PREDICTIVE ANALYSIS OF LIFE EXPECTANCY IN COUNTRIES

Adekola Adepoju, **Harsha Kamineni**, **Lakshmi Ravindranathan Menon**, **Lavanya Pragada** and **Pooja Manikonda**.
Indiana University-Purdue University, Indianapolis, IN 46202, USA,

agadepoj@iu.edu, hskamine@iu.edu, lmenon@iu.edu, lpragada@iu.edu, pomani@iu.edu.

Abstract. This project was carried out to analyzes various attributes pertaining to a country and its citizens. The goal is to identify how these attributes can help predict life expectancy. Through data analysis, we analyzed how lifestyle choices, disease prevalence and socio-economic status of a country has life expectancy over a period of years. This will enable the community to improve life expectancy by understanding the major factors that improve or decrease life expectancy.

Keywords: Country, Status, Year, Infant Death, Measles, Diphtheria, Polio, Under five deaths, Percentage expenditure, Schooling, BMI, Life expectancy.

1 Project Scope

1.1 Introduction

According to the Centers for Disease Control and Prevention, live expectancy can be defined as the average number of years a person is expected to live after attaining a given age (*Life expectancy*, 2023).

There are numerous factors that can affect life expectancy, with a few being based on gender, genetics, perinatal and childhood conditions, education, socio-economic status etc. (Beckman, 2016)

Being able to estimate life expectancy is important as it helps to plan treatment strategies to focus on the critical areas that can help improve life expectancy eventually. There are a lot of factors that need to be considered when predicting life expectancy for example, the economic status of the country the individual is living, the lifestyle choices like lack of exercise, smoking, drinking, exposure to diseases and an individual's economic status to mention a few.

1.2 Aim

The aim of this project is to analyze the life expectancy across the different countries based on the lifestyle choices, disease prevalence and the economic status of each country by comparing their progress yearly.

Based on our research questions, our two hypotheses are below:

- Null Hypothesis – The attributes such as the country's status, percentage expenditure, BMI, infant and under-five deaths, measles, polio, diphtheria, schooling do not have any effect on life expectancy of the people.
- Alternate Hypothesis – The attributes such as the country's status, percentage expenditure, BMI, infant and under-five deaths, measles, polio, diphtheria, schooling do have any effect on life expectancy of the people.

1.3 Purpose

- The project will analyze if a country's status, percentage expenditure, BMI, infant and under-five deaths, measles, polio, diphtheria, schooling play a significant role in determining life expectancy of it's citizens. The goal of this project is to help identify how these attributes can help predict life expectancy.

2 Methodology

2.1 Steps of the Project

The main focus of our project was comparing the effects country's status, percentage expenditure, BMI, infant and under-five deaths, measles, polio, diphtheria, schooling on life expectancy using database technologies such as SQL and Python. The tools we have used are Python Jupyter notebook, phpMyAdmin. The methodology of our project is listed below:

- **Data Collection**
- **Exploratory Data Analysis**
- Descriptive statistics
- Data pre-processing
- Data visualization
- Normality and statistical tests
- Correlation matrix
- **Train and test split of data**
- **Developing A Model**
- **Performance analysis of the model**
- **Results**

2.2 Original Team Members and Responsibilities

The team worked on the project using background skills and knowledge making the project team diverse. Below was the original list of team members that worked on the project and the assigned responsibilities at the beginning of the project.

Name	Responsibilities
Adekola Adepoju	Data collection
Harsha Kamineni	Data extraction and cleaning
Lakshmi Ravindranathan Menon	Exploratory Data analysis
Lavanya Pragada	Binary classification model development
Pooja Manikonda	Data visualization and Data evaluation

Fig. 1. Original Responsibilities of Team Members for This Project

2.3 Actual Contributions from Individual Team Members

As the project commenced, we had to revisit and reassign responsibilities according to the strength of the team members. Below is the actual contribution of individual team members.

Name	Responsibilities
Adekola Adepoju	Data collection, Data extraction, performance analysis of the model
Harsha Kamineni	Exploratory Data analysis, performance analysis of the model
Lakshmi Ravindranathan Menon	Statistical Testing, performance analysis of the model
Lavanya Pragada	Data cleaning, Data Pre-processing, performance analysis of the model
Pooja Manikonda	Developing a Machine learning Model, performance analysis of the model.

Fig. 2. Revised Responsibilities of Team Members for This Project

2.4 Project Challenges

We did not face a lot of challenges during this project, but a few notable ones are listed below:

After submitting out project proposal, we found out that another team chose the exact same data set that we chose, and this was a cause of concern I and my team members as we had to start looking for a new dataset to work with because we wanted our project to be unique. However, Prof allowed our team and the other team to continue with the dataset by splitting the variables equally so each project even though working on the same dataset could have unique identities.

We also initially had doubts on how to extract our data and connect it to jupyter hub from phpMyAdmin but that was resolved with the aid of our assigned TA (Pallavi) who assured us to progress with our project.

We also had various ideas on how to go about our methodology but once again, thanks to Pallavi (our assigned TA), she helped us come to a conclusion on the best methodology to use and then we proceeded with the agreed choice.

Having regular in person meetings was a struggle but because of the dedication the team had from day 1, we were able to strive and ensure we were all available when needed.

We had no null values of na values in our data, but we had a lot of outliers which we had to treat in order not to affect the accuracy of out machine learning model.

3 Data Collection

Our dataset was collected from Kaggle - https://www.kaggle.com/datasets/arunjangir245/life-expectancy-data. This dataset provides key health and development indicators on global insights like BMI, country status and more.

4 Data Extraction and Storage Data Extraction

4.1 Data Extraction

We extracted and stored the data on SQL using phpMyAdmin**.** Below are descriptions of the attributes we used in our data analysis:

Independent Variable

- Categorical Data

• Country

• Status

- Numerical Data

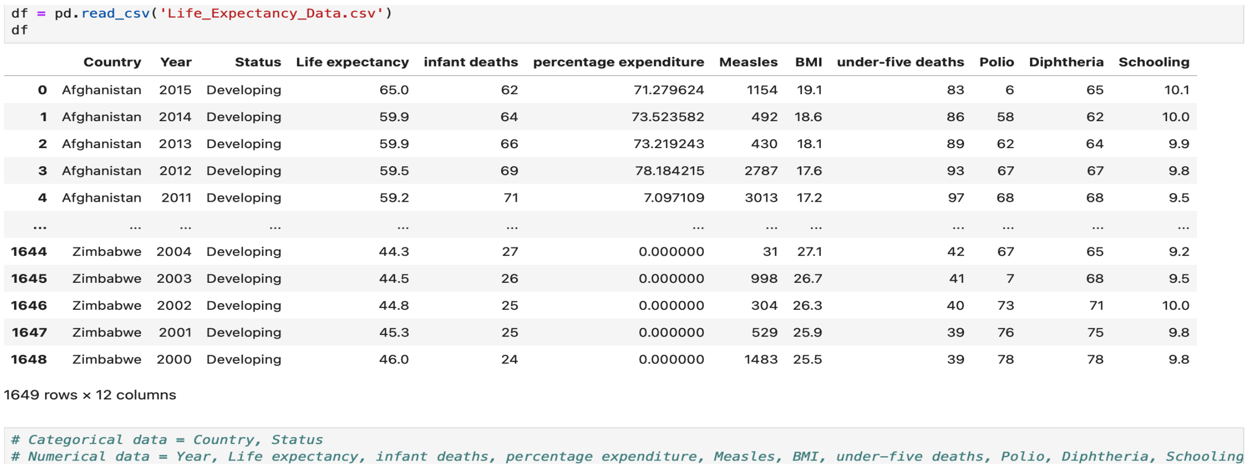
• Under five deaths

- Year
- Percentage expenditure
- Infant Death
- Polio
- Measles
- Diphtheria
- BMI
- Schooling

Dependent variable: Life Expectancy

4.2 Data Cleaning & Pre-processing

We connected and read the csv file on Python Jupyter Notebook and then described the attributes we selected by calculating the statistical values. We also used Python to check for null values and na values. We checked for outliers and then treated the outliers with Inter quartile range.



