UNDERSTANDING & PREDICTING LIFE EXPECTANCY

Scott Schirkofsky
Data Science Practicum II
Regis University

OVERVIEW

The Global Health Observatory indicates that life expectancy globally increased from 66.8 years in 2000 to 73.4 in 2019. Life expectancy is also regarded as the key metric for assessing population's health.

This project seeks to better understand the various factors that affect life expectancy and their relationship with each other and life expectancy itself. The project will also use various machine learning techniques to predict various life expectancy by country and other indicators and provide the metrics needed to evaluate the predictions.

Blog: https://sschirko.edublogs.org/
Github: https://github.com/schirko

UNDERSTANDING & PREDICTING LIFE EXPECTANCY

Life Expectancy	"Life expectancy is a statistical measure of the average time an organism is expected to live, based on the year of its birth, its current age, and other demographic factors including biological sex." - Wikipedia
Factors	HIV, Polio, Socio-economic, Alcohol & Drug Use, Income, Education, BMI, Country GDP, Population etc.

THE DATA

Original datasets from The Global Health Observatory (GHO), the data repository for the World Health Organization (WHO). This data tracks the health status and more than health related 1000 indicators for the 193 countries that are members of the United Nations.

- 2938 rows and 22 columns.
- Collected over 15-year period from 2000 to 2015.
- Sourced from Kaggle.

DATA SCIENCE TASKS

MULTIPLE LINEAR REGRESSION

DECISION TREE

RANDOM FOREST

ANALYZING THE DATA

Data Cleaning	Pandas
Data Exploration & Visualization	Pandas, Numpy, Matplotlib, Dataframe, Seaborn,
Splitting Data/ Applying Models	Sklearn, Linear Reg., Multi Reg., Decision Tree, Random Forest, Polynomial, T-test, Accuracy and Evaluation Tests

Matplotlib, Pandas, Seaborn, Tableau

DATASET

This dataset was obtained from Kaggle challenges via the World Health Organization. It is comprised of data from around the world and includes many indicators for each country during the time frame of 2000-2015. The data essentially represents a times series for the countries and features included.

Feature	Description	Type
country	the country of origin	String
year	year data were collected	Integer
life_expectancy	life expectancy of the country's people in years from 2000-1015	String
adult_mortality	adult mortality rate per 1000 population	Integer
infant_deaths	population's infant deaths per 1000	Integer
alcohol	a country's alcohol consumption rate in liters per capita	Integer
percentage_expenditure	health expenditure as a percentage of Gross Domestic Product	Integer
hepatitis_b	number of 1 year olds with Hepatitis B immunization over all 1 year olds in population	Integer
measles	number of reported Measles cases per 1000 population	Integer
bmi (Interval/Ordinal)	average Body Mass Index (BMI) of a country's total population	Integer
under-five_deaths	number of people under the age of five deaths per 1000 population	Integer
polio	number of 1 year olds with Polio immunization over the number of all 1 year olds in population	Integer
total_expenditure	government expenditure on health as a percentage of total government expenditure	Integer
diphtheria	diphtheria tetanus toxoid and pertussis (DTP3) immunization rate of 1 year olds	Integer
hiv/aids	deaths per 1000 live births caused by HIV/AIDS for people under 5; number of people under 5 who die due to HIV/AIDS per 1000 births	Integer
gdp	Gross Domestic Product per capita	Integer
population	population of a country	Integer
thinness_1-19years	rate of thinness among people aged ***10-19***	Integer
thinness_5-9_years	rate of thinness among people aged 5-9	Integer
income_composition_of_resources	human development Index in terms of income composition of resources (index ranging from 0 to 1)	Integer
schooling	average number of years of schooling of a population	Integer

EDA – RENAMING COLUMNS

Before Renaming Columns

```
dtypes():
 Country                          object
Year                              int64
Status                           object
Life expectancy                 float64
Adult Mortality                   int64
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
dtype: object
```

After Renaming Columns

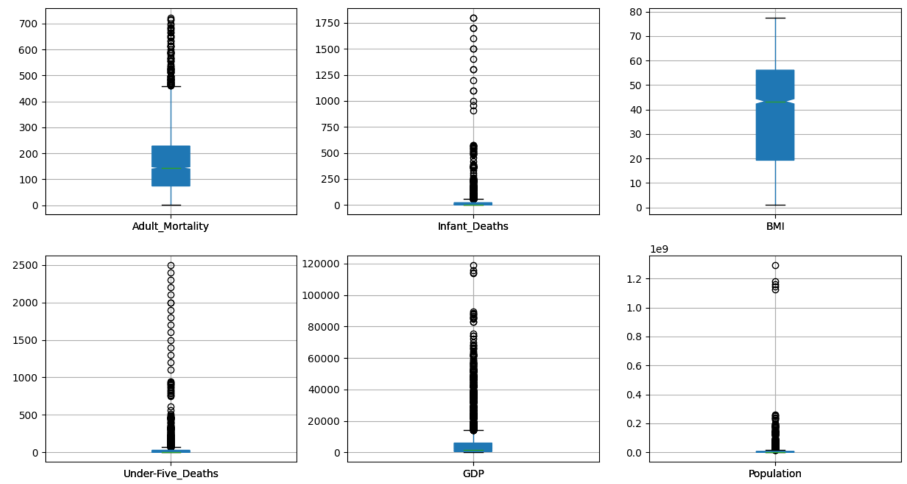
```
Data columns (total 22 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Country                    2928 non-null   object
 1   Year                       2928 non-null   int64
 2   Status                     2928 non-null   object
 3   Life_Expectancy            2928 non-null   float64
 4   Adult_Mortality            2928 non-null   int64
 5   Infant_Deaths              2090 non-null   float64
 6   Alcohol                    2735 non-null   float64
 7   Pct_Expenditure            2928 non-null   float64
 8   Hepatitis_B                2375 non-null   float64
 9   Measles                    2928 non-null   int64
 10  BMI                        2896 non-null   float64
 11  Under-Five_Deaths          2153 non-null   float64
 12  Polio                      2909 non-null   float64
 13  Ttl_Expend                 2702 non-null   float64
 14  Diphtheria                 2909 non-null   float64
 15  HIV_AIDS                   2928 non-null   float64
 16  GDP                        2485 non-null   float64
 17  Population                 2284 non-null   float64
 18  Thinness_10-19_Years       2896 non-null   float64
 19  Thinness_5-9_Years         2896 non-null   float64
 20  Income_Comp_Of_Resources   2768 non-null   float64
 21  Schooling                  2768 non-null   float64
```

EDA – MISSING VALUES (INEXPLICIT NULLS)

describe():

	Life_Expectancy	Adult_Mortality	Infant_Deaths	Alcohol	Pct_Expenditure	Hepatitis_B	Measles	BMI	Under-Five_Deaths
count	2928.000000	2928.000000	2928.000000	2735.000000	2928.000000	2375.000000	2928.000000	2896.000000	2928.000000
mean	69.224932	164.796448	30.407445	4.614856	740.321185	80.960842	2427.855874	38.235394	42.179303
std	9.523867	124.292079	118.114450	4.050749	1990.930605	25.018337	11485.970937	19.959590	160.700547
min	36.300000	1.000000	0.000000	0.010000	0.000000	1.000000	0.000000	1.000000	0.000000
25%	63.100000	74.000000	0.000000	0.905000	4.853964	77.000000	0.000000	19.300000	0.000000
50%	72.100000	144.000000	3.000000	3.770000	65.611455	92.000000	17.000000	43.350000	4.000000
75%	75.700000	228.000000	22.000000	7.715000	442.614322	97.000000	362.250000	56.100000	28.000000
max	89.000000	723.000000	1800.000000	17.870000	19479.911610	99.000000	212183.000000	77.600000	2500.000000

Box Plot Outliers



- Adult mortality has a 1 value for min.
- Infant Deaths has a min of 0 and max of 1800.
- BMI has min of 1.
- Under-Five_Deaths has min of 0.

