Technological Institute of the Philippines	Quezon City - Computer Engineering
Course Code:	CPE 019
Code Title:	Emerging Technologies in CpE 2
2nd Semester	AY 2023-2024

Prelim Examination Name | Buenafe, Dhafny S. **Name** | Serrano, Vincent Gon B. **Section** | CPE32S3 **Date Performed**: |March 1, 2024 **Date Submitted**: |March 6, 2024 **Instructor**: | Engr. Roman Richard

Prelim Examination

Choose any dataset applicable for classification and/or prediction analysis problems.

Show the application of the following algorithms:

Linear Regression:

- Singular LR
- Multiple LR
- Polynomial LR

Logistic Regression

Decision Tree

Random Forest

Provide Evaluation reports for all models

NOTE: Submit the github link that contains all files (pdf report, dataset and python notebooks).

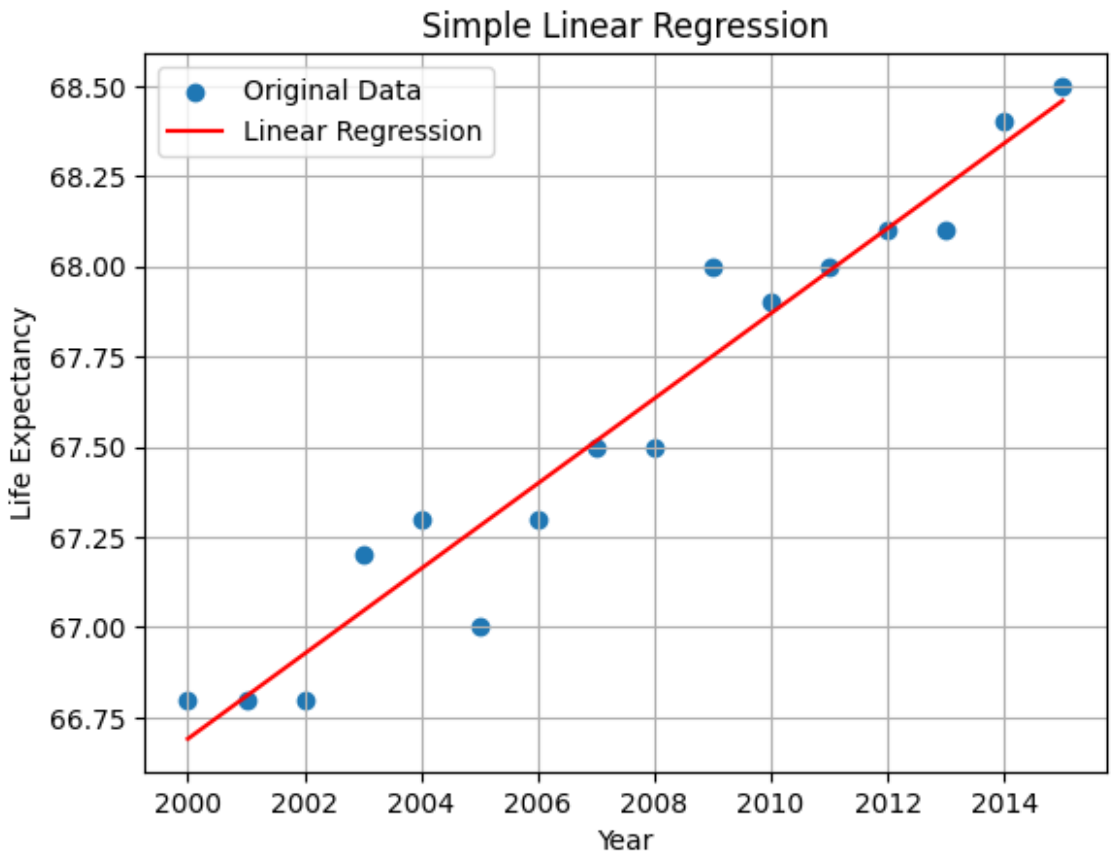
LINEAR REGRESSION

```
# Import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Step 1: Read the CSV file
df = pd.read_csv('/content/Life Expectancy Data.csv')

df_philippines = df[df['Country'] == 'Philippines']

# Display the resulting DataFrame
df_philippines.head(16)
```

