Analyzing and Predicting Life Expectancies

By Canon Pham

My name is Canon Pham, and I am an intended Data Science major at Stanford University looking to gain more practice and experience with machine learning and data visualization skills. For this first personal data science project, I have decided to use a dataset on Kaggle that utilizes various demographic and vaccination statistics to determine how healthcare expenditure and schooling in a country impacts its life expectancy. The official description of the dataset is found below:

Step into the world of global health and demographics with our rich and comprehensive dataset. It's your passport to unraveling the secrets of life expectancy and understanding the pulse of population health. Dive into a treasure trove of valuable information for public health research and epidemiology, where each column tells a unique story about a nation's health journey.

Discover the Gems in Our Dataset:

Country: Explore the global tapestry with data from diverse nations.

Year: Unlock the passage of time and its impact on health trends.

Status: Understand the development status, whether "Developed" or "Developing," that shapes the course of health.

Life Expectancy: Peer into the crystal ball of population health, revealing how long people can expect to live.

Adult Mortality: Gauge the probabilities of survival between ages 15 and 60 per 1,000 population.

Infant Deaths: Delve into infant health with the number of infant deaths per 1,000 live births.

Alcohol: Raise a glass to insights on average alcohol consumption in liters per capita.

Percentage Expenditure: Unearth health expenditure as a percentage of a country's GDP.

Hepatitis B: Measure immunization coverage for Hepatitis B.

Measles: Examine the impact of this preventable disease with the number of reported cases per 1,000 population.

BMI: Step onto the scales of national health with the average Body Mass Index.

Under-Five Deaths: Shine a spotlight on child mortality with the number of deaths under age five per 1,000 live births.

Polio: Inspect immunization coverage for Polio.

Total Expenditure: Track the total health expenditure as a percentage of GDP.

Diphtheria: Assess immunization coverage for Diphtheria.

HIV/AIDS: Witness the prevalence of HIV/AIDS as a percentage of the population.

GDP: Follow the financial pulse of a nation with Gross Domestic Product data.

Population: Witness the ebb and flow of a nation's populace.

Thinness 1-19 Years: Explore the prevalence of thinness among children and adolescents aged 1-19.

Thinness 5-9 Years: Zoom in on thinness among children aged 5-9.

Income Composition of Resources: Decode the composite index reflecting income distribution and resource access.

Schooling: Measure the gift of knowledge with data on average years of schooling.

Using the information in this dataset, I decided to create a statistical study of how various demographic and health-related features correlate to life expectancy in a country and use the features of the dataset to build a model that can predict life expectancies.

Reading in the Data

Firstly, I needed to read in the dataset from Kaggle and import various packages for data visualizations and reading.

In [1]:
```
# This Python 3 environment comes with many helpful analytics libraries instal
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/d
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
from plotnine import * # data visualizations using ggplot
import matplotlib.pyplot as plt # data visualizations using matplotlib
import seaborn as sns # dta visualizations using seaborn

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will lis

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
# You can write up to 20GB to the current directory (/kaggle/working/) that ge
# You can also write temporary files to /kaggle/temp/, but they won't be saved
```

In [3]:
```
# Reading in the dataset
df = pd.read_csv("/content/Life_Expectancy_Data.csv")
df
```

Out[3]:

	Country	Year	Status	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepa
0	Afghanistan	2015	Developing	65.0	263	62	0.01	71.279624	
1	Afghanistan	2014	Developing	59.9	271	64	0.01	73.523582	
2	Afghanistan	2013	Developing	59.9	268	66	0.01	73.219243	
3	Afghanistan	2012	Developing	59.5	272	69	0.01	78.184215	
4	Afghanistan	2011	Developing	59.2	275	71	0.01	7.097109	
...	...	...	...	...	...	...	...	...	
1644	Zimbabwe	2004	Developing	44.3	723	27	4.36	0.000000	
1645	Zimbabwe	2003	Developing	44.5	715	26	4.06	0.000000	
1646	Zimbabwe	2002	Developing	44.8	73	25	4.43	0.000000	
1647	Zimbabwe	2001	Developing	45.3	686	25	1.72	0.000000	
1648	Zimbabwe	2000	Developing	46.0	665	24	1.68	0.000000	

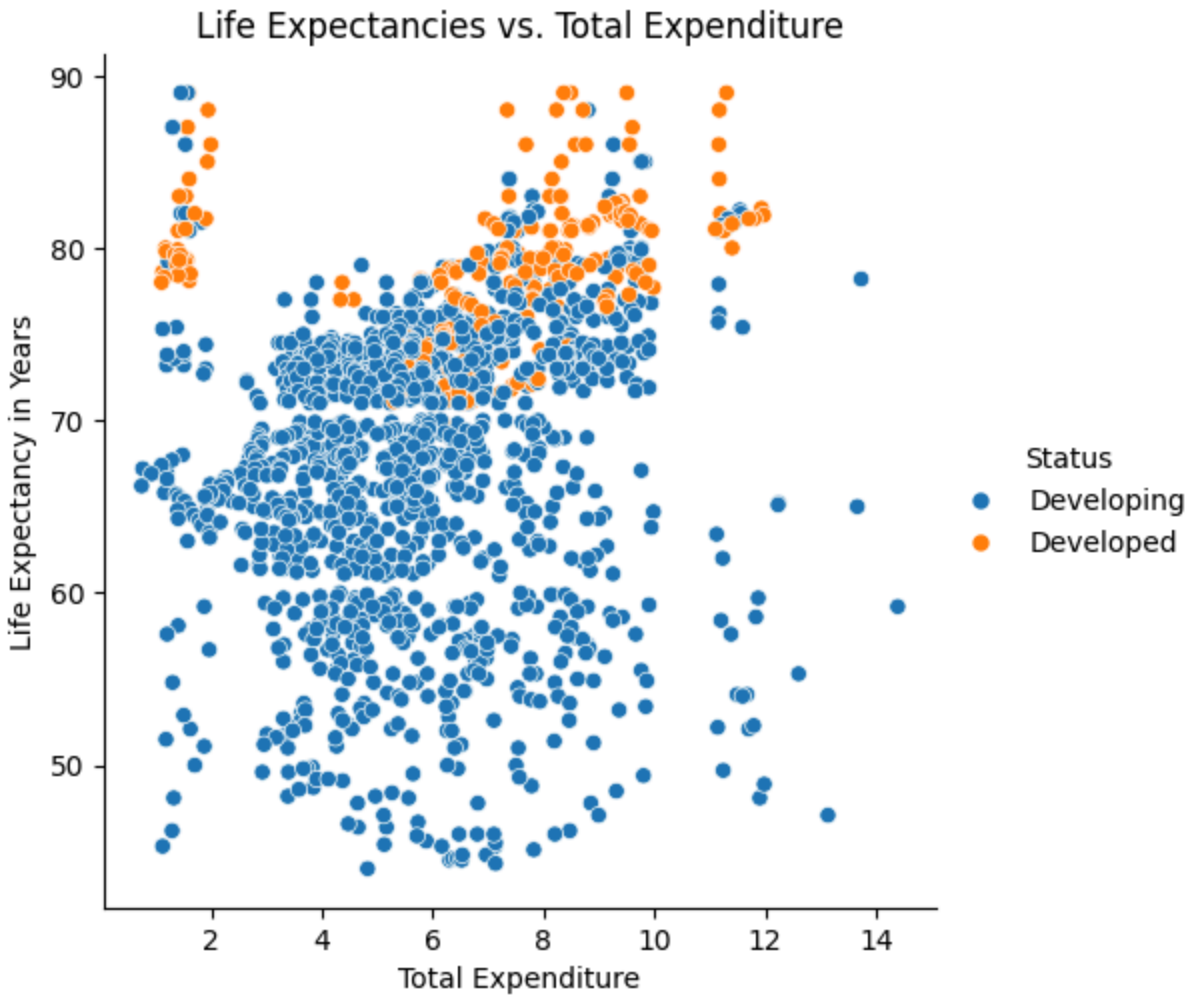
1649 rows × 22 columns

Exploratory Data Analysis

I have divided this project into 2 parts: the first being a visual exploration of the features in the dataset and the second being the creation and testing of a predictive model in order to predict Life Expectancies.

