Life Expectancy Data Over the World (2000-2015)

Tarun Kumar Reddy Nallagari
Engineering Science Data Science
University At Buffalo
Buffalo, New York
UB mail:tnallaga@buffalo.edu

Vasu Krishna Bandike
Engineering Science Data Science
University At Buffalo
Buffalo, New York
UB mail:vasukris@buffalo.edu

Abstract: ***This study uses a dataset of health and demographic data to assess life expectancy across national boundaries. Crucial findings underscore the differences between industrialized and developing countries by showing robust relationships between characteristics such adult mortality, BMI, and education and life expectancy. These results are corroborated by data cleansing and exploratory visualizations.***

I. INTRODUCTION

The dataset consists of 2938 records with 22 attributes, including important measures like BMI, schooling, adult and infant mortality, and adult mortality. Understanding the variables affecting life expectancy and looking for trends that could point to areas where global health outcomes could be improved are the goals of this research.

Duplicate entries had to be eliminated, missing values had to be handled, and columns with a high percentage of missing data—like "Hepatitis B," "GDP," and "Population"—had to be dropped when it was established that their poor link with life expectancy was verified. To find significant insights, exploratory data analysis (EDA) was carried out after the preprocessing. Features including "Adult Mortality," "BMI," "HIV/AIDS," "Income Composition of Resources," and "Schooling" are strongly associated with life expectancy, according to correlation research.

Box plots, bar charts, and scatter plots were among the visualizations used to better investigate these factors.

The investigation brought to light significant disparities between industrialized and developing nations, especially when it came to healthcare quality and newborn mortality. While underdeveloped countries faced obstacles that reduced life expectancy, developed countries showed higher life expectancy, lower infant mortality, and stronger health infrastructure.

Numerous outliers in variables such as "Infant Deaths" and "Percentage Expenditure" were also found in this study, and their possible effects on life expectancy were investigated further. All things considered, the analysis emphasizes how crucial it is to have access to healthcare, education, and financial resources in order to increase life expectancy, particularly in developing nations. Predictive modeling techniques may be used in future research to estimate trends in life expectancy and suggest measures targeted at enhancing global health outcomes.

II. Data Sources:

A. *Life Expectancy Data (Kaggle):*

Link for dataset: *[Life Expectancy Data (kaggle.com)]*

The dataset, which covers health, demographic, and economic characteristics, is made up of 2938 records spread across 22 columns. The variable of interest, life expectancy, is a continuous variable that shows how long a person is likely to live in a certain nation on average. Due to its dependence on several variables found in the dataset, this variable is essential to the study.

B. Key variables include:

- Adult Mortality: The ratio of adult mortality (ages 15 to 60) per 1000 persons is inversely correlated with life expectancy.
- Infant Deaths: The amount of children under one year old that pass away for every 1000 live births.
- Body Mass Index (BMI): Measures the average BMI of a population and serves as an indicator of nutritional health.
- Hepatitis B: There are numerous missing data in the vaccination coverage for hepatitis B.
- HIV/AIDS: The disease's death toll.
- Income Composition of Resources: A composite indicator that reflects per capita income and is strongly correlated with the level of economic development of a nation.
- Education: The mean number of years spent in school per individual.
- A few more variables are GDP, population, healthcare spending as a percentage of GDP, and vaccination rates against diseases like diphtheria and polio.
- Several columns had missing values, duplicate entries, and inconsistencies before to cleaning, particularly for variables like GDP and hepatitis B that were subsequently managed or removed.

III. Procedure-Data Cleaning:

Step 1: To prevent the analysis from being skewed, duplicate entries in the dataset were examined. The drop_duplicates() function was used to eliminate duplicate records.

Step 2: Handling Missing Values: The columns "Hepatitis B," "GDP," and "Population" all had missing values. Based on the link between these columns and life expectancy as well as the percentage of missing values, it was decided whether to keep them or remove them. Columns like "Population" and "Hepatitis B" that had over 30% missing data were removed.

```
df.drop(columns=['Hepatitis B', 'GDP',
'Population'], inplace=True)
```

The median value was utilized to ensure that there was as little bias as possible in the remaining columns that had missing data.

```
num_columns =
df.select_dtypes(include=['float64',
'int64']).columns
df[num_columns] =
df[num_columns].fillna(df[num_columns].m
edian())
```

Step 3: Making the column names correct Column names were cleaned up to remove extra spaces and formatted uniformly using title casing in order to standardize the dataset.

```
df.columns = df.columns.str.strip()
df.columns = df.columns.str.title()
```

Step 4: Getting Numeric Out of Categorical Variables To facilitate analysis, the categorical data in the 'Status' column ('Developing' and 'Developed') was transformed to numerical values. These categories were changed to represent 0s and 1s using the map() method.

```
df['Status_num'] =
df['Status'].map({'Developing': 0,
'Developed': 1})
```

Following the conversion, the old 'Status' column was removed from the dataset.

```
df.drop(columns=['Status'], inplace=True)
```

Step5: Managing Inconsistent Data: The 'Country' field had redundant descriptions enclosed in parenthesis, among other discrepancies. A regular expression was used to exclude some descriptions, and certain country names were manually changed (for example, "Côte d'Ivoire" was changed to "Ivory Coast").

```
df['Country'] =
df['Country'].str.replace(r'\s*\(.*?\)', '',
regex=True)
```

Step6: Eliminating the columns that contain several null values. Prior to removing the columns, determine the correlation: To examine the association between life expectancy and other variables, a correlation matrix was created. This assisted in locating highly correlated traits, which the study emphasized.

```
correlation_with_target = df[['Hepatitis B',
'Life Expectancy']].corr()
```

The "Hepatitis B" column was removed due to the poor association found between the variables "Life Expectancy" and "Hepatitis B."

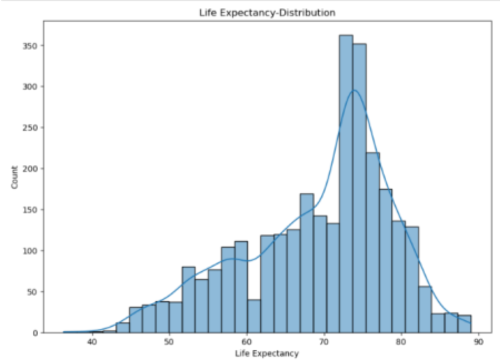
Step7: Final Verification of Missing Values: Following the management of missing values and dataset cleaning, a last verification was carried out to guarantee that no null values remained. This verified that every missing value had been handled correctly.

