PART A

Topic of Interest

Our topic of interest for this project is to analyse how the wealth of a country affects various qualities of its population. In particular, we look at how its wealth is correlated to its death rate per capita, its literacy rate, and its suicide rates.

Dataset

The one common dataset all members of the project team used was a revised version of the stage 1 & stage 2 dataset. This new dataset, named "dataStage3.csv" includes all the columns present in the first dataset, with all metadata and schema remaining the same. Simple cleaning was performed on the dataset to remove quality issues that prevented the production of a predictive model. This included removing any null values. See below the schema for the dataset:

Column Name	Column Description	Column type
Country Code	ISO 3166-1 alpha-3 country codes	String
Literacy Value	Literacy rate of a country	Float
Region	Grouping of countries based on region by the World Bank	String
IncomeGroup	The World Bank's classification of income group	String
Death-Crude-XXXX	The crude death rate per one thousand people for XXXX year. Years include from range 2010-2020	Float
Suicide-XXXX	The suicide mortality rate per 100,000 population per country for XXXX year. Years include from range 2000-2019	Float
Income-XXXX	The GDP (billion) of a country for XXXX year. Years include from range 2017-2021	Float

Preliminary Notes

- The attribute to be **predicted** is the quantitative Death Crude rate
- The dataset has been split into a **training** set and a **testing** set at a 90:10 split.

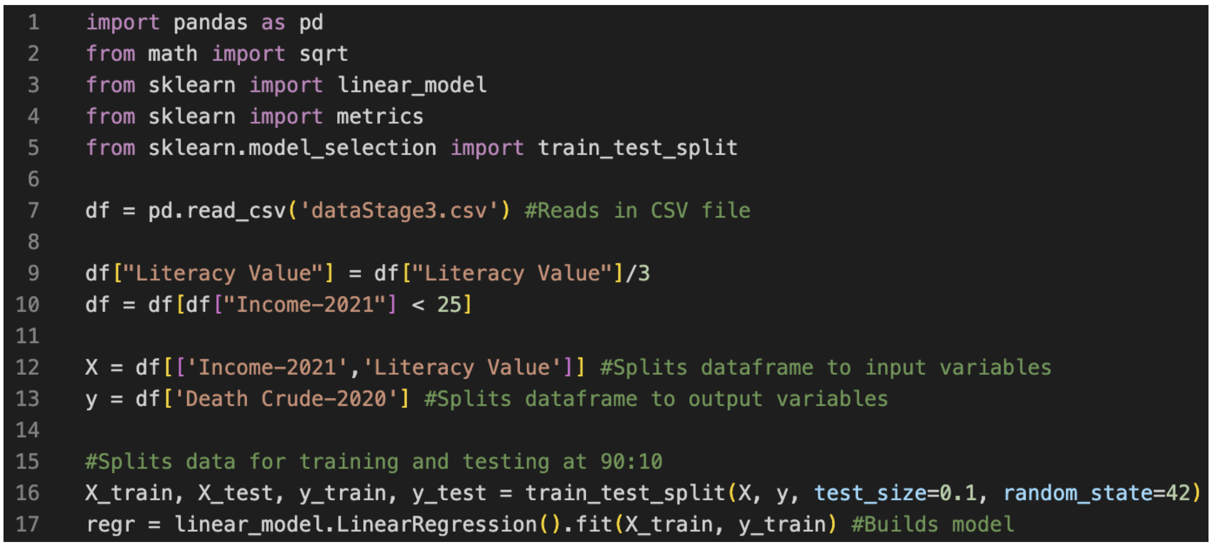
Member 1 - jali6584

Linear Regression - Predicting Death Crude Rate from Income and Literacy Value

Linear Regression is a machine learning technique that is often regarded as the most basic and common predictive analysis. This technique models the relationship between two variables to predict a target value using a line-of-best-fit. In this model, a given increase in an attribute results in the same identical change with the target value. This occurs irrelevant of what the value the target attribute has, nor the values of any other attributes. See below details about the produced linear regression predictive model:

- Input Variables: Income-2021, Literacy Value
  - The Literacy value has been rescaled from 0-100 to one third its original scale. This has been done to have a similar scale between the input variables so they are comparable. Without rescalling, the literacy value with a much higher variation would greatly influence the result.
- Output Variable: Death Crude-2020
- Allocation of Data: the data in the dataset has been split into training and testing data at a 90:10 split as agreed upon above.

