Life Expectancy: A Data-Driven Exploration and Prediction

Milestone: Implementation of Selected Models

Group 13

Sai Varun Kumar Namburi

Prajwal Srinivas

namburi.sai@northeastern.edu

srinivas.pra@northeastern.edu

Percentage of Effort Contributed by Student1: 50%

Percentage of Effort Contributed by Student2: 50%

Signature of Student 1: Sai Varun

Signature of Student 2: Prajwal

Date of Submission: 2nd Apr 2023

Data Source: This dataset is taken from kaggle.com
https://www.kaggle.com/code/philbowman212/life-expectancy-exploratory-data-analysis/data

Data Description:

The World Health Organization's Global Health Observatory (GHO) maintains records of all countries' health statuses and related factors. The data concerning life expectancy and health factors for 193 countries were obtained from the WHO's Global Health Observatory data repository. It was noted that over the past 15 years, there has been significant progress in the health sector, leading to a significant improvement in human mortality rates, particularly in developing nations compared to the last 30 years. In this project, data from the years 2000 to 2015 for 193 countries were selected for further analysis. In this dataset, we have 22 columns as below
(Country, Year, Status, Life expectancy, Adult Mortality, infant deaths, Alcohol, percentage expenditure, Hepatitis B, Measles, BMI, under-five deaths, Polio, Total expenditure, Diphtheria, HIV/AIDS, GDP, Population, thinness 1-19 years, thinness 5-9 years, Income composition of resources, Schooling)

Implementation of Selected Model:

We have decided to use Linear Regression, a Mixed Effect Model, and a Neural Network

Linear Regression:

Linear Regression is a simple and widely used statistical model for predicting a continuous outcome variable based on one or more predictor variables. It assumes a linear relationship between the outcome variable and the predictors, which can be limiting in cases where the relationship is more complex.

In this model, life expectancy would be the dependent variable or the outcome variable, and the predictor variables would be the independent variables. The goal is to find a linear relationship between the outcome variable and the predictor variables.

To build a linear regression model for life expectancy, you would need a dataset that includes observations of individuals along with their ages, sex, education, income, health behaviors, and life expectancy. You would use this dataset to estimate the coefficients of the linear regression equation and to make predictions for new individuals.

It is important to note that linear regression assumes that the relationship between the outcome variable and the predictor variables is linear. Additionally, it assumes that the errors or residuals are normally distributed and have constant variance. Therefore, you should assess the assumptions of the linear regression model before using it to make predictions.

LINEAR REGRESSION

```
1  from sklearn.linear_model import LinearRegression
2  model = LinearRegression(fit_intercept=True)
3  x = final.loc[:,feature_df]
4  y = final.Life_Expectancy
5  model.fit(x, y)
```

[66] ✓ 0.3s

```
LinearRegression()
```

```
1  print("Model slopes:    ", model.coef_)
2  print("Model intercept:", model.intercept_)
```

[67] ✓ 0.0s

```
Model slopes:     [ 0.40621688 -0.9355618   0.2132231  -0.48777945  0.37412096  0.09314029
 -0.17081718 -3.55105439]
Model intercept: 67.38142442920507
```

```
1  y_predict = model.predict(x.values)
2  RMSE = np.sqrt(((y-y_predict)**2).values.mean())
3
4  results = pd.DataFrame()
5  results["Method"] = ["Linear Regression"]
6  results["RMSE"] = RMSE
7  results
```

[68] ✓ 0.0s

```
c:\Users\18572\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\base.py:450: UserWarning: X does
  warnings.warn(
```

	Method	RMSE
0	Linear Regression	4.932611

```
1  np.min(final['Life_Expectancy']), np.max(final['Life_Expectancy'])
```

[69] ✓ 0.0s

```
(36.3, 89.0)
```

Mixed Effect Model:

A mixed-effects model can be a useful approach to analyze life expectancy data when there are repeated measurements on the same individuals, or when individuals are nested within groups (e.g., cities, countries, or families).

In a mixed-effects model, there are both fixed effects and random effects. Fixed effects are the variables that have a consistent effect on the outcome variable across all levels of the factor. Random effects are variables that have an effect on the outcome variable that varies across different levels of the factor.

For example, if you are analyzing life expectancy data for individuals in different cities, the city can be a random effect because life expectancy may differ across different cities, but the effect of the city on life expectancy may not be the same for all individuals.

To build a mixed-effects model for life expectancy, you would need to identify the fixed and random effects and include them in the model. The fixed effects can include variables such as age, sex, education, and income. The random effects can include variables such as city, family, or other group identifiers.