EDA - MISSING VALUES (EXPLICIT NULLS)

- 14 columns that contain nulls, that's 63.64%!
- Big contributors toward remaining null values in the features BMI, Population, GDP, and Infant Deaths.

```
[iloc = 5] Infant_Deaths has 838 null values: 28.62% null
[iloc = 6] Alcohol has 193 null values: 6.59% null
[iloc = 8] Hepatitis_B has 553 null values: 18.89% null
[iloc = 10] BMI has 1446 null values: 49.39% null
[iloc = 11] Under-Five_Deaths has 775 null values: 26.47% null
[iloc = 12] Polio has 19 null values: 0.65% null
[iloc = 13] Ttl_Expend has 226 null values: 7.72% null
[iloc = 14] Diphtheria has 19 null values: 0.65% null
[iloc = 16] GDP has 443 null values: 15.13% null
[iloc = 17] Population has 644 null values: 21.99% null
[iloc = 18] Thinness_10-19_Years has 32 null values: 1.09% null
[iloc = 19] Thinness_5-9_Years has 32 null values: 1.09% null
[iloc = 20] Income_Comp_Of_Resources has 160 null values: 5.46% null
[iloc = 21] Schooling has 160 null values: 5.46% null
14 out of 22 columns contain null values; 63.64% columns contain null values.
```

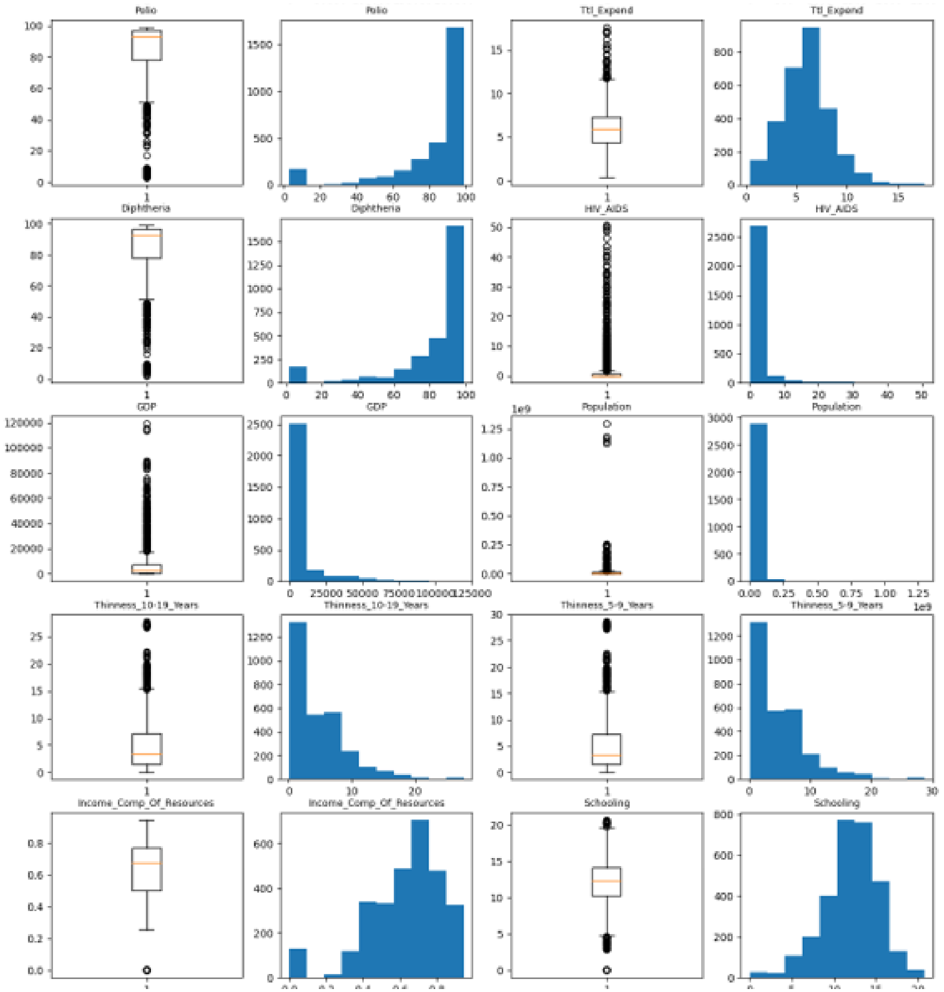
EDA – MISSING VALUES (IMPUTATION)

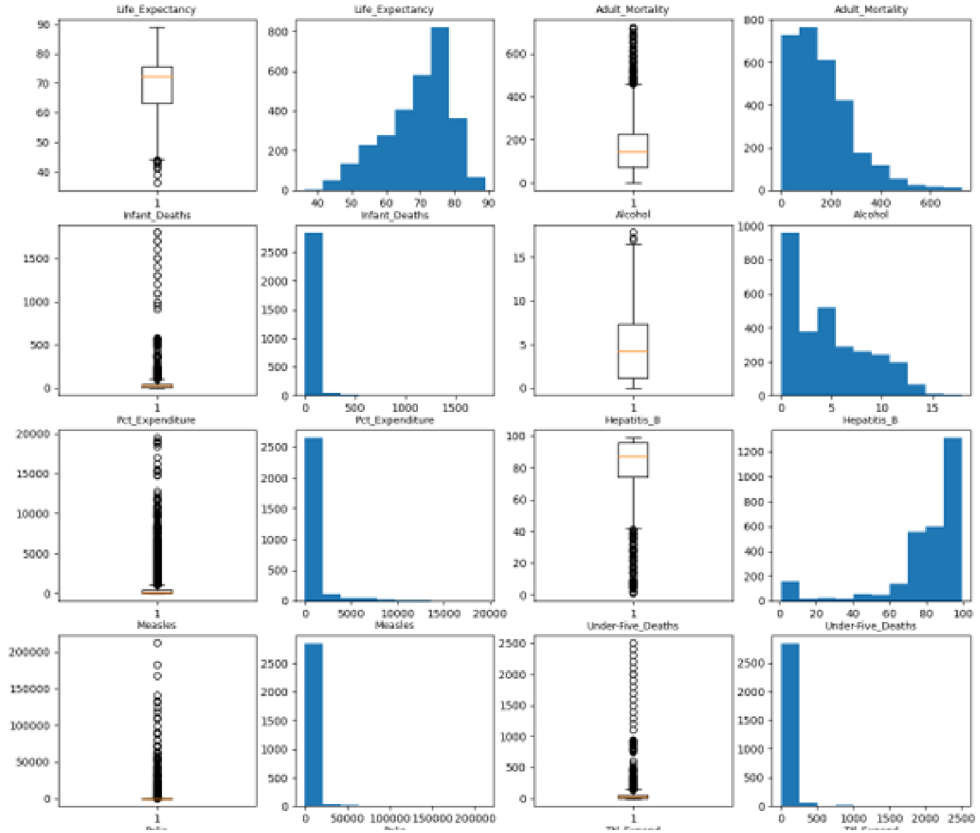
Imputation can be used to estimate new values for the missing values

```
def imputed_data():
    global df
    imputed_data = []
    for year in list(df.Year.unique()):
        year_data = df[df.Year == year].copy()
        for col in list(year_data.columns)[3:]:  # Get numerical features
            year_data[col] = year_data[col].fillna(year_data[col].dropna().mean()).copy()  # Impute the data
        imputed_data.append(year_data)
    df = pd.concat(imputed_data).copy()  # Update global dataframe var
    null_explicit_details()  # View null breakdown after imputing
```

```
0 out of 21 columns contain null values; 0.0% columns contain null values.
```