The image below shows the python code used to produce the linear regression predictive model. Simple comments have been used where necessary to explain the workings of relatively complex lines. See below the python code complete breakdown of the code



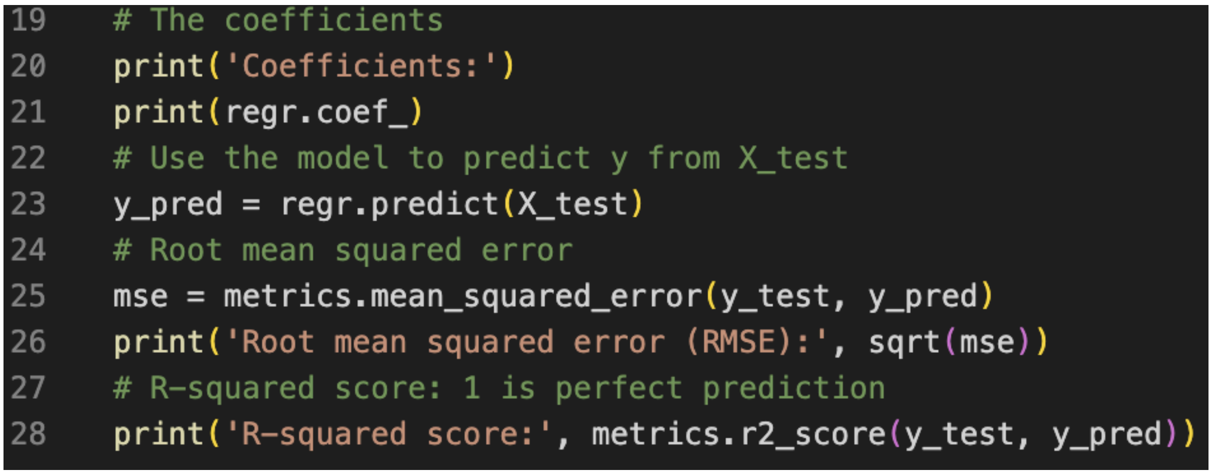
```python
1   import pandas as pd
2   from math import sqrt
3   from sklearn import linear_model
4   from sklearn import metrics
5   from sklearn.model_selection import train_test_split
6
7   df = pd.read_csv('dataStage3.csv') #Reads in CSV file
8
9   df["Literacy Value"] = df["Literacy Value"]/3
10  df = df[df["Income-2021"] < 25]
11
12  X = df[['Income-2021','Literacy Value']] #Splits dataframe to input variables
13  y = df['Death Crude-2020'] #Splits dataframe to output variables
14
15  #Splits data for training and testing at 90:10
16  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=42)
17  regr = linear_model.LinearRegression().fit(X_train, y_train) #Builds model
```

- Line 1-5: Imports the relevant libraries
- Line 7: Reads in the ‘dataStage3.csv’ as a pandas DataFrame stored in the variable df
- Line 9: Rescaling the Literacy Value to one third its original scale
- Line 10: Removing any entries where the value in the ‘Income-2021’ is greater than 25. This was performed to remove the outliers, which was necessary as linear models are quite fragile if data contains any outliers or errors
- Line 12: Extracts the two input variables (‘Income-2021’, ‘Literacy Value’) and store it in the variable X
- Line 13: Extracts the target variable (‘Death Crude-2020’) and store it in the variable Y

- Line 16: Splits the dataset into a training set and a testing set with a 90:10 split respectively. This split occurs randomly, but with the inclusion of the parameter *random_state* maintains the same random splitting of data across runs
- Line 17: Builds the linear regression model based on the dependant and independent training data