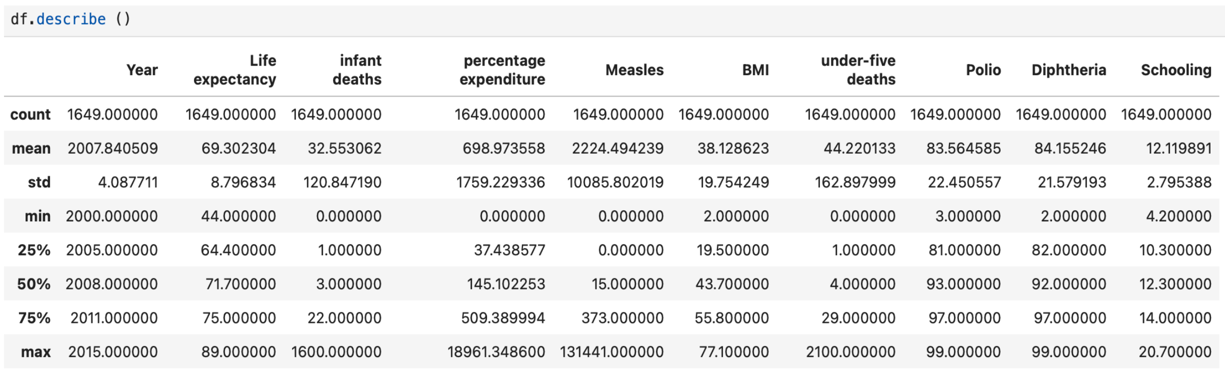
```
df = pd.read_csv('Life_Expectancy_Data.csv')
df
```

	Country	Year	Status	Life expectancy	infant deaths	percentage expenditure	Measles	BMI	under-five deaths	Polio	Diphtheria	Schooling
0	Afghanistan	2015	Developing	65.0	62	71.279624	1154	19.1	83	6	65	10.1
1	Afghanistan	2014	Developing	59.9	64	73.523582	492	18.6	86	58	62	10.0
2	Afghanistan	2013	Developing	59.9	66	73.219243	430	18.1	89	62	64	9.9
3	Afghanistan	2012	Developing	59.5	69	78.184215	2787	17.6	93	67	67	9.8
4	Afghanistan	2011	Developing	59.2	71	7.097109	3013	17.2	97	68	68	9.5
...	...	...	...	...	...	...	...	...	...	...	...	...
1644	Zimbabwe	2004	Developing	44.3	27	0.000000	31	27.1	42	67	65	9.2
1645	Zimbabwe	2003	Developing	44.5	26	0.000000	998	26.7	41	7	68	9.5
1646	Zimbabwe	2002	Developing	44.8	25	0.000000	304	26.3	40	73	71	10.0
1647	Zimbabwe	2001	Developing	45.3	25	0.000000	529	25.9	39	76	75	9.8
1648	Zimbabwe	2000	Developing	46.0	24	0.000000	1483	25.5	39	78	78	9.8

1649 rows × 12 columns

```
# Categorical data = Country, Status
# Numerical data = Year, Life expectancy, infant deaths, percentage expenditure, Measles, BMI, under-five deaths, Polio, Diphtheria, Schooling
```

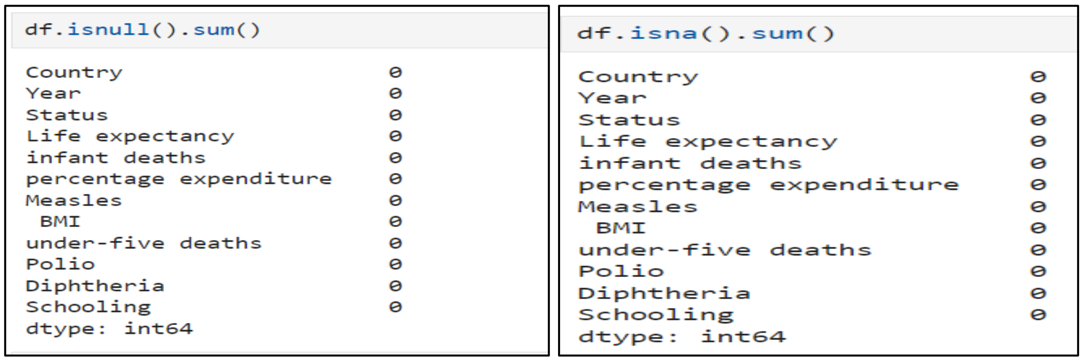
Fig. 3. Data frame



```
df.describe ()
```

	Year	Life expectancy	infant deaths	percentage expenditure	Measles	BMI	under-five deaths	Polio	Diphtheria	Schooling
count	1649.000000	1649.000000	1649.000000	1649.000000	1649.000000	1649.000000	1649.000000	1649.000000	1649.000000	1649.000000
mean	2007.840509	69.302304	32.553062	698.973558	2224.494239	38.128623	44.220133	83.564585	84.155246	12.119891
std	4.087711	8.796834	120.847190	1759.229336	10085.802019	19.754249	162.897999	22.450557	21.579193	2.795388
min	2000.000000	44.000000	0.000000	0.000000	0.000000	2.000000	0.000000	3.000000	2.000000	4.200000
25%	2005.000000	64.400000	1.000000	37.438577	0.000000	19.500000	1.000000	81.000000	82.000000	10.300000
50%	2008.000000	71.700000	3.000000	145.102253	15.000000	43.700000	4.000000	93.000000	92.000000	12.300000
75%	2011.000000	75.000000	22.000000	509.389994	373.000000	55.800000	29.000000	97.000000	97.000000	14.000000
max	2015.000000	89.000000	1600.000000	18961.348600	131441.000000	77.100000	2100.000000	99.000000	99.000000	20.700000

