CHRIST
(DEEMED TO BE UNIVERSITY)
PUNE LAVASA CAMPUS
The Hub of Analytics

An Analysis Report on

Life Expectancy Prediction using Linear Regression

Submitted By:

Abhinav Khandelwal

24122002

Introduction:

Life expectancy is a measure that indicates a person's average lifespan. Several factors influence a person's life expectancy, such as whether the person lives in a developed or developing country, their economic condition, the country's population, GDP, and other social and geographical factors.

This study aims to build a regression model to predict the average life expectancy based on economic, social, geographical, and mortality factors.

Dependent And Independent Columns:

In this study the Dependent Column will be Life Expectancy which will be predicted and mainly the independent variable are status of the country like the country is developed or developing, Alcohol, Infant Deaths, Adult Mortality and etc.

Collect and Understand the Dataset:

The Life Expectancy Dataset, which was initially based on information gathered by the World Health Organization (WHO), was obtained from Kaggle and used in this investigation. It includes comprehensive data on mortality rates over a number of years, socioeconomic variables, and health indicators for different nations.

- Features like life expectancy, adult mortality, infant mortality, alcohol use, GDP, education, BMI, vaccination rates, and population are all included in the dataset.
- The inclusion of both developed and developing nations enables comparisons across various developmental stages.

The information is meant to aid in the analysis of the ways in which social, economic, and health-related factors affect the population's average lifespan.

```
df = pd.read_csv(r"C:\Users\Abhinav Khandelwal\Desktop\MS DS\Trimester 3\Regression Modelling\Project Life Expectancy\Life Expectancy Data.csv")
df.head()
```

	Country	Year	Status	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles	BMI	under-five deaths	Polio	Total expenditure	Diphtheria	HIV/AIDS
0	Afghanistan	2015	Developing	65.0	263.0	62	0.01	71.279624	65.0	1154	19.1	83	6.0	8.16	65.0	0.1
1	Afghanistan	2014	Developing	59.9	271.0	64	0.01	73.523582	62.0	492	18.6	86	58.0	8.18	62.0	0.1
2	Afghanistan	2013	Developing	59.9	268.0	66	0.01	73.219243	64.0	430	18.1	89	62.0	8.13	64.0	0.1
3	Afghanistan	2012	Developing	59.5	272.0	69	0.01	78.184215	67.0	2787	17.6	93	67.0	8.52	67.0	0.1
4	Afghanistan	2011	Developing	59.2	275.0	71	0.01	7.097109	68.0	3013	17.2	97	68.0	7.87	68.0	0.1

Dataset Metadeta

```
#Number of Records in the datase
print("Total number of records in the dataset are: ", df.shape[0])
```

```
Total number of records in the dataset are:  2938
```

```
#Number of Variables in the dataset
print("Total number of variable in the dataset are: ", df.shape[1])
```

```
Total number of variable in the dataset are:  22
```

```
#Nature of the variables
print("Nature of the variables are: ")
print("    ")
df.info()
```

```
Nature of the variables are:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2938 entries, 0 to 2937
Data columns (total 22 columns):
 #   Column                           Non-Null Count  Dtype  
---  ------                           --------------  -----  
 0   Country                          2938 non-null   object 
 1   Year                             2938 non-null   int64  
 2   Status                           2938 non-null   object 
 3   Life expectancy                  2928 non-null   float64
 4   Adult Mortality                  2928 non-null   float64
 5   infant deaths                    2938 non-null   int64  
 6   Alcohol                          2744 non-null   float64
 7   percentage expenditure           2938 non-null   float64
 8   Hepatitis B                      2385 non-null   float64
 9   Measles                          2938 non-null   int64  
 10   BMI                             2904 non-null   float64
 11  under-five deaths                2938 non-null   int64  
 12  Polio                            2919 non-null   float64
 13  Total expenditure                2712 non-null   float64
 14  Diphtheria                       2919 non-null   float64
 15   HIV/AIDS                        2938 non-null   float64
 16  GDP                              2490 non-null   float64
 17  Population                       2286 non-null   float64
 18   thinness  1-19 years            2904 non-null   float64
 19   thinness 5-9 years             2904 non-null   float64
 20  Income composition of resources  2771 non-null   float64
 21  Schooling                        2775 non-null   float64
dtypes: float64(16), int64(4), object(2)
memory usage: 505.1+ KB
```

Exploratory Data Analysis and Data Preprocessing:

After reviewing the dataset, we can remove the year and the country column and not contributing to the analysis. Instead of a country we will focus on Status of the country whether it is developed or developing.

```
df.drop(columns={"Country","Year"}, inplace=True)
list(df.columns)
```

```
['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']
```

Encoding Categorical Column:

There is a Categorical feature in the dataset which is status of the country which tells about whether the country is developed or developing. **One hot encoding** technique will be used to encode the categorical column to numerical column for the machine learning to train.