Mixed-effects models have the advantage of accounting for the correlation between observations within the same individual or group, which can lead to more accurate estimates of the effects of the predictor variables on the outcome variable. However, the interpretation of the model coefficients may be more complex compared to a simple linear regression model.

It is important to note that mixed-effects models assume that the random effects are normally distributed and have a constant variance across the different levels of the factor. Therefore, you should assess the assumptions of the mixed-effects model before using it to make predictions.

```
#!pip install -q statsmodels
import statsmodels.api as sm
import statsmodels.formula.api as smf
md = smf.mixedlm("Life_Expectancy ~ Adult_Mortality_scaled + Alcohol + Polio_scaled + hivaids + BMI + thinness_1to19_years + Developing",
                 final,
                 groups=final["sum_countries_embedding"], re_formula="~Adult_Mortality_scaled + Alcohol + Polio_scaled + hivaids + BMI + thinness_1to19_years + Developing")
#re_formula To ensure that each country has its own random slope
```

[73] ✓ 1.2s Python

```
mdf = md.fit(method=["lbfgs"])
print(mdf.summary())
```

[74] ✓ 23.1s Python

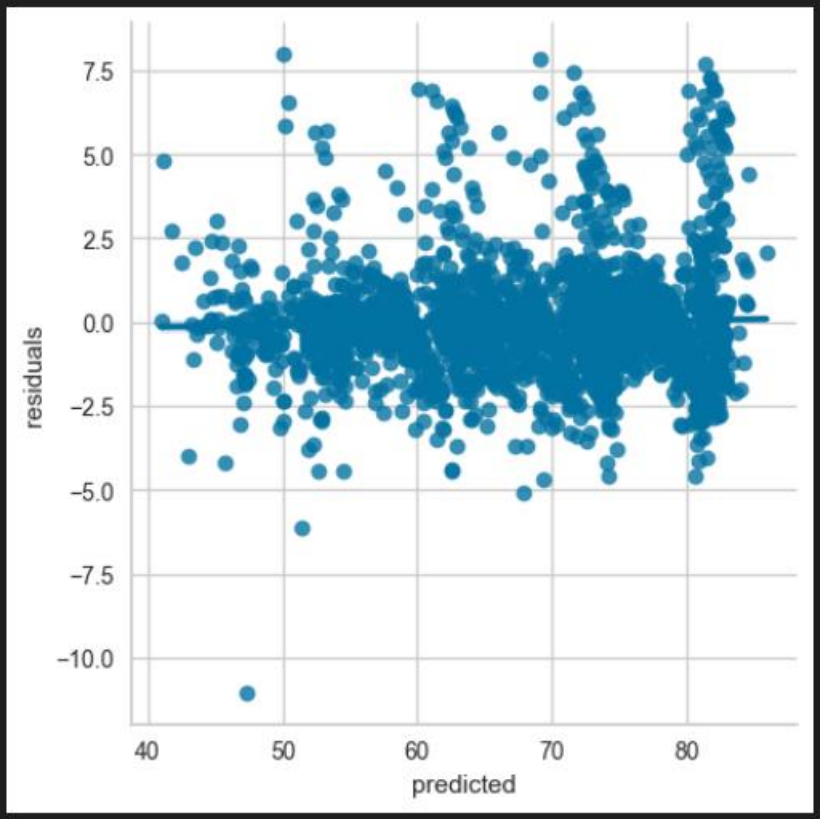
```
                                                  Coef.    Std.Err.     z    P>|z|  [0.025 0.975]
-------------------------------------------------------------------------------------------------
Intercept                                        79.581       1.574 50.569 0.000  76.496 82.665
Adult_Mortality_scaled                           -0.051       0.024 -2.153 0.031  -0.097 -0.005
Alcohol                                          -0.015       0.100 -0.147 0.883  -0.211  0.182
Polio_scaled                                      0.169       0.034  4.914 0.000   0.102  0.237
hivaids                                          -1.659
BMI                                               0.025       0.052  0.491 0.623  -0.076  0.126
thinness_1to19_years                             -0.616       0.199 -3.100 0.002  -1.006 -0.227
Developing                                      -11.848       1.684 -7.036 0.000 -15.148 -8.547
Group Var                                         4.665    9266.963
Group x Adult_Mortality_scaled Cov               -0.060       0.350
Adult_Mortality_scaled Var                        0.025       0.003
Group x Alcohol Cov                               0.068       0.724
Adult_Mortality_scaled x Alcohol Cov              0.012
Alcohol Var                                       1.038       0.087
Group x Polio_scaled Cov                         -0.017       0.286
Adult_Mortality_scaled x Polio_scaled Cov          0.009
Alcohol x Polio_scaled Cov                        -0.239
Polio_scaled Var                                  0.217
Group x hivaids Cov                               0.354   46334.804
Adult_Mortality_scaled x hivaids Cov             -0.007       0.016
Alcohol x hivaids Cov                             0.020
Polio_scaled x hivaids Cov                       -0.004
hivaids Var                                       3.475       0.351
Group x BMI Cov                                  -0.022       0.469
Adult_Mortality_scaled x BMI Cov                  0.000
Alcohol x BMI Cov                                -0.013
Polio_scaled x BMI Cov                            0.011       0.017
hivaids x BMI Cov                                 0.023
BMI Var                                           0.465
Group x thinness_1to19_years Cov                 -0.120       1.534
Adult_Mortality_scaled x thinness_1to19_years Cov  0.000
Alcohol x thinness_1to19_years Cov               -0.047
Polio_scaled x thinness_1to19_years Cov           0.137
hivaids x thinness_1to19_years Cov               -0.280
BMI x thinness_1to19_years Cov                    0.024
thinness_1to19_years Var                          3.000
Group x Developing Cov                            0.948  255351.363
Adult_Mortality_scaled x Developing Cov          -0.010       0.355
Alcohol x Developing Cov                          0.062       0.697
Polio_scaled x Developing Cov                    -0.014       0.312
hivaids x Developing Cov                          0.373   46334.804
BMI x Developing Cov                              0.003       0.485
```

```
performance = pd.DataFrame()
performance["residuals"] = mdf.resid.values
performance["predicted"] = mdf.fittedvalues

sns.lmplot(x = "predicted", y = "residuals", data = performance)
```