Fig. 4. Describing the Data frame with statistical values

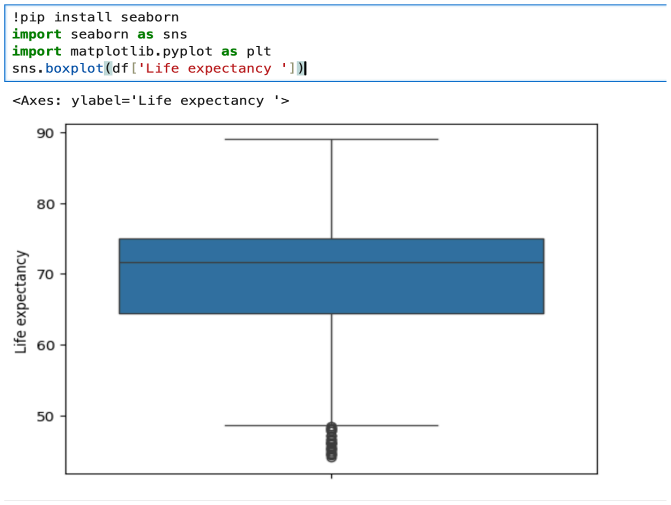


```
df.isnull().sum()

Country                   0
Year                      0
Status                    0
Life expectancy           0
infant deaths             0
percentage expenditure    0
Measles                   0
 BMI                      0
under-five deaths         0
Polio                     0
Diphtheria                0
Schooling                 0
dtype: int64
```

```
df.isna().sum()

Country                   0
Year                      0
Status                    0
Life expectancy           0
infant deaths             0
percentage expenditure    0
Measles                   0
 BMI                      0
under-five deaths         0
Polio                     0
Diphtheria                0
Schooling                 0
dtype: int64
```

Fig. 5. Using Python code to check for null values and na values.

4.3 Outlier analysis using Boxplot

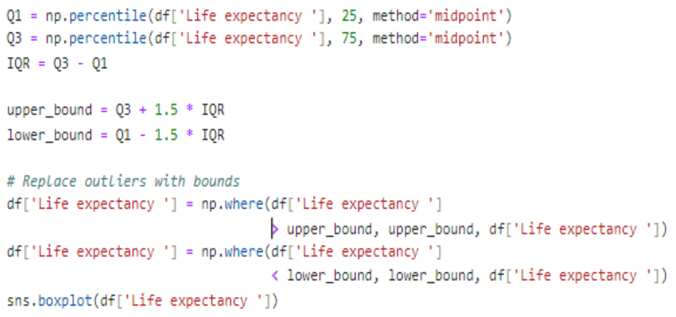


```
!pip install seaborn
import seaborn as sns
import matplotlib.pyplot as plt
sns.boxplot(df['Life expectancy '])
```

```
<Axes: ylabel='Life expectancy '>
```

Fig. 6. Using Python code to check for outliers on life expectancy data.

4.4 Treating outlier using Inter-Quartile Range



```
Q1 = np.percentile(df['Life expectancy '], 25, method='midpoint')
Q3 = np.percentile(df['Life expectancy '], 75, method='midpoint')
IQR = Q3 - Q1

upper_bound = Q3 + 1.5 * IQR
lower_bound = Q1 - 1.5 * IQR

# Replace outliers with bounds
df['Life expectancy '] = np.where(df['Life expectancy ']
                                  > upper_bound, upper_bound, df['Life expectancy '])
df['Life expectancy '] = np.where(df['Life expectancy ']
                                  < lower_bound, lower_bound, df['Life expectancy '])
sns.boxplot(df['Life expectancy '])
```

4.5 Boxplot vizualization after treating with IQR

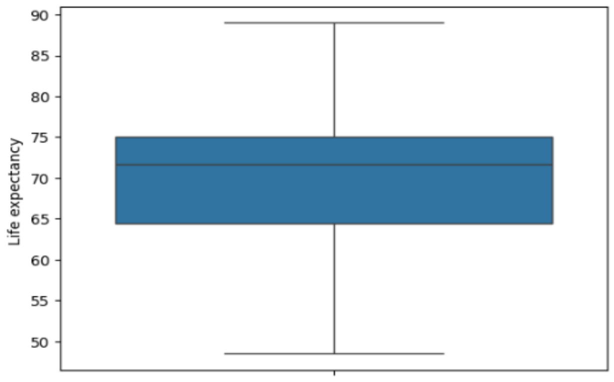


Fig. 7. Boxplot of life expectancy data confirming the treatment of outliers using IQR

5 Data Visualization

After treating the outliers, we worked on creating several visualizations for our data on python by pie chart, graphical representation and histogram. Below are the representations and codes:

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

plt.figure(figsize=(4, 4))
plt.pie(status_counts, labels=status_counts.index,
        autopct='%1.1f%%', startangle=140, colors=['coral', 'lightskyblue'])
plt.title('Distribution of Status in the Dataset')

plt.show()
```

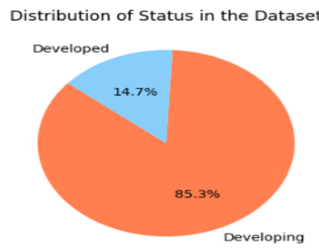


Fig. 8. Python code and Bar chart representation for Distribution of Status in our dataset

```
plt.figure(figsize=(8, 4))
sns.lineplot(x='Year', y='Life expectancy ', data=df)
plt.xlabel('Year')
plt.ylabel('Life Expectancy')
plt.title('Trend of Life Expectancy Over Years')
plt.show()
```

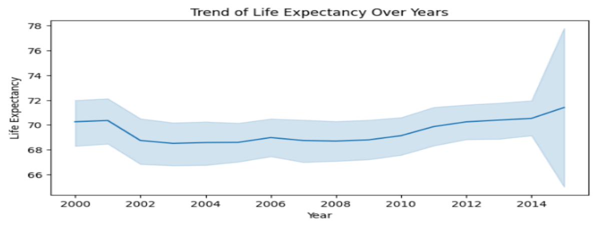


Fig. 9. Python code and graphical representation of the trend of life expectancy over the years

```
from scipy.stats import skew
numeric_cols = df.select_dtypes(include=['int64','float64']).columns
skewness = df[numeric_cols].apply(lambda x: skew(x.dropna()))
skewness = skewness[abs(skewness) > 0.5]  # You can adjust the threshold as needed
print("Numeric variables with significant skewness:")
print(skewness)
num_variables = len(skewness)
num_rows = int(np.ceil(num_variables / 2))
plt.figure(figsize=(15, 5 * num_rows))

for i, col in enumerate(skewness.index):
    plt.subplot(num_rows, 2, i + 1)  # Adjust the number of columns
    sns.histplot(df[col], bins=20, kde=True, color='blue', edgecolor='black')
    plt.title(f'Distribution of {col}')
    plt.xlabel(col)
    plt.ylabel('Density')