Before Encoding:

```
df["Status"]
```

```
0       Developing
1       Developing
2       Developing
3       Developing
4       Developing
           ...
2933    Developing
2934    Developing
2935    Developing
2936    Developing
2937    Developing
Name: Status, Length: 2938, dtype: object
```

After Encoding:

```
df = pd.get_dummies(df,dtype= int)
df[["Status_Developed","Status_Developing"]]
```

	Status_Developed	Status_Developing
0	0	1
1	0	1
2	0	1
3	0	1
4	0	1
...	...	...
2933	0	1
2934	0	1
2935	0	1
2936	0	1
2937	0	1

Correlation Matrix:

Correlation Matrix is used to plot to check the correlation between the independent variables to check if there is Multicollinearity and then heat map is used to visualize the correlation matrix for clear understanding.

```
#Correlation matrix to check the relationship between the variables
sns.heatmap(df.corr())
```

```
<Axes: >
```

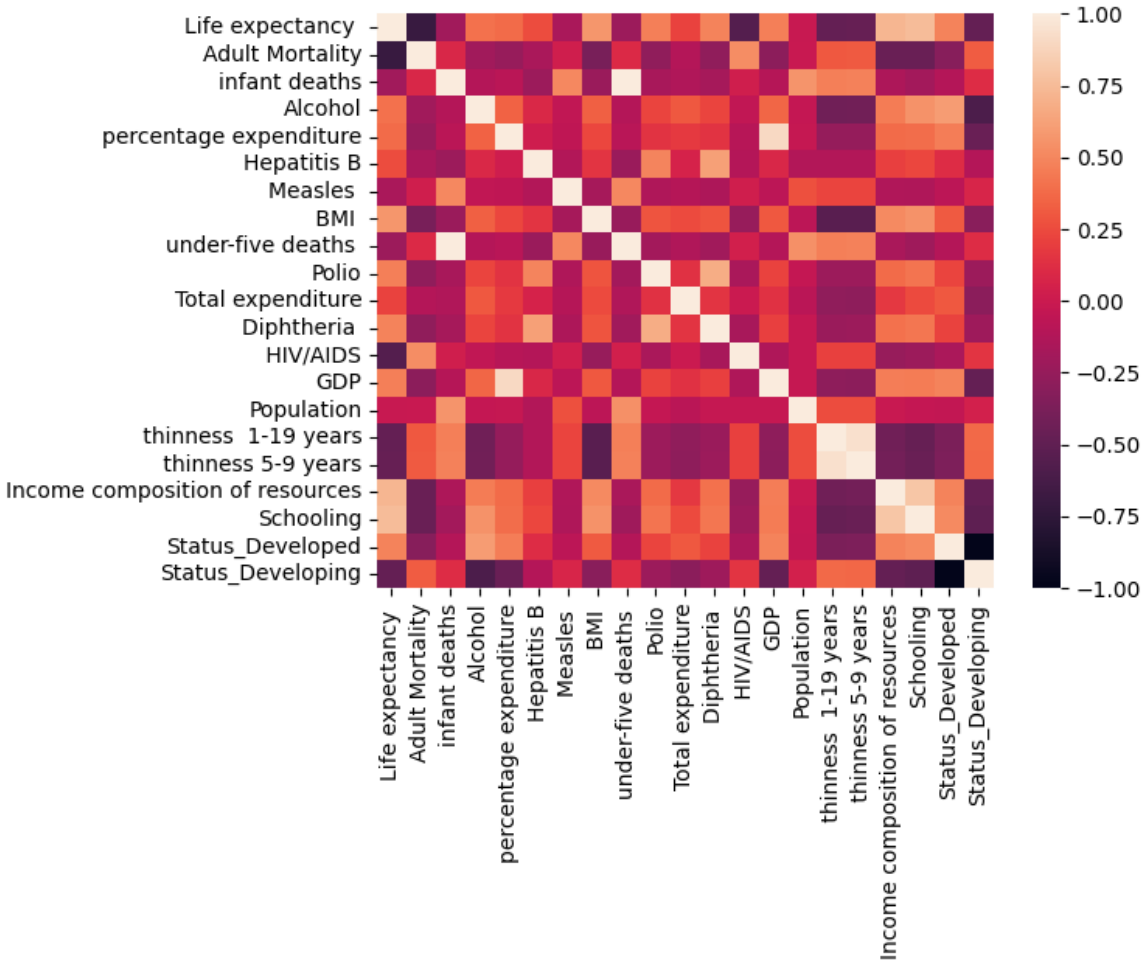

Figure 1: Correlation Matrix

From the above visualization we can conclude that there are some variables (independent variables) which are highly correlated having the correlation greater than 0.5 in both positive and negative direction which shows close to strong correlation this can affect out model.