{"type":"dataframe","variable_name":"df_philippines"}

1. SINGULAR LINEAR REGRESSION

```
from sklearn.linear_model import LinearRegression

# x = independent, y dependent
X = df_philippines['Year'].values.reshape(-1, 1)
y = df_philippines['Life Expectancy'].values

# fit the LR model
model = LinearRegression()
model.fit(X, y)

# predictions using model
predictions = model.predict(X)

# plotting
plt.scatter(X, y, label='Original Data')
plt.plot(X, predictions, color='red', label='Linear Regression')
plt.xlabel('Year')
plt.ylabel('Life Expectancy')
plt.title('Simple Linear Regression')
plt.legend()
plt.grid()
plt.show()
```



Evaluation Report:

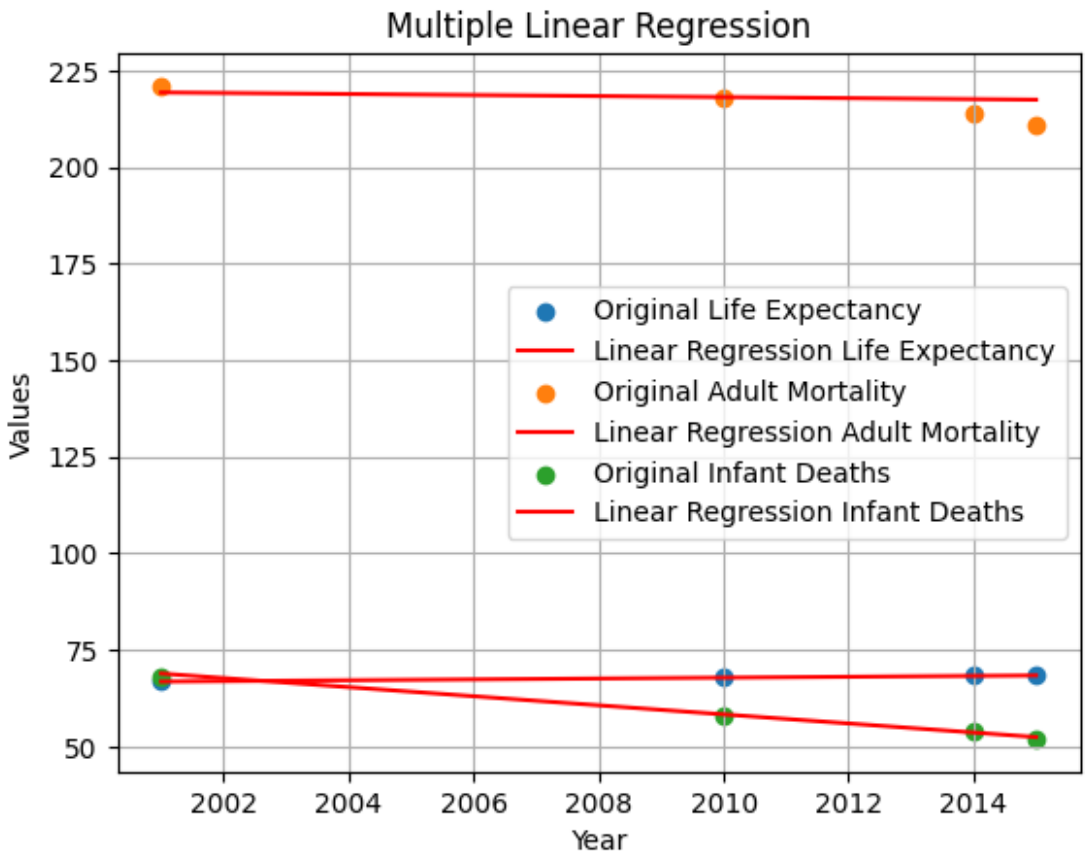
- The dataset that has been utilized in the Linear regression was the Life expectance dataset from the WHO. The singular linear regression specified a dataframe particularly in the Philippines. The dataframe can be seen above for further reference. In preparation for this kind of linear regression, we took year as the independent variable and life expectance column as its dependent variable. From there, the model is trained through the LinearRegression() for the provided variables. Next, the .predict() were used to make predictions for the given data. Afterwards, plotting the original data points (blue dots) along with the linear regression line (red, prediction of values) is necessary to represent linear relationship between the columns Year and Life Expectancy.