EDA – OUTLIERS (CORRELATION MATRIX)





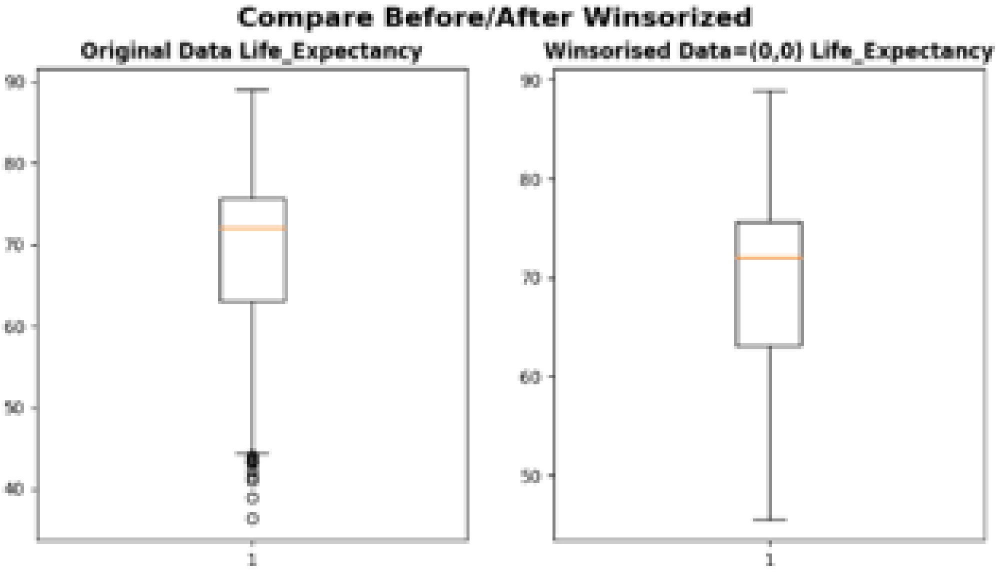
EDA – OUTLIERS (SUMMARY)

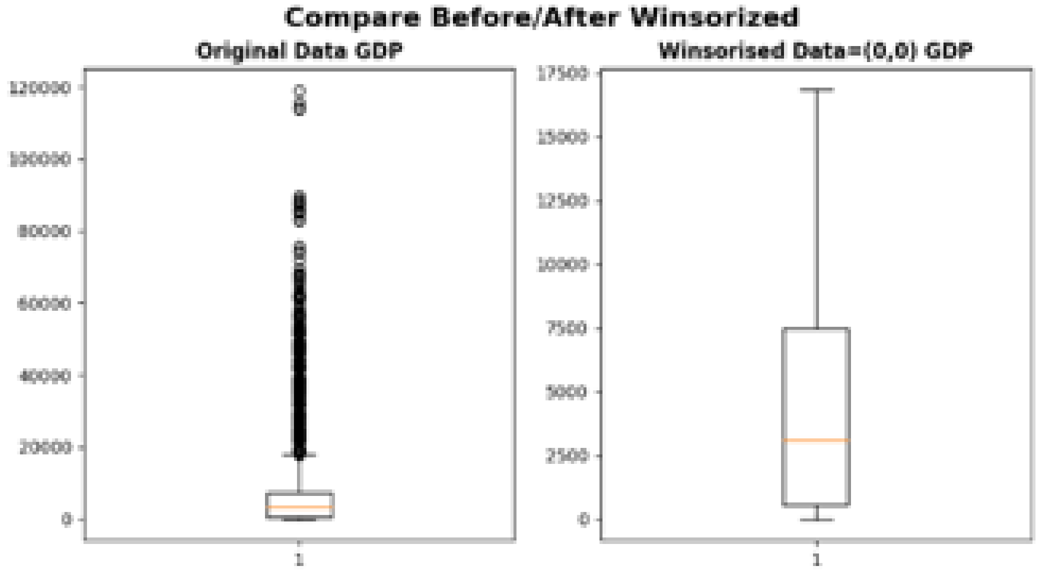
```
-------Life_Expectancy Outliers------------------------
Count: 10
Percentage of Data: 0.34%

-------Adult_Mortality Outliers------------------------
Count: 82
Percentage of Data: 2.8%

-------Infant_Deaths Outliers--------------------------
Count: 135
Percentage of Data: 4.61%

-------Alcohol Outliers--------------------------------
Count: 3
Percentage of Data: 0.1%

-------Pct_Expenditure Outliers------------------------
Count: 388
Percentage of Data: 13.25%

-------Hepatitis_B Outliers----------------------------
Count: 220
Percentage of Data: 7.51%

-------Measles Outliers--------------------------------
Count: 542
Percentage of Data: 18.51%

-------Under-Five_Deaths Outliers----------------------
Count: 142
Percentage of Data: 4.85%

-------Polio Outliers----------------------------------
Count: 278
Percentage of Data: 9.49%
```

```
-------Ttl_Expend Outliers----------------------------
Count: 49
Percentage of Data: 1.67%

-------Diphtheria Outliers----------------------------
Count: 297
Percentage of Data: 10.14%

-------HIV_AIDS Outliers------------------------------
Count: 542
Percentage of Data: 18.51%

-------GDP Outliers-----------------------------------
Count: 300
Percentage of Data: 10.25%

-------Population Outliers----------------------------
Count: 203
Percentage of Data: 6.93%

-------Thinness_10-19_Years Outliers------------------
Count: 100
Percentage of Data: 3.42%

-------Thinness_5-9_Years Outliers--------------------
Count: 99
Percentage of Data: 3.38%

-------Income_Comp_Of_Resources Outliers--------------
Count: 130
Percentage of Data: 4.44%

-------Schooling Outliers-----------------------------
Count: 75
Percentage of Data: 2.56%
```

