Data Preparation/Feature Engineering

1. Overview

Data preparation and feature engineering are critical phases in a machine learning project. These phases involve transforming raw data into a format suitable for modeling. Proper data preparation ensures that the machine learning algorithms can effectively learn patterns and make accurate predictions.

2. Data Collection

The dataset used in this project is sourced from the World Health Organization (WHO), containing country-level statistics on various factors that influence life expectancy. The data includes variables such as healthcare expenditure, GDP per capita, immunization coverage, prevalence of diseases, and environmental factors. During data collection, preprocessing steps included downloading the dataset in CSV format and initial checks for data integrity.

3. Data Cleaning

- **Handling Missing Values:** Missing values were identified and handled appropriately. For numerical features, missing values were imputed using the mean or median of the respective column. Categorical features were imputed with the mode.
- **Outliers:** Outliers were identified using statistical methods such as z-score or IQR (Interquartile Range). Outliers were either corrected if data errors were confirmed, or winsorized (replaced with the nearest non-outlier value) to prevent them from skewing the model.
- **Data Quality Issues:** Checks were performed for data consistency, ensuring that values fell within expected ranges and formats.

4. Exploratory Data Analysis (EDA)

- **Univariate Analysis:** Histograms and boxplots were used to visualize distributions and detect outliers.
- **Bivariate Analysis:** Scatter plots and correlation matrices were generated to identify relationships between features and the target variable (life expectancy).
- **Key Insights:** Significant correlations were observed between life expectancy and factors such as healthcare expenditure, sanitation access, and education levels. Countries with higher GDP per capita generally exhibited higher life expectancy.

5. Feature Engineering

- **Creation of New Features:** Features such as GDP per capita multiplied by healthcare expenditure to reflect the economic impact on health outcomes.

- **Transformation of Features:** Log transformations were applied to skewed variables like healthcare expenditure to improve normality.

- **Selection of Relevant Features:** Features were selected based on correlation analysis and domain knowledge, prioritizing those most likely to impact life expectancy.

6. Data Transformation

DataFrame Shape

In [4]:

```
#print number of rows and columns in the dataset

print("Number of Rows:",df.shape[0])
print("Number of Features:",df.shape[1])
```

```
Number of Rows: 2938
Number of Features: 22
```

Handling Outliers

First I will draw boxplot to check outliers

In [14]:

```
# Loop through each column and create a box plot
for column in df.columns:
    fig = px.box(df, y=column, title=f'Box Plot for {column}')

    # Update layout to center the title and make it bold
    fig.update_layout(
        title=dict(text=f'<b>Box Plot for {column}</b>', x=0.5),
        boxmode='group'
    )

    fig.show()
```

The transformation starts by visualizing the dataframe shape, finding outlier and normalizing the data then encoding categorical variables such as country names. These steps ensure that the dataset is ready for model training, optimizing the performance and interpretability of the machine learning models used to predict life expectancy.

Model Exploration

1. Model Selection

For this project on predicting life expectancy using machine learning, an Artificial Neural Network (ANN) is chosen as the primary model. ANNs are well-suited for handling complex relationships in data and can capture non-linear patterns effectively, which is beneficial given the diverse set of factors influencing life expectancy.

Strengths:

- **Non-linear relationships:** ANNs can model complex non-linear relationships between input features and the target variable.
- **Feature learning:** They can automatically learn relevant features from the data, reducing the need for extensive feature engineering.
- **Scalability:** ANNs can handle large datasets with many features.
- **Versatility:** Suitable for both regression and classification tasks.

Weaknesses:

- **Computational complexity:** Training ANNs can be computationally expensive, especially with large datasets and complex architectures.
- **Black-box nature:** Interpretability of results can be challenging compared to simpler models like linear regression.
- **Requires large amounts of data:** ANNs generally require a large amount of data to generalize well and avoid overfitting.

2. Model Training

The ANN was implemented using TensorFlow and Keras in Python. Key aspects of model training include:

- **Architecture:** A feedforward neural network with multiple hidden layers was used.
- **Activation function:** ReLU (Rectified Linear Unit) activation for hidden layers and linear activation for the output layer (since it's a regression task).
- **Loss function:** Mean Squared Error (MSE) to measure the difference between predicted and actual life expectancy values.
- **Optimizer:** Adam optimizer for efficient gradient descent.
- **Hyperparameters:** Tuned parameters such as number of hidden layers, neurons per layer, and learning rate.