plt.tight_layout()
plt.show()
```

```
Numeric variables with significant skewness:
Life expectancy          -0.554532
infant deaths             1.246955
percentage expenditure    1.206865
Measles                   1.190954
under-five deaths         1.234256
Polio                    -1.140948
Diphtheria               -1.108366
dtype: float64
```

Fig. 9. Python code showing the numeric valuables with significant skewness.

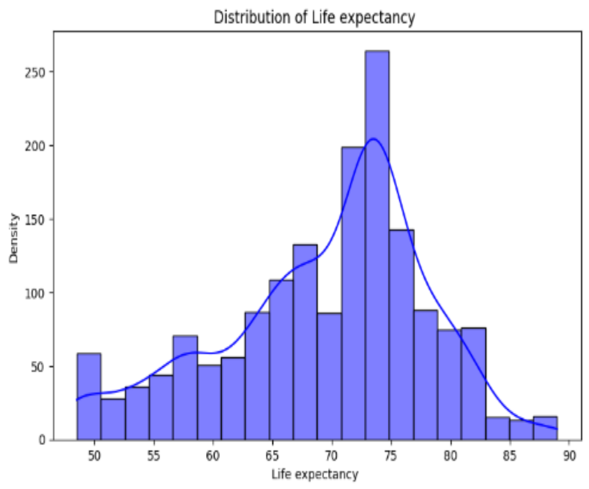
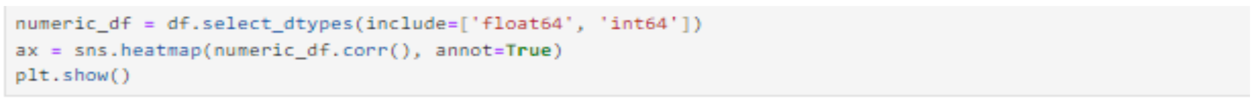
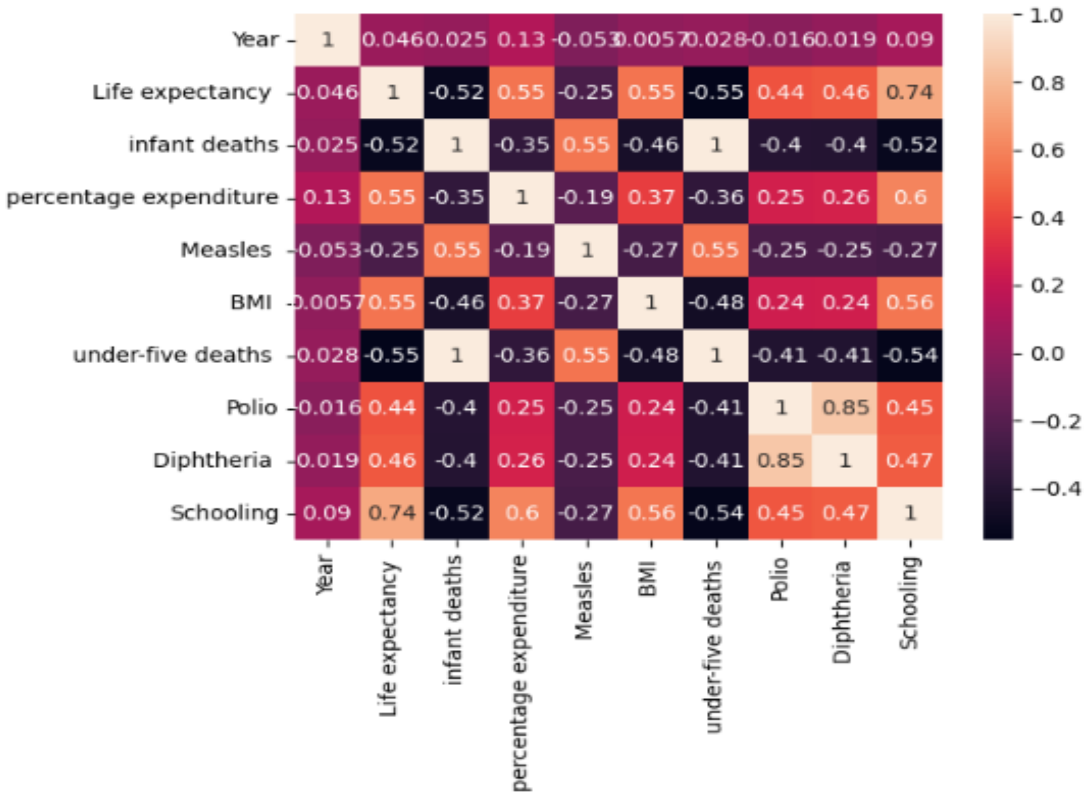


Fig. 10. Histogram representation showing the distribution of life expectancy with tail is towards the left meaning it is negatively skewed.



```
numeric_df = df.select_dtypes(include=['float64', 'int64'])
ax = sns.heatmap(numeric_df.corr(), annot=True)
plt.show()
```



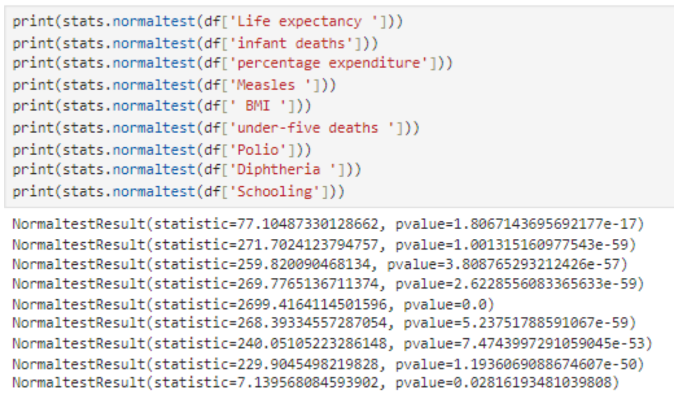
	Year	Life expectancy	infant deaths	percentage expenditure	Measles	BMI	under-five deaths	Polio	Diphtheria	Schooling
Year	1	0.046	0.025	0.13	-0.053	0.0057	0.028	-0.016	0.019	0.09
Life expectancy	0.046	1	-0.52	0.55	-0.25	0.55	-0.55	0.44	0.46	0.74
infant deaths	0.025	-0.52	1	-0.35	0.55	-0.46	1	-0.4	-0.4	-0.52
percentage expenditure	0.13	0.55	-0.35	1	-0.19	0.37	-0.36	0.25	0.26	0.6
Measles	-0.053	-0.25	0.55	-0.19	1	-0.27	0.55	-0.25	-0.25	-0.27
BMI	0.0057	0.55	-0.46	0.37	-0.27	1	-0.48	0.24	0.24	0.56
under-five deaths	0.028	-0.55	1	-0.36	0.55	-0.48	1	-0.41	-0.41	-0.54
Polio	-0.016	0.44	-0.4	0.25	-0.25	0.24	-0.41	1	0.85	0.45
Diphtheria	0.019	0.46	-0.4	0.26	-0.25	0.24	-0.41	0.85	1	0.47
Schooling	0.09	0.74	-0.52	0.6	-0.27	0.56	-0.54	0.45	0.47	1

Fig. 11. Python code and Heatmap showing correlation between attributes and Life expectancy.

5 Data Analysis

5.1 Normality Test

- **Stats.normaltest**

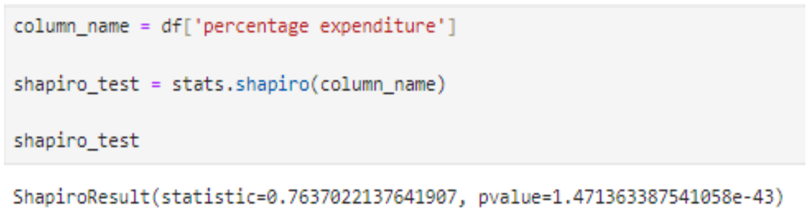