Evaluation

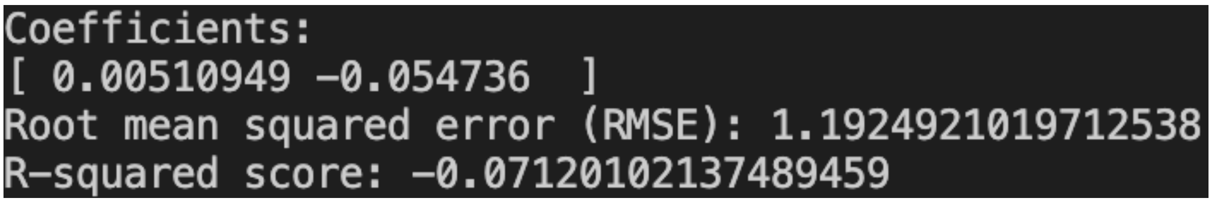
As there are a number of ways to produce different predictive models for a given dataset, it is necessary to measure the overall effectiveness of the model. To evaluate how well this predictive model does in predicting, I chose to calculate the root square mean error (RMSE) and the R-squared value. The score of the RMSE is the sum of the difference between the predicted and actual value, which is averaged on the items in the test dataset. The difference is squared and the overall result is square rooted so the different signs of values don't cancel out. The R-Squared value is a measurement that indicates how much the degree of variation of the dependent variable is explained by the independent variable. A lower RMSE score indicates a better fit, while an R-Square score closer to 1 is better. See below the code snippet for calculating the two evaluation scores, and the output.



```
19   # The coefficients
20   print('Coefficients:')
21   print(regr.coef_)
22   # Use the model to predict y from X_test
23   y_pred = regr.predict(X_test)
24   # Root mean squared error
25   mse = metrics.mean_squared_error(y_test, y_pred)
26   print('Root mean squared error (RMSE):', sqrt(mse))
27   # R-squared score: 1 is perfect prediction
28   print('R-squared score:', metrics.r2_score(y_test, y_pred))
```

- Line 21: Retrieves the coefficients in the model
- Line 23: Predicts the target with the model supplied with the dependant testing data
- Line 25: Calculates the RMSE score
- Line 28: Calculates and prints out the R-Squared score

Output



```
Coefficients:
[ 0.00510949 -0.054736  ]
Root mean squared error (RMSE): 1.1924921019712538
R-squared score: -0.07120102137489459
```

From the output, the coefficient of the *Income-2021* is higher than the coefficient of *Literacy Value*. This signifies that the *Income-2021* has a greater association with the target *Death Crude-2020* than *Literacy Value*. The root mean squared error of this model is 1.19.

This score is relatively high, with a good RMSE score generally being between 0.2-0.5. The R-Squared score is -0.07 which is quite low. The results of these two evaluations signifies that this model isn't very good at being able to predict the data accurately, and alluding to possible overfitting of the data. Adequate transformations of the data such as rescaling and removing outliers have been performed in removing high variance of a singular attribute. A possible point of entry for factors affecting the evaluation score is through the characteristics of the dataset. Although the dataset has been cleaned, it cannot be guaranteed the dataset is completely clean, with new quality issues being found early on the original cleaned dataset. Additionally, the volume of the dataset is relatively low, with more data being ideal to better see the trends and correlation. Overall further investigation is required to appropriately determine the effectiveness of this model, but current results indicate this model isn't quite good. Experiments on different predictive model types and on better data should occur to generate a better predictive model to predict the death crude rate from literacy value and income.