2. MULTIPLE LINEAR REGRESSION

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

X = df_philippines[['Year']]
y = df_philippines[['Life Expectancy', 'Adult Mortality', 'Infant
Deaths']]

# test and train sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
```

```
test_size=0.2, random_state=42)

# fit LR
model = LinearRegression()
model.fit(X_train, y_train)

predictions = model.predict(X_test)

# evaluate model by mse
mse = mean_squared_error(y_test, predictions)
print(f'Mean Squared Error: {mse}')

# results
print('Coefficients:', model.coef_)
print('Intercept:', model.intercept_)

# plotting, original data and linear regression of multiple variables
for i, col in enumerate(y.columns):
    plt.scatter(X_test, y_test[col], label=f'Original {col}')
    plt.plot(X_test, predictions[:, i], label=f'Linear Regression
{col}', color='red')

plt.xlabel('Year')
plt.ylabel('Values')
plt.title('Multiple Linear Regression')
plt.legend()
plt.grid(True)
plt.show()
```

```
Mean Squared Error: 4.9107779110028
Coefficients: [[ 0.11299639]
 [-0.13357401]
 [-1.17509025]]
Intercept: [-159.27942238  486.62184116 2420.26444043]
```



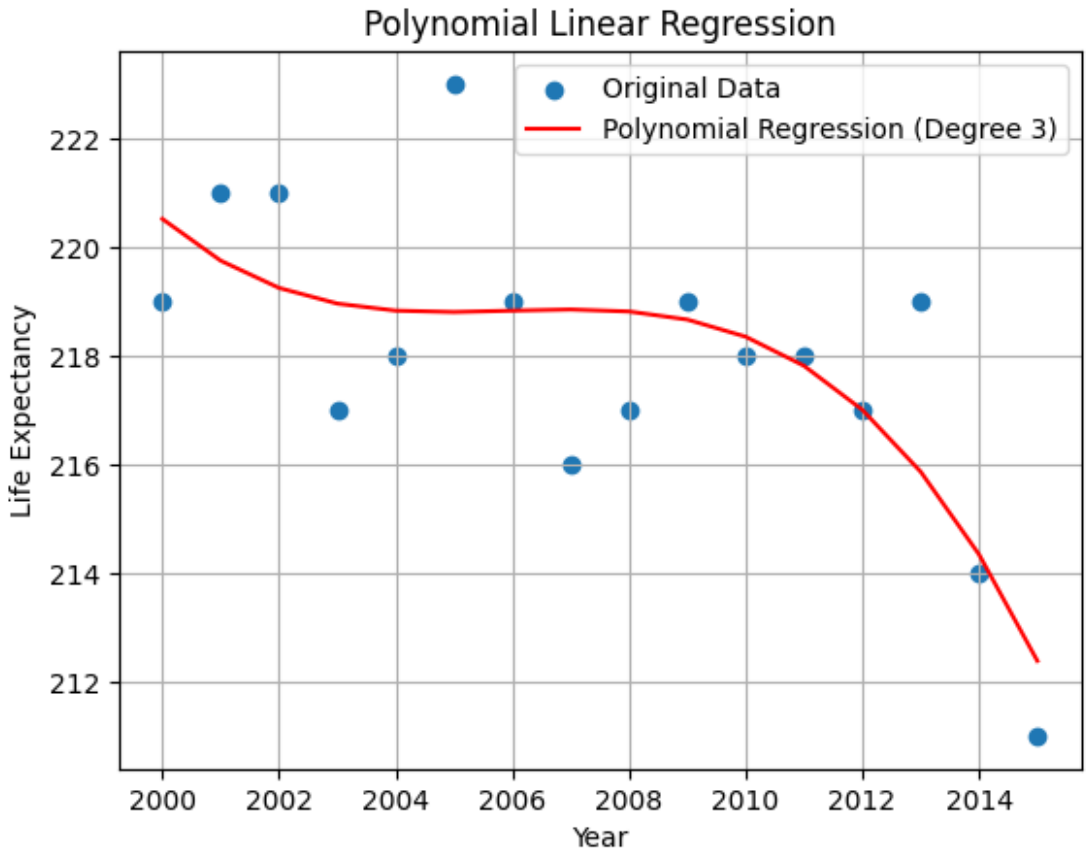
EVALUATION REPORT:

- The Multiple Linear Regression also utilized the same dataframe, particularly the Philippines dataframe (df_philippines). Except that the Y variable has three elements namely the columns of Life Expectancy, Adult Mortality, and Infant Deaths. The dataset was split into test and train sets. The Linear Regression model was once again fit to the given data. Predictions are made on to the testing set and computed for the Mean Squared Error which measures the average difference between the actual and predicted values. The MSE for this model was 4.9108. Taking note to lower MSE indicates better predictive performance.

3. POLYNOMIAL LINEAR REGRESSION

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
import matplotlib.pyplot as plt

df = pd.read_csv('/content/Life Expectancy Data.csv')
df_philippines = df[df['Country'] == 'Philippines']

# independent and dependent var
X = df_philippines['Year'].values.reshape(-1, 1)
```

```
y = df_philippines['Adult Mortality'].values

# polynomial degree and regression line
degree = 3
poly_features = PolynomialFeatures(degree=degree)
X_poly = poly_features.fit_transform(X)

# fit the PLR
poly_model = LinearRegression()
poly_model.fit(X_poly, y)

poly_predictions = poly_model.predict(X_poly)

# plotting
plt.scatter(X, y, label='Original Data')
plt.plot(X, poly_predictions, color='red', label=f'Polynomial
Regression (Degree {degree})')
plt.xlabel('Year')
plt.ylabel('Life Expectancy')
plt.title('Polynomial Linear Regression')
plt.legend()
plt.grid(True)
plt.show()
```