```
print(stats.normaltest(df['Life expectancy ']))
print(stats.normaltest(df['infant deaths']))
print(stats.normaltest(df['percentage expenditure']))
print(stats.normaltest(df['Measles ']))
print(stats.normaltest(df[' BMI ']))
print(stats.normaltest(df['under-five deaths ']))
print(stats.normaltest(df['Polio']))
print(stats.normaltest(df['Diphtheria ']))
print(stats.normaltest(df['Schooling']))
```

```
NormaltestResult(statistic=77.10487330128662, pvalue=1.8067143695692177e-17)
NormaltestResult(statistic=271.7024123794757, pvalue=1.001315160977543e-59)
NormaltestResult(statistic=259.820090468134, pvalue=3.808765293212426e-57)
NormaltestResult(statistic=269.7765136711374, pvalue=2.6228556083365633e-59)
NormaltestResult(statistic=2699.4164114501596, pvalue=0.0)
NormaltestResult(statistic=268.39334557287054, pvalue=5.23751788591067e-59)
NormaltestResult(statistic=240.05105223286148, pvalue=7.4743997291059045e-53)
NormaltestResult(statistic=229.9045498219828, pvalue=1.1936069088674607e-50)
NormaltestResult(statistic=7.139568084593902, pvalue=0.02816193481039808)
```

Fig. 11. Python code for performing normality test between attributes and Life expectancy.

- **Shapiro-Wilk Test**



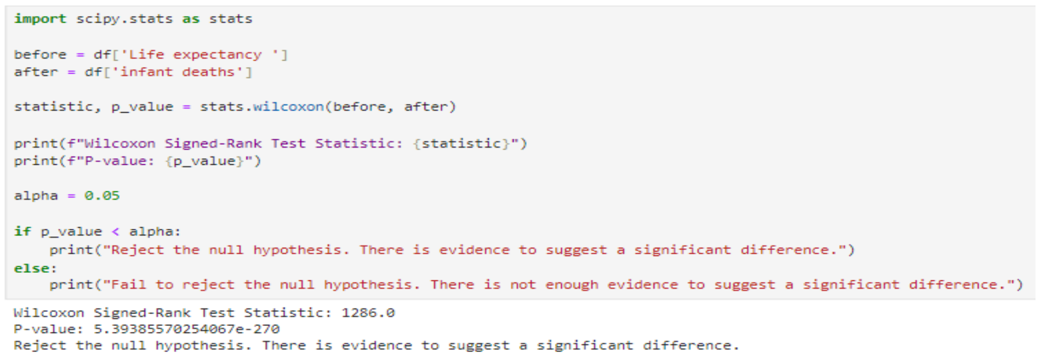
```
column_name = df['percentage expenditure']

shapiro_test = stats.shapiro(column_name)

shapiro_test
```

```
ShapiroResult(statistic=0.7637022137641907, pvalue=1.471363387541058e-43)
```

Fig. 12. Python code for Shapiro wilk test done on percentage expenditure data.

5.2 Statistical Test ((Non-Parametric)

- Wilcoxon signed rank test

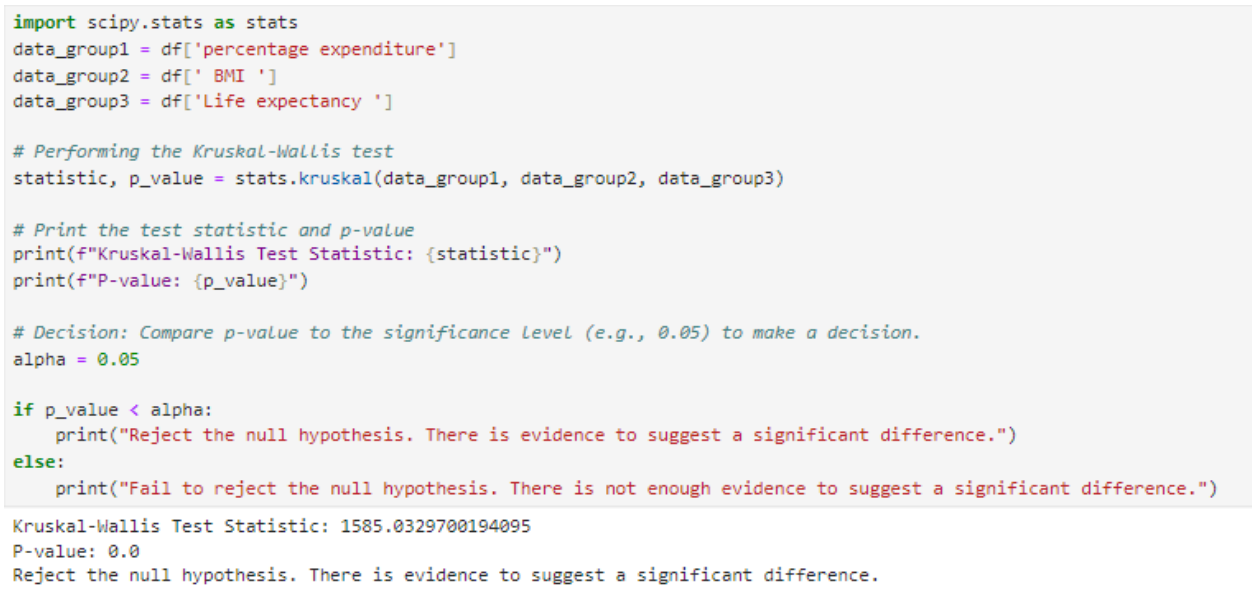


```
import scipy.stats as stats

before = df['Life expectancy ']
after = df['infant deaths']

statistic, p_value = stats.wilcoxon(before, after)

print(f"Wilcoxon Signed-Rank Test Statistic: {statistic}")
print(f"P-value: {p_value}")

alpha = 0.05

if p_value < alpha:
    print("Reject the null hypothesis. There is evidence to suggest a significant difference.")
else:
    print("Fail to reject the null hypothesis. There is not enough evidence to suggest a significant difference.")
```