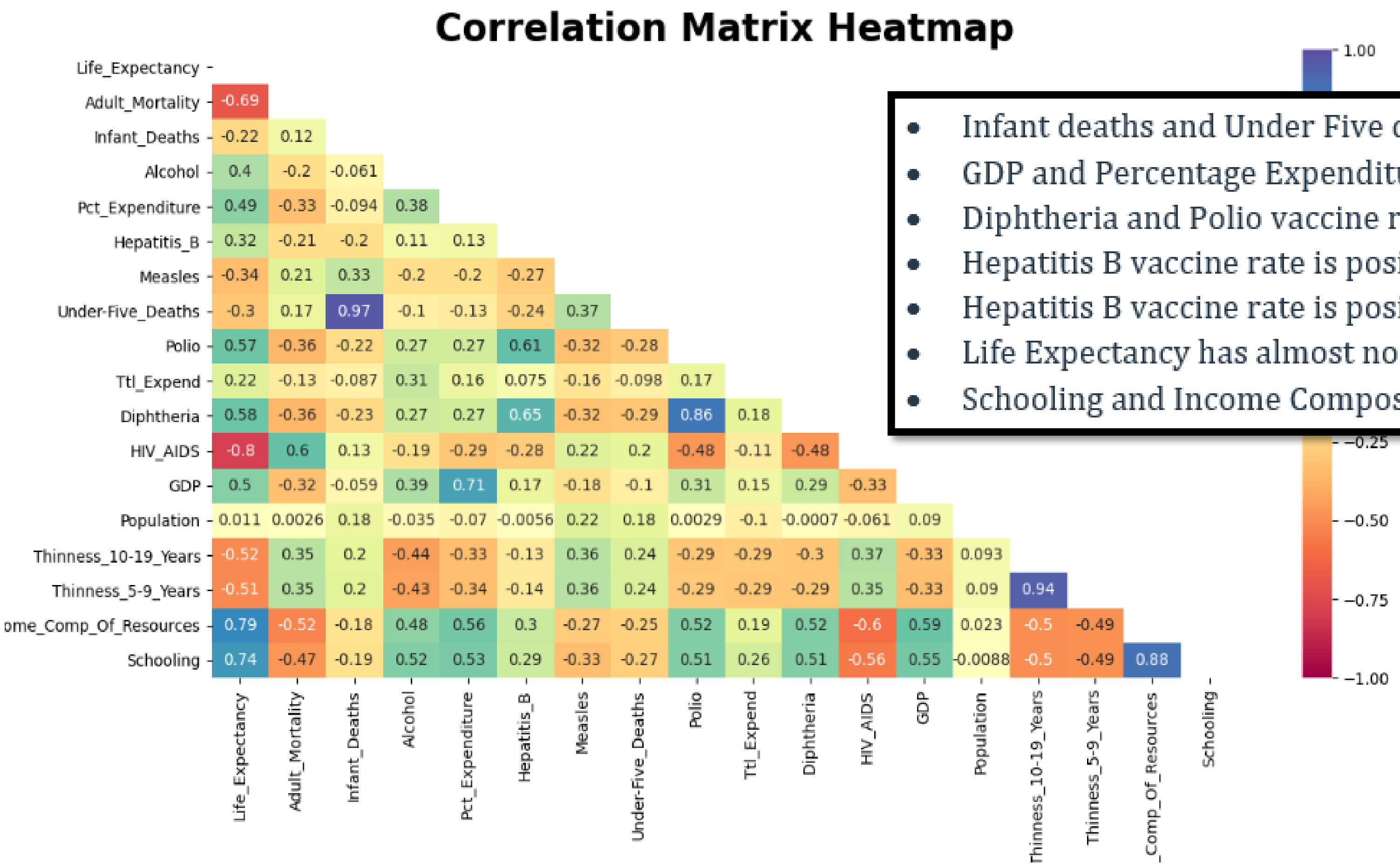
EDA – OUTLIERS (WINSORIZING)

Remove outliers by setting upper and lower limits.