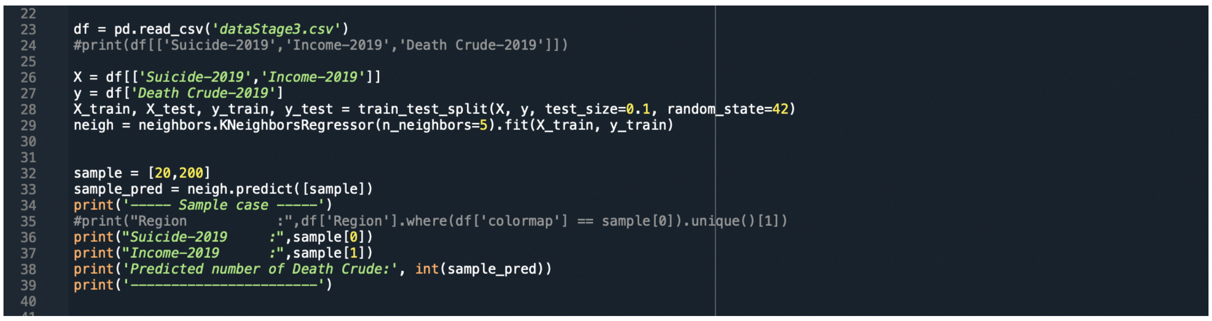
Member 2 (ager5893)

K-Nearest Neighbors Regression - Predicting Death Crude with Suicide and Income data

The *K-Nearest Neighbors Regression (KNN)* machine learning technique is a non-parametric method which is used to estimate continuous variables and is weighted by the inverse of their distance. KNN finds the distance between a query and all the examples in the data, selects the data closest to the query, and votes on the average or most frequent value. It can be used to solve classification and regression problems, which is why this method might be suitable for death crude prediction. The breakdown of the details on the KNN regression model can be seen below:

- Input attributes = Suicide-2019 and Income-2019
- Output attribute = Death Crude-2019
- The data has been split to a training dataset and testing dataset with a ratio of 10:90.

The code below shows the python code to produce the sample prediction for variable death crude based on the suicide and income data in 2019:

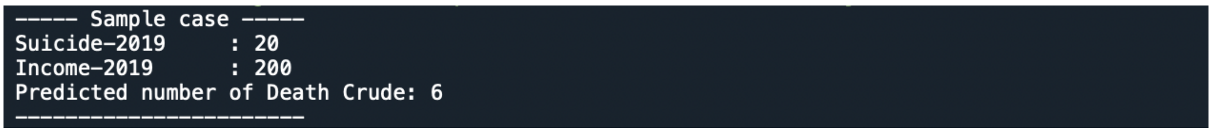


```
22
23      df = pd.read_csv('dataStage3.csv')
24      #print(df[['Suicide-2019','Income-2019','Death Crude-2019']])
25
26      X = df[['Suicide-2019','Income-2019']]
27      y = df['Death Crude-2019']
28      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=42)
29      neigh = neighbors.KNeighborsRegressor(n_neighbors=5).fit(X_train, y_train)
30
31
32      sample = [20,200]
33      sample_pred = neigh.predict([sample])
34      print('----- Sample case -----')
35      #print("Region           :",df['Region'].where(df['colormap'] == sample[0]).unique()[1])
36      print("Suicide-2019     :",sample[0])
37      print("Income-2019      :",sample[1])
38      print('Predicted number of Death Crude:', int(sample_pred))
39      print('----------------------')
40
```

Code description :

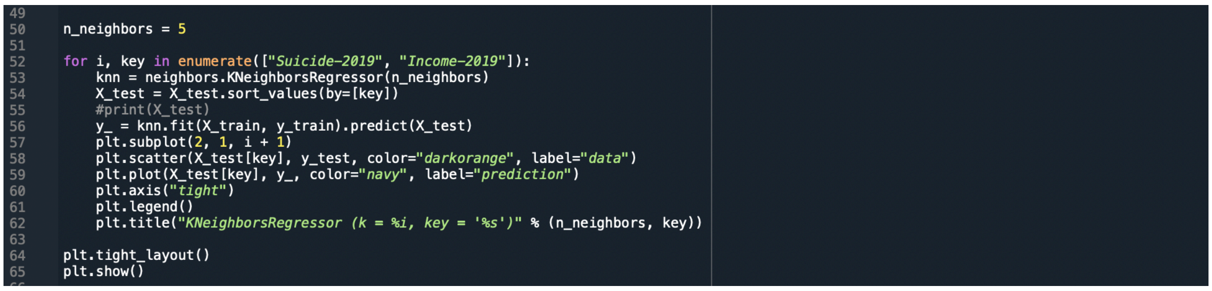
- Line 23: Read the dataset *dataStage3.csv.*
- Line 26: Slice data frame for the input attributes.
- Line 27: Slice data frame for the target or predicted attribute.
- Line 29: Setting the random to 42 and setting test size to 0.1, splitting the training and test data to a ratio of 10:90.
- Line 32: Creating one sample from input attributes and predicting the death crude.
- Line 36: Printing the sample suicide value.
- Line 37: Printing the sample income value.
- Line 38: Printing the result of the prediction for death crude based on the sample.