In [6]:
```
# Life Expectancy and Total Health Expenditure in Developed vs. Developing Cou
dplot = sns.relplot(data=df, x = "Total expenditure", y = "Life expectancy ", h
dplot.set_axis_labels("Total Expenditure", "Life Expectancy in Years")
```

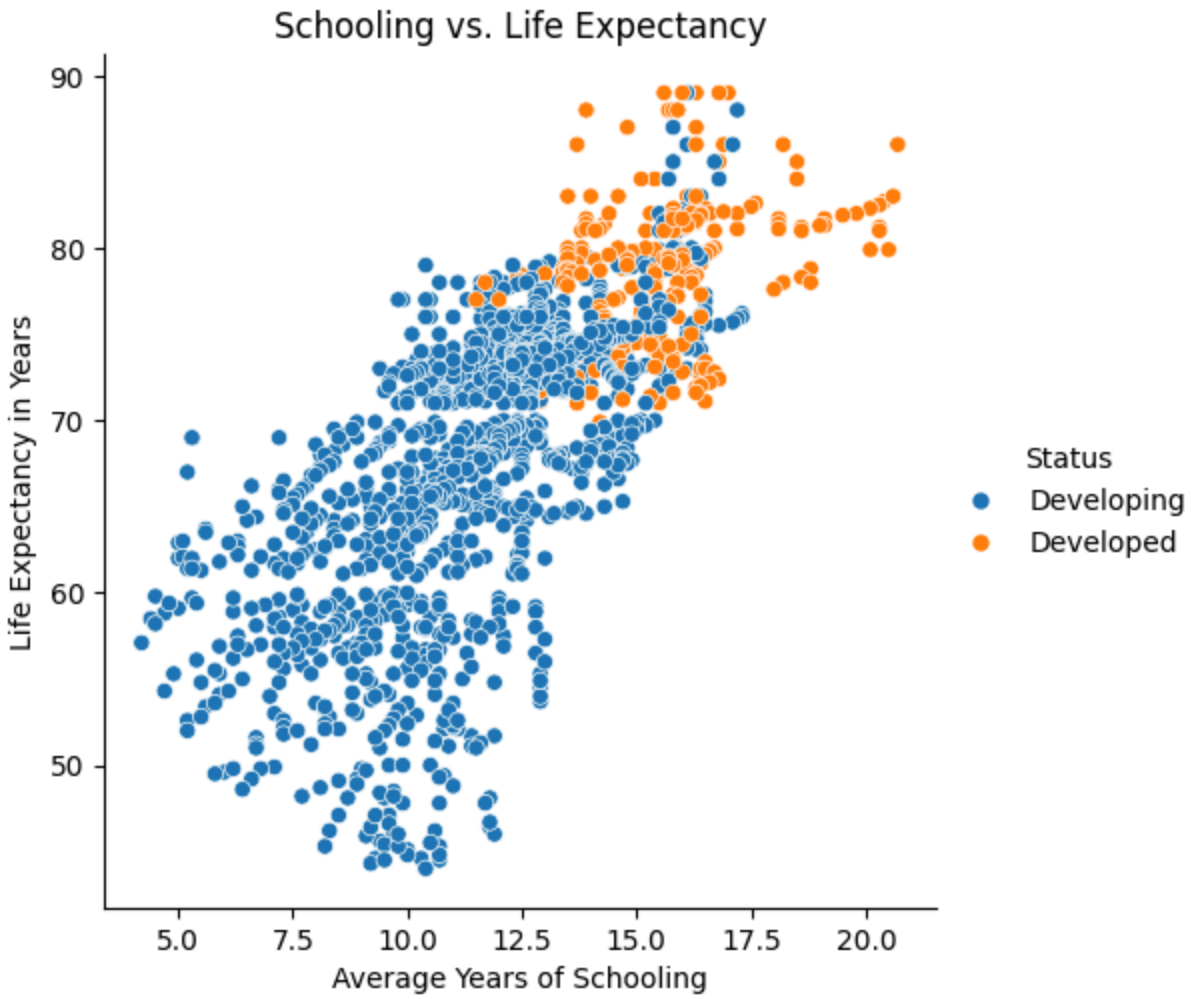
Out[6]:
```
<seaborn.axisgrid.FacetGrid at 0x792aa899aaa0>
```



The above scatterplot demonstrates the very weak correlation between the total that a country spends on healthcare and the life expectancy in that country. As seen in the distribution of the points on the scatterplot, developed countries tend to have higher life expectancies and spend a greater amount of their GPD on healthcare (there is a cluster of developed countries toward the higher limit of Life Expectancy and Total Expenditure).

In [5]:
```
schooling = sns.relplot(data=df, x = "Schooling", y = "Life expectancy ", hue =
schooling.set(title = "Schooling vs. Life Expectancy")
schooling.set_axis_labels("Average Years of Schooling", "Life Expectancy in Yea
```

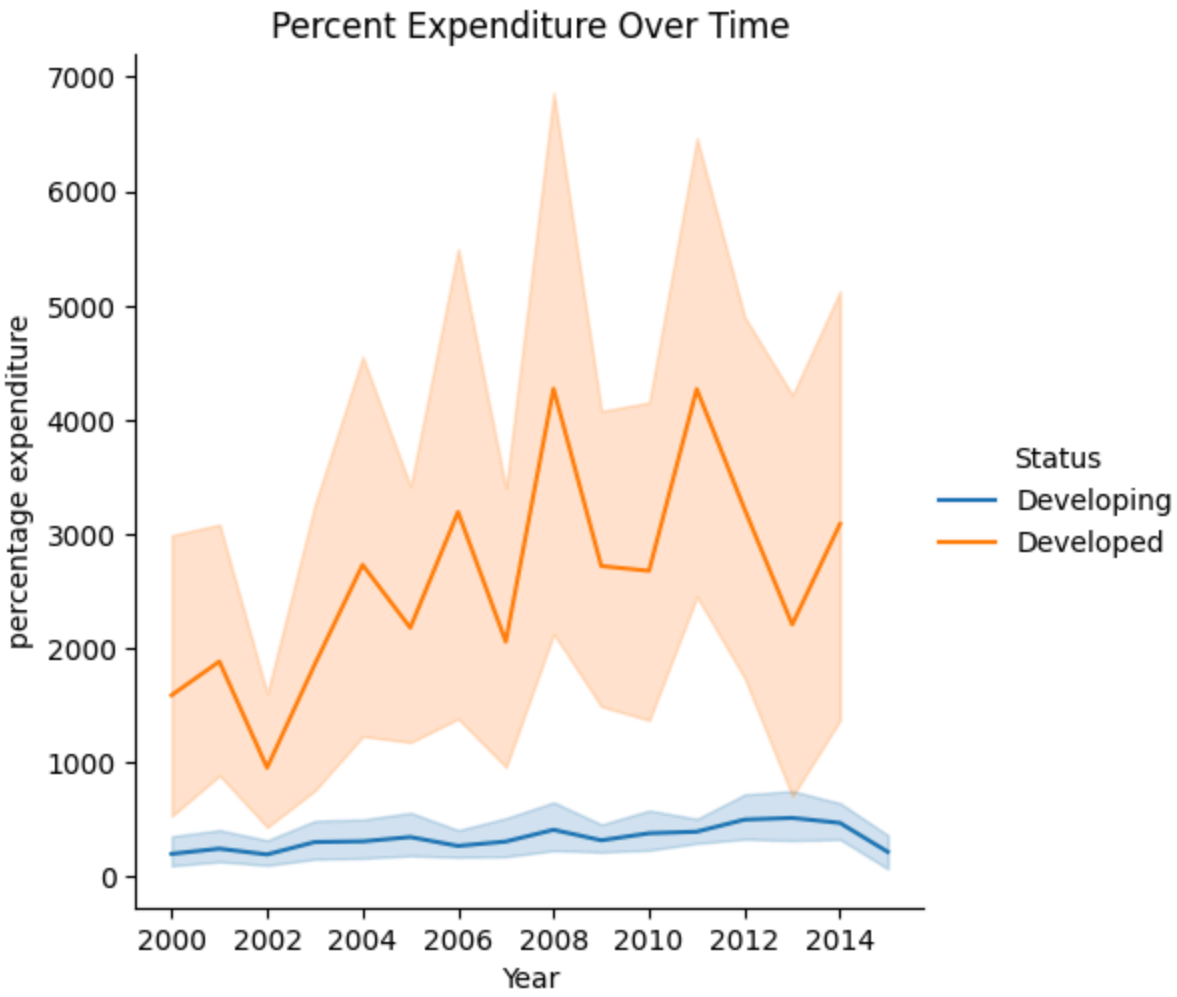
Out[5]:
```
<seaborn.axisgrid.FacetGrid at 0x792ae28a3280>
```



This next scatterplot visually demonstrates the moderately strong correlation between the average years of schooling in a country and its life expectancy. Similar to the first scatterplot, developed countries tend to have much higher life expectancies and average years of formal education. Based on the distribution of these points, all of the developed countries in this dataset have an average of a least 10 years of schooling and a life expectancy of at least 65.

