
plt.tight_layout()
plt.show()
```



According to the heatmap, life expectancy is positively correlated with education, GDP, and BMI, but it is highly negatively correlated with newborn and adult mortality. This validates the models' use of these variables as predictors.

5. Baseline Model: Multiple Linear Regression

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import numpy as np

# Target and features
target = "Life expectancy"
features = ["Adult Mortality", "Alcohol", "BMI", "GDP", "Schooling", "infant deaths"]

# Filter only existing columns
features = [f for f in features if f in df.columns]
print("Features used:", features)

X = df[features]
y = df[target]

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# Model
lin_model = LinearRegression()
lin_model.fit(X_train, y_train)

# Predictions
y_train_pred = lin_model.predict(X_train)
y_test_pred = lin_model.predict(X_test)

# Metrics
mae = mean_absolute_error(y_test, y_test_pred)
rmse = np.sqrt(mean_squared_error(y_test, y_test_pred))
r2_train = r2_score(y_train, y_train_pred)
r2_test = r2_score(y_test, y_test_pred)

print("\n--- Linear Regression Performance ---")
print("Train R²:", round(r2_train, 3))
print("Test  R²:", round(r2_test, 3))
print("Test MAE (years):", round(mae, 2))
```

```
print("Test RMSE (years):", round(rmse, 2))


Features used: ['Adult Mortality', 'Alcohol', 'BMI', 'GDP', 'Schooling', 'infant deaths']

--- Linear Regression Performance ---
Train R²: 0.721
Test  R²: 0.731
Test MAE (years): 3.34
Test RMSE (years): 4.83
```

6. Decision Tree Regressor

```
from sklearn.tree import DecisionTreeRegressor

tree_model = DecisionTreeRegressor(random_state=42)
tree_model.fit(X_train, y_train)

y_train_tree = tree_model.predict(X_train)
y_test_tree = tree_model.predict(X_test)

mae_tree  = mean_absolute_error(y_test, y_test_tree)
rmse_tree = np.sqrt(mean_squared_error(y_test, y_test_tree))
r2_train_tree = r2_score(y_train, y_train_tree)
r2_test_tree  = r2_score(y_test, y_test_tree)

print("\n--- Decision Tree Performance ---")
print("Train R²:", round(r2_train_tree, 3))
print("Test  R²:", round(r2_test_tree, 3))
print("Test MAE (years):", round(mae_tree, 2))
print("Test RMSE (years):", round(rmse_tree, 2))
```

```
--- Decision Tree Performance ---
Train R²: 1.0
Test  R²: 0.926
Test MAE (years): 1.5
Test RMSE (years): 2.53
```

7. Random forest regressor

```
from sklearn.ensemble import RandomForestRegressor

rf_model = RandomForestRegressor(
    n_estimators=200,
    random_state=42,
    n_jobs=-1
)

rf_model.fit(X_train, y_train)

y_train_rf = rf_model.predict(X_train)
y_test_rf  = rf_model.predict(X_test)

mae_rf  = mean_absolute_error(y_test, y_test_rf)
rmse_rf = np.sqrt(mean_squared_error(y_test, y_test_rf))
r2_train_rf = r2_score(y_train, y_train_rf)
r2_test_rf  = r2_score(y_test, y_test_rf)

print("\n--- Random Forest Performance ---")
print("Train R²:", round(r2_train_rf, 3))
print("Test  R²:", round(r2_test_rf, 3))
print("Test MAE (years):", round(mae_rf, 2))
print("Test RMSE (years):", round(rmse_rf, 2))
```

```
--- Random Forest Performance ---
Train R²: 0.994
Test  R²: 0.958
Test MAE (years): 1.22
Test RMSE (years): 1.9
```

8. Model Comparison

```
import pandas as pd

results = pd.DataFrame({
    "Model": ["Linear Regression", "Decision Tree", "Random Forest"],
    "Train R²": [r2_train, r2_train_tree, r2_train_rf],
    "Test R²": [r2_test, r2_test_tree, r2_test_rf],
    "MAE": [mae, mae_tree, mae_rf],
    "RMSE": [rmse, rmse_tree, rmse_rf]
})

results
```

	Model	Train R²	Test R²	MAE	RMSE
0	Linear Regression	0.720911	0.730708	3.339229	4.831032
1	Decision Tree	0.999898	0.926230	1.504762	2.528531
2	Random Forest	0.993665	0.958161	1.218593	1.904232

9. Stability Check: Cross-Validation

```
from sklearn.model_selection import cross_val_score

cv_scores = cross_val_score(
    rf_model, X, y, cv=5, scoring='r2'
)

print("Cross-Validation R² Scores:", np.round(cv_scores, 3))
print("Mean CV R²:", round(cv_scores.mean(), 3))
```

```
Cross-Validation R² Scores: [0.916 0.89  0.918 0.922 0.904]
Mean CV R²: 0.91
```