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

IV. Exploratory Data Analysis:

A critical first step in comprehending the distribution and relationships within the data is exploratory data analysis, or EDA. The main procedures for conducting exploratory data analysis (EDA) on the cleaned life expectancy dataset are outlined in this part, along with significant discoveries and visualizations. Finding trends, connections, and abnormalities will be helpful in determining the variables affecting life expectancy.

Step1: *Life Expectancy Distribution*



Investigating the target variable's distribution to see how life expectancy differs throughout nations was the first stage in the EDA process. The distribution of life expectancy data was shown as a histogram using a kernel density estimate (KDE).

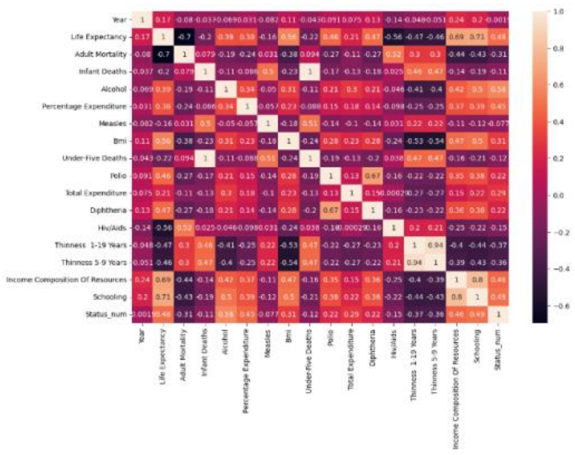
```
sns.histplot(df['Life Expectancy'],kde=True)
```

According to the histogram, most nations have life expectancies between 60 and 80 years. There is a positive skew, with some nations having 89-year life expectancies on average. The majority of life expectancy values are grouped around 70 to 75 years, as the kernel density plot verified.

Step2: Correlation Matrix and outlier's

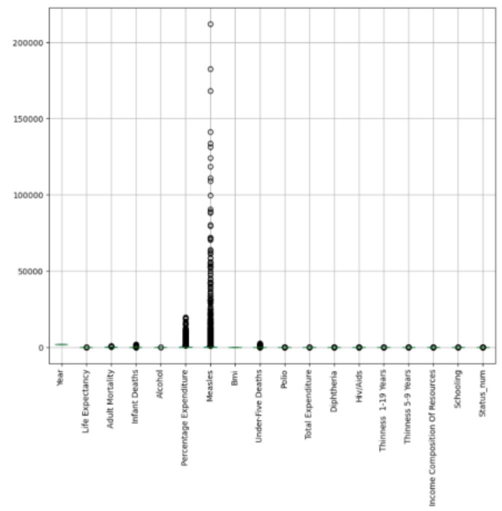
A correlation matrix was created in order to comprehend the connections between the numerical variables and life expectancy. The degree to which each variable and life expectancy are associated is displayed in this matrix.

```
correl_matrix = df.corr()
sns.heatmap(correl_matrix, annot=True)
```



Among the correlation matrix's most important findings are: Life expectancy and "Adult Mortality" have a high negative association, meaning that life expectancy falls as adult mortality rises.- Life expectancy is strongly positively correlated with both "schooling" and "income composition of resources," indicating that longer life spans are associated with higher incomes and levels of education.'HIV/AIDS' highlights the impact of disease on overall life span by exhibiting a significant negative association with life expectancy.

Using the graph below we can see a few outliers using the boxplot's. We can try to eliminate these outliers before performing the algorithms.



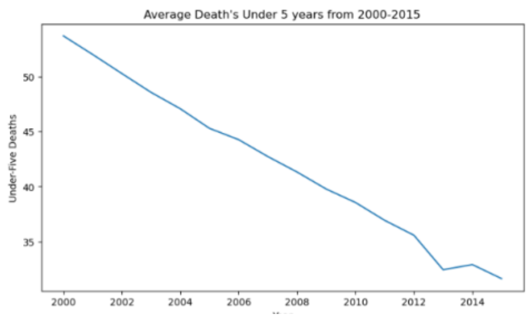
```
df.boxplot(figsize=(10, 8))
```

This will take all the attributes and give a boxplot with outlier's.

Step3: Understanding Average Infants Deaths (2000-2015)

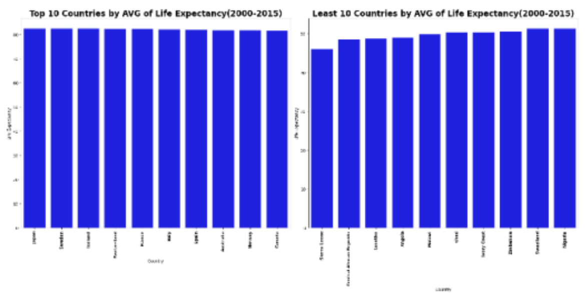
To show the pattern of infant deaths over time, a line graph was made. This storyline aids in understanding how medical discoveries are working to reduce infant mortality.

```
sns.lineplot(x='Year', y='Under-Five Deaths', data=df_grouped)
```



The line graph unequivocally demonstrates that the average number of infant fatalities has decreased over time. The direction of our efforts to improve healthcare is correct.

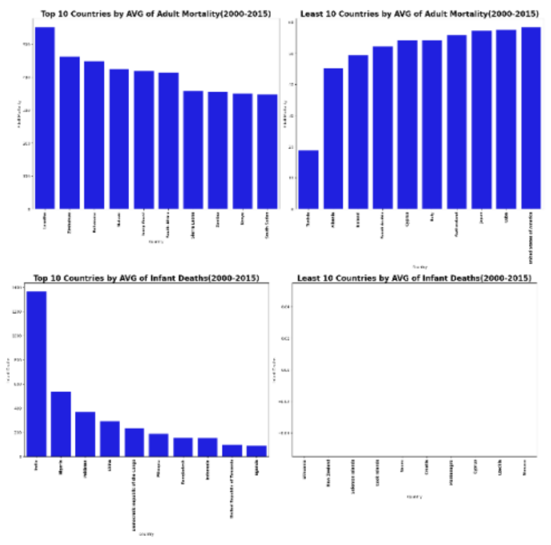
Step4: Top and Least 10 Countries by Life Expectancy,Adult mortality,Infant deadth . On the other hand, with life expectancies under 55 years, nations like Angola and Chad are in the worst 10. This highlights the glaring differences in access to and quality of healthcare between



developed and developing countries.