In [7]:
```
# This is the Total Health expenditure over the 2000s for developed versus deve
te_time = sns.relplot(
    data=df, kind="line",
    x="Year", y="percentage expenditure", hue="Status",
)

te_time.set(title = "Percent Expenditure Over Time")
```

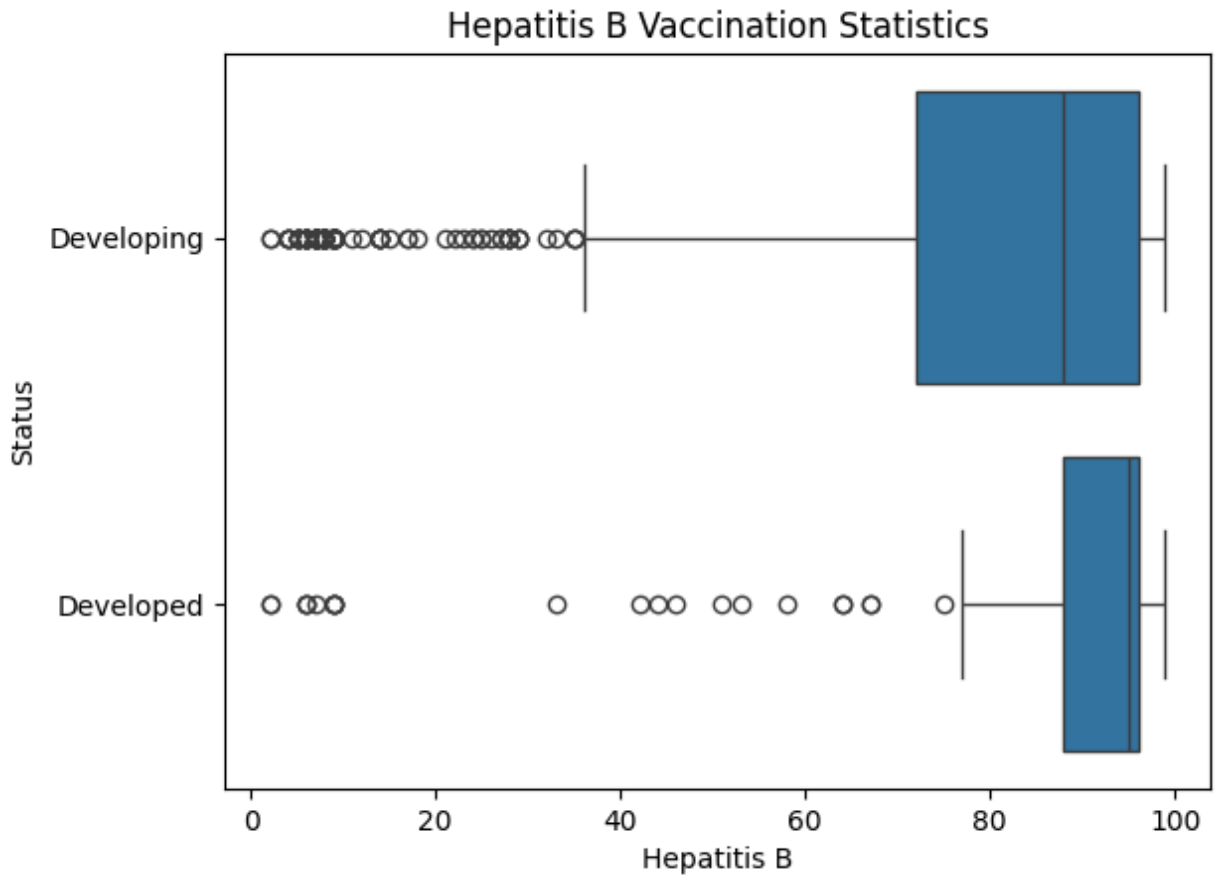
Out[7]:
```
<seaborn.axisgrid.FacetGrid at 0x792aa8a13bb0>
```



This line graph visualizes the percentage of GPD that a country spends on healthcare between 2000 and 2015 for both developed and developing countries. Based on this data, developing countries consistently spent a low percentage of their GPD on healthcare in the 2000s while that expenditure tended to fluctuate in developed countries. The period during which developed countries spent the greatest percentage of their GPD on healthcare was between 2008 and 2012.

In [8]:
```
# This demonstrates the main descriptive statistics of the proportion of the p
Hepatitis_B_box = sns.boxplot(data = df, x="Hepatitis B", y="Status")
Hepatitis_B_box.set(title = "Hepatitis B Vaccination Statistics")
```

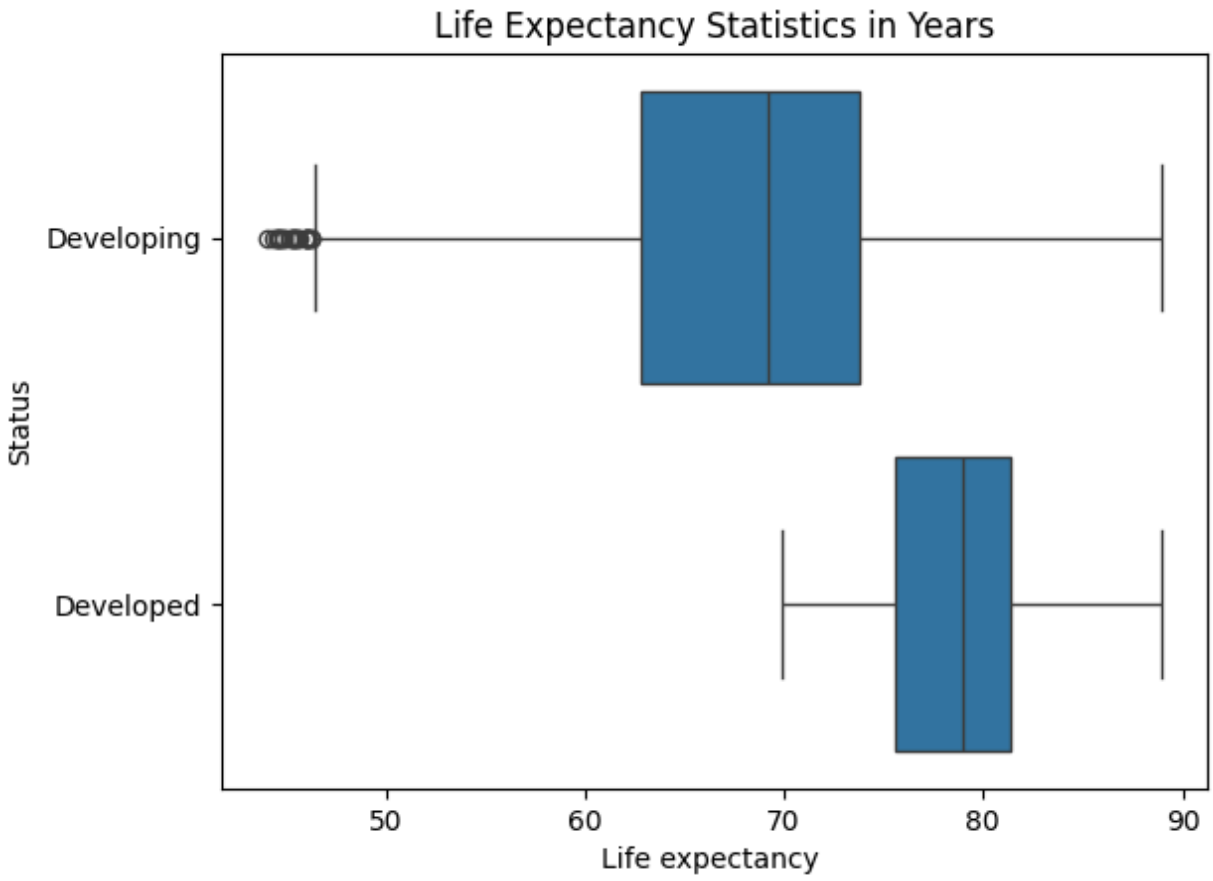
Out[8]:
```
[Text(0.5, 1.0, 'Hepatitis B Vaccination Statistics')]
```



This boxplot provides the medians, quartiles, and mininum and maximum percentages of the population that are vaccinated against Hepatitis B in developed versus developing countries. Over the 2000s, developed countries had a higher median Hepatitis B coverage with approximately 96% being vaccinated in these countries compared to 87.5% in developing countries. The median percentage of those vaccinated against Hepatitis B in developing countries was equal to the lowest 25% of developed countries, meaning that while 75% of developed countries were 87.5% vaccinated against Hepatitis B, only 25% of developing countries were.

