Life Expectancy: A Data-Driven Exploration and Prediction

Milestone: Performance Evaluation and Interpretation

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: 16th Apr 2023

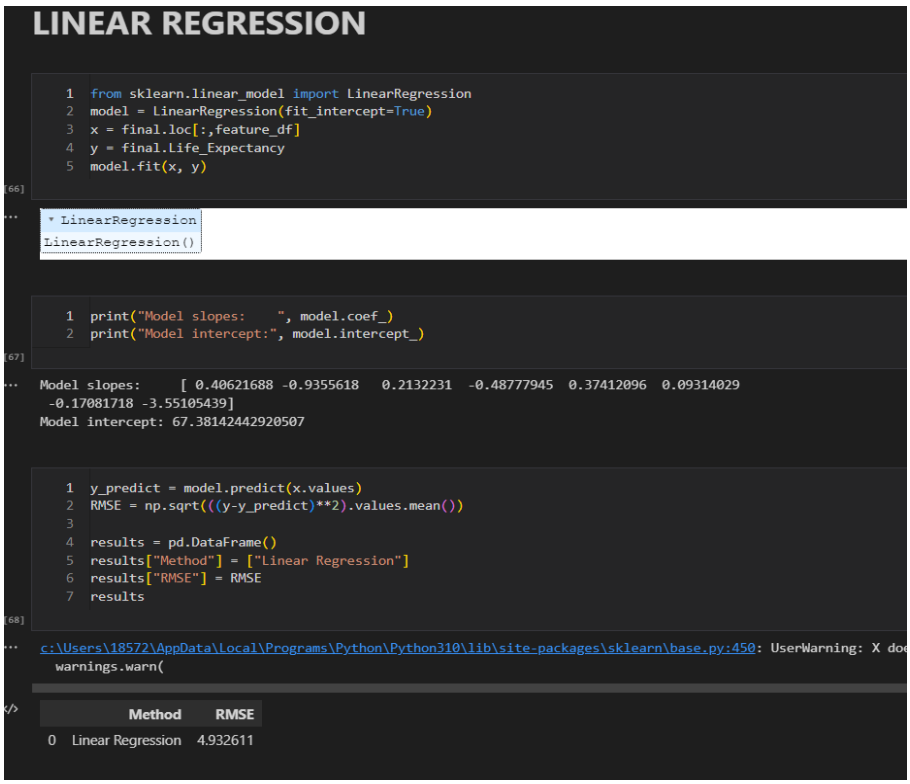
Data Source: This dataset is taken from kaggle.com
https://www.kaggle.com/code/philbowman212/life-expectancy-exploratory-data-analysis/data

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)

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.



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.

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).

```
#!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
```

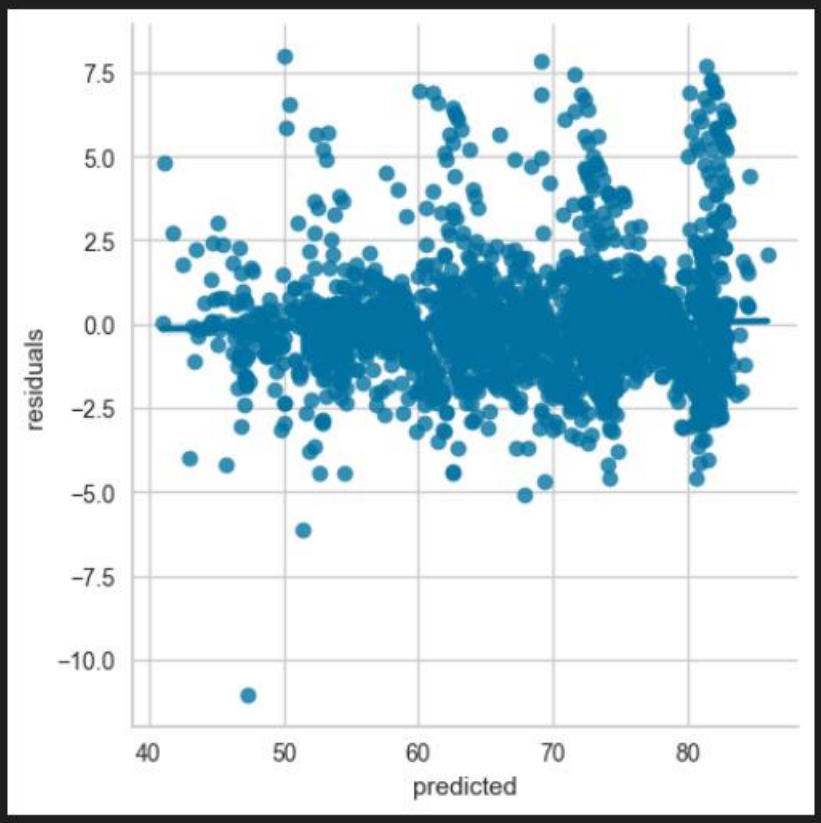
```
mdf = md.fit(method=["lbfgs"])
print(mdf.summary())
```