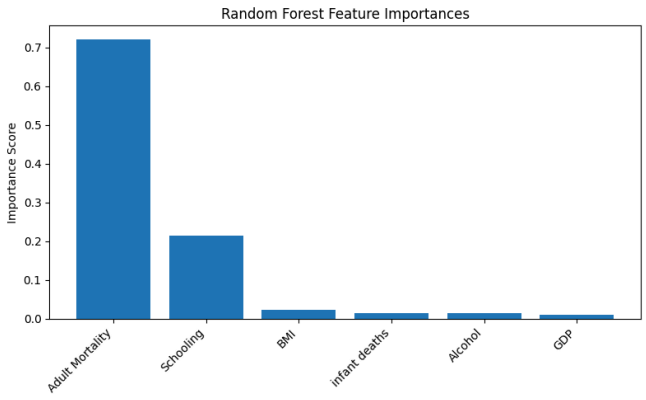
10. Feature Importance (Random Forest)

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

importances = rf_model.feature_importances_
indices = np.argsort(importances)[::-1]

plt.figure(figsize=(8,5))
```

```
plt.bar(range(len(features)), importances[indices])
plt.xticks(range(len(features)), np.array(features)[indices], rotation=45, ha="right")
plt.title("Random Forest Feature Importances")
plt.ylabel("Importance Score")
plt.tight_layout()
plt.show()
```



11. Summary of Model Findings

In this analysis, I tested three different models to predict life expectancy: Linear Regression, Decision Tree, and Random Forest. The Linear Regression model worked as a simple baseline and achieved a test R^2 of about 0.73, with the predictions being off by around 3–5 years. The Decision Tree model performed much better and reached a test R^2 of 0.93, although the training score was 1.0, which shows that it overfitted the training data.

The best results came from the Random Forest model. It achieved a test R^2 of 0.96 with much lower errors (MAE ~1.2 years, RMSE ~2 years). This means the Random Forest model was able to predict life expectancy very accurately across countries. When I checked the model's stability using cross-validation, the average score stayed around 0.91, which shows that the model is consistent and not dependent on only one train-test split.

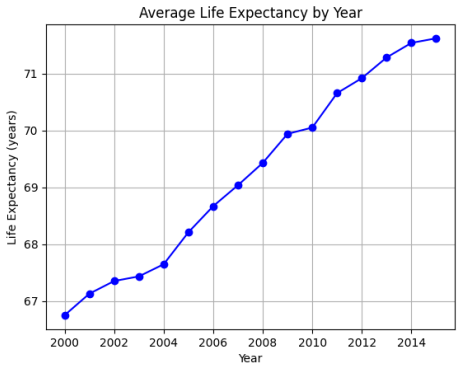
The feature importance results showed that adult mortality had the biggest impact on life expectancy, followed by schooling. The remaining features such as BMI, infant deaths, alcohol, and GDP had smaller influences. Overall, these results make sense because higher mortality and lower education levels are usually linked to lower life expectancy.

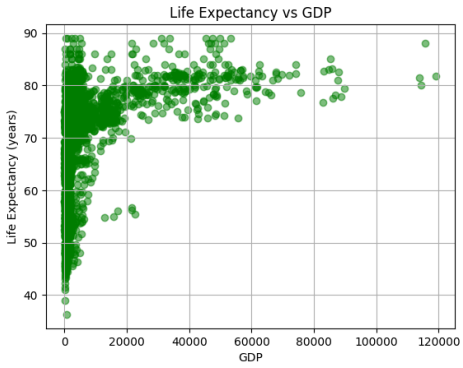
12. Reproducibility Notes

- This notebook was created in Google Colab using Python 3 and the following libraries: pandas, numpy, matplotlib, scikit-learn.
- To reproduce the results, upload the dataset to the Colab environment and run all cells in order.
- All preprocessing, exploration, model training, and evaluation steps are included in this notebook.
- The Random Forest model provides the final recommended predictions due to its higher accuracy and stability.

```
import matplotlib.pyplot as plt
# Plot 1 Average Life Expectancy by Year
if "Year" in df.columns and "Life expectancy" in df.columns:
    avg_by_year = df.groupby("Year")["Life expectancy"].mean()
    avg_by_year.plot(kind="line", marker='o', color='blue')
    plt.title("Average Life Expectancy by Year")
    plt.xlabel("Year")
    plt.ylabel("Life Expectancy (years)")
    plt.grid(True)
    plt.show()
# Plot 2 GDP vs Life Expectancy
if "GDP" in df.columns and "Life expectancy" in df.columns:
    plt.scatter(df["GDP"], df["Life expectancy"], alpha=0.5, color='green')
    plt.title("Life Expectancy vs GDP")
    plt.xlabel("GDP")
    plt.ylabel("Life Expectancy (years)")
    plt.grid(True)
    plt.show()

    print("These visuals help confirm that economic and social factors influence how long people live.")
```





These visuals help confirm that economic and social factors influence how long people live.

```
from sklearn.model_selection import train_test_split
```