In [9]:
```
# Visualizing the descriptive statistics of Life Expectancy in developing versu
Life_box = sns.boxplot(data = df, x="Life expectancy ", y="Status")
Life_box.set(title = "Life Expectancy Statistics in Years")
```

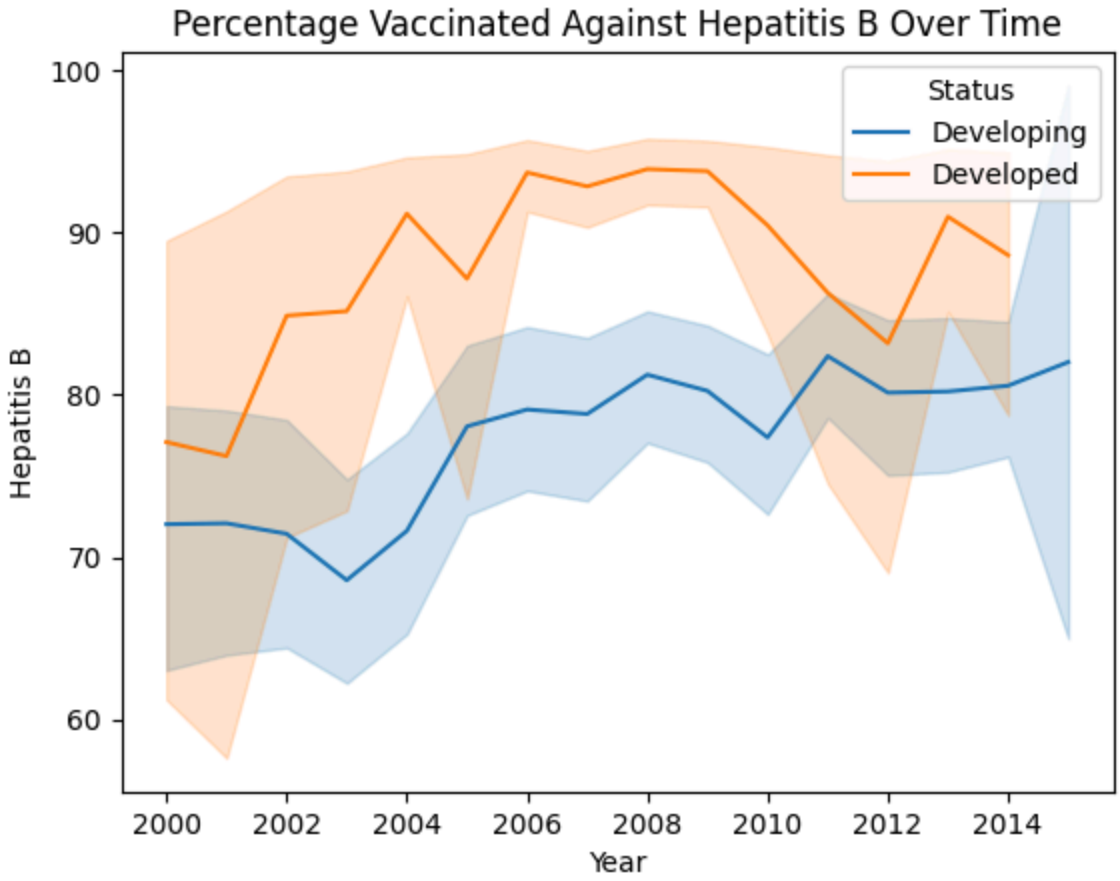
Out[9]:
```
[Text(0.5, 1.0, 'Life Expectancy Statistics in Years')]
```



The above boxplot is a visual representation of life expectancies between 2000 and 2015 in developing versus developed countries. Similar to the Hepatitis B vaccinations data, developed countries had a higher median life expectancy with approximately 78 years compared to approximately 69.5 years in developing countries. Additionally, the variation in life expectancies for developing countries was much greater than those in developed countries. While all of the developed countries had a life expectancy between 70 and 90 years of age, life expectancies in developing countries dropped below 50 years.

In [10]:
```
# Plotting the percentage of a country's population vaccinated against Hepatit
HepB = sns.lineplot(data = df, x="Year", y="Hepatitis B", hue="Status")
HepB.set(title = "Percentage Vaccinated Against Hepatitis B Over Time")
```

Out[10]:
```
[Text(0.5, 1.0, 'Percentage Vaccinated Against Hepatitis B Over Time')]
```



The above line graph demonstrates the percentage of a country's population covered against Hepatitis B between 2000 and 2015 in developing versus developed countries. In developing countries, there was generally an increase in Hepatitis B coverage over time, though there was a significant drop in Hepatitis B coverage between 2009 and 2012 for developed countries. This is interesting because according to earlier analysis, developed countries spent the greatest percentage of their GPD on healthcare between 2008 and 2012.

Creating a Model to Predict Life Expectancies

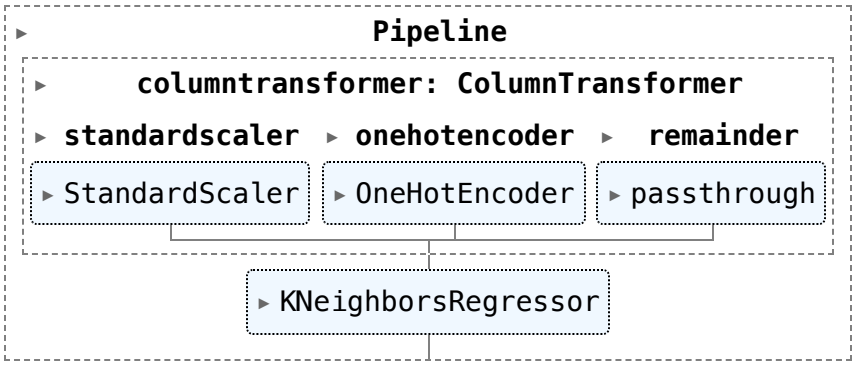
After seeing the relationships between several variables in the dataset and Life Expectancy, my next step was creating a machine learning model to predict Life Expectancies. For my first attempt at predicting life expectancies, I decided to try a K-Nearest Neighbors Regressor model splitting 20% of the original dataset into test data and using the other 80% as training. I ran the model with various features selected as independent variables before arriving at the following list of features:

- Adult Mortality
- Infant Deaths
- Alcohol Consumption
- Percentage Expenditure on Healthcare
- Average Years of Schooling

- Country Name
- Status
- Year
- Income Composition of Resources

In [11]:
```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.compose import make_column_transformer
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.neighbors import KNeighborsRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LinearRegression
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import cross_val_score

# Setting the independent and dependent variables in order to predict life expe
X = df[["Adult Mortality", "infant deaths", "Alcohol", "percentage expenditure"
y = df["Life expectancy "]

# Using Train-test-split to make 20% of the dataset test data
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                    random_state = 104,
                                    test_size = 0.2)

# Standard scaling the numerical variables to put them on the same metric and (
ct_k = make_column_transformer(
    (StandardScaler(), ["Adult Mortality", "infant deaths", "Alcohol", "percen
    (OneHotEncoder(handle_unknown="ignore"), ["Country", "Status"]),
    remainder="passthrough"
)

# Establishing the KNN Regressor
pipeline_k = make_pipeline(
    ct_k,
    KNeighborsRegressor(n_neighbors=5, metric = "manhattan")
)

# Using GridSearch CV to find the best comobination of neighbors and scoring me
grid_search_k = GridSearchCV(pipeline_k,
                             param_grid = {
                                 "kneighborsregressor__n_neighbors": [5, 10, 15
                                 "kneighborsregressor__metric": ["euclidean", "
                             },
                             cv=25, scoring="neg_mean_squared_error", n_jobs=1

# Fitting GridSearchCV on the training data to find the best parameters.
grid_search_k.fit(X_train, y_train)
grid_search_k.best_estimator_
```

Out[11]:



In [12]:
```
# The best model to use would be with 5 neighbors and using the Manhattan dista
grid_search_k.best_params_
```

Out[12]:
```
{'kneighborsregressor__metric': 'manhattan',
 'kneighborsregressor__n_neighbors': 5}
```

In [13]:
```
# Fitting the pipeline on the data for which we do know the Life Expectancies
pipeline_k.fit(X_train, y_train)

# Predicting the Life Expectancies for the 20% of test data based on the values
predictions_k = pipeline_k.predict(X_test)
predictions_k
```