EVALUATION REPORT:

- The Polynomial Linear Regression has undergone with the same process as the prior two linear regressions. It also represent the relationship between Year and Adult Mortality in the Philippines using the same dataset. The function PolynomialFeatures() were used to trasnform the regression line to have a polynomial feature. It is also notable that the degree of the polynomial feature, which was a inputted into the aforementioned function) was set to 3. This is a cubic polynomial.

LOGISTIC REGRESSION

```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn import linear_model
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix


sna = pd.read_csv('/content/Social_Network_Ads.csv')

data = sna.drop(columns=['User ID'])
data = pd.get_dummies(data)

predictions = ['Age', 'EstimatedSalary','Gender_Female',
'Gender_Male']
x = data[predictions]
y = data['Purchased']

scaler = StandardScaler()
scaler.fit(x)
scaled_data = scaler.transform(x)
scaled_data = pd.DataFrame(scaled_data, columns=x.columns)
scaled_data.head()
```


```
{"summary":"{\n  \"name\": \"scaled_data\",\n  \"rows\": 400,\n
\"fields\": [\n    {\n      \"column\": \"Age\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
1.0012523486435176,\n        \"min\": -1.8773105578331641,\n
\"max\": 2.134240875847471,\n        \"num_unique_values\": 43,\n
\"samples\": [\n          1.1791095821139865,\n
0.12846515900715358,\n          0.03295202963380511\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"EstimatedSalary\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
1.0012523486435179,\n        \"min\": -1.6075056615492507,\n
\"max\": 2.3567499772898386,\n        \"num_unique_values\": 117,\n
\"samples\": [\n          1.387709710018061,\n
0.18375059007433778,\n          0.5361288690822568\n        ],\n
```

```
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Gender_Female\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
1.0012523486435176,\n        \"min\": -1.020204061220407,\n
\"max\": 0.9801960588196068,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          0.9801960588196068,\n          -
1.020204061220407\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Gender_Male\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 1.0012523486435176,\n        \"min\": -
0.9801960588196068,\n        \"max\": 1.020204061220407,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          -
0.9801960588196068,\n          1.020204061220407\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"scaled_data"}

x_train, x_test, y_train, y_test = train_test_split(scaled_data, y,
test_size=0.2, random_state=1)

model = linear_model.LogisticRegression()
model.fit(x_train, y_train)

y_pred = model.predict(x_test)
accuracy_score = round(accuracy_score(y_pred, y_test), 4)
conf_matrix = confusion_matrix(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)
print('Accuracy: ', accuracy_score)
print("Confusion Matrix:\n", conf_matrix)
print("Classification Report:\n", classification_rep)

Accuracy:  0.825
Confusion Matrix:
 [[40  8]
 [ 6 26]]
Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.83      0.85        48
           1       0.76      0.81      0.79        32

    accuracy                           0.82        80
   macro avg       0.82      0.82      0.82        80
weighted avg       0.83      0.82      0.83        80
```