	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
```

```
]  1  hist = pd.DataFrame(history.history)
   2  hist['epoch'] = history.epoch
   3  hist.tail()
```

	loss	root_mean_squared_error	val_loss	val_root_mean_squared_error	epoch
35	5.904938	2.430008	8.257162	2.873528	35
36	5.372705	2.317910	8.346706	2.889067	36
37	5.738933	2.395607	8.139395	2.852962	37
38	5.235527	2.288127	8.433536	2.904055	38
39	5.269314	2.295499	8.206532	2.864705	39

```
[91]
1  predictions = model.predict(test_ds)
2  y = np.concatenate([y for x, y in test_ds], axis=0)
```

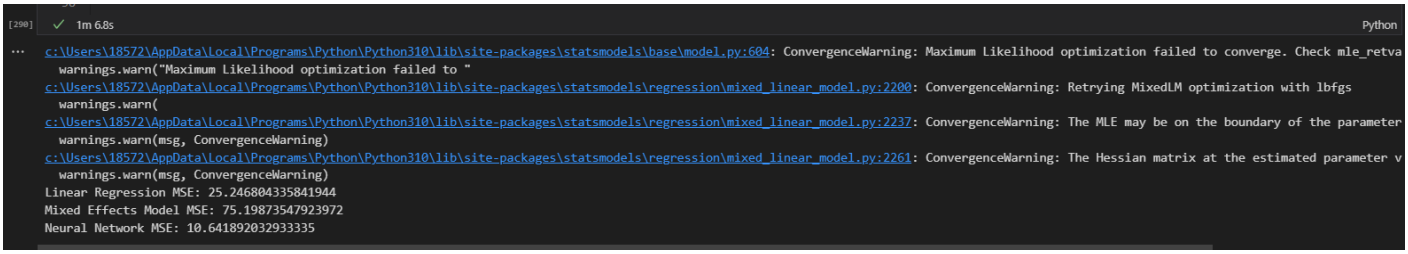
```
...  IING:tensorflow:Layers in a Sequential model should only have a single input tensor. Receiv
     36 [==============================] - 0s 1ms/step
```

```
[92]
1  compare = pd.DataFrame({'predictions':predictions.reshape((-1,)), 'True': y})
2  compare.tail()
```

	predictions	True
568	63.425800	63.9
569	70.844452	73.0
570	74.340126	75.9
571	72.930397	74.8
572	84.465088	83.3

After comparing accuracy with the three models

```
[290]
# split the data into training and testing sets
import statsmodels.api as sm
import statsmodels.formula.api as smf
from sklearn.metrics import mean_squared_error
from sklearn.neural_network import MLPRegressor
from sklearn.linear_model import LinearRegression

# df = pd.read_csv('Life Expectancy Data.csv')
X = final.loc[:,feature_df]
y = final.Life_Expectancy
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# train and evaluate linear regression model
lr = LinearRegression()
lr.fit(X_train, y_train)
lr_pred = lr.predict(X_test)
lr_mse = mean_squared_error(y_test, lr_pred)

# train and evaluate mixed effects model
# (assuming the data has a nested structure with country as a random effect)
mixed_model = 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")
mixed_results = mixed_model.fit()
mixed_pred = mixed_results.predict(X_test)
mixed_mse = mean_squared_error(y_test, mixed_pred)

# train and evaluate neural network model
nn = MLPRegressor(hidden_layer_sizes=(100,), activation="relu", solver="adam", max_iter=1000)
nn.fit(X_train, y_train)
nn_pred = nn.predict(X_test)
nn_mse = mean_squared_error(y_test, nn_pred)

# print the MSE for each model
print("Linear Regression MSE:", lr_mse)
print("Mixed Effects Model MSE:", mixed_mse)
print("Neural Network MSE:", nn_mse)

✓ 1m 6.8s                                                                    Python
```



In this particular case, the mixed effect model is superior to both normal linear regression and the used neural network architecture, since it takes into account the dependence of the data.