Out[13]:
```
array([57.74, 75.18, 71.2 , 72.22, 53.26, 81.06, 65.36, 71.4 , 50.94,
       55.7 , 73.82, 65.88, 63.2 , 72.42, 73.54, 72.5 , 80.78, 81.76,
       54.68, 57.9 , 74.18, 79.78, 64.54, 76.74, 82.8 , 73.3 , 77.1 ,
       51.96, 70.32, 65.74, 74.74, 48.96, 69.62, 72.36, 53.5 , 74.22,
       62.72, 74.16, 67.04, 74.02, 76.7 , 76.14, 80.66, 68.32, 56.88,
       73.4 , 72.88, 68.08, 78.48, 70.7 , 49.66, 82.74, 65.5 , 60.42,
       73.84, 56.62, 77.16, 71.9 , 73.58, 69.46, 59.96, 67.86, 73.42,
       75.92, 63.16, 72.22, 56.3 , 71.04, 67.34, 59.54, 68.44, 61.96,
       75.68, 72.36, 78.52, 76.5 , 49.06, 65.3 , 67.02, 48.76, 84.04,
       72.76, 64.46, 51.3 , 72.94, 81.6 , 66.38, 67.3 , 82.42, 72.88,
       67.52, 71.28, 69.04, 63.16, 73.24, 61.24, 72.7 , 75.34, 58.78,
       74.76, 66.5 , 69.74, 60.16, 63.02, 74.44, 59.46, 80.32, 73.12,
       75.28, 61.44, 80.32, 72.72, 49.44, 83.12, 70.58, 79.82, 73.6 ,
       82.64, 73.52, 72.16, 56.62, 74.22, 73.38, 62.4 , 62.22, 74.74,
       81.92, 45.72, 62.1 , 71.26, 73.84, 72.62, 73.  , 72.34, 75.56,
       74.48, 81.16, 72.58, 77.16, 73.74, 67.32, 73.4 , 69.12, 84.04,
       75.98, 67.56, 76.3 , 51.98, 62.52, 50.32, 70.54, 71.4 , 84.16,
       76.2 , 72.88, 50.52, 77.9 , 74.44, 58.36, 82.8 , 73.08, 71.78,
       73.6 , 73.  , 81.1 , 74.74, 65.42, 84.16, 75.92, 60.48, 72.08,
       73.42, 79.26, 54.36, 72.16, 61.36, 68.82, 68.1 , 52.88, 55.4 ,
       75.94, 65.9 , 81.54, 56.22, 71.8 , 72.22, 68.38, 69.36, 70.14,
       82.9 , 72.96, 79.96, 67.94, 66.38, 72.92, 54.48, 63.24, 73.22,
       68.34, 72.12, 72.5 , 63.18, 78.08, 58.  , 79.42, 74.62, 63.46,
       73.66, 72.16, 54.42, 60.58, 52.36, 76.6 , 71.44, 73.72, 81.98,
       76.72, 76.42, 62.72, 54.98, 63.16, 75.98, 81.88, 80.32, 68.1 ,
       69.02, 80.7 , 62.72, 55.88, 73.52, 67.3 , 73.72, 66.44, 63.4 ,
       64.96, 68.08, 81.24, 57.32, 68.62, 79.62, 57.78, 68.24, 71.24,
       65.5 , 82.04, 58.2 , 79.72, 61.44, 75.94, 61.12, 71.78, 54.8 ,
       76.24, 84.16, 71.8 , 82.16, 51.56, 51.5 , 74.36, 67.54, 81.1 ,
       62.74, 74.44, 81.1 , 66.1 , 72.7 , 82.34, 74.74, 63.74, 75.72,
       68.32, 72.66, 75.96, 68.02, 67.86, 73.94, 68.52, 79.78, 81.76,
       62.52, 70.7 , 72.2 , 66.3 , 64.02, 73.4 , 64.06, 67.56, 74.74,
       73.94, 65.58, 57.84, 56.76, 72.92, 76.08, 84.04, 63.88, 72.16,
       74.84, 72.22, 75.5 , 81.78, 65.06, 53.74, 79.68, 67.54, 51.26,
       75.12, 64.82, 65.88, 65.76, 78.64, 82.34, 67.86, 62.52, 63.48,
       65.14, 79.82, 72.14, 54.22, 68.1 , 73.54, 79.54, 58.36, 65.56,
       70.2 , 54.26, 55.24, 73.82, 54.88, 65.86])
```

Evaluating the KNN Regression Model

To determine the accuracy of the model, I used mean squared error to find the variation between the actual and predicted Life Expectancies on average. I obtained a mean squared error of 3.765, meaning that the squared variation between the true and predicted Life Expectancies was approximately 3.765.

In [14]:
```
# Calculating the mean squared error between the predicted Life Expectancies a
Error = mean_squared_error(y_test, predictions_k)
Error
```

Out[14]:
```
3.7651721212121223
```

In [15]:
```
# Creating a new column of the dataframe with the predictions
df["Predictions_K"] = pd.DataFrame(predictions_k)
df["Predictions_K"]
```

Out[15]:
```
0       57.74
1       75.18
2       71.20
3       72.22
4       53.26
        ...
1644      NaN
1645      NaN
1646      NaN
1647      NaN
1648      NaN
Name: Predictions_K, Length: 1649, dtype: float64
```

In [16]:
```
# Creating a dataframe with only the test data for which we predicted the Life
Predictions_df_k = df.loc[df["Predictions_K"].notna()].reset_index()
Predictions_df_k
```

Out[16]:

	index	Country	Year	Status	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure
0	0	Afghanistan	2015	Developing	65.0	263	62	0.01	71.279624
1	1	Afghanistan	2014	Developing	59.9	271	64	0.01	73.523582
2	2	Afghanistan	2013	Developing	59.9	268	66	0.01	73.219243
3	3	Afghanistan	2012	Developing	59.5	272	69	0.01	78.184215
4	4	Afghanistan	2011	Developing	59.2	275	71	0.01	7.097109
...	...	...	...	...	...	...	...	...	...
325	325	Canada	2003	Developing	79.7	78	2	7.70	4687.845565
326	326	Central African Republic	2014	Developing	58.0	437	15	0.01	53.439643
327	327	Central African Republic	2013	Developing	49.9	451	16	0.01	52.377666
328	328	Central African Republic	2012	Developing	53.0	439	16	0.01	7.344808
329	329	Central African Republic	2011	Developing	49.8	443	16	1.66	58.529475