```
Wilcoxon Signed-Rank Test Statistic: 1286.0
P-value: 5.39385570254067e-270
Reject the null hypothesis. There is evidence to suggest a significant difference.
```

Fig. 12. Python code and result of Wilcoxon signed rank test

- Kruskal-Wallis Test



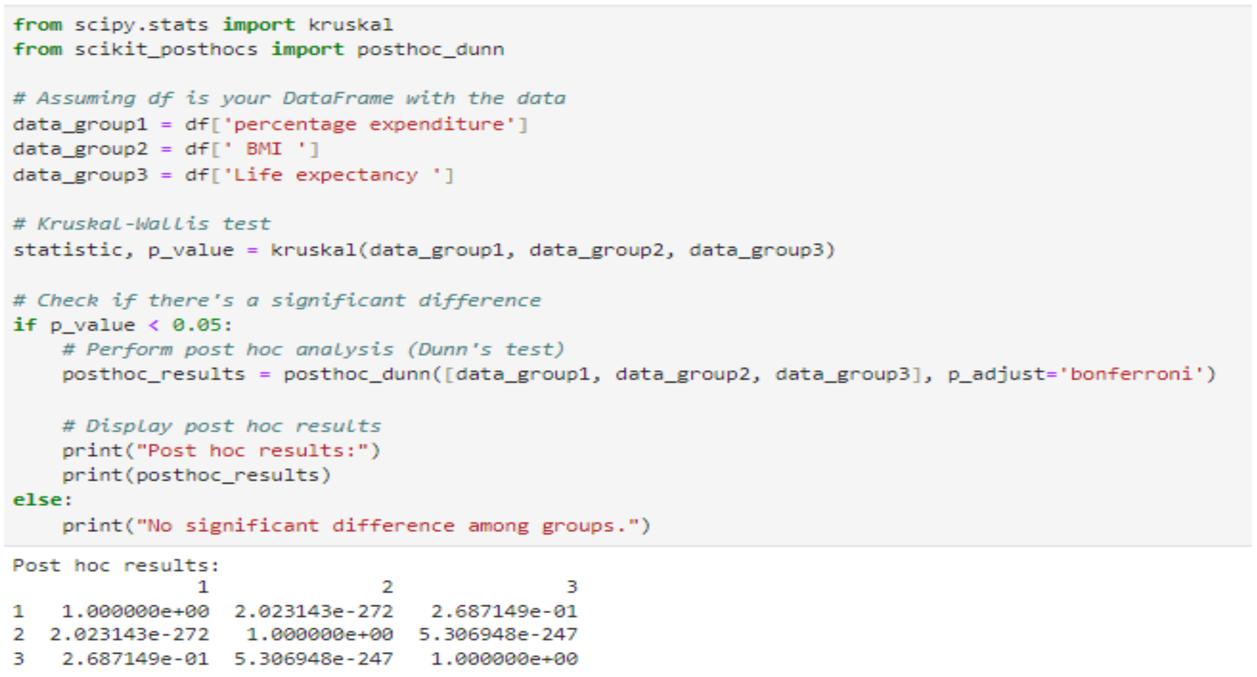
```python
import scipy.stats as stats
data_group1 = df['percentage expenditure']
data_group2 = df[' BMI ']
data_group3 = df['Life expectancy ']

# Performing the Kruskal-Wallis test
statistic, p_value = stats.kruskal(data_group1, data_group2, data_group3)

# Print the test statistic and p-value
print(f"Kruskal-Wallis Test Statistic: {statistic}")
print(f"P-value: {p_value}")

# Decision: Compare p-value to the significance level (e.g., 0.05) to make a decision.
alpha = 0.05

if p_value < alpha:
    print("Reject the null hypothesis. There is evidence to suggest a significant difference.")
else:
    print("Fail to reject the null hypothesis. There is not enough evidence to suggest a significant difference.")
```

```
Kruskal-Wallis Test Statistic: 1585.0329700194095
P-value: 0.0
Reject the null hypothesis. There is evidence to suggest a significant difference.
```

Fig. 13. Python code and result of Kruskal-Wallis test

The Kruskal-Wallis test confirms any statistically significant differences between the medians of three or more independent groups. It is like Mann-Whitney U test (Wilcoxon rank-sum test) which is for comparing two groups.

- Post Hoc Analysis

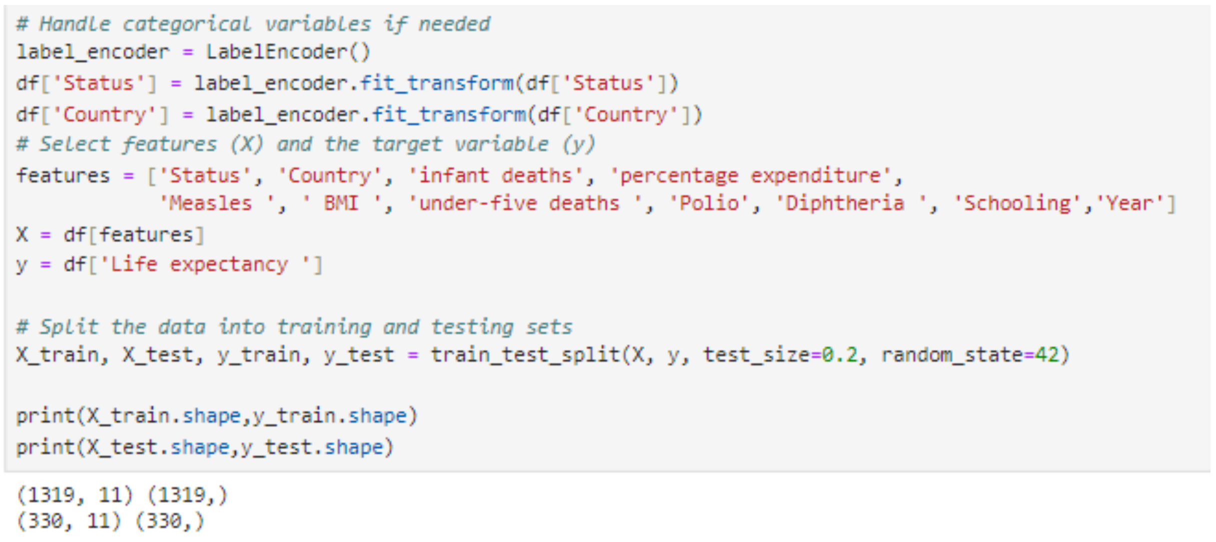


```python
from scipy.stats import kruskal
from scikit_posthocs import posthoc_dunn

# Assuming df is your DataFrame with the data
data_group1 = df['percentage expenditure']
data_group2 = df[' BMI ']
data_group3 = df['Life expectancy ']

# Kruskal-Wallis test
statistic, p_value = kruskal(data_group1, data_group2, data_group3)

# Check if there's a significant difference
if p_value < 0.05:
    # Perform post hoc analysis (Dunn's test)
    posthoc_results = posthoc_dunn([data_group1, data_group2, data_group3], p_adjust='bonferroni')

    # Display post hoc results
    print("Post hoc results:")
    print(posthoc_results)
else:
    print("No significant difference among groups.")
```