Distribution of the Dependent Variable:

```
#Distribution of Life Expectancy (Dependent Column)
sns.displot(data = df, x = df.columns[0], kind = "kde")
```

```
<seaborn.axisgrid.FacetGrid at 0x22211938990>
```

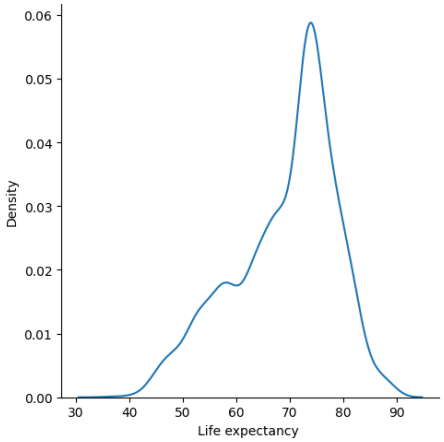

Figure 2 Life Expectancy Distribution

From the distribution we can clearly that the distribution is negatively skewed which shows that the majority of the people has average age in between 62 and 85 but let's see how this distribution is when we check Developing vs Developed Countries.

```
warnings.filterwarnings(action = "ignore")
sns.displot(data = df, x = df.columns[3], kind = "kde", hue = "Status")
```

```
<seaborn.axisgrid.FacetGrid at 0x1f6cea795d0>
```

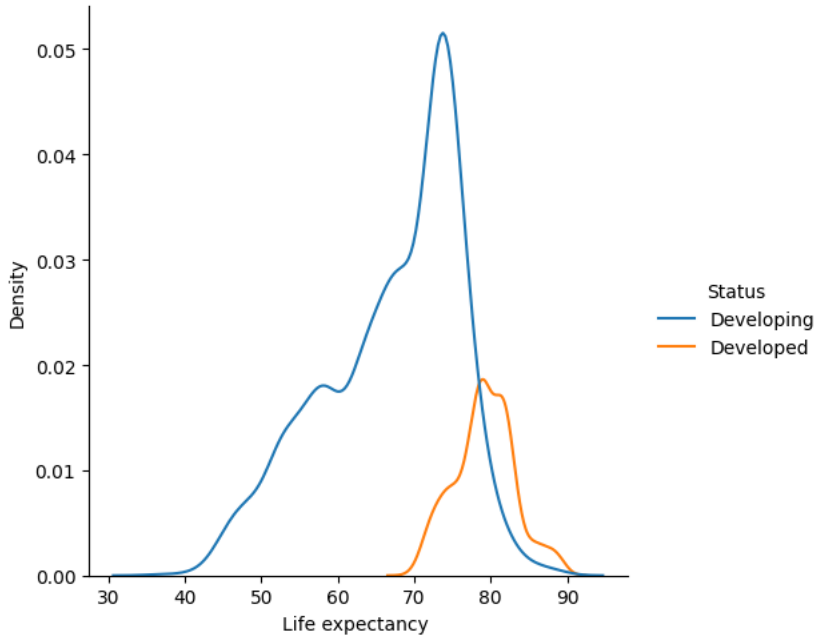

Figure 3 Life Expectancy based on Developed and Developing Countries

From the above distribution we can see that the distribution of Developed Countries are close to normal and they have the average age is between 65 – 90 and majority are in between 78 – 85. While in the case of Developing countries they have the range from 30 – 90 and majority are lying in between 60 – 80. We can say that the average age in the Developed countries are higher and also not highly spread.

Identifying Outlier:

Outlier are the datapoints which are away from the natural behaviour of the data. They affect the model performance and linear regression is highly sensitive to the outliers because of the loss function it is used to train the model. It is important to identify the outliers and handled it.

Box plot is used to identify the outliers.

```
fig, ax = plt.subplots(nrows = 5, ncols=4, figsize = (40,40))
ax = ax.flatten()
plot_ax = 0
ignore_columns = ["Life expectancy ","Status_Developed","Status_Developing"]
for i in df.columns:
    if i in ignore_columns:
        continue
    else:
        sns.boxplot(data = df, x = i, ax = ax[plot_ax])
        ax[plot_ax].tick_params(axis='x', labelsize=20)
        ax[plot_ax].set_title(f'Box plot of {i}', fontsize = 25)
        plot_ax = plot_ax + 1

plt.tight_layout()
plt.savefig('C:\\Users\\Abhinav Khandelwal\\Desktop\\MS DS\\Trimester 3\\Regression Modelling\\Project Life Expectancy\\boxplot_output.png', dpi=300, bb

plt.show()
```