Output :



```
----- Sample case -----
Suicide-2019     : 20
Income-2019      : 200
Predicted number of Death Crude: 6
----------------------
```

A *K-Nearest Neighbors Regression Model* can also be done to clearly show the result of the prediction while comparing it with the existing data, as well as to show any correlation between predictions from each attribute. The model shows how the KNN algorithm chose the precise value for its prediction. The python code for the K-Nearest Neighbors Regression Model can be seen below:



```
49
50      n_neighbors = 5
51
52      for i, key in enumerate(["Suicide-2019", "Income-2019"]):
53          knn = neighbors.KNeighborsRegressor(n_neighbors)
54          X_test = X_test.sort_values(by=[key])
55          #print(X_test)
56          y_ = knn.fit(X_train, y_train).predict(X_test)
57          plt.subplot(2, 1, i + 1)
58          plt.scatter(X_test[key], y_test, color="darkorange", label="data")
59          plt.plot(X_test[key], y_, color="navy", label="prediction")
60          plt.axis("tight")
61          plt.legend()
62          plt.title("KNeighborsRegressor (k = %i, key = '%s')" % (n_neighbors, key))
63
64      plt.tight_layout()
65      plt.show()
```

Code description

- Line 50: Restating the n_neighbors.
- Line 52: Calling the input attributes used as the base for the prediction of death crude.
- Line 54: Sorting the test data.
- Line 56: Creating y_ .
- Line 58: Customising the features of the data model such as adding colour and labels.
- Line 59: Customising the features of the prediction model such as adding colour and labels.
- Line 62: Adding title for the predictive models in each variable.
- Line 65: Showing and printing the K-Nearest Neighbors Regression Model.

Output :

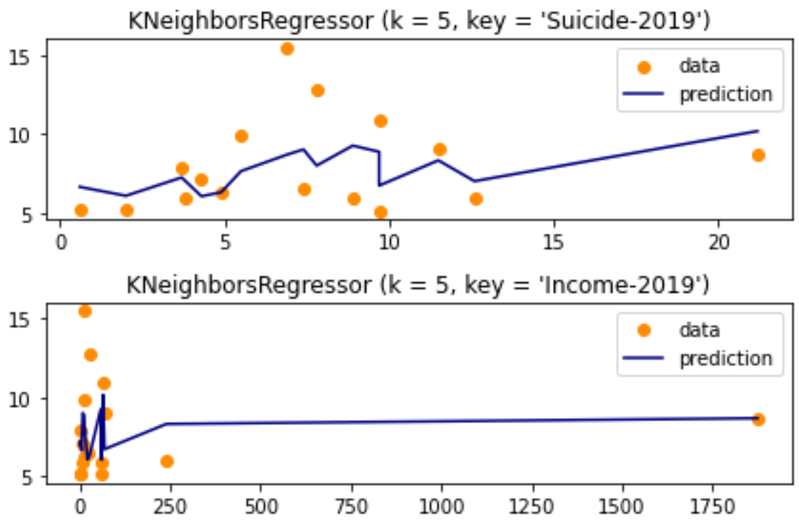
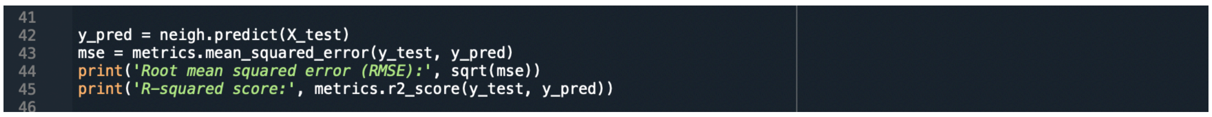