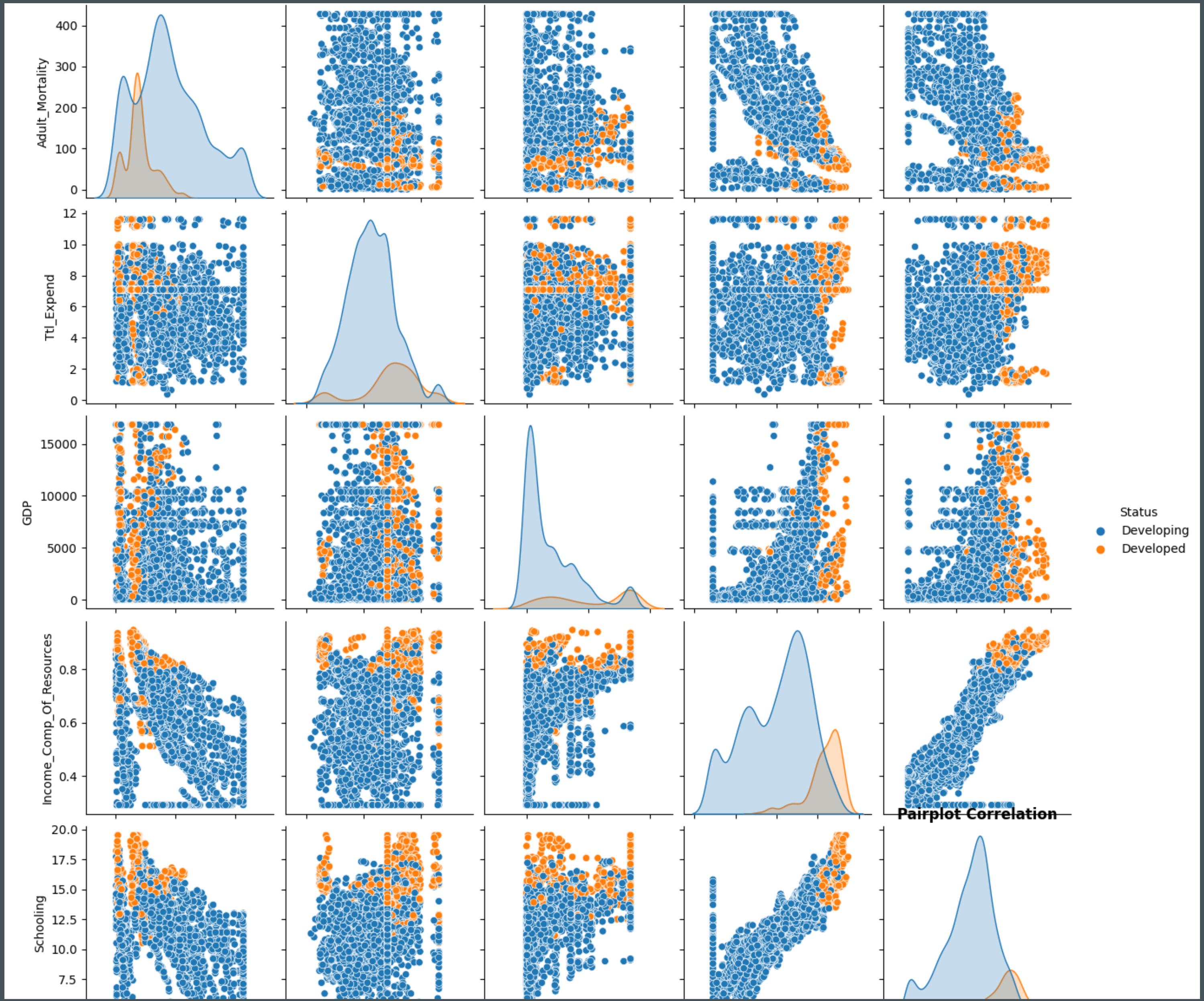


EDA – CORRELATION MATRIX HEATMAP

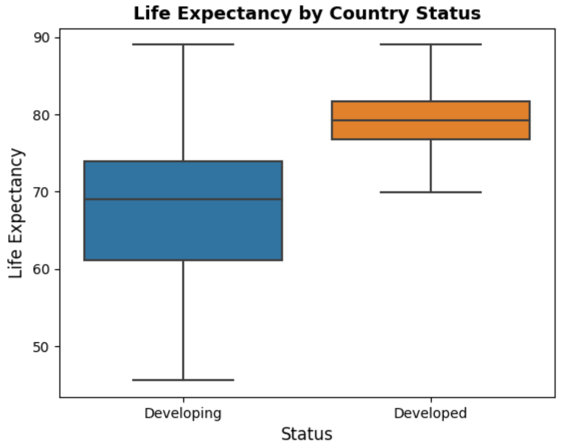


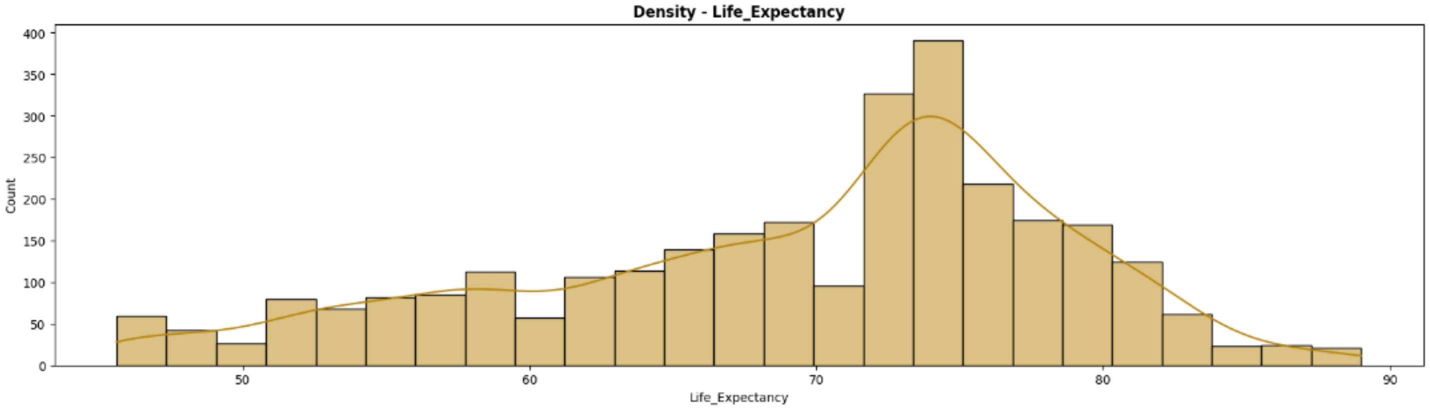
- Infant deaths and Under Five deaths are extremely highly correlated
- GDP and Percentage Expenditure are fairly highly correlated
- Diphtheria and Polio vaccine rate are highly positively correlated
- Hepatitis B vaccine rate is positively correlated with Polio
- Hepatitis B vaccine rate is positively correlated with and Diphtheria vaccine rates
- Life Expectancy has almost no correlation to Population.
- Schooling and Income Composition of Resources are highly correlated