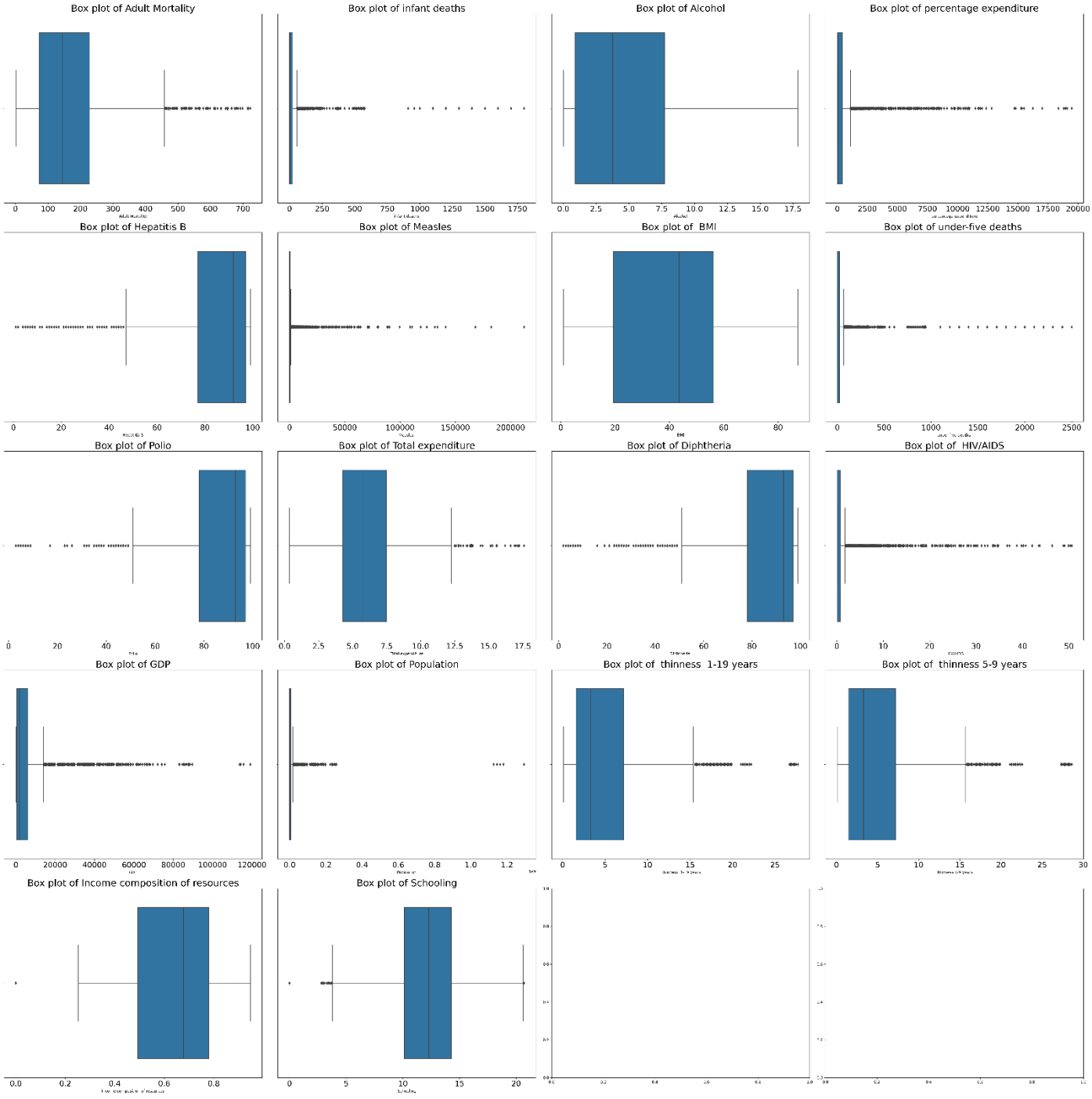

Figure 4 Box Plot of Independent Variables

From the above plots it is very clear that all the independent variables have some outlier's columns like GDP, thinness 5-9 years, HIV/AIDS. Measles, percentage expenditure these variables have a lot of outliers which is not good for the analysis. So these outliers need to be handled there are many techniques but, in this Transformation, will be used because if we remove the outliers then we will end up losing the dataset.