EVALUATION REPORT:

- In the logistic regression used a new datasets that are most applicable in the logistic regression. The values in the data are widely dispersed, and applying scaling helps to improve the performance of the model. The precision and recall are not significantly different based on the F1-score.

DECISION TREE

```
# importing libraries
import pandas as pd
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error

#importing datasets
df = pd.read_csv('/content/Life Expectancy Data.csv')
df
```

```
{"type":"dataframe","variable_name":"df"}
```

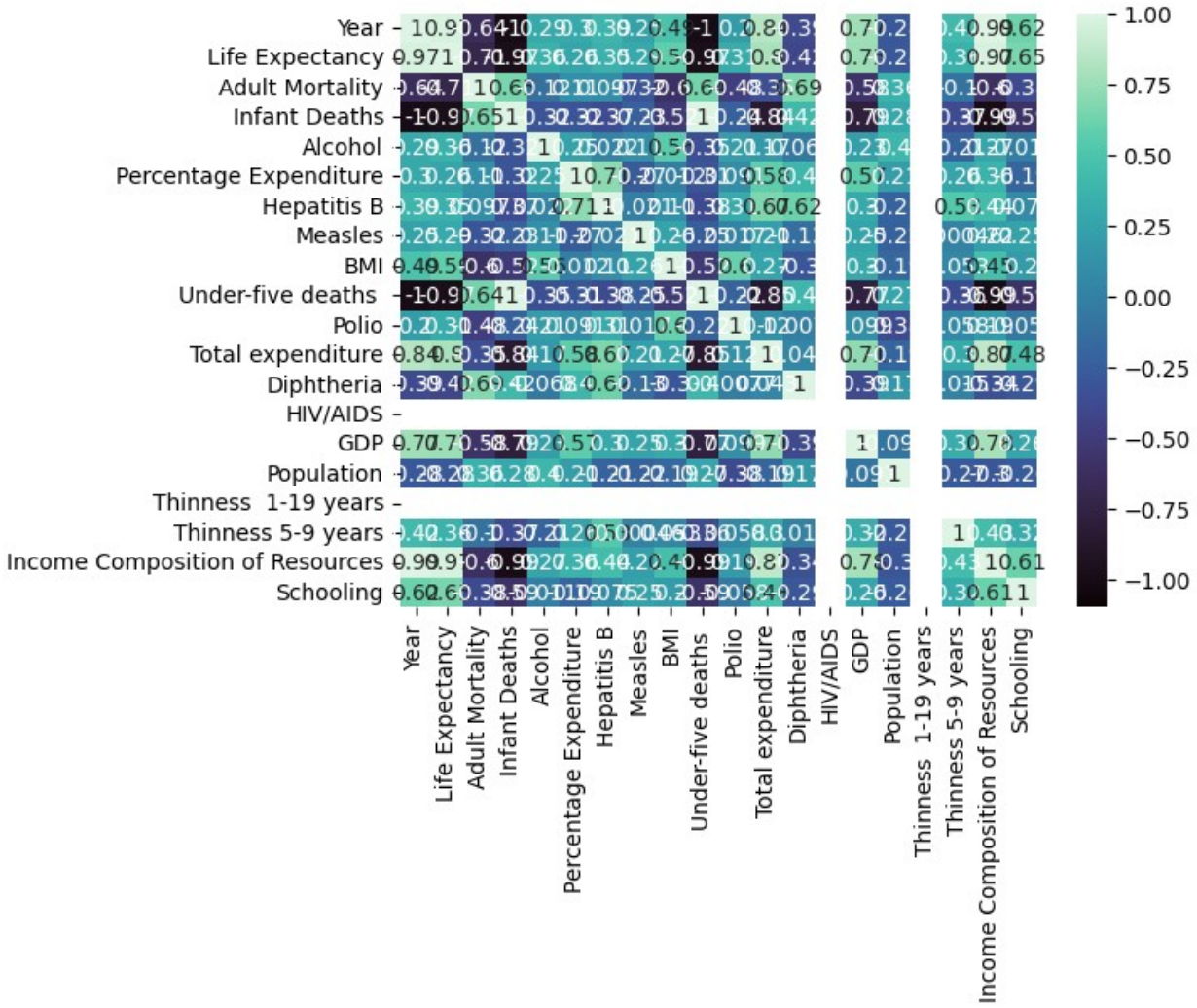
```
df_philippines = df[df['Country'] == 'Philippines']

# Display the resulting DataFrame
df_philippines.head(16)
```

```
{"type":"dataframe","variable_name":"df_philippines"}
```

```
import seaborn as sns

sns.heatmap(data=df_philippines.corr(), vmin=1, vmax=-1, annot=True,
cmap='mako')
plt.figure(figsize=(20, 6))
```

```
<ipython-input-269-aeeb62062da4>:3: FutureWarning: The default value
of numeric_only in DataFrame.corr is deprecated. In a future version,
it will default to False. Select only valid columns or specify the
value of numeric_only to silence this warning.
  sns.heatmap(data=df_philippines.corr(), vmin=1, vmax=-1, annot=True,
cmap='mako')
```

```
<Figure size 2000x600 with 0 Axes>
```




```
<Figure size 2000x600 with 0 Axes>

df_philippines.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 16 entries, 2023 to 2038
Data columns (total 22 columns):
 #   Column                           Non-Null Count  Dtype  
---  ------                           --------------  -----  
 0   Country                          16 non-null     object 
 1   Year                             16 non-null     int64  
 2   Status                           16 non-null     object 
 3   Life Expectancy                  16 non-null     float64
 4   Adult Mortality                  16 non-null     float64
 5   Infant Deaths                    16 non-null     int64  
 6   Alcohol                          15 non-null     float64
 7   Percentage Expenditure           16 non-null     float64
 8   Hepatitis B                      16 non-null     float64
```

```
 9   Measles                          16 non-null     int64
 10  BMI                              16 non-null     float64
 11  Under-five deaths                16 non-null     int64
 12  Polio                            16 non-null     float64
 13  Total expenditure                15 non-null     float64