EDA – FEATURE PAIR PLOTS

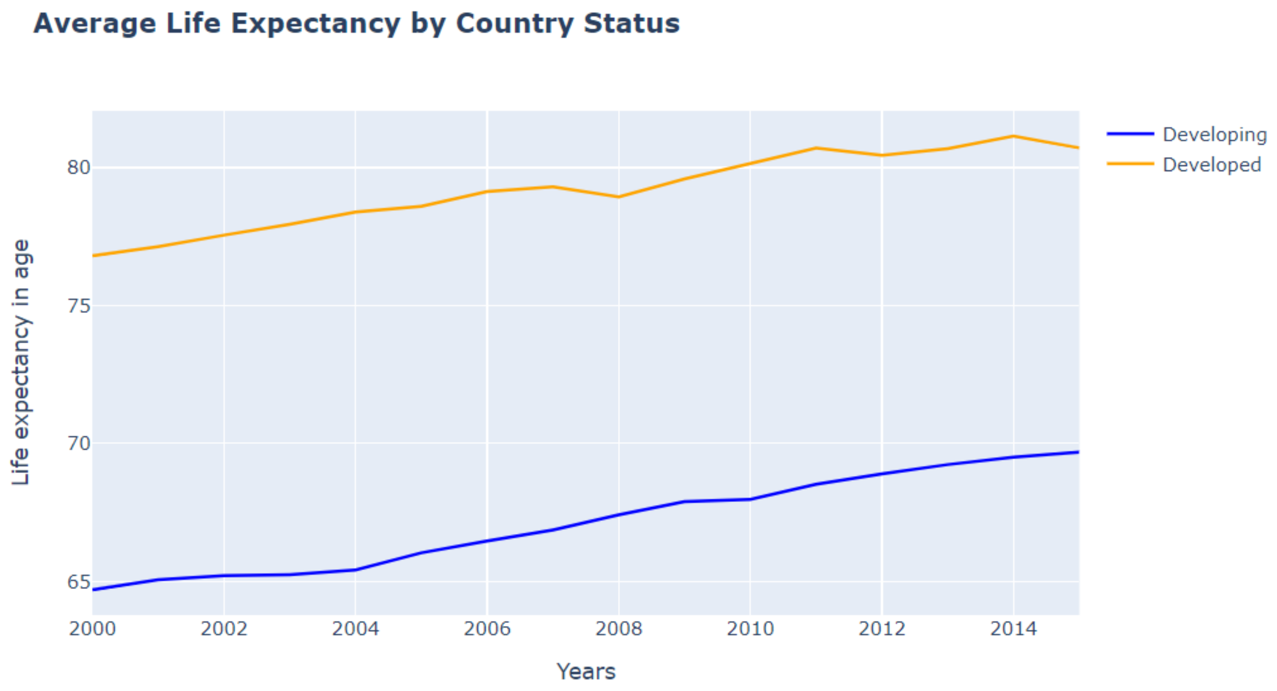


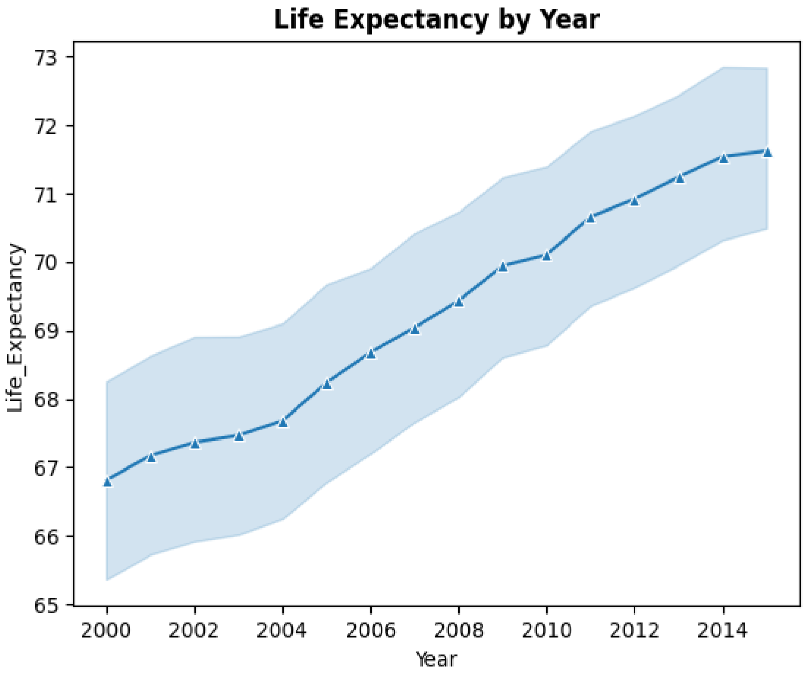
EDA – LIFE EXPECTANCY



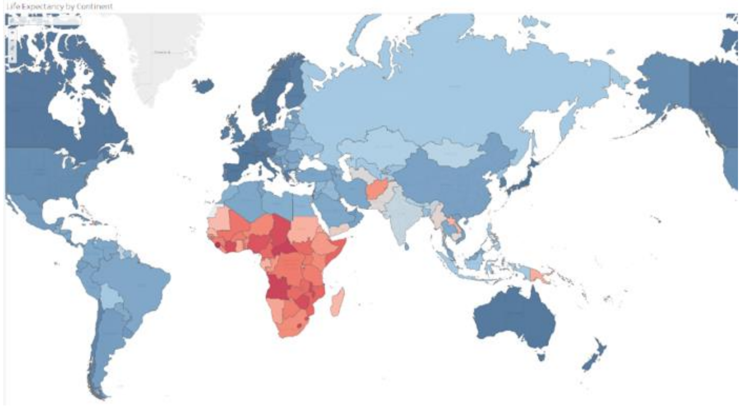


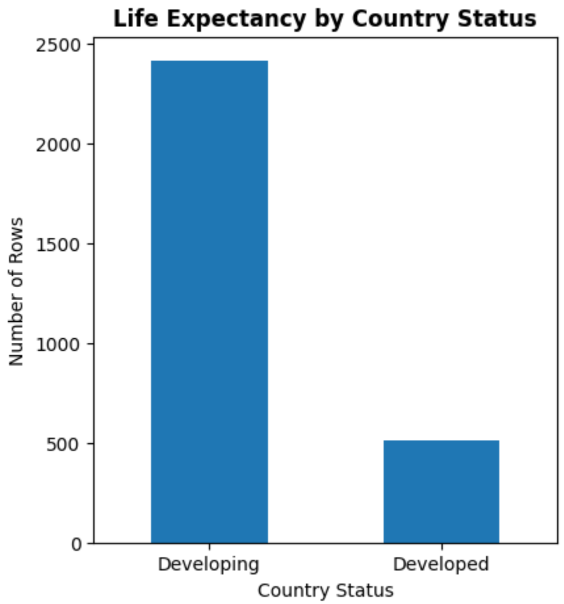
EDA – LIFE EXPECTANCY

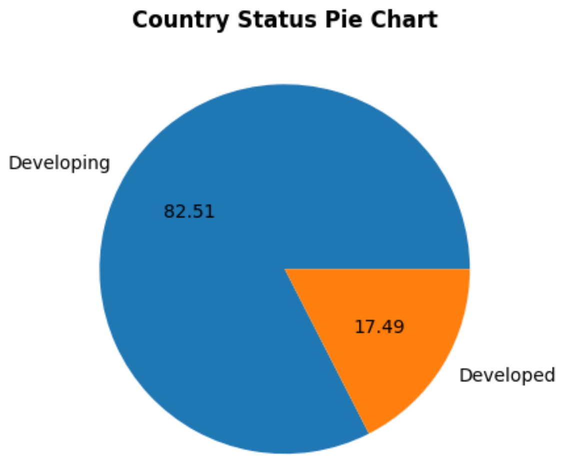




EDA – LIFE EXPECTANCY







Top Life 15 Expectancy by Country (MEAN)

Country	Life_Expectancy
Japan	82.537500
Sweden	82.518750
Iceland	82.443750
Switzerland	82.331250
France	82.218750
Italy	82.187500
Spain	82.068750
Australia	81.812500
Norway	81.793750
Canada	81.687500
Austria	81.481250
Singapore	81.475000
New Zealand	81.337500
Israel	81.300000
Greece	81.218750

Top 15 Life Excpectancy by Country (Median)

Country	Life_Expectancy
Japan	82.550000
Switzerland	82.200000
Iceland	81.950000
Singapore	81.850000
Sweden	81.800000
Australia	81.800000
Spain	81.750000
Italy	81.700000
Israel	81.600000
France	81.600000
Canada	81.350000
Norway	81.200000
Netherlands	81.100000
Austria	81.050000
New Zealand	81.000000

Bottom 15 Life Expectancy by Country (MEAN)

Country	Life_Expectancy
Sierra Leone	47.518750
Central African Republic	48.512500
Lesotho	48.925000
Angola	49.037500
Malawi	50.375000
Côte d'Ivoire	50.387500
Chad	50.387500
Zimbabwe	50.781250
Swaziland	51.325000
Nigeria	51.356250
Somalia	53.318750
Mozambique	53.393750
South Sudan	53.875000
Cameroon	54.018750
Zambia	54.087500

Bottom 15 Life Expectancy by Country (MEDIAN)