Yeo-Johnson Power Transformation:

Before applying Yeo-Johnson, log transformation, sqrt transformation and other transformation was tried but the performance of the model didn't improve. Yeo-Johnson Transformation gave the good result and it worked with negative values very well.

```
#Applying Variable Transformation to handle the outliers
power_transform = PowerTransformer()
x = df2.drop(columns = "remainder__Life expectancy ")
y = df2["remainder__Life expectancy "]
x = pd.DataFrame(power_transform.fit_transform(x), columns= x.columns)
```

```
fig, ax = plt.subplots(nrows = 5, ncols=4, figsize = (40,40))
ax = ax.flatten()
plot_ax = 0
ignore_columns = ["remainder__Life expectancy ","remainder__Status_Developed","remainder__Status_Developing"]
for i in df2.columns:
    if i in ignore_columns:
        continue
    else:
        sns.boxplot(data = x, x = i, ax = ax[plot_ax])
        ax[plot_ax].tick_params(axis='x', labelsize=20)
        ax[plot_ax].set_title(f'Box plot of {i}', fontsize = 25)
        plot_ax = plot_ax + 1

plt.tight_layout()
plt.savefig('C:\\Users\\Abhinav Khandelwal\\Desktop\\MS DS\\Trimester 3\\Regression Modelling\\Project Life Expectancy\\box_plot_after_transform.png',

plt.show()
```

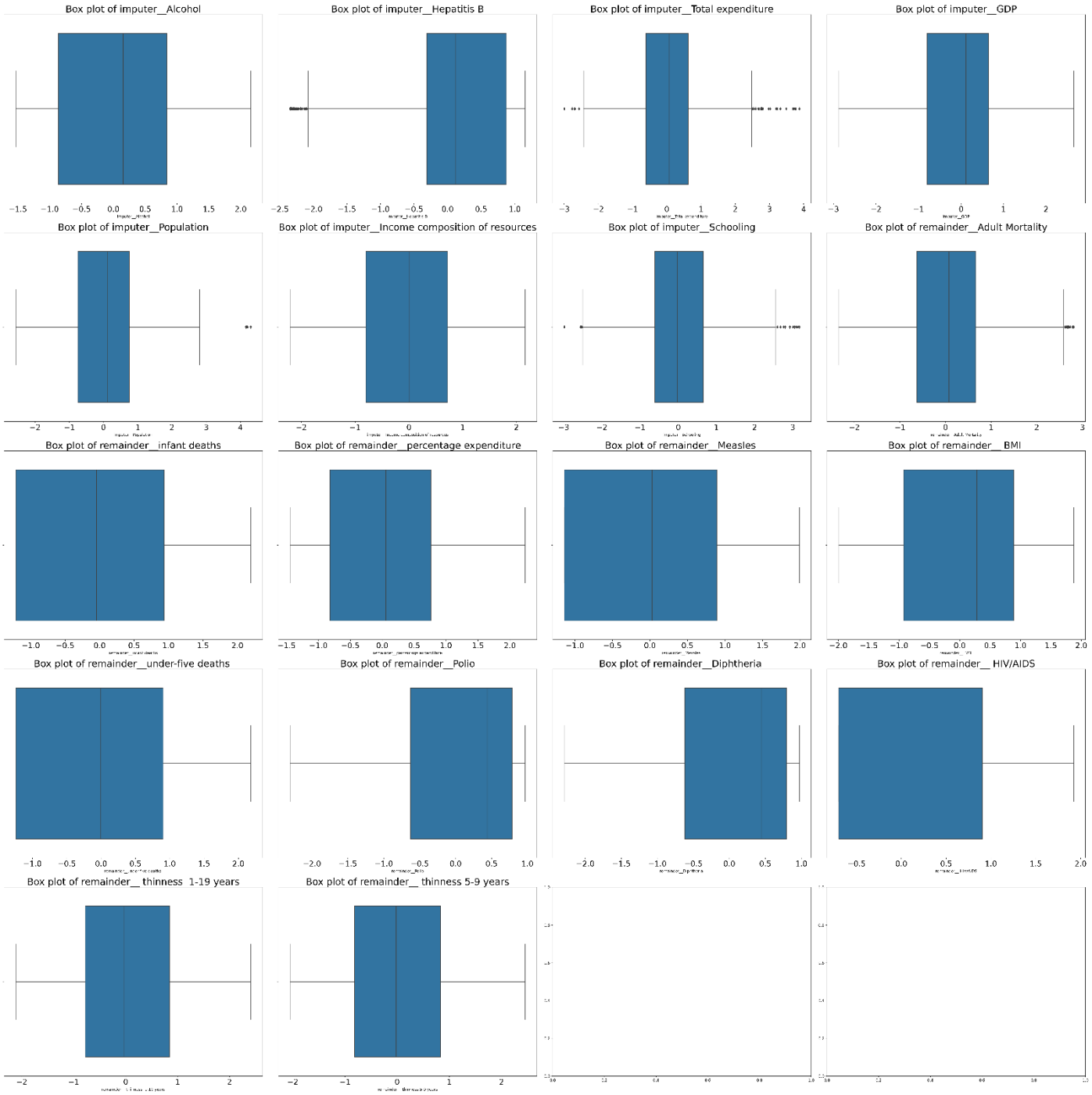

Figure 5 Box Plot after Yeo-Johnson Transformation

After applying the Yeo-Johnson Transformation we can see in the same columns like GDP, thinness 5-9 years, HIV/AIDS. Measles, percentage expenditure the outliers in these columns are handled very well.