Cross-validation: K-fold cross-validation (typically 5 or 10 folds) was employed to assess the model's performance robustly and prevent overfitting.

3. Model Evaluation

Evaluation Metrics:

- **Mean Squared Error (MSE):** Measures the average squared difference between predicted and actual life expectancy values.
- **R-squared (R2) Score:** Indicates the proportion of the variance in the dependent variable (life expectancy) that is predictable from the independent variables.
- **Visualizations:** Scatter plots of predicted vs. actual values, residual plots, and possibly learning curves to assess model performance and convergence.

4. Code Implementation

In [2]:

```
df=pd.read_csv('/kaggle/input/life-expectancy-who/Life Expectancy Data.csv')
df
```

Out[2]:

	Country	Year	Status	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles	...
0	Afghanistan	2015	Developing	65.0	263.0	62	0.01	71.279624	65.0	1154	...
1	Afghanistan	2014	Developing	59.9	271.0	64	0.01	73.523582	62.0	492	...
2	Afghanistan	2013	Developing	59.9	268.0	66	0.01	73.219243	64.0	430	...
3	Afghanistan	2012	Developing	59.5	272.0	69	0.01	78.184215	67.0	2787	...
4	Afghanistan	2011	Developing	59.2	275.0	71	0.01	7.097109	68.0	3013	...
...	...	...	...	...	...	...	...	...	...	...	...
2933	Zimbabwe	2004	Developing	44.3	723.0	27	4.36	0.000000	68.0	31	...
2934	Zimbabwe	2003	Developing	44.5	715.0	26	4.06	0.000000	7.0	998	...
2935	Zimbabwe	2002	Developing	44.8	73.0	25	4.43	0.000000	73.0	304	...
2936	Zimbabwe	2001	Developing	45.3	686.0	25	1.72	0.000000	76.0	529	...
2937	Zimbabwe	2000	Developing	46.0	665.0	24	1.68	0.000000	79.0	1483	...

Feature	Description
Country	countries has been collected from the same WHO data repository website
Year	year 2013-2000
Status	Status of country **Developing** or **Developed**
Life expectancy	Life Expectancy in age **our target**
Adult Mortality	Adult Mortality Rates of both sexes (probability of dying between 15 and 60 years per 1000 population)
infant deaths	Number of Infant Deaths per 1000 population
Alcohol	Alcohol, recorded per capita (15+) consumption (in litres of pure alcohol)
percentage expenditure	Expenditure on health as a percentage of Gross Domestic Product per capita(%)
Hepatitis B	Hepatitis B (HepB) immunization coverage among 1-year-olds (%)
Measles	Measles - number of reported cases per 1000 population
BMI	Average Body Mass Index of entire population
under-five deaths	Number of under-five deaths per 1000 population
Polio	Polio (Pol3) immunization coverage among 1-year-olds (%)
Total expenditure	General government expenditure on health as a percentage of total government expenditure (%)
Diphtheria	Diphtheria tetanus toxoid and pertussis (DTP3) immunization coverage among 1-year-olds (%)
HIV/AIDS	Deaths per 1 000 live births HIV/AIDS (0-4 years)
GDP	Gross Domestic Product per capita (in USD)
Population	Population of the country

Exploring Categorical Features

'Country' Feature

In [9]:

```
df['Country'].value_counts()
```

Out[9]:

```
Country
Afghanistan             16
Peru                    16
Nicaragua               16
Niger                   16
Nigeria                 16
                        ..
Niue                     1
San Marino               1
Nauru                    1
Saint Kitts and Nevis    1
Dominica                 1
Name: count, Length: 193, dtype: int64
```

'Status' Feature

In [10]:

```
df['Status'].value_counts()
```

Out[10]:

```
Status
Developing    2426
Developed      512
Name: count, dtype: int64
```