 14  Diphtheria                       16 non-null     float64
 15  HIV/AIDS                         16 non-null     float64
 16  GDP                              16 non-null     float64
 17  Population                       16 non-null     float64
 18  Thinness  1-19 years             16 non-null     float64
 19  Thinness 5-9 years               16 non-null     float64
 20  Income Composition of Resources  16 non-null     float64
 21  Schooling                        16 non-null     float64
dtypes: float64(16), int64(4), object(2)
memory usage: 2.9+ KB
```

```
df_philippines["Total expenditure"].fillna(df_philippines["Total
expenditure"].mean(), inplace=True)
```

```
<ipython-input-220-cdc4b0d23d0c>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  df_philippines["Total expenditure"].fillna(df_philippines["Total
expenditure"].mean(), inplace=True)
```

```
df_philippines["Alcohol"].fillna(df_philippines["Alcohol"].mean(),
inplace=True)
```

```
<ipython-input-221-1fe1b6b0a60b>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  df_philippines["Alcohol"].fillna(df_philippines["Alcohol"].mean(),
inplace=True)
```

```
#create the array for the target values
y_target = df_philippines["Life Expectancy"].values
```

```
columns = ['Alcohol',  'Hepatitis B' , 'Polio','Adult
Mortality' ,'HIV/AIDS']
#create the variable to hold the features that the classifier will use
X_input = df_philippines[list(columns)].values
```

```
from sklearn.preprocessing import LabelEncoder
from sklearn import tree
```

```
label_encoder = LabelEncoder()
y_target_encoded = label_encoder.fit_transform(y_target)
clf_philippines = tree.DecisionTreeClassifier(criterion="entropy",
max_depth = 3)
clf_philippines = clf_philippines.fit(X_input, y_target_encoded)

from sklearn.metrics import mean_squared_error

y_pred = clf_philippines.predict(X_input)
mse = mean_squared_error(y_target, y_pred)
print(f"Mean Squared Error: {mse}")
```

```
Mean Squared Error: 4087.0049999999997
```

```
from sklearn.metrics import accuracy_score
accuracy_score(y_pred, y_test)
```

```
0.825
```

```
pip install scikit-learn
```

```
Requirement already satisfied: scikit-learn in
/usr/local/lib/python3.10/dist-packages (1.2.2)
Requirement already satisfied: numpy>=1.17.3 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.25.2)
Requirement already satisfied: scipy>=1.3.2 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.11.4)
Requirement already satisfied: joblib>=1.1.1 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn) (3.3.0)
```