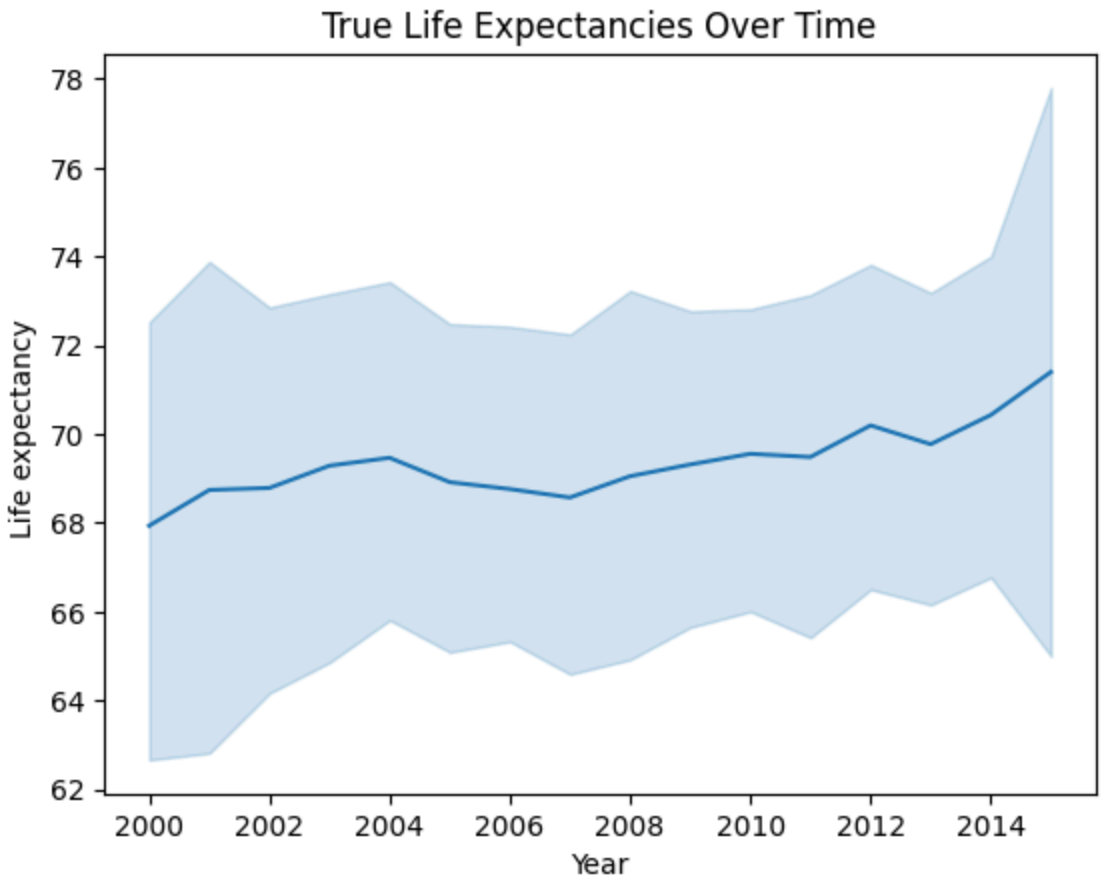
330 rows × 24 columns

Once I calculated the mean squared error, I independently graphed the actual Life Expectancies over time and the predicted Life Expectancies over time for the test data. These 2 graphs demonstrated similar relationships between Life Expectancy and time, though the predicted values had significantly sharper fluctuations between 2000 and 2008.

In [17]:
```
# Plotting the Life Expectancies over the years
True_Life= sns.lineplot(data = Predictions_df_k, x="Year", y="Life expectancy "
```

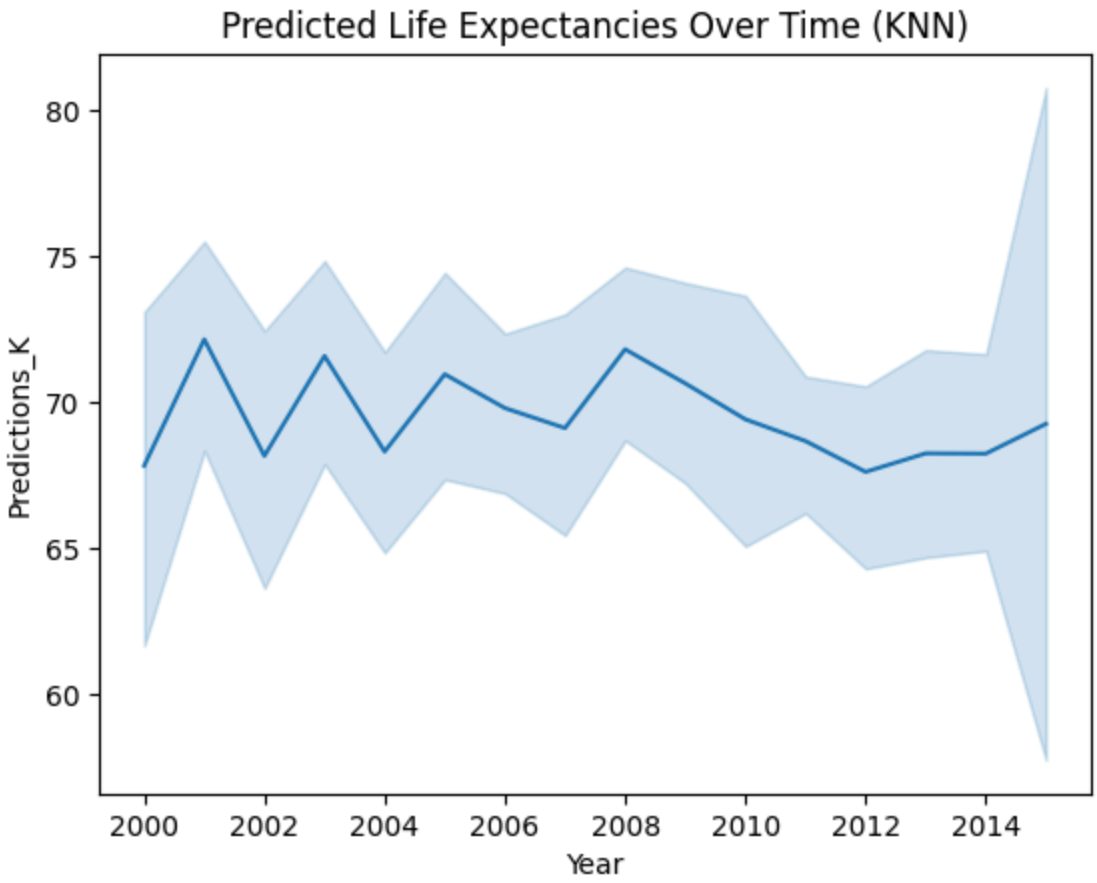
```
True_Life.set(title = "True Life Expectancies Over Time")
```

Out[17]:
```
[Text(0.5, 1.0, 'True Life Expectancies Over Time')]
```



In [18]:
```
# Plotting the predicted Life Expectancies and comparing them to the true valu
preds_k= sns.lineplot(data = Predictions_df_k, x="Year", y="Predictions_K")
preds_k.set(title = "Predicted Life Expectancies Over Time (KNN)")
```

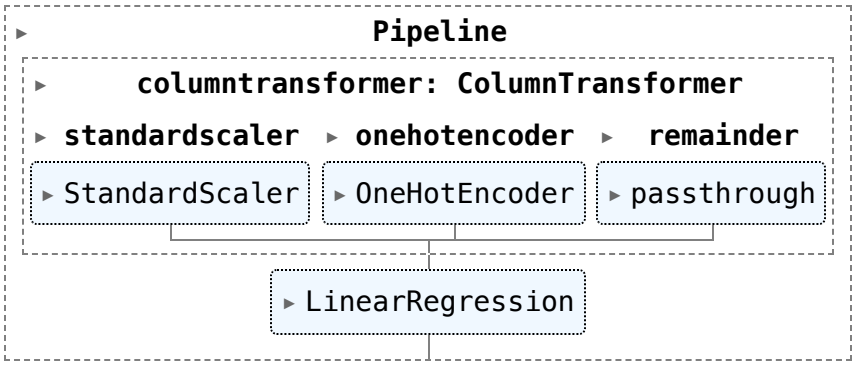
Out[18]:
```
[Text(0.5, 1.0, 'Predicted Life Expectancies Over Time (KNN)')]
```



After viewing the results for the K-Nearest Neighbors model, I decided to try a Linear Regression model using the same independent variable features and test size to predict Life Expectancies. I used the same pre-processing with standard scaling and OneHotEncoding.

In [19]:
```
# Using the same features as independent variables
X = df[["Adult Mortality", "infant deaths", "Alcohol", "percentage expenditure"
y = df["Life expectancy "]

# Creating a 20/80 test and training split for the data
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                    random_state = 104,
                                    test_size = 0.2)

# Standard scaling quantitative variables and OneHotEncoding categorical varia
ct_lin = make_column_transformer(
    (StandardScaler(), ["Adult Mortality", "infant deaths", "Alcohol", "percen
    (OneHotEncoder(handle_unknown="ignore"), ["Country", "Status"]),
    remainder="passthrough"
)

# Using linear regression
pipeline_lin = make_pipeline(
    ct_lin,
    LinearRegression()
)

# Fitting the pipeline on the data for which we do know the Life Expectancies
pipeline_lin.fit(X_train, y_train)
```

Out[19]:



In [20]:
```
# Making predictions on the 20% of test data using the features and model sele
predictions_lin = pipeline_lin.predict(X_test)
predictions_lin
```