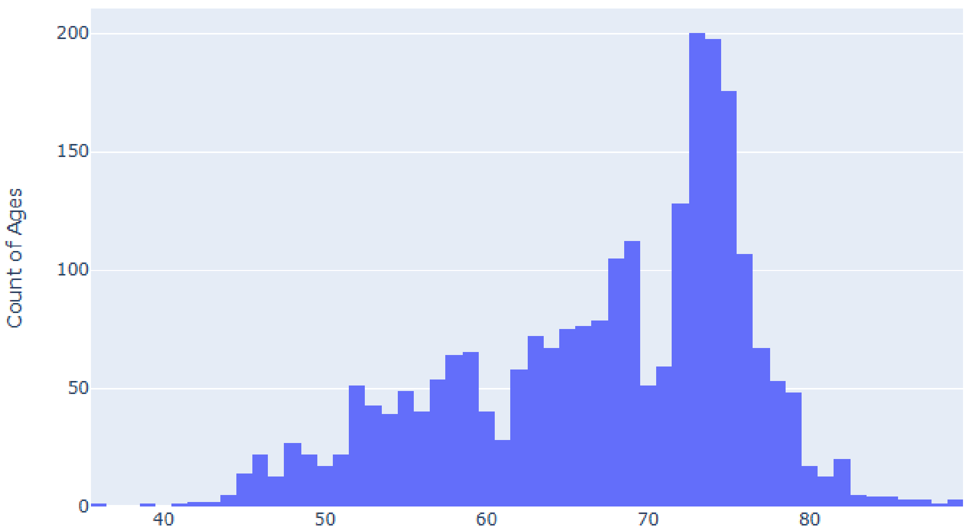
Developing

In [20]:

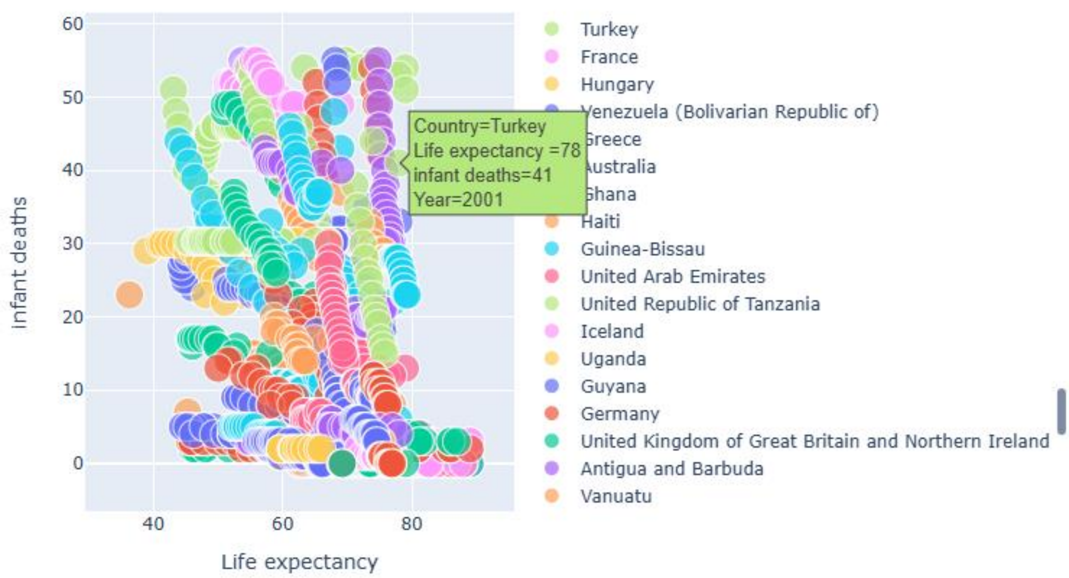
```
# Filter DataFrame for 'Developing' status
developing_df = df[df['Status'] == 'Developing']

# Create a histogram
fig = px.histogram(developing_df, x='Life expectancy ', title="Life Expectancy of Developi
ng Nations")
fig.update_layout(
    xaxis_title='',
    yaxis_title='Count of Ages',
    title_text='<b>Life Expectancy of Developing Countries</b>',
    title_x=0.5,  # Center title
)
fig.show()
```

Life Expectancy of Developing Countries



Life expectancy vs Infant deaths for Countries over Years



Splitting data into Train Test

In [37]:

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

In [38]:

```
print(f"Shape of X_train is: {X_train.shape}")
print(f"Shape of Y_train is: {y_train.shape}\n")
print(f"Shape of X_test is: {X_test.shape}")
print(f"Shape of Y_test is: {y_test.shape}")
```

```
Shape of X_train is: (2350, 21)
Shape of Y_train is: (2350,)

Shape of X_test is: (588, 21)
Shape of Y_test is: (588,)
```

Model Structure

In [39]:

```
model = Sequential([
        Dense(64, activation='relu', input_dim=21),
        Dense(64, activation='relu'),
        Dense(64, activation='relu'),
        Dense(1, activation='linear')
])
```

Model Compiling ¶

In [40]:

```
model.compile(optimizer='adam', loss='mean_squared_error', metrics=['mean_absolute_erro
r','mean_squared_error'])
```

Model Summary