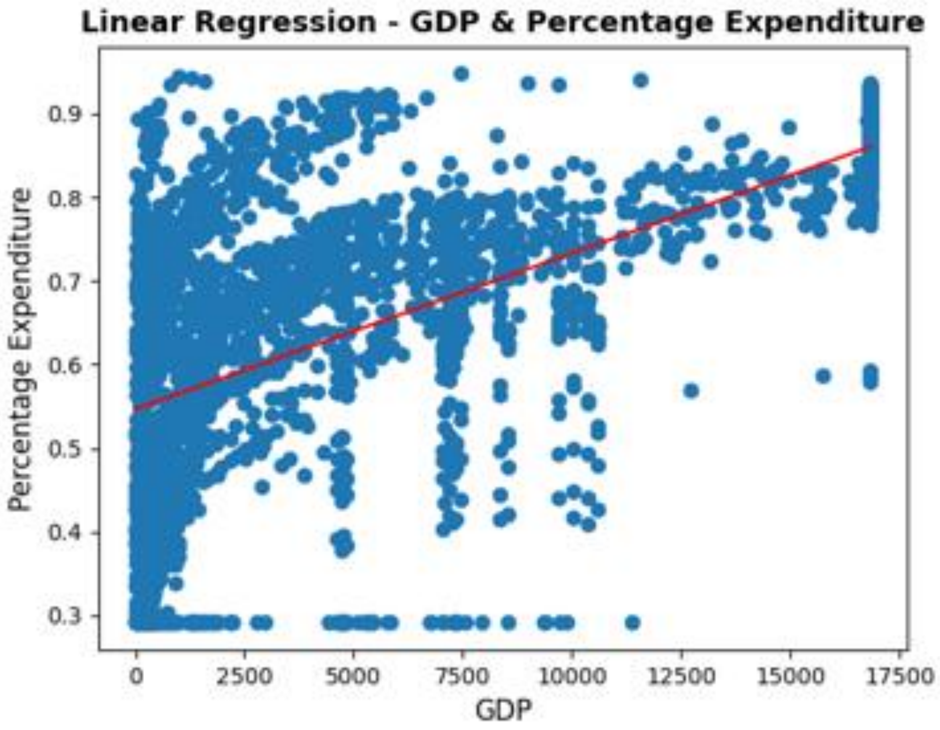
Country	Life_Expectancy
Sierra Leone	46.650000
Central African Republic	47.200000
Zimbabwe	47.400000
Angola	48.450000
Lesotho	48.550000
Malawi	49.250000
Chad	49.500000
Côte d'Ivoire	50.450000
Swaziland	50.700000
Nigeria	51.800000
Somalia	52.750000
Mozambique	53.900000
Cameroon	53.900000
South Sudan	53.950000
Equatorial Guinea	55.200000

EDA – RANKINGS

SCOTT SCHIRKOFSKY | REGIS UNIVERSITY

LINEAR REGRESSION

```
Linear regr.predict: [[1351.02549826]]
Linear lr_coef: [[0.14705833]]
Linear r2 Score: 0.920254296978608
Mean Absolute Error: 50943.94783778521
Mean Squared Error: 3456291221.6604424
Root Mean Squared Error: 58790.23066514063
```



MULTIPLE LINEAR REGRESSION

```
mlr_intercept_: [-2735.63211318]
intercept_coef_: [[1755.9938428   195.23412791]]
Mean squared error regression loss: 1656.8188428933993

MLR r2 Error: 0.09821875898006727
```

```python
def multiple_linear_regression():
    mlr_df = wins_df.dropna().copy()  # Copy winsorized dataframe

    mlr_df.drop(["Country", "Status"], axis=1, inplace=True)  # Drop because the are categorical
    x = mlr_df.iloc[:, [-2, -1]].values  # The dependent variables/features
    print("print X: ", x)
    y = mlr_df['Pct_Expenditure'].values.reshape(-1, 1)  # The independent variable/features

    x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20, random_state=42)  # split data
    regr = LinearRegression()
    model = regr.fit(x_train, y_train)  # Fit the LR model

    print("mlr_intercept_: ", regr.intercept_)
    print("intercept_coef_: ", regr.coef_)

    new_data = [[0.4, 8], [0.5, 10]]  # Random data to test
    new_data = pd.DataFrame(new_data).T
    model.predict(new_data)

    # Mean squared error regression loss
    mserl = np.sqrt(mean_squared_error(y_train, model.predict(x_train)))
    print("Mean squared error regression loss: ", mserl)

    # Train and predict model
    model.score(x_train, y_train)  # Coefficient of determination
    cross_val_score(model, x_train, y_train, cv=10, scoring="r2").mean()  # Evaluate the score by cross-validation
    y_head = model.predict(x_test)
    y_head[0:5]  # Get first 5

    # Calculate r2 score
    r2_degeri = r2_score(y_test, y_head)
    print("\nTest r2 Error = ", r2_degeri)
```

POLYNOMIAL REGRESSION

```
Polynomial r2 Value: 0.6496357950943772
```

```
def polynomial_regression():
    poly_df = wins_df.dropna().copy()  # Copy winsorized dataframe
    regr = LinearRegression()
    x = poly_df.GDP.values.reshape(-1, 1)
    y = poly_df['Pct_Expenditure'].values.reshape(-1, 1)
    regr.fit(x, y)  # Fit linear model first

    x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20, random_state=42)  # Split data

    poly_regr = PolynomialFeatures(degree=15)
    x_polynomial = poly_regr.fit_transform(x)
    regr.fit(x_polynomial, y)  # Fit the polynomial features model
    y_head = regr.predict(x_polynomial)

    poly_features = PolynomialFeatures(degree=8)
    level_poly = poly_features.fit_transform(x_train)
    regr.fit(level_poly, y_train)   # Fit the trained model
    y_head = regr.predict(poly_features.fit_transform(x_train))
    y_test = np.array(range(0, len(y_train)))

    r2 = r2_score(y_train, y_head)
    print("r2 Value: ", r2)  # percentage of significance

    plt.scatter(y_test, y_train, color="blue")
    plt.scatter(y_test, y_head, color="orange")
    plt.xlabel("GDP")
    plt.ylabel("Percentage Expenditure")
    plt.title("Polynomial Regression - Percentage Expenditure", fontweight='bold', fontsize=13)
    plt.savefig('charts/Polynomial_Regression.png', dpi=None, facecolor='w', edgecolor='g', orientation='landscape',
                format=None, transparent=False, bbox_inches=None, pad_inches=0.10, metadata=None)
    plt.show()
```

DECISION TREE REGRESSION

```
Decision Tree Prediction: [98.68367951]
```

```
def decision_tree_regression():
    dtr_df = wins_df.dropna().copy()  # Copy winsorized dataframe

    x = dtr_df.GDP.values.reshape(-1, 1)
    y = dtr_df['Pct_Expenditure'].values.reshape(-1, 1)
    x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20, random_state=42)  # split data

    regr = DecisionTreeRegressor()  # created model
    regr.fit(x_train, y_train)  # fitted model according to train values

    print("Decision Tree Prediction: ", regr.predict([[1000]]))

    x_array = np.arange(min(x), max(x), 0.01).reshape(-1, 1)  # line information to be drawn as a predict
    y_head = regr.predict(x_array)      # percentage of spend estimate
```