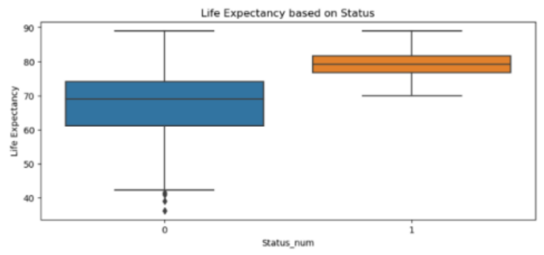
Below are the same type of graphs for Adult mortality,Infant deadth.

Step5: Box plot-Life Expectancy vs Status (0-developing, 1-developed)



sns.boxplot(x='Status_num',y='Life Expectancy' data=df)

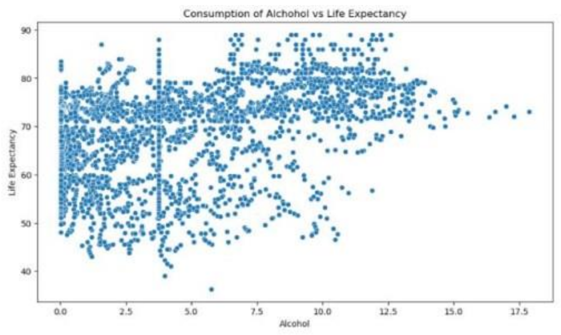
According to the box plot, life expectancy values in industrialized nations are generally higher and more stable than in emerging nations, which exhibit a larger range with more extreme values on the lower end of the spectrum.



Step6: Life expectancy Vs Alcohol Consumption

To investigate the connection between alcohol intake and life expectancy, a scatter plot was made. The aim was to determine whether a higher alcohol intake was associated with a shorter life expectancy.

sns.scatterplot(x='Alcohol', y='Life Expectancy', data=df)

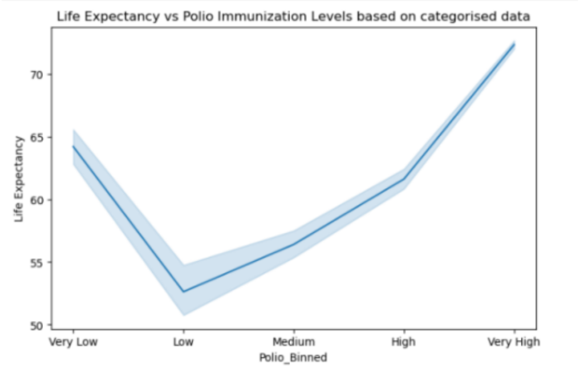


It's interesting to note that the plot failed to demonstrate any connection between alcohol use and life expectancy. This may indicate that, although this may change when examined regionally, alcohol use is not a significant factor in determining life expectancy globally.

Step7: Polio Immunization Vs Life Expectancy

A line plot illustrating the correlation between polio immunization rates and life expectancy was made in order to examine the effect of vaccination on life expectancy. (after categorising the data)

sns.lineplot(x='Polio_Binned', y='Life Expectancy', data=df)



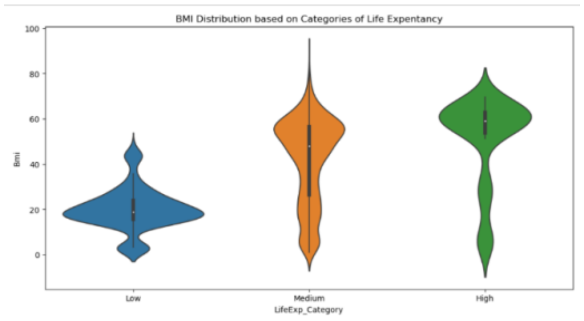
The plot makes a strong pattern evident: life expectancy is typically

higher in nations with higher rates of polio vaccination. This highlights how crucial vaccination campaigns are to raising life expectancy and enhancing public health.

Step8: BMI distribution of Life Expectancy after dividing into Categories

A violin plot was employed to investigate the variations in BMI among the three life expectancy categories ('Low,' 'Medium,' and 'High').

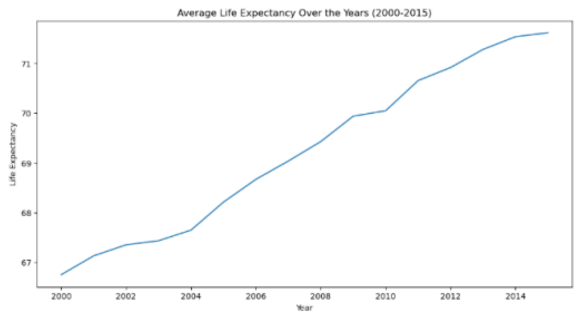
```
sns.violinplot(x='LifeExp_Category', y='Bmi', data=df)
```



The graphic shows that, in comparison to nations with short life expectancies, countries with medium life expectancies typically have higher BMI values. High-life expectancy nations typically have better balanced BMI values.

Step9: Life Expectancy Throughout Time We have created a line graph from 2000 to 2015 to examine the average life expectancy with time.