```
Post hoc results:
               1              2              3
1   1.000000e+00  2.023143e-272   2.687149e-01
2  2.023143e-272   1.000000e+00  5.306948e-247
3   2.687149e-01  5.306948e-247   1.000000e+00
```

Fig. 14. Python code and result of POST HOC analysis

6.0 Train and Test Split



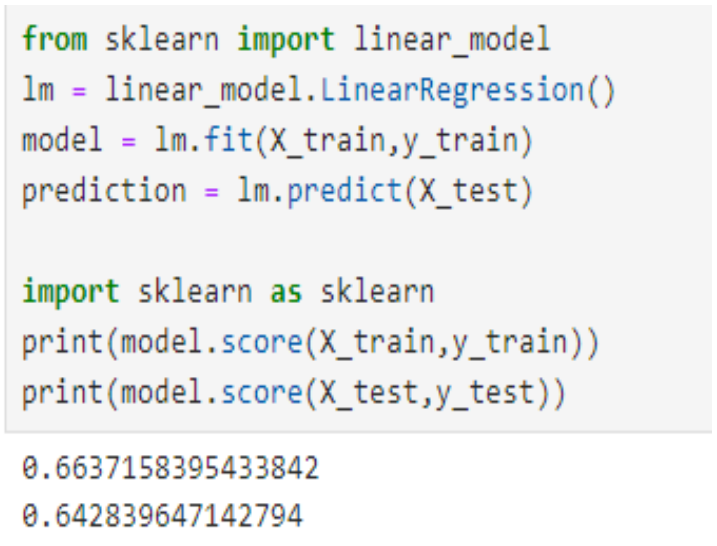
```
# Handle categorical variables if needed
label_encoder = LabelEncoder()
df['Status'] = label_encoder.fit_transform(df['Status'])
df['Country'] = label_encoder.fit_transform(df['Country'])
# Select features (X) and the target variable (y)
features = ['Status', 'Country', 'infant deaths', 'percentage expenditure',
            'Measles ', ' BMI ', 'under-five deaths ', 'Polio', 'Diphtheria ', 'Schooling','Year']
X = df[features]
y = df['Life expectancy ']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

print(X_train.shape,y_train.shape)
print(X_test.shape,y_test.shape)
```

```
(1319, 11) (1319,)
(330, 11) (330,)
```

Fig. 15. Python code and result of the data being split into 80% and 20% train test split.

7.0 Developing A Model

- **Linear Regression** : This model shows linear relationship between features and target, it predicts numerical outcomes, interprets feature impact linearly.

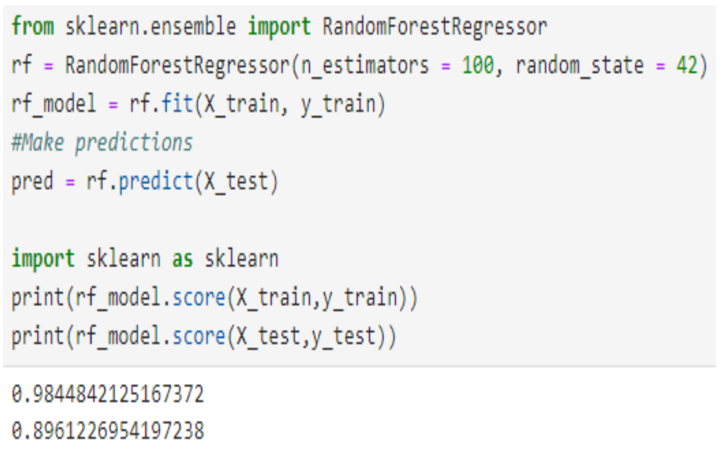


```
from sklearn import linear_model
lm = linear_model.LinearRegression()
model = lm.fit(X_train,y_train)
prediction = lm.predict(X_test)

import sklearn as sklearn
print(model.score(X_train,y_train))
print(model.score(X_test,y_test))
```

```
0.6637158395433842
0.642839647142794
```

Fig. 16. Python code and result of Linear regression model stating that the model exhibits promising performance as the training and test results demonstrate close alignment with an accuracy rate of 64%.

- **Random Forest Regression**: A method using multiple decision trees for improved accuracy, it captures non-linear patterns, effective for large datasets.

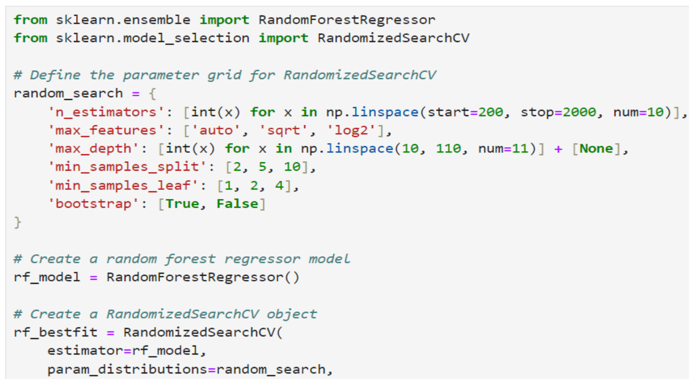


```
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor(n_estimators = 100, random_state = 42)
rf_model = rf.fit(X_train, y_train)
#Make predictions
pred = rf.predict(X_test)

import sklearn as sklearn
print(rf_model.score(X_train,y_train))
print(rf_model.score(X_test,y_test))
```

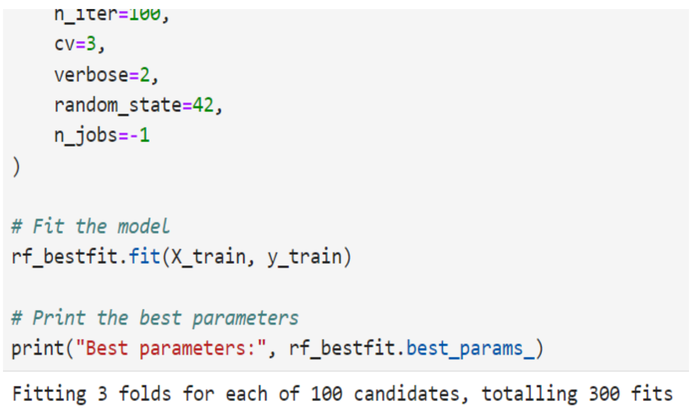
```
0.9844842125167372
0.8961226954197238
```

Fig. 17. Python code and result of Linear regression model stating that the model's training and test result are not as closely related as linear regression with an accuracy rate of 89%.

- Hyperparameter tuning (randomized search) and model building



```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import RandomizedSearchCV

# Define the parameter grid for RandomizedSearchCV
random_search = {
    'n_estimators': [int(x) for x in np.linspace(start=200, stop=2000, num=10)],
    'max_features': ['auto', 'sqrt', 'log2'],
    'max_depth': [int(x) for x in np.linspace(10, 110, num=11)] + [None],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'bootstrap': [True, False]
}

# Create a random forest regressor model
rf_model = RandomForestRegressor()

# Create a RandomizedSearchCV object
rf_bestfit = RandomizedSearchCV(
    estimator=rf_model,
    param_distributions=random_search,
```

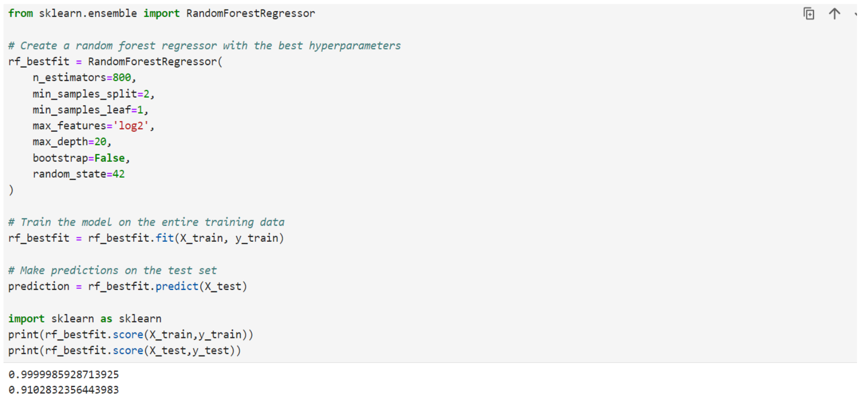