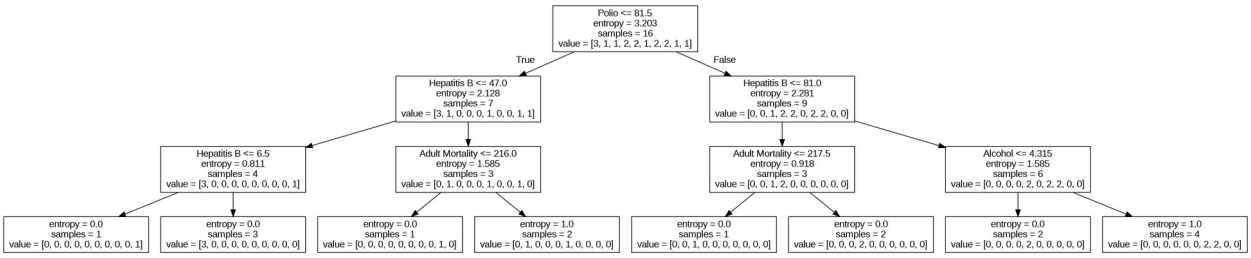
```
from six import StringIO
from sklearn import tree

# Create a Graphviz dot file to export the results of the classifier
with open("/content/Life Expectancy Data.csv", 'w') as f:
    f = tree.export_graphviz(clf_philippines, out_file=f,
feature_names=columns)
```

```
!apt-get install graphviz
```

```
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
graphviz is already the newest version (2.42.2-6).
0 upgraded, 0 newly installed, 0 to remove and 35 not upgraded.
```

```
!dot -Tpng '/content/Life Expectancy Data.csv' -o .philippines.png
```

```
from IPython.display import Image

Image(".philippines.png")
```



Evaluation Report:

- Training a decision tree classifier on your dataset, where the target variable (y_target) is encoded into numerical values for compatibility with the machine learning algorithm. The decision tree is designed to make predictions based on the input features (X_input) and the relationships captured during the training process. The accuracy score is 0.825.