Figure 1 - KNN Regression Model for death crude prediction

The KNN machine learning algorithm applies feature similarity to predict the value of each data point meaning that a new point is assigned a specific value based on how it resembles the point in the training dataset. Therefore the coefficient might vary based on the data or the point. For the model above, the Y-axis is the death crude being predicted while the X-axis are suicide and income data from 2019. All three data used are taken from the same year to prevent any inaccuracy and to make the prediction more credible. Based on the model, it shows that there is a higher death rate when the income is lower and when suicide rate is higher.

Evaluation

In order to assess the accuracy and proximity of the model, we could find its root mean squared error (RMSE) and R-squared value. Both methods can measure how well a specific regression model fits a dataset, RMSE estimates the error by its value and the R-squared method calculates the error in terms of percentage. For RMSE, lower scores indicate better fit, as for the R-squared method, values close to 1 would be considered a good score, meaning the model is suitable for the data predicted.

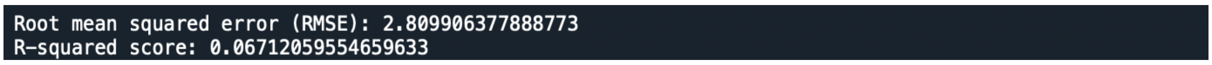


```
41
42  y_pred = neigh.predict(X_test)
43  mse = metrics.mean_squared_error(y_test, y_pred)
44  print('Root mean squared error (RMSE):', sqrt(mse))
45  print('R-squared score:', metrics.r2_score(y_test, y_pred))
46
```

Code description

- Line 42: Using the model to predict the X testing data
- Line 43: Finding the root mean squared error (RMSE)
- Line 44: Printing the RMSE result.
- Line 45: Printing R-squared result.

Output :



```
Root mean squared error (RMSE): 2.809906377888773
R-squared score: 0.06712059554659633
```