Relationship between Independent Variables and Dependent Variable:

```
warnings.filterwarnings(action = "ignore")
fig, ax = plt.subplots(nrows = 5, ncols=4, figsize = (40,40))
ax = ax.flatten()
plot_ax = 0
ignore_columns = ["Life expectancy ","Status_Developed","Status_Developing"]
for i in df.columns:
    if i in ignore_columns:
        continue
    else:
        sns.scatterplot(data = df, x = i, y = "Life expectancy ", ax = ax[plot_ax])
        ax[plot_ax].tick_params(axis='x', labelsize=20)
        ax[plot_ax].set_title(f'Scatter plot of {i} and Life Expectancy', fontsize = 25)

        plot_ax = plot_ax + 1
plt.tight_layout()
plt.savefig('C:\\Users\\Abhinav Khandelwal\\Desktop\\MS DS\\Trimester 3\\Regression Modelling\\Project Life Expectancy\\scatter_output.png', dpi=300, b

plt.show()
```

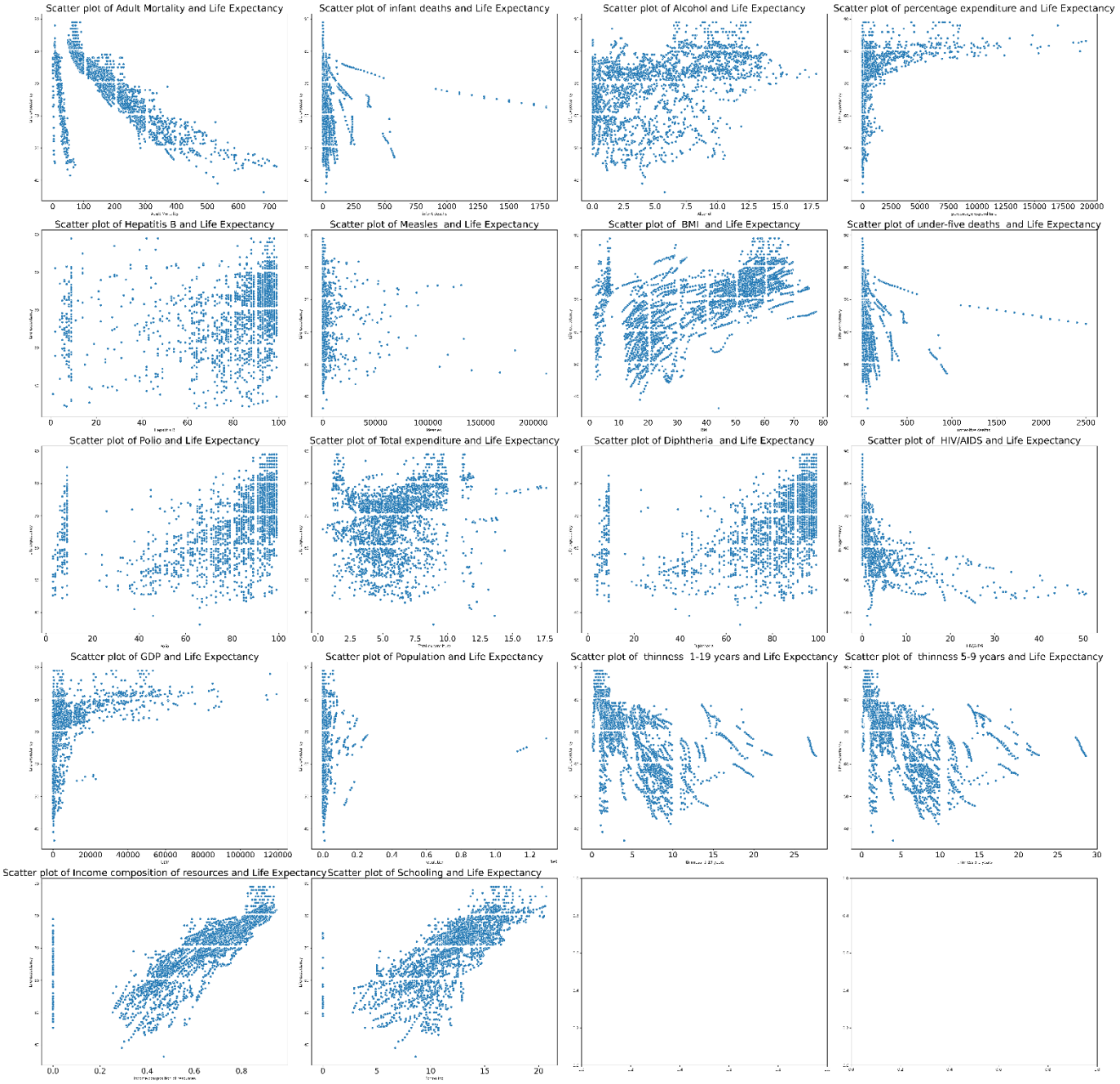

Figure 6 Scatter Plot of the Independent Variables with Dependent Variable

From the scatter plot between independent variable and dependent variable we can observe that not all are having linear relationship with the dependent variable. Some variables like Income composition of resources, Alcohol, Schooling line some sort of linear relationship.

Identifying and Handling Null Values:

```
#Check for null values
df.isnull().sum()/df.shape[0]*100
```

```
Life expectancy                   0.340368
Adult Mortality                   0.340368
infant deaths                     0.000000
Alcohol                           6.603131
percentage expenditure            0.000000
Hepatitis B                      18.822328
Measles                           0.000000
 BMI                              1.157250
under-five deaths                 0.000000
Polio                             0.646698
Total expenditure                 7.692308
Diphtheria                        0.646698
 HIV/AIDS                         0.000000
GDP                              15.248468
Population                       22.191967
 thinness  1-19 years             1.157250
 thinness 5-9 years               1.157250
Income composition of resources   5.684139
Schooling                         5.547992
Status_Developed                  0.000000
Status_Developing                 0.000000
dtype: float64
```

Figure 7 Column with Null Values Percentage

Columns with null values less than 5% the rows will be dropped and other columns null values will be imputed through Simple Imputer.

Dropping the columns having null values less than 5%:

```
#Handling null values
drop_rows = df.columns[((df.isnull().sum()/df.shape[0] * 100 > 0) & (df.isnull().sum()/df.shape[0] * 100 <= 5)).values]
for i in drop_rows:
    df.dropna(subset = i,inplace = True)
```

Imputing columns having null values greater than 5%:

```
impute_col = df.columns[(df.isnull().sum()/df.shape[0] * 100 > 5).values]
```

```
imputing = [
    ("imputer", SimpleImputer(), impute_col)
]
imputer_transf = ColumnTransformer(imputing, remainder="passthrough")
df2 = pd.DataFrame(imputer_transf.fit_transform(df), columns= imputer_transf.get_feature_names_out())
```

```
df2.isnull().sum()/df.shape[0] * 100
```

```
imputer__Alcohol                          0.0
imputer__Hepatitis B                      0.0
imputer__Total expenditure                0.0
imputer__GDP                              0.0
imputer__Population                       0.0
imputer__Income composition of resources  0.0
imputer__Schooling                        0.0
remainder__Life expectancy                0.0
remainder__Adult Mortality                0.0
remainder__infant deaths                  0.0
remainder__percentage expenditure         0.0
remainder__Measles                        0.0
remainder__ BMI                           0.0
remainder__under-five deaths              0.0
remainder__Polio                          0.0
remainder__Diphtheria                     0.0
remainder__ HIV/AIDS                      0.0
remainder__ thinness  1-19 years          0.0
remainder__ thinness 5-9 years            0.0
remainder__Status_Developed               0.0
remainder__Status_Developing              0.0
dtype: float64
```

All the null values have been handled as the percentage of the null values are 0.

Model Building:

Feature Selection:

Instead of working on all the feature, feature selection technique will be applied to identify 10 important features in the dataset. In feature selection technique SelectKBest is used with f_regression score. F_regression can handle the negative values efficiently as compared to chisq test.

```
fea = SelectKBest(score_func=f_regression)
```

```
fea.fit(x,y)
```

```
SelectKBest(score_func=<function f_regression at 0x000001F6C47C59E0>)
```

As per the test below are the important features for our analysis.

```
list(fea.get_feature_names_out())
```

```
['imputer__Income composition of resources',
 'imputer__Schooling',
 'remainder__Adult Mortality',
 'remainder__infant deaths',
 'remainder__ BMI ',
 'remainder__under-five deaths ',
 'remainder__Polio',
 'remainder__Diphtheria ',
 'remainder__ HIV/AIDS',
 'remainder__ thinness 5-9 years']
```

Splitting the dataset into training and testing:

```
#Splitting the dataset into training and testting
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2, random_state=42)
```