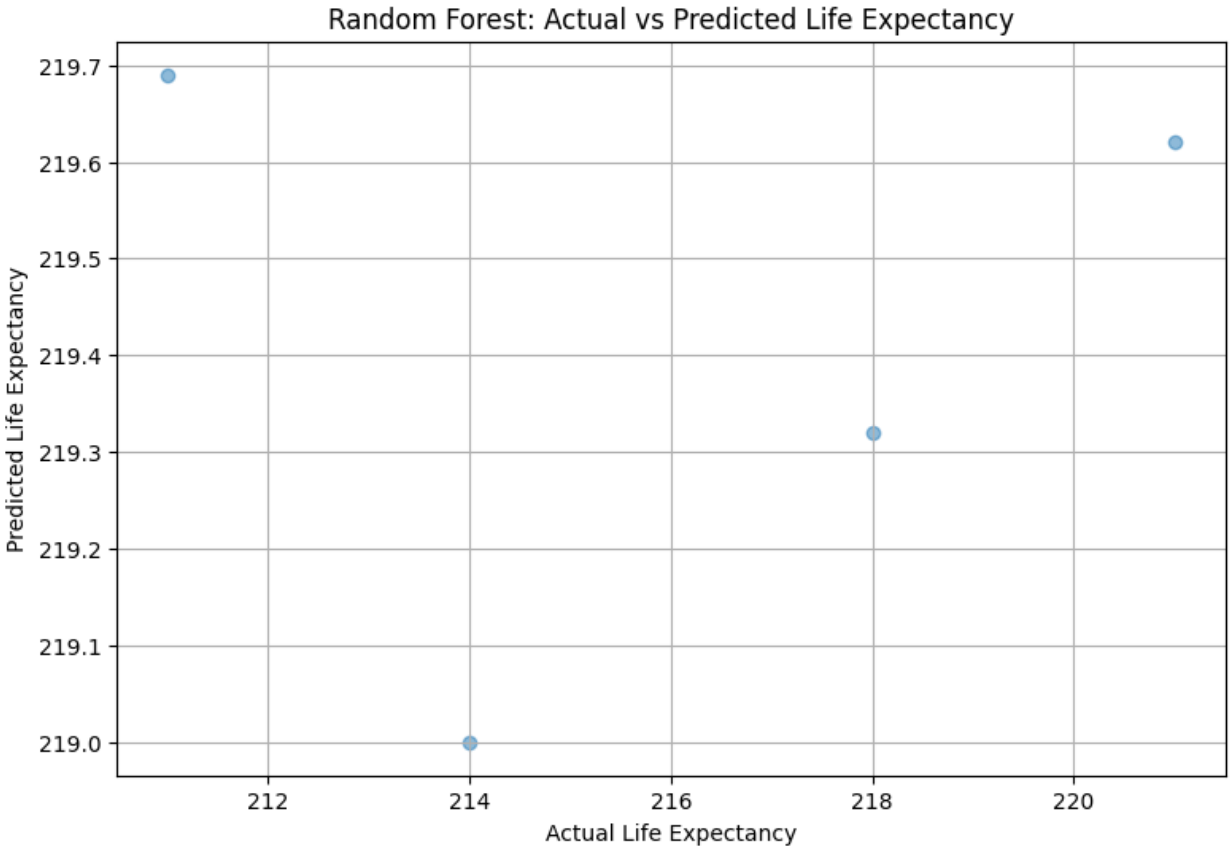
##RANDOM FOREST

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error


df_philippines = df_philippines.fillna(0)

# Select features and target variable
features = ['Alcohol', 'Hepatitis B', 'Measles', 'Polio','Diphtheria',
'HIV/AIDS' ]
target = 'Adult Mortality'

# Split the data into features and target variable
X = df_philippines[features]
y = df_philippines[target]

# 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)

# Create the Random Forest model
rf_model = RandomForestRegressor(random_state=42)
rf_model.fit(X_train, y_train)

# Make predictions on the test set
rf_predictions = rf_model.predict(X_test)
```

```
# Evaluate the Random Forest model
rf_mae = mean_absolute_error(y_test, rf_predictions)
print(f'Random Forest Mean Absolute Error: {rf_mae}')
```

```
Random Forest Mean Absolute Error: 4.097499999999997
```

```
plt.figure(figsize=(9, 6))
plt.scatter(y_test, rf_predictions, alpha=0.5)
plt.title('Random Forest: Actual vs Predicted Life Expectancy')
plt.xlabel('Actual Life Expectancy')
plt.ylabel('Predicted Life Expectancy')
plt.grid(True)
plt.show()
```



Evaluation Report:

- The graph includes both a numerical evaluation metric (MAE) and a scatter plot to assess the performance of the Random Forest model in predicting adult mortality based on the selected features. The scatter plot visualizes the relationship between the actual life expectancy (y) and the predicted life expectancy based on the given features. The scatter plot output indicates that the Random Forest model's predictions are not perfectly aligned with the actual life expectancy values.