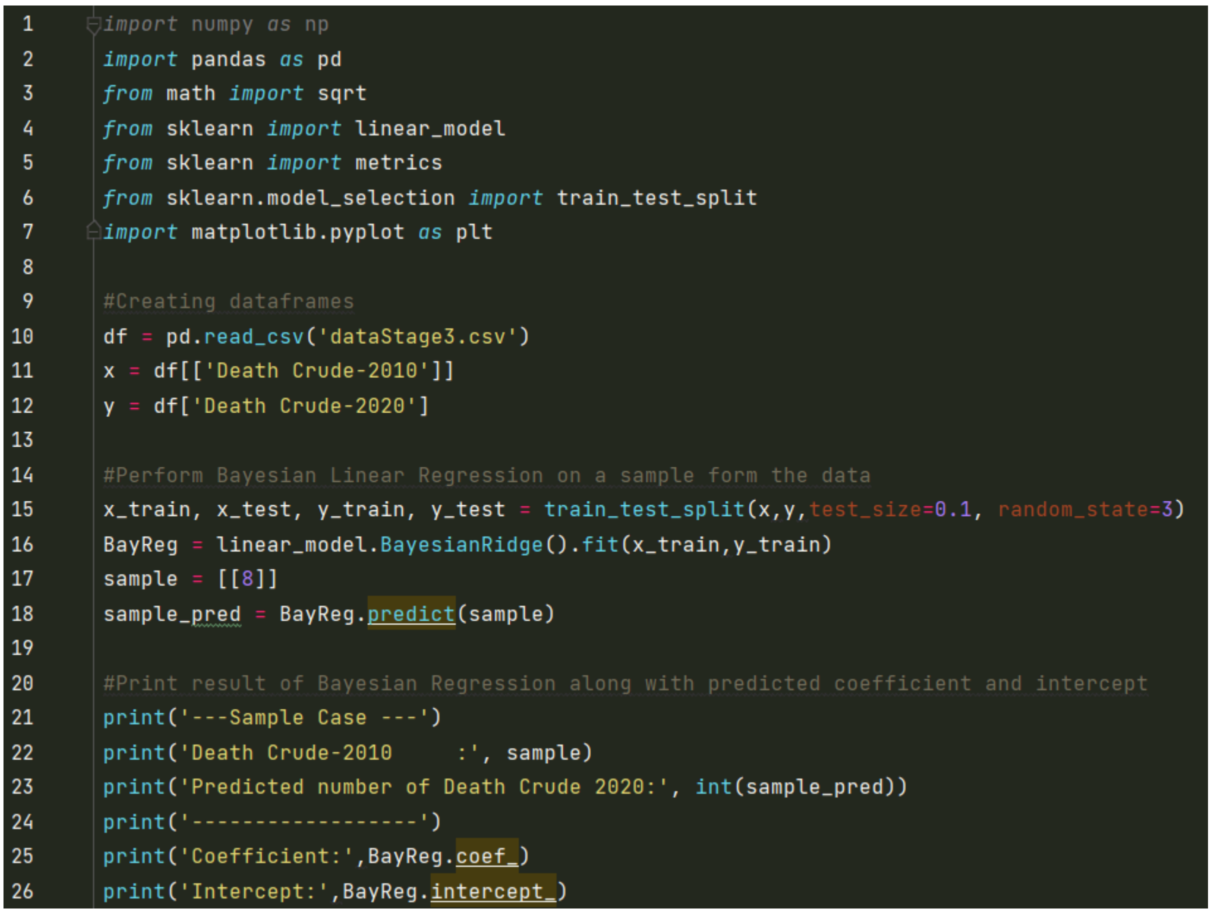
Based on the result, it can be seen that the value is high for the RMSE value and very low for the R-Square percentage, this might indicate overfitting of data and inaccuracy in data prediction. Overfitting could be caused by high variance in the model and data, small training dataset, or complicated data. It can also be said that the model being used, the KNN model, is not explaining a lot of variance in the dependent variable of the data and the model might not be reliable. A solution to this issue might be switching to different models, adjusting the training and test dataset ratio so that there is not too much and too little data for each set, and decreasing the complexity of the model.

There are some limitations which possibly affected the result. The first one being limitations from the dataset itself, although some more cleaning has been done to the dataset used for this stage, there might still be some flaws in the data, an example would be incorrect values or data. Another possible reason as to why the error values are high might also be due to a couple variables having a higher range compared to the other datas, this could be seen in the income data where there is a huge gap on the last few points. Variables presented don't always affect each other, meaning there is necessarily no strong causation or correlation among each variable. Overall, KNN is a simple yet powerful machine learning technique which works well with huge amounts of datas, yet it has some weaknesses including using a distance based learning making it complicated to determine the distance and also attributes used to produce the best fitting result, another one would be high in terms of computation cost.

Member 3 - khol3902

<u>Bayesian Linear Regression - Predicting a Country's Crude Death Rate in 2020 using their Crude Death Rate from 2010.</u>