```
    n_iter=100,
    cv=3,
    verbose=2,
    random_state=42,
    n_jobs=-1
)

# Fit the model
rf_bestfit.fit(X_train, y_train)

# Print the best parameters
print("Best parameters:", rf_bestfit.best_params_)
```

```
Fitting 3 folds for each of 100 candidates, totalling 300 fits
```

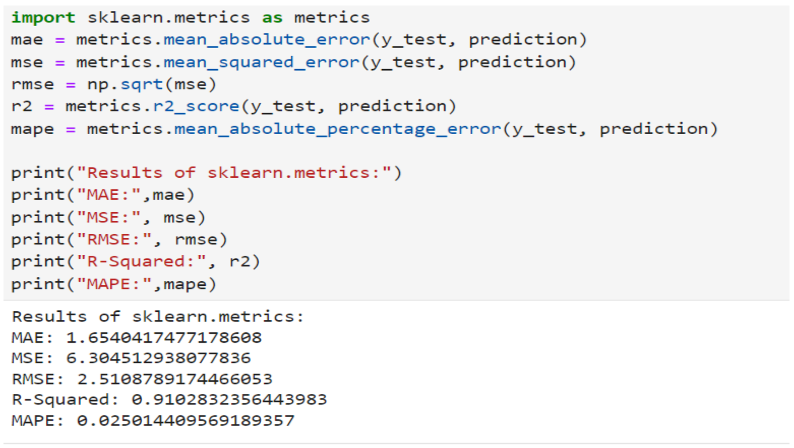
Fig. 18. Python code and result showing performance of hyperparameter tuning by identifying the best parameters for tuning the model. Best parameters are {'n_estimators': 800, 'min_samples_split': 2, 'min_samples_leaf': 1, 'max_features': 'log2', 'max_depth': 20, 'bootstrap': False}

- Random Forest Regression after hyperparameter tuning



```
from sklearn.ensemble import RandomForestRegressor

# Create a random forest regressor with the best hyperparameters
rf_bestfit = RandomForestRegressor(
    n_estimators=800,
    min_samples_split=2,
    min_samples_leaf=1,
    max_features='log2',
    max_depth=20,
    bootstrap=False,
    random_state=42
)

# Train the model on the entire training data
rf_bestfit = rf_bestfit.fit(X_train, y_train)

# Make predictions on the test set
prediction = rf_bestfit.predict(X_test)

import sklearn as sklearn
print(rf_bestfit.score(X_train,y_train))
print(rf_bestfit.score(X_test,y_test))
```

```
0.9999985928713925
0.9102832356443983
```

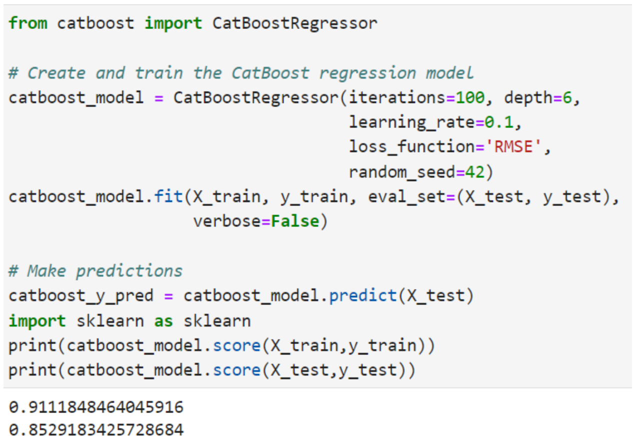


```
import sklearn.metrics as metrics
mae = metrics.mean_absolute_error(y_test, prediction)
mse = metrics.mean_squared_error(y_test, prediction)
rmse = np.sqrt(mse)
r2 = metrics.r2_score(y_test, prediction)
mape = metrics.mean_absolute_percentage_error(y_test, prediction)

print("Results of sklearn.metrics:")
print("MAE:",mae)
print("MSE:", mse)
print("RMSE:", rmse)
print("R-Squared:", r2)
print("MAPE:",mape)
```

```
Results of sklearn.metrics:
MAE:  1.6540417477178608
MSE:  6.304512938077836
RMSE:  2.5108789174466053
R-Squared:  0.9102832356443983
MAPE:  0.025014409569189357
```

Fig. 19. Python code and result stating that after tuning the model, the model's training and test results are now more closely related than before with an accuracy rate of 91%.

- Cat Boost Model



```
from catboost import CatBoostRegressor

# Create and train the CatBoost regression model
catboost_model = CatBoostRegressor(iterations=100, depth=6,
                                   learning_rate=0.1,
                                   loss_function='RMSE',
                                   random_seed=42)
catboost_model.fit(X_train, y_train, eval_set=(X_test, y_test),
                   verbose=False)

# Make predictions
catboost_y_pred = catboost_model.predict(X_test)
import sklearn as sklearn
print(catboost_model.score(X_train,y_train))
print(catboost_model.score(X_test,y_test))
```

```
0.9111848464045916
0.8529183425728684
```

Fig. 20. Python code and result of Cat Boost Model stating that the model's training and test results are not closely related with the accuracy rate being 85%.

8.0 Performance Analysis of the Model

Evaluation metrics	**Linear Regression**	**Random Forest Regressor**	**Tuned Random Forest Regressor**	**Cat Boost**
MAE	**3.8908185963409267**	**1.8476787878787886**	**1.6540417477178608**	**2.404382179289595**
MSE	**25.098119417590308**	**7.299592393939397**	**6.304512938077836**	**10.33561808502055**
RMSE	**5.009802333185443**	**2.7017757852825977**	**2.5108789174466053**	**3.2149056105927203**
R-Squared	**0.642839647142794**	**0.8961226954197238**	**0.9102832356443983**	**0.8529183425728684**
MAPE	**0.05894272333165045**	**0.027624652276114842**	**0.025014409569189357**	**0.0362597686087901**

Fig. 21. Comparison of the evaluation metrics of all three machine learning models.

9.0 Result

- When all three models were compared, a hyperparameter-tuned Random Forest Regressor model performed the best (R squared- value: 91%).
- Based on all the observations, we have rejected the null hypothesis and accepted the alternate hypothesis which states that the attributes being analyzed are associated with life expectancy.

10.0 Limitations

- Inconsistencies in the data
- Creating meaningful features was challenging and incorrect feature engineering can adversely affect the model performance.
- The data was not normally distributed
- We detected outliers and treated them.
- Our models were not performing well due to which we had to perform hyperparameter tuning to increase the model performance

REFRENCES

1. *Life expectancy*. (2023, November 30). Cdc.gov.

https://www.cdc.gov/nchs/nvss/life-expectancy.htm

2. Beckman, K. (2016, May 27). *9 factors that affect longevity*. ThinkAdvisor.

https://www.thinkadvisor.com/2016/05/27/9-factors-that-affect-longevity/