Creating the model:

```
x_train = sm.add_constant(x_train)
model = sm.OLS(y_train,x_train).fit()
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:     remainder__Life expectancy   R-squared:                       0.834
Model:                            OLS   Adj. R-squared:                  0.834
Method:                 Least Squares   F-statistic:                     1159.
Date:                Wed, 30 Apr 2025   Prob (F-statistic):               0.00
Time:                        06:55:01   Log-Likelihood:                -6398.3
No. Observations:                2310   AIC:                         1.282e+04
Df Residuals:                    2299   BIC:                         1.288e+04
Df Model:                          10
Covariance Type:            nonrobust
================================================================================================================
                                       coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------------------------------------
const                                69.3696      0.081    861.208      0.000      69.212      69.528
imputer__Income composition of resources     2.5382      0.163     15.528      0.000       2.218       2.859
imputer__Schooling                    0.3620      0.165      2.191      0.029       0.038       0.686
remainder__Adult Mortality           -1.3021      0.097    -13.373      0.000      -1.493      -1.111
remainder__infant deaths              2.6460      0.792      3.341      0.001       1.093       4.199
remainder__ BMI                      -0.0722      0.106     -0.683      0.495      -0.279       0.135
remainder__under-five deaths         -3.6682      0.807     -4.548      0.000      -5.250      -2.086
remainder__Polio                      0.2128      0.159      1.342      0.180      -0.098       0.524
remainder__Diphtheria                 0.9368      0.157      5.977      0.000       0.629       1.244
remainder__ HIV/AIDS                 -3.8478      0.116    -33.116      0.000      -4.076      -3.620
remainder__ thinness 5-9 years       -0.9864      0.107     -9.254      0.000      -1.195      -0.777
==============================================================================
Omnibus:                       48.979   Durbin-Watson:                   1.954
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               99.956
Skew:                          -0.075   Prob(JB):                     1.97e-22
Kurtosis:                       4.008   Cond. No.                         32.6
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Figure 8 Trained Model Parameters

From the OLS Table we can observe that R2 value is 0.834 which means that the independent variables in the model is able to capture 83% of the variance of the dependent variable and Adjusted R2 is also 0.834 which indicated that R2 is not inflated because of increase of the variables.

Interpretation of p-values:

p-value less than 0.05 means we have enough evidence to reject the null hypothesis and accept the alternate hypothesis. Below are the columns which has p-value <0.05 and are significant for the analysis.

```
#Columns whose p-value is less than 0.05
significant_col = x.columns[(model.pvalues < 0.05).values[1:]]
```

```
list(significant_col)
```

```
['imputer__Income composition of resources',
 'imputer__Schooling',
 'remainder__Adult Mortality',
 'remainder__infant deaths',
 'remainder__under-five deaths ',
 'remainder__Diphtheria ',
 'remainder__ HIV/AIDS',
 'remainder__ thinness 5-9 years']
```

Figure 9 Significant Columns as per p-value

Below are the columns which are not significant has p-value greater than 0.05:

```
insignificant_col = x.columns[(model.pvalues > 0.05).values[1:]]
list(insignificant_col)
```

```
['remainder__ BMI ', 'remainder__ Polio']
```

Interpretation of Coefficients:

Variables like **Income composition of resources, Adult Mortality, infant deaths, under-five deaths, HIV/AIDS** these variables have high impact on the dependent variables as their coefficient values are higher between the range of 1 and 3 as we have transformed the data so we can't interpret the exact impact. Other variables which are significant has less impact on the Life Expectancy.

Conclusion and Recommendations:

Predicting life expectancy is a sensitive and complex task influenced by multiple socio-economic and health-related factors. This project explored various features, including adult mortality, infant deaths, alcohol consumption, GDP, education, BMI, vaccination rates, and population metrics.

To ensure data quality, outliers in the independent variables were treated using the Yeo-Johnson transformation, and missing values were addressed using a simple imputation technique. After preprocessing, feature selection techniques were employed to identify the top 10 most influential features for predicting life expectancy. The final model was trained on these selected features.

The analysis revealed that the most significant contributors to life expectancy include:

- Income composition of resources
- Schooling
- Adult mortality
- Infant deaths
- Under-five deaths
- Diphtheria immunization rate
- HIV/AIDS prevalence
- Thinness among children aged 5–9

These findings can help policymakers and health organizations focus on the most impactful areas to improve life expectancy in various populations.

The model performance can potentially be improved with access to a larger dataset, as the current model was trained on approximately 2,800 data points, which limited its predictive power and resulted in a modest R^2 score. In future data collection efforts, focusing on the most significant features identified in this analysis—rather than gathering a broad range of variables—could enhance model accuracy and efficiency.