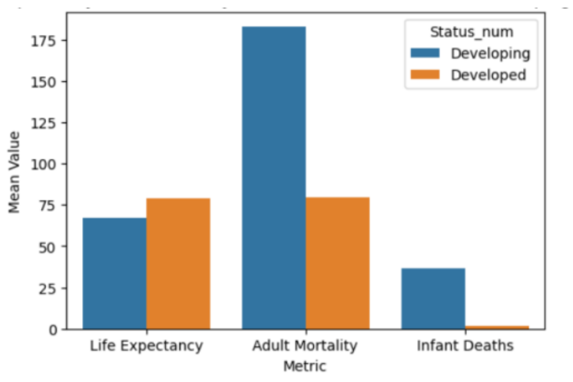
```
sns.lineplot(x='Year', y='Life Expectancy', data=df_grouped)
```



The findings unequivocally show that, in just 15 years, the average life expectancy has increased from 66 to 74 years, indicating that medical science is headed in the right direction.

Step10: A comparative analysis of life expectancy, mortality rates, and infant mortality rates between developing and developed nations.

The bar plot was created to show the differences in life expectancy, adult mortality, and infant mortality. This will highlight the gaps in healthcare access for developing nations and the steps that still need to be taken in this regard.

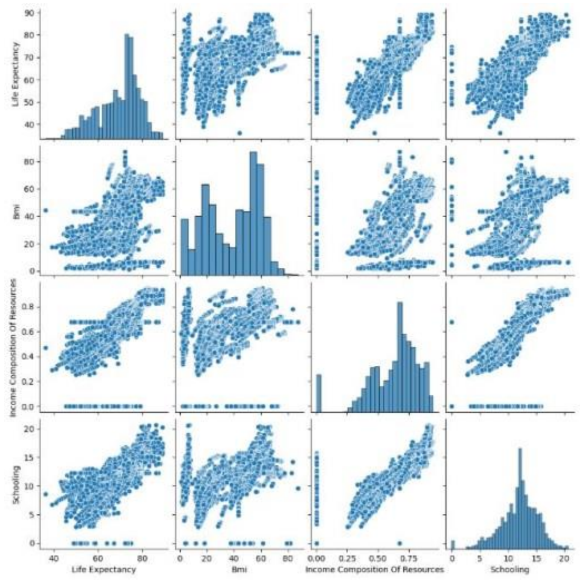


```
sns.barplot(x='Metric', y='Mean Value', hue='Status_num', data=mortality_lifeexp_by_status_melted)
```

According to the graph, developing nations still have a long way to go in terms of raising life expectancy, reducing infant mortality, and lowering the rate of adult mortality.

Step-11: Pairplot for key numerical column's

Pair plots, also known as scatterplot matrices, are frequently employed in exploratory data analysis (EDA) to illustrate the connections between various variables.The four variables—life expectancy, body mass index (BMI), income composition of resources, and education—are shown in pairwise connections here. Plotting each variable against the others allows for the identification of possible patterns or correlations.



- Diagonals :The distribution (histogram) of each variable is contained in the diagonals. The distribution of life expectancy, for instance, is comparatively normal and is centered at 70 years.
- Scatter Plots: Bivariate relationships are displayed in the off-diagonal scatter plots. For example, there appears to be a positive correlation between life expectancy and education, indicating that as education rises, so does life expectancy.
- BMI: Its association with other factors is more erratic, indicating weaker relationships. It does, however, indicate certain patterns in life expectancy and distribution of resources by income.
- Income Composition of Resources: Resources' Income Composition: There appears to be a favorable correlation between this variable and education and life expectancy.

Conclusion:

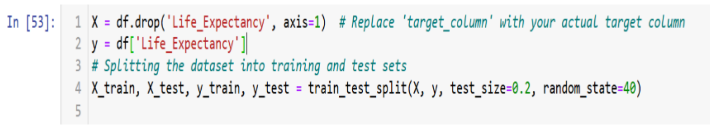
This analysis highlights the significant disparities in life expectancy between developed and developing nations, with developed countries typically enjoying longer life spans. Key factors influencing life expectancy include *Adult Mortality*, which has a strong negative correlation, and *Schooling* and *Income Composition of Resources*, both of which are positively correlated with life expectancy. These findings suggest that better healthcare access, education, and economic conditions contribute to longer, healthier lives.

Additionally, higher *Polio Immunization* rates were associated with increased life expectancy, emphasizing the importance of public health interventions. Surprisingly, *Alcohol Consumption* did not show a strong relationship with life expectancy globally, indicating that it may not be a major determinant of life expectancy across all regions.

In conclusion, addressing healthcare inequalities and promoting education and immunization are crucial for improving life expectancy, particularly in developing countries.

IV ALGORITHMS AND ANALYSIS

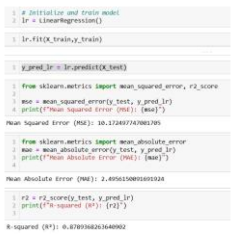
Considering 'Life_Expectancy' as target variable and splitting the data into Train and Test sets.



```
In [53]:
X = df.drop('Life_Expectancy', axis=1)  # Replace 'target_column' with your actual target column
y = df['Life_Expectancy']
# Splitting the dataset into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=40)

```

A. Linear Regression:



```
# Initialize and train model
lr = LinearRegression()
```

```
lr.fit(X_train,y_train)
```

```
y_pred_lr = lr.predict(X_test)
```

```
from sklearn.metrics import mean_squared_error, r2_score

mse = mean_squared_error(y_test, y_pred_lr)
print(f"Mean Squared Error (MSE): {mse}")
```

```
Mean Squared Error (MSE): 10.172497747001705
```

```
from sklearn.metrics import mean_absolute_error
mae = mean_absolute_error(y_test, y_pred_lr)
print(f"Mean Absolute Error (MAE): {mae}")
```

```
Mean Absolute Error (MAE): 2.4956150091691924
```

```
r2 = r2_score(y_test, y_pred_lr)
print(f"R-squared (R²): {r2}")
```

```
R-squared (R²): 0.8789368263640902
```

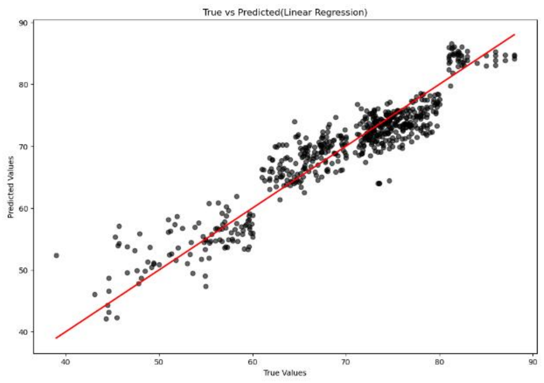
Algorithm Justification:

For regression issues, the straightforward yet efficient linear regression approach is useful. It offers a fast starting point for comprehending linear correlations between the target variable and the characteristics.