Out[20]:
```
array([57.10916342, 75.25682833, 71.33841144, 73.42818457, 50.91017241,
       81.00614122, 65.01179912, 70.82395465, 48.56869142, 55.42736029,
       72.94313596, 65.86053373, 61.37268377, 71.95034287, 75.01022024,
       72.08366871, 80.51374639, 83.02610688, 54.29702855, 58.97033779,
       72.6422166 , 80.96650959, 63.69345703, 77.43322845, 82.44136463,
       72.80873392, 77.52752642, 53.70292985, 68.98196126, 66.30911549,
       75.76361986, 51.06301831, 69.33566502, 72.3503174 , 54.08847004,
       73.35236069, 63.14116706, 75.20870328, 67.21458083, 73.16713204,
       77.47782584, 75.14933857, 82.6309075 , 70.25296074, 57.56540713,
       72.4747687 , 73.84863944, 67.5869025 , 79.24518883, 70.16415393,
       49.34417907, 83.48800573, 65.8903893 , 59.43743764, 75.14355459,
       56.9619653 , 77.43561325, 70.60752785, 74.76160778, 69.35790134,
       56.47766428, 67.17118672, 72.98148937, 75.9157563 , 60.42377547,
       73.73099255, 57.01415342, 70.60759008, 67.87274264, 56.42771207,
       65.91727356, 63.4135369 , 78.4985292 , 71.65457628, 79.55057128,
       76.40031388, 51.30552396, 64.13300484, 66.6649678 , 48.72712113,
       82.5939018 , 72.83128569, 59.65174777, 49.91325011, 75.05470843,
       81.9088407 , 67.74783753, 66.82496242, 82.03915576, 72.1060214 ,
       68.10852051, 70.42433547, 68.43108955, 60.83906152, 73.23819527,
       61.12041372, 71.24119677, 75.8245244 , 57.99265137, 74.55845123,
       63.80165119, 70.19319949, 58.4642678 , 63.56816471, 71.6542951 ,
       59.70900347, 81.01520656, 72.9958634 , 73.3334807 , 60.22352916,
       78.02765628, 72.8838482 , 49.74143291, 82.02774962, 69.27180648,
       80.21410002, 75.37103218, 81.88688569, 73.13041433, 73.19837675,
       55.56350475, 74.66225649, 72.64079979, 63.3462359 , 59.34885851,
       76.37426882, 83.27731099, 48.41441788, 63.48986358, 70.89106156,
       74.21568895, 71.61532231, 73.26534775, 74.49529563, 75.04027153,
       75.85914662, 81.40828067, 72.92689338, 79.4479472 , 74.71565239,
       66.208078  , 74.16833066, 66.96969908, 82.61544868, 76.00262644,
       69.02520645, 78.071317  , 50.84354942, 62.73039516, 46.57517854,
       69.98071478, 71.47303342, 81.84203235, 76.38034673, 73.16122792,
       50.03306606, 79.22049113, 73.96902937, 57.14326358, 82.77892161,
       73.95960334, 69.72559243, 72.66579128, 72.59104161, 80.42474597,
       75.37182443, 65.8570911 , 82.63510953, 76.56341302, 60.25560912,
       71.2248083 , 73.74713275, 79.83837885, 54.3589344 , 70.49207037,
       61.93173618, 67.86352769, 68.48183078, 54.32322688, 47.39946121,
       75.25810043, 67.41407887, 79.9094221 , 55.43255588, 71.1370418 ,
       74.02385958, 69.17228021, 68.68158082, 70.28351553, 80.7542053 ,
       70.52486976, 80.61513133, 69.04818947, 67.87619545, 73.32388006,
       55.930066  , 61.38593717, 73.68096187, 67.83745919, 71.23277751,
       72.23132092, 64.1187617 , 78.60978237, 49.93934624, 80.11626601,
       74.36474383, 63.7364885 , 74.71284027, 72.07333366, 57.12200189,
       58.68214507, 55.74358558, 77.99817936, 69.95876181, 72.791119  ,
       80.48358317, 77.12064978, 76.71403144, 62.03166606, 54.8002646 ,
       61.14658876, 76.87077357, 84.28262816, 80.53003908, 65.83347755,
       68.21583855, 80.01186693, 63.41057046, 56.42967484, 73.20690704,
       66.48073633, 74.65202326, 66.61928153, 64.35842964, 63.73101504,
       69.01457114, 79.96076074, 57.45014654, 68.55670276, 80.1269642 ,
       50.39488282, 68.74005814, 70.28275434, 66.50489683, 83.82845141,
       58.53915391, 79.84156616, 59.49772764, 75.56181571, 60.14130465,
       69.40099924, 55.59024657, 76.62558077, 82.46405932, 71.47499855,
       83.04248023, 47.65509967, 50.90036883, 75.28686815, 67.9113767 ,
       81.34073593, 61.4798707 , 75.20062801, 81.01881294, 62.94977231,
       70.87944392, 80.32343286, 73.88047244, 64.78223103, 77.16931649,
       69.59271368, 71.39067981, 75.31508299, 68.06231184, 66.26275216,
       74.31846927, 68.49653506, 81.52355566, 81.4408225 , 61.0561171 ,
       70.69594404, 70.20307675, 66.86042591, 62.33983614, 74.48417837,
       64.18104809, 69.22295829, 74.824846  , 73.45112266, 66.63865561,
       58.09127036, 55.52016436, 70.75652126, 75.79829525, 82.9222777 ,
       63.5407834 , 70.1699857 , 76.01538973, 72.77183066, 75.68932343,
```

```
81.66438827, 65.213421  , 51.41697746, 80.29995621, 67.51772495,
51.59613749, 74.66797858, 65.65902194, 65.98520364, 63.32988577,
78.58967226, 81.14134129, 67.44513136, 60.71569731, 62.57753219,
67.49115713, 80.45526802, 72.93606509, 55.82391382, 67.14295512,
73.95675368, 78.81698549, 58.01918649, 65.35266094, 69.50811451,
55.76510512, 51.13298811, 73.83622653, 59.377855  , 65.44026641])
```

Evaluating the Linear Regression Model

Based on the mean squared error, the linear regression model was slightly more accurate because it had a lower average variation between the actual and predicted Life Expectancies, with 3.735 versus 3.765 for the KNN Regression model.

In [21]:
```
# Calculating the mean_squared error between the true and predicted Life Expec
Error_lin = mean_squared_error(y_test, predictions_lin)
Error_lin
```

Out[21]:
```
3.735134279537227
```

For linear regression models, it is also possible to calculate an r2 score to determine the accuracy of a model. This measures what percentage of variation between true and predicted values can be accounted for by independent variables.

In [22]:
```
# Calculating how much of the error is based on the relationships between indep
from sklearn.metrics import r2_score
error_r2 = r2_score(y_test, predictions_lin)
error_r2
```

Out[22]:
```
0.9558449609486388
```

For this particular model, approximately 95.58% of variation between the actual and predicted Life Expectancies is accounted for by the independent variables used in the model. A high r2 score (between 0 and 1) indicates a more accurate model.

In [23]:
```
# Finding the linear regression coefficients
pd.DataFrame(zip(X.columns, pipeline_lin.named_steps["linearregression"].coef_
```

Out[23]:

	0	1
0	Adult Mortality	-0.360594
1	infant deaths	-0.341724
2	Alcohol	-0.591621
3	percentage expenditure	0.071598
4	Schooling	0.383894
5	Country	1.230378
6	Status	0.044666
7	Year	-8.063229
8	Income composition of resources	7.974602

Next, I calculated the regression coefficients for each of the features used as independent variables. These coefficients indicate how much the predicted Life Expectancy would change if the variable was shifted by 1 standard deviation. For instance, the most significant positive factor in this model was the income composition of resources, as the Life Expectancy would increase by 7.97 if the income composition of resources increased by 1 standard deviation. On the other hand, Year was also a significant factor because the Life Expectancy will decrease by about 8.06 when the Year increases by 1 standard deviation.

In [24]:
```
# Creating a new column of the dataframe with the linear regression predictions
df["Predictions_Lin"] = pd.DataFrame(predictions_lin)
df["Predictions_Lin"]
```

Out[24]:
```
0       57.109163
1       75.256828
2       71.338411
3       73.428185
4       50.910172
          ...
1644          NaN
1645          NaN
1646          NaN
1647          NaN
1648          NaN
Name: Predictions_Lin, Length: 1649, dtype: float64
```

In [25]:
```
# Making a dataframe of only the 20% of test data with the predictions from bo
Lin_Preds = df.loc[df["Predictions_Lin"].notna()].reset_index()
Lin_Preds
```

Out[25]:

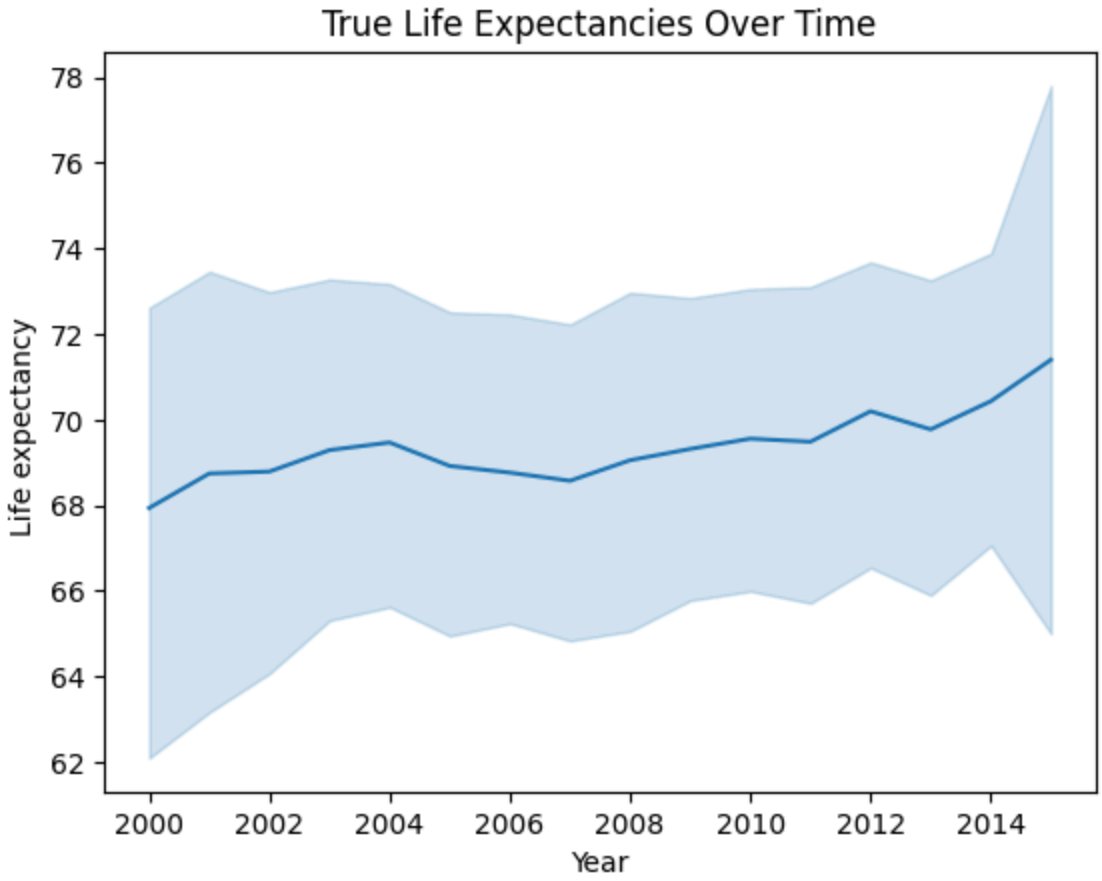
	index	Country	Year	Status	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure
0	0	Afghanistan	2015	Developing	65.0	263	62	0.01	71.279624
1	1	Afghanistan	2014	Developing	59.9	271	64	0.01	73.523582
2	2	Afghanistan	2013	Developing	59.9	268	66	0.01	73.219243
3	3	Afghanistan	2012	Developing	59.5	272	69	0.01	78.184215
4	4	Afghanistan	2011	Developing	59.2	275	71	0.01	7.097109
...	...	...	...	...	...	...	...	...	...
325	325	Canada	2003	Developing	79.7	78	2	7.70	4687.845565
326	326	Central African Republic	2014	Developing	58.0	437	15	0.01	53.439643
327	327	Central African Republic	2013	Developing	49.9	451	16	0.01	52.377666
328	328	Central African Republic	2012	Developing	53.0	439	16	0.01	7.344808
329	329	Central African Republic	2011	Developing	49.8	443	16	1.66	58.529475

330 rows × 25 columns

After calculating the scores and coefficients, I separately graphed the actual Life Expectancies over time and the predicted Life Expectancies over time for the 330 values in the test data. These 2 graphs demonstrated similar relationships between Life Expectancy and time, though the predicted values had sharper fluctuations.

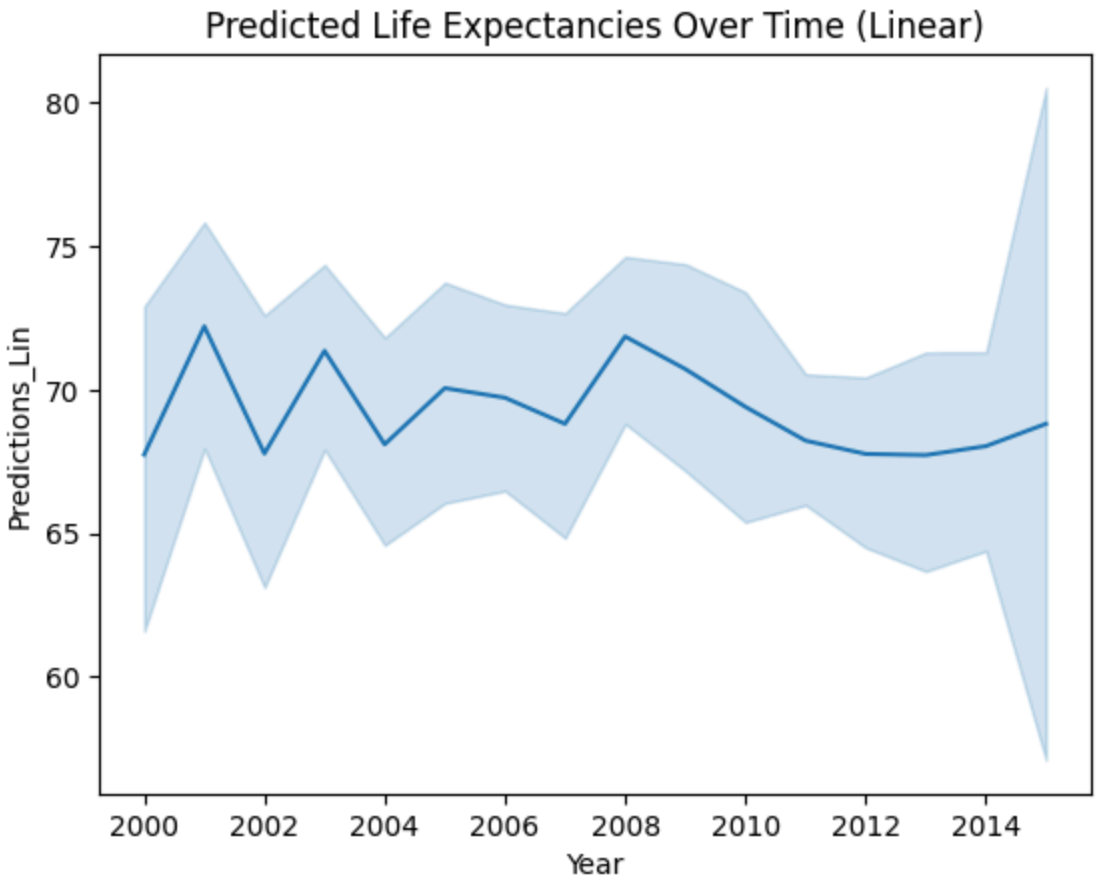
In [26]:
```
# Plotting the true Life Expectancies over time
Actual_life= sns.lineplot(data = Lin_Preds, x="Year", y="Life expectancy ")
Actual_life.set(title = "True Life Expectancies Over Time")
```

Out[26]:
```
[Text(0.5, 1.0, 'True Life Expectancies Over Time')]
```



In [27]:
```
# Plotting the predicted Life Expectancies over time
predicted_life = sns.lineplot(data = Lin_Preds, x="Year", y="Predictions_Lin")
predicted_life.set(title = "Predicted Life Expectancies Over Time (Linear)")
```

Out[27]:
```
[Text(0.5, 1.0, 'Predicted Life Expectancies Over Time (Linear)')]
```



What to Take Away From This Study

Exploring the relationships between healthcare demographics and life expectancy in a country can reveal which factors most highly influence life expectancy and how trends have existed over time. Using this information to build a model can also work to provide future experimentation with new demographic information and life expectancy. Additionally, a country may be able to set a goal for which percentge of their population will be vaccinated against a certain illness based on how these statistics have developed in the recent past and how it would impact the predicted life expectancy in that country.

In [ ]:
```
# @markdown Run this cell to download this notebook as a webpage, `_NOTEBOOK.h

import google, json, nbformat

# Get the current notebook and write it to _NOTEBOOK.ipynb
raw_notebook = google.colab._message.blocking_request("get_ipynb",
                                                      timeout_sec=30)["ipynb"]
with open("_NOTEBOOK.ipynb", "w", encoding="utf-8") as ipynb_file:
  ipynb_file.write(json.dumps(raw_notebook))

# Use nbconvert to convert .ipynb to .html.
!jupyter nbconvert --to html --log-level WARN _NOTEBOOK.ipynb

# Download the .html file.
google.colab.files.download("_NOTEBOOK.html")
```