RANDOM FOREST REGRESSION

```
Random Forest Prediction: [73.33903837]
```

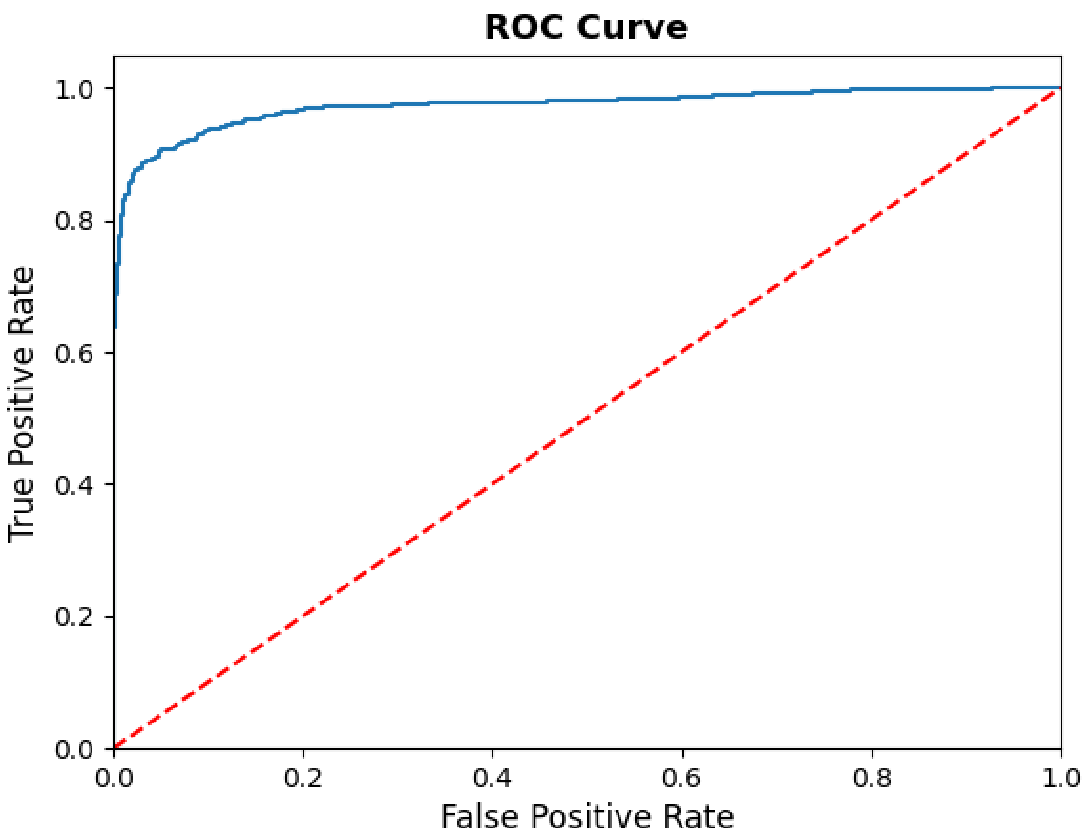
```
def random_forest_regression():
    rfr_df = wins_df.dropna().copy()

    x = rfr_df.GDP.values.reshape(-1, 1)
    y = rfr_df['Pct_Expenditure'].values
    regr = RandomForestRegressor(n_estimators=100, random_state=42)
    regr.fit(x, y)  # The ideal fit line

    print("Random Forest Prediction: ", regr.predict([[1000]]))
    print("\n")

    x_array = np.arange(min(x), max(x), 0.01).reshape(-1, 1)   # line information to be drawn as a predict
    y_head = regr.predict(x_array)  # percentage of spend predict
```

```
Confusion Matrix:
 array([[ 169,  73]
       [  45, 1362]])

Accuracy Score:  .9284414796846574

Classification Report:

              precision    recall  f1-score   support

           0       0.79      0.70      0.74       242
           1       0.95      0.97      0.96      1407

    accuracy                           0.93      1649
   macro avg       0.87      0.83      0.85      1649
weighted avg       0.93      0.93      0.93      1649

Cross Value Score:  0.9330426854925429
```

PREDICTION/ TUNE MODEL

ROC CURVE

Visualize the True False Positives and True False Negatives



SUMMARY

- The dataset began with 21 variables that were processed down to 12 independent variables (features) that describe Life Expectancy, the dependent variable.
- The data was cleaned by first cleaning up the headings, then detecting and dealt with missing values, both inexplicit and explicit. The data was also imputed and winsorized to address various missing values and outliers. From this a model was born.
- A number of machine learning methods were applied to the model including Linear Regression, Multiple Regression, Decision Tree regression, Random Forest Regression, and t-test.
- Accuracy metrics like Confusion Matrix, Accuracy Score, Classification Report, and Cross Value Score were run and the model scored an excellent 93% accuracy.

FUTURE PROJECTS/ QUESTIONS

While this project was more focused on predictions and the exploratory data process, future projects would benefit from what was learned here.

Additional questions to be answered in future projects:

- What is the impact of disease in developed versus developing nations?
- What are the impacts of vaccines on Life Expectancy in developing nations?
- Which immunizations have the highest correlation to greater life expectancy?
- What 5 features would be most impactful on life expectancy in developing countries?

REFERENCES

CODING AND OTHER VALUABLE RESOURCES

I wish I could acknowledge each piece of code I used for inspiration but there are far too many to list. So, I've listed a few of the more useful resources I used below.

Topic	Sources
Regression	• https://scikit-learn.org/stable/auto_examples/linear_model/plot_ols.html#sphx-glr-auto-examples-linear-model-plot-ols-py • https://www.kaggle.com/mathchi/life-expectancy-who-with-several-ml-techniques#Logistic-Regression-Model
Winsorize	• https://www.kaggle.com/philbowman212/life-expectancy-exploratory-data-analysis/
Matpltlib	• https://stackoverflow.com/questions/46664082/python-how-to-save-statsmodels-results-as-image-file
EDA	• https://www.kaggle.com/philbowman212/life-expectancy-exploratory-data-analysis/ • https://towardsdatascience.com/an-extensive-guide-to-exploratory-data-analysis-ddd99a03199e
Training & Testing Model	• https://www.kaggle.com/rishavchowdhury0123/life-expectancy-eda-and-prediction#Training-and-Testing-the-Model • https://www.kaggle.com/mathchi/life-expectancy-who-with-several-ml-techniques#Logistic-Regression-Model
Plotting	• https://towardsdatascience.com/understanding-boxplots-5e2df7bcbd51 • https://stackabuse.com/matplotlib-box-plot-tutorial-and-examples/ • https://thecleverprogrammer.com/2021/01/06/life-expectancy-analysis-with-python/
Predicting	• https://www.datasciencesociety.net/using-machine-learning-to-explain-and-predict-the-life-expectancy-of-different-countries/ • https://www.kaggle.com/mathchi/life-expectancy-who-with-several-ml-techniques#Logistic-Regression-Model