[75] ✓ 0.5s

```
<seaborn.axisgrid.FacetGrid at 0x250bba79180>
```



Neural Networks Model:

A neural network can be a powerful approach for predicting life expectancy based on a wide range of predictor variables. Neural networks are a type of machine learning model that can learn complex non-linear relationships between input and output variables.

To build a neural network model for life expectancy, you would need a dataset that includes observations of individuals along with their ages, sex, education, income, health behaviors, and life expectancy. You would use this dataset to train the neural network to predict life expectancy based on the predictor variables.

USING A NEURAL NETWORK

```
[77]  1  final = pd.read_csv('./final.csv')
      ✓ 0.0s
```

```
[78]  1  final.head()
      ✓ 0.0s
```

	Country	Adult_Mortality	Alcohol	HIV/AIDS	Polio	BMI	thinness_1to19_years	Life_Expectancy	Developing	Adult_Mortality_scaled	Polio_scaled
0	Afghanistan	263.0	0.01	0.1	6.0	19.1	17.2	65.0	1	7.257618	0.625000
1	Afghanistan	271.0	0.01	0.1	58.0	18.6	17.5	59.9	1	7.479224	11.458333
2	Afghanistan	268.0	0.01	0.1	62.0	18.1	17.7	59.9	1	7.396122	12.291667
3	Afghanistan	272.0	0.01	0.1	67.0	17.6	17.9	59.5	1	7.506925	13.333333
4	Afghanistan	275.0	0.01	0.1	68.0	17.2	18.2	59.2	1	7.590028	13.541667

```
[79]  1  final['Status'] = final['Developing'].map(lambda x: 'Developing' if x==1 else 'Developed')
      ✓ 0.0s
```

```
[80]  1  train, test = train_test_split(final, test_size=0.2)
      2  train, val = train_test_split(train, test_size=0.2)
      3  print(len(train), 'train examples')
      4  print(len(val), 'validation examples')
      5  print(len(test), 'test examples')
      ✓ 0.1s
```

```
1832 train examples
459 validation examples
573 test examples
```

The neural network would consist of layers of interconnected nodes, with each node representing a mathematical function that combines the input variables to produce an output. The weights between the nodes are adjusted during the training process to minimize the difference between the predicted and actual life expectancy values.

One advantage of neural networks is that they can capture complex interactions and patterns in the data that may be difficult to capture using traditional statistical models such as linear regression or mixed-effects models. However, they can be prone to overfitting if the model is too complex or if there is not enough data to support the complexity of the model.