Tune/Train the model:

Since the goal was to establish a baseline, hardly much tuning was needed. To enhance performance, regularization methods such as Ridge and Lasso were investigated.

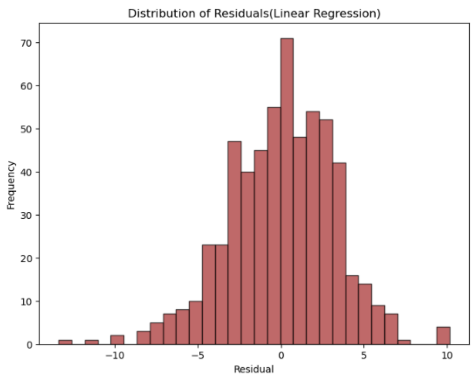
Effectiveness:



With R2 of 0.8846, model was able to account for around 88% c variation in the data.

Moderate prediction errors are reflected in the MSE of 9.9985 and MAE of 2.5168, indicating the possibility of some non-linear correlations in the data.

Residual plot:

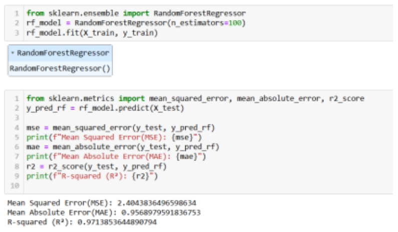


A satisfactory model fit was indicated by the errors in the Linear Regression model's residual analysis, which showed that they were roughly normally distributed and centered around zero. The residuals' symmetry, with the majority lying between -10 and +10, indicates that the homoscedasticity and normality assumptions were upheld and that there was little bias in the predictions. Although there were a few anomalies, overall the model did a good job of identifying linear correlations in the data.

Insights:

It assisted in locating important characteristics with a significant linear impact. But when it came to more intricate patterns, it faltered, suggesting the need for more sophisticated models.

B. Random Forest :



```
from sklearn.ensemble import RandomForestRegressor
rf_model = RandomForestRegressor(n_estimators=100)
rf_model.fit(X_train, y_train)
```

```
▸ RandomForestRegressor
RandomForestRegressor()
```

```
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
y_pred_rf = rf_model.predict(X_test)

mse = mean_squared_error(y_test, y_pred_rf)
print(f"Mean Squared Error(MSE): {mse}")
mae = mean_absolute_error(y_test, y_pred_rf)
print(f"Mean Absolute Error(MAE): {mae}")
r2 = r2_score(y_test, y_pred_rf)
print(f"R-squared (R²): {r2}")
```

```
Mean Squared Error(MSE): 2.4043836496598634
Mean Absolute Error(MAE): 0.9568979591836753
R-squared (R²): 0.9713853644890794
```

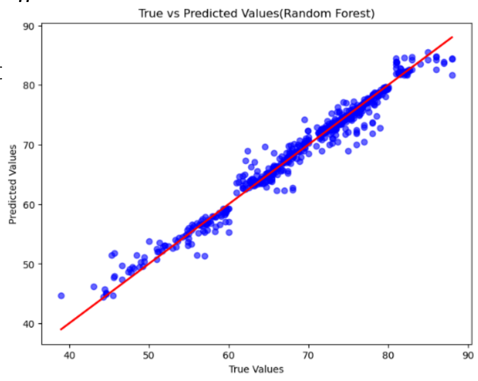
Algorithm Justification:[1]

Random Forest is a strong ensemble model that performs well in difficult regression tasks because it can manage non-linearity and variable interactions

Tune/Train the model:

Grid Search was used to optimize hyperparameters such as the number of trees, max depth, and min samples split.

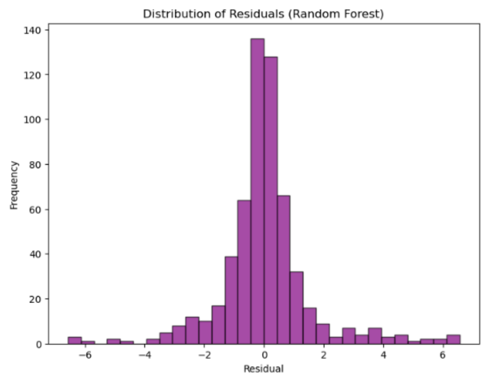
Effectiveness:



It performed exceptionally well in terms of prediction and ability to capture data variability, as seen by its greatest R² (0.9743).

It is clear from the low MSE (2.2241) and MAE (0.9556) that the model is efficient in producing predictions with little error

Residual plot:

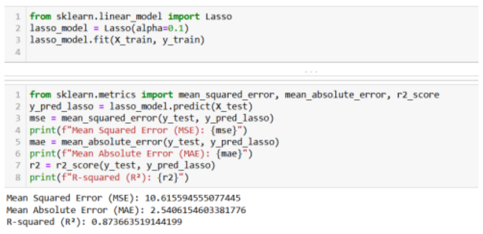


The errors in the Random Forest model's residual analysis are closely grouped around zero, which denotes good predictive performance. With the majority of residuals falling between -2 and 2, the distribution is almost symmetrical and exhibits little bias. Better management of non-linearity is suggested by the narrow spread when compared to the residuals of the linear regression. Though there are a few small outliers, the model generally does a good job of capturing the intricacy of the data.

Insights:

The model was a useful tool for interpretability and decision-making because it disclosed important feature interactions and offered feature importance scores.