It is important to note that neural networks require a large amount of data and computational resources to train effectively. Additionally, the model architecture and hyperparameters need to be carefully chosen and tuned to avoid overfitting and achieve good performance on the test data.

```
[81]
def df_to_dataset(dataframe, shuffle=True, batch_size=32):
    dataframe = dataframe.copy()
    labels = dataframe.pop('Life_Expectancy')
    ds = tf.data.Dataset.from_tensor_slices((dict(dataframe), labels))
    if shuffle:
        ds = ds.shuffle(buffer_size=len(dataframe))
    ds = ds.batch(batch_size)
    return ds
✓ 0.0s
```

```
[82]
train_ds = df_to_dataset(train)
val_ds = df_to_dataset(val, shuffle=False, batch_size=16)
test_ds = df_to_dataset(test, shuffle=False, batch_size=16)
✓ 0.1s
```

```
[83]
final_batch = next(iter(train_ds))[0]
✓ 0.2s
```

```
[84]
def demo(feature_column):
    feature_layer = layers.DenseFeatures(feature_column)
    return feature_layer(final_batch).numpy()
✓ 0.0s
```

```
[85]
feature_columns = []
# numeric cols
for header in ['Alcohol', 'HIV/AIDS', 'Polio_scaled','BMI','thinness_1to19_years']:
    feature_columns.append(feature_column.numeric_column(header))
✓ 0.0s
```

```
[86]
#catgorical cols
status = feature_column.categorical_column_with_vocabulary_list(
      'Status', ['Developing', 'Developed'])

status = feature_column.indicator_column(status)
feature_columns.append(status)
✓ 0.0s
```

```
[87]
# embedding columns
country = feature_column.categorical_column_with_vocabulary_list(
      'Country', final.Country.unique())
country_embedding = feature_column.embedding_column(country, dimension=6)
feature_columns.append(country_embedding)
✓ 0.0s
```

```
[88]
#Input layer
feature_layer = tf.keras.layers.DenseFeatures(feature_columns)
#Model architecture
model = tf.keras.Sequential([
  feature_layer,
  layers.Dense(128, activation='relu'),
  layers.Dense(64, activation='relu'),
  layers.Dense(32, activation='relu'),
  layers.Dense(16, activation='relu'),
  layers.Dense(1)
])
model.compile(loss='mse', optimizer=tf.keras.optimizers.Adam(0.001), metrics=[tf.keras.metrics.RootMeanSquaredError()])
✓ 0.1s
```

```
[89]
#Fitting
history = model.fit(train_ds, validation_data=val_ds, epochs=40)
✓ 10.8s
... Epoch 1/40
```

Logistic Regression:

Logistic regression is typically used to model binary outcomes, such as the presence or absence of a disease, and is not well-suited for predicting life expectancy, which is a continuous variable. However, logistic regression could be used to predict a binary outcome related to life expectancy, such as the likelihood of an individual surviving to a certain age.

For example, you could use logistic regression to model the probability of an individual surviving to a certain age based on predictor variables such as age, sex, education, income, and health behaviors. In this case, the outcome variable would be binary, with 1 indicating survival to the specified age and 0 indicating death before that age.

To build a logistic regression model for this problem, you would need a dataset that includes observations of individuals along with their ages, sex, education, income, health behaviors, and survival status (1 or 0) at the specified age. You would use this dataset to estimate the coefficients of the logistic regression equation and to make predictions for new individuals.

It is important to note that logistic regression assumes that the relationship between the predictor variables and the outcome variable is linear on the logit scale. Additionally, it assumes that the errors or residuals are independent and have a constant variance. Therefore, you should assess the assumptions of the logistic regression model before using it to make predictions.

```
14  # Select predictor variables
15  X = data[['Adult Mortality', 'infant deaths', 'Alcohol', 'percentage expenditure',
16            'Hepatitis B', 'Measles ', ' BMI ', 'under-five deaths ', 'Polio',
17            'Total expenditure', 'Diphtheria ', ' HIV/AIDS', 'GDP', 'Population',
18            ' thinness  1-19 years', ' thinness 5-9 years',
19            'Income composition of resources', 'Schooling']]
20
21  # Split data into training and test sets
22  X_train, X_test, y_train, y_test = train_test_split(X, data['survival'], test_size=0.2, random_state=42)
23
24  # Fit logistic regression model
25  clf = LogisticRegression()
26  clf.fit(X_train, y_train)
27
28  # Predict on test data
29  y_pred = clf.predict(X_test)
30
31  # Evaluate model performance
32  accuracy = accuracy_score(y_test, y_pred)
33  print('Accuracy:', accuracy*100)
34
[107]  ✓ 0.1s

...  Accuracy: 81.81818181818183
```