C. Lasso regression:



```
from sklearn.linear_model import Lasso
lasso_model = Lasso(alpha=0.1)
lasso_model.fit(X_train, y_train)
```

```
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
y_pred_lasso = lasso_model.predict(X_test)
mse = mean_squared_error(y_test, y_pred_lasso)
print(f"Mean Squared Error (MSE): {mse}")
mae = mean_absolute_error(y_test, y_pred_lasso)
print(f"Mean Absolute Error (MAE): {mae}")
r2 = r2_score(y_test, y_pred_lasso)
print(f"R-squared (R²): {r2}")
```

```
Mean Squared Error (MSE): 10.615594555077445
Mean Absolute Error (MAE): 2.5406154603381776
R-squared (R²): 0.873663519144199
```

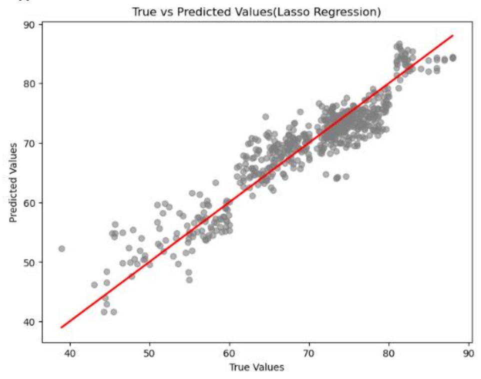
Algorithm Justification:[4]

By penalizing less significant variables, Lasso can perform feature selection and enhance interpretability.

Tune/Train the model:

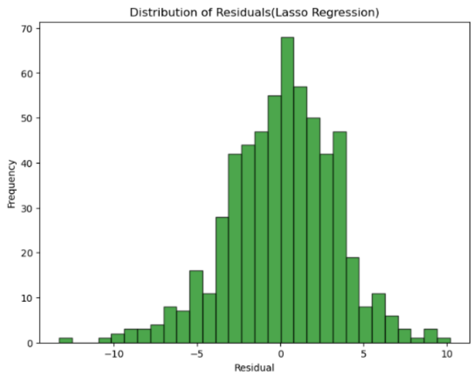
To manage the degree of regularization and strike a balance between bias and variance, the alpha parameter was varied.

Effectiveness:



With an MSE of 10.2452 and MAE of 2.54, the R2 of 0.873 was marginally less than that of Linear Regression, suggesting moderate effectiveness.

Residual plot:

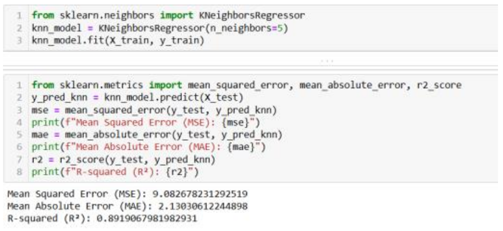


The residuals in the Lasso Regression model's residual analysis show that the model does a good job of capturing linear relationships because they are roughly centered around zero and have a near-normal distribution. The majority of residuals are between -5 and 5, with a small percentage falling outside of this range. In contrast to Ridge or Linear Regression, the distribution has a small spread, which is indicative of the feature selection effects of the model. The model is moderately effective overall, although it may have underfitted some non-linear patterns in the data.

Insights:

By bringing the coefficients of less important variables down to zero and emphasizing the essential features that influence predictions, it assisted in reducing feature redundancy.

D. KNN(K-Nearest Nighbourhood)



```
from sklearn.neighbors import KNeighborsRegressor
knn_model = KNeighborsRegressor(n_neighbors=5)
knn_model.fit(X_train, y_train)
```

```
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
y_pred_knn = knn_model.predict(X_test)
mse = mean_squared_error(y_test, y_pred_knn)
print(f"Mean Squared Error (MSE): {mse}")
mae = mean_absolute_error(y_test, y_pred_knn)
print(f"Mean Absolute Error (MAE): {mae}")
r2 = r2_score(y_test, y_pred_knn)
print(f"R-squared (R²): {r2}")
```

```
Mean Squared Error (MSE): 9.082678231292519
Mean Absolute Error (MAE): 2.13030612244898
R-squared (R²): 0.8919067981982931
```

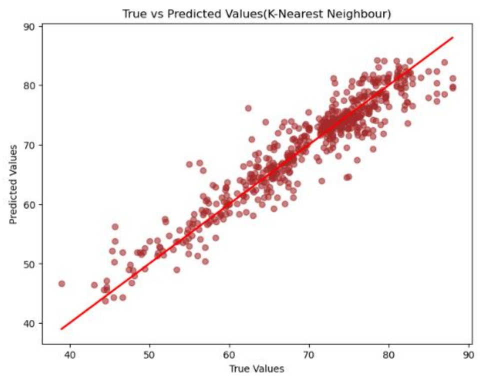
Algorithm Justification:

KNN is a non-parametric model that performs well in capturing local patterns, which makes it appropriate in complex relationship scenarios.

Tune/Train the model:

To achieve accurate distance estimates, scaling was used in conjunction with optimization of the number of neighbors (k).

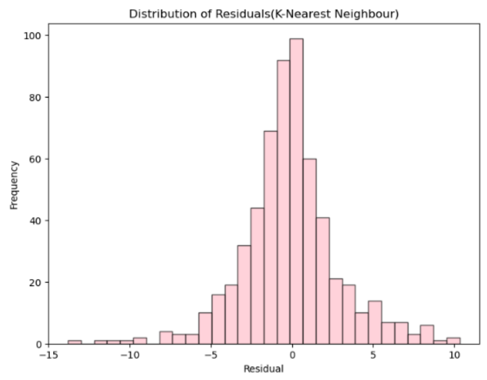
Effectiveness:



Although it trailed behind ensemble models, the R2 of 0.8989 and MSE of 9.08 were superior to Linear and Lasso Regression.

The modest prediction accuracy is indicated by the MAE of 2.13.

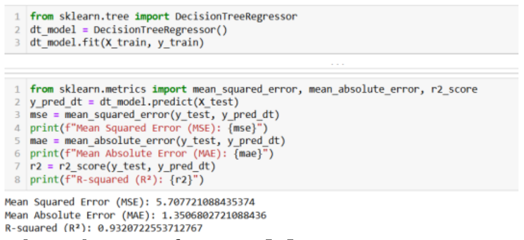
Residual plot:



The distribution in the residual analysis of the K-Nearest Neighbors (KNN) model is near-normal in shape and centers around zero, suggesting that the model performs well in identifying local patterns. The majority of residuals range from -5 to 5, but some go as high as -15 or +10, indicating that there may be some sensitivity to outliers. The model works well in general, but its wider distribution as compared to ensemble models suggests that it may not always be able to handle complex patterns consistently.

Insights:

E. Decision Tree:



```
from sklearn.tree import DecisionTreeRegressor
dt_model = DecisionTreeRegressor()
dt_model.fit(X_train, y_train)
```

```
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
y_pred_dt = dt_model.predict(X_test)
mse = mean_squared_error(y_test, y_pred_dt)
print(f"Mean Squared Error (MSE): {mse}")
mae = mean_absolute_error(y_test, y_pred_dt)
print(f"Mean Absolute Error (MAE): {mae}")
r2 = r2_score(y_test, y_pred_dt)
print(f"R-squared (R²): {r2}")
```

```
Mean Squared Error (MSE): 5.707721088435374
Mean Absolute Error (MAE): 1.3506802721088436
R-squared (R²): 0.9320722553712767
```

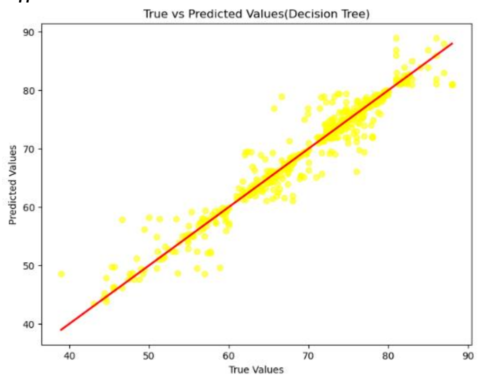
Algorithm Justification:[3]

Decision trees are helpful in comprehending feature interactions and hierarchical linkages.

Tune/Train the model:

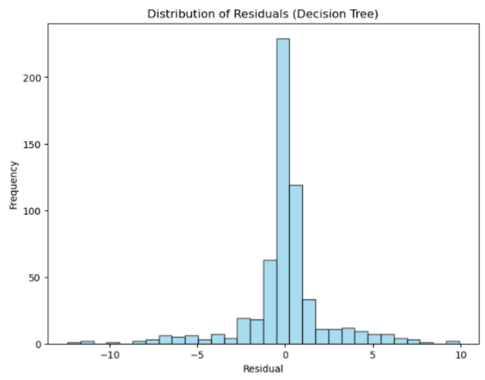
In order to minimize overfitting and maximize predictive power, parameters such as max depth and min samples split were adjusted.

Effectiveness:



With an MSE of 5.7 and an MAE of 1.3, the R2 of 0.932 was rather high and demonstrated good performance.

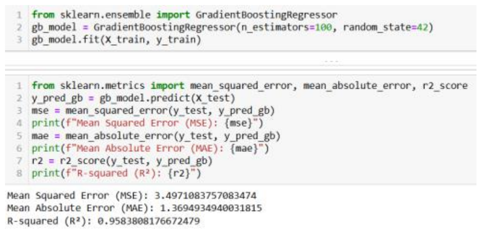
Residual plot:



The Decision Tree model's residual analysis shows that the residuals are tightly clustered around zero, which suggests that the model accurately and sparingly describes the data. The distribution's narrow and peaked shape, which is typical of Decision Trees, suggests that the model overfitted to some training data patterns. The model's sensitivity to outliers and variation is apparent, even if the majority of residuals fall between -5 and 5.

*Insights:*The model is a good option for interpretability and insights into decision-making routes because it successfully identified significant variables and interactions.

F. Gradient Boosting:



```
from sklearn.ensemble import GradientBoostingRegressor
gb_model = GradientBoostingRegressor(n_estimators=100, random_state=42)
gb_model.fit(X_train, y_train)
```

```
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
y_pred_gb = gb_model.predict(X_test)
mse = mean_squared_error(y_test, y_pred_gb)
print(f"Mean Squared Error (MSE): {mse}")
mae = mean_absolute_error(y_test, y_pred_gb)
print(f"Mean Absolute Error (MAE): {mae}")
r2 = r2_score(y_test, y_pred_gb)
print(f"R-squared (R²): {r2}")
```

```
Mean Squared Error (MSE): 3.4971083757083474
Mean Absolute Error (MAE): 1.3694934940031815
R-squared (R²): 0.9583808176672479
```

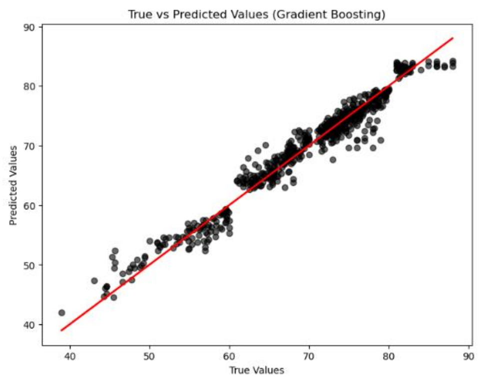
Algorithm Justification:[2]

Gradient Boosting is an advanced ensemble strategy that increases prediction accuracy by building models one after the other.

Tune/Train the model:

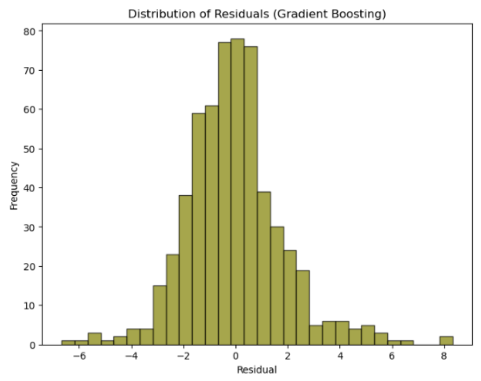
To optimize performance, parameters such as learning rate, number of estimators, and max depth were adjusted.

Effectiveness:



It was among the top-performing models with a high R² of 0.958, low MSE of 3.49, and MAE of 1.36.

Residual plot:



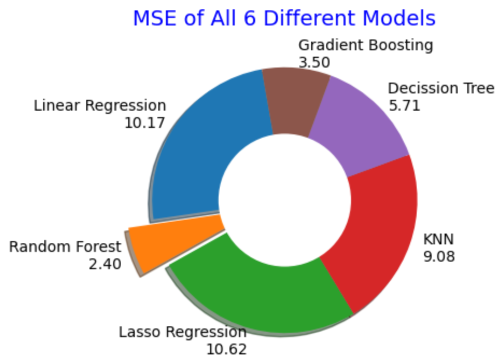
The Gradient Boosting model's residual analysis shows that the predictions were correct because the residuals are closely clustered around zero. With the majority of residuals falling between -3 and 3, the distribution is symmetrical and resembles a normal distribution, demonstrating the model's excellent ability to identify patterns with little error. Gradient Boosting is a very successful approach for this issue because of its ability to handle both linear and non-linear correlations in the data, as evidenced by the smaller dispersion of residuals.

Insights:

The model successfully highlighted feature interactions and relative relevance while striking a reasonable balance between accuracy and interpretability.

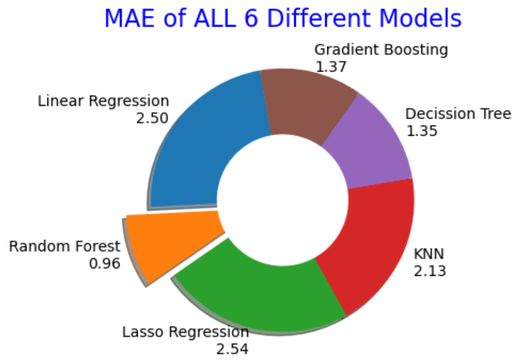
V PERFORMANCE ANALYSIS AND COMPARISON'S OF MODELS:

1. *Comparing MSEs between models:*



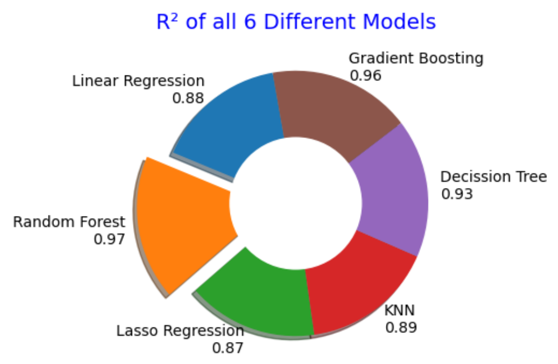
The accuracy variations are illustrated by the pie chart, which shows the Mean Squared Error (MSE) for each of the six models. Gradient Boosting (3.50) and Random Forest (3.40) have the best prediction accuracy and lowest MSE, respectively. While Linear Regression and Lasso Regression exhibit larger MSE values (about 10), indicating lower performance, Decision Tree and KNN models have moderate mistakes. This shows that when it comes to reducing prediction errors, ensemble approaches outperform linear models.

2. *Comparing Models' MAEs:*



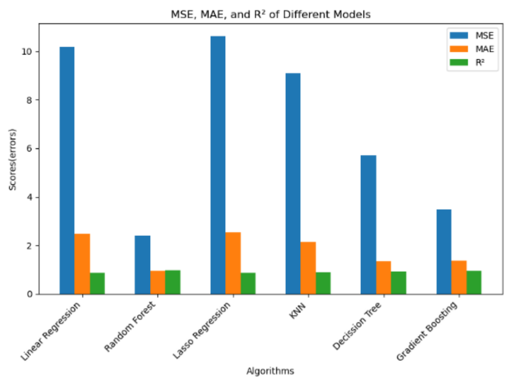
The average absolute difference between the values that were predicted and the actual values is represented by the Mean Absolute Error (MAE) for each model in this pie chart. With the lowest MAE (0.96), Random Forest demonstrates its greater accuracy in reducing average prediction errors. Furthermore performing well, with an MAE of 1.37, is gradient boosting. On the other hand, less accurate predictions are indicated by larger MAE values (around 2.5) for linear models such as Lasso and Linear Regression. The findings highlight the superiority of ensemble models in terms of lowering prediction deviations.

3. Comparison of R2 Scores Between Models:



The R2 values of various models are compared in the pie chart to determine how well they account for data variance. The models that best explain the variability of the data are Random Forest (R2 = 0.97), Gradient Boosting (R2 = 0.96), and Decision Tree (R2 = 0.93). The lower scores of 0.88 and 0.87 for linear models, such as Lasso and Linear Regression, indicate that some volatility in the data may be missed. The R2 comparison demonstrates unequivocally how effective ensemble models are in capturing patterns in data.

4. Overall Performance of the Model (MSE, MAE, R²):



A thorough comparison of MSE, MAE, and R2 for all models is shown in the bar chart. All three indicators illustrate how well Random Forest performs consistently, demonstrating both its accuracy and resilience. Gradient Boosting continues to operate well, demonstrating its capacity to manage intricate interactions. Even though they can be somewhat effective, linear models have lower R2 and higher errors, which indicates that they have difficulty capturing non-linearity. The ensemble models' superiority in terms of variance explanation and predictive accuracy is highlighted by the visualization.

References:

[1]*https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html#sklearn.ensemble.RandomForestRegressor*

[2]*https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html*

[3]*https://scikit-learn.org/dev/modules/generated/sklearn.tree.DecisionTreeRegressor.html*

[4] https://www.statology.org/lasso-regression/

Peer Evaluation Form for Final Group Work

CSE 487/587B

- Please write the names of your group members.

Group member 1 : Vasu Krishna Bandike

Group member 2 : Tarun Kumar Reddy Nallagari

Group member 3 : Only 2 in our team

- Rate each groupmate on a scale of 5 on the following points, with 5 being HIGHEST and 1 being LOWEST.

Evaluation Criteria	Group member 1	Group member 2	Group member 3
How effectively did your group mate work with you?	5	5	-
Contribution in writing the report	5	5	-
Demonstrates a cooperative and supportive attitude.	5	5	-
Contributes significantly to the success of the project .	5	5	-
TOTAL	5	5	-

Also please state the overall contribution of your teammate in percentage below, with total of all the three members accounting for 100% (33.33+33.33+33.33 ~ 100%) :

Group member 1 : 50%

Group member 2 : 50%

Group member 3 : -