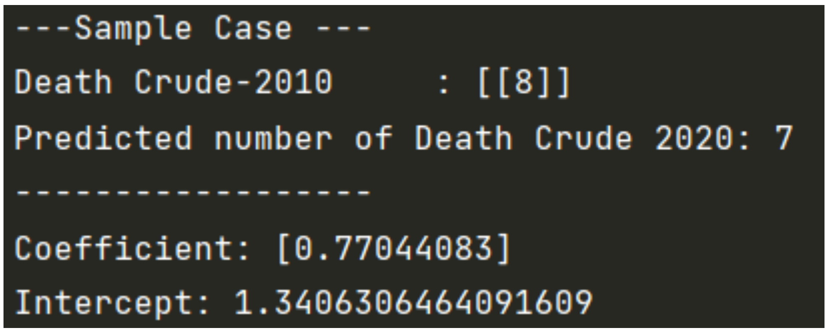
The Bayesian Linear Regression machine-learning technique assumes that the input variables *x* exist as probabilities in a normal distribution and using linear combinations of these we produce an output variable *y* whose values exist as probabilities in a normal distribution. This means that the uncertainty of our model is already accounted for by the probability distribution of *y*. In small datasets a normal linear regression would not account for the high amount of uncertainty that a lack of data brings, and so it becomes more difficult to determine if the observed trendline is accurate. With Bayesian regression, since our model is already probabilistic, the trend that is 'most likely' to be an accurate fit of the data can be calculated. This is good for our dataset where we have a relatively small amount of data, with each variable only having 152 data points. If our dataset was to approach infinity, then this 'most likely' trend would approach the trend found using normal linear regression. The code belows shows a sample prediction for Death Crude-2020 using Death Crude-2010:



```
1   import numpy as np
2   import pandas as pd
3   from math import sqrt
4   from sklearn import linear_model
5   from sklearn import metrics
6   from sklearn.model_selection import train_test_split
7   import matplotlib.pyplot as plt
8
9   #Creating dataframes
10  df = pd.read_csv('dataStage3.csv')
11  x = df[['Death Crude-2010']]
12  y = df['Death Crude-2020']
13
14  #Perform Bayesian Linear Regression on a sample form the data
15  x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.1, random_state=3)
16  BayReg = linear_model.BayesianRidge().fit(x_train,y_train)
17  sample = [[8]]
18  sample_pred = BayReg.predict(sample)
19
20  #Print result of Bayesian Regression along with predicted coefficient and intercept
21  print('---Sample Case ---')
22  print('Death Crude-2010     :', sample)
23  print('Predicted number of Death Crude 2020:', int(sample_pred))
24  print('------------------')
25  print('Coefficient:',BayReg.coef_)
26  print('Intercept:',BayReg.intercept_)
```

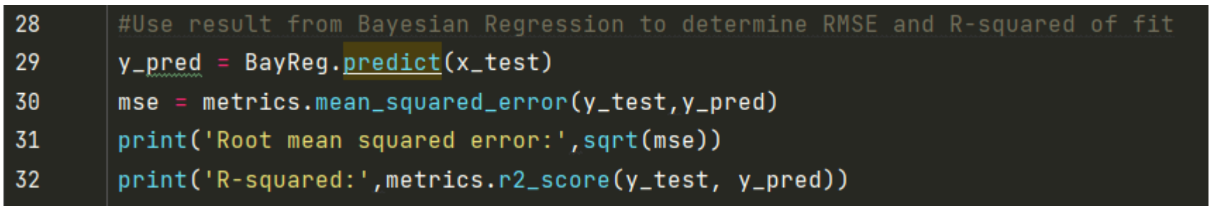
- Lines 1-7: importing relevant packages
- Lines 10-12: Create dataframe object from dataStage3.csv and then create variables x and y from columns Death Crude-2010 and Death Crude-2020 respectively.

- Lines 15-18: Splitting the training and test data to a 1:9 ratio, using Bayesian Regression to fit data, taking a sample number and then predicting the y-value of this sample in the Bayesian model
- Lines 21-26: Print the prediction from the sample, along with the coefficient and intercept of the Bayesian Regression. Output shown below



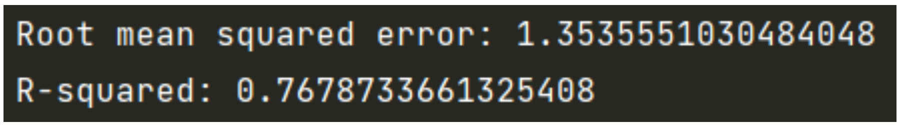
```
---Sample Case ---
Death Crude-2010      : [[8]]
Predicted number of Death Crude 2020: 7
------------------
Coefficient: [0.77044083]
Intercept: 1.3406306464091609
```

With this Bayesian Linear Regression model we can determine the root mean squared error (RMSE) and R-squared values to determine how well the model fits the data.



```
28   #Use result from Bayesian Regression to determine RMSE and R-squared of fit
29   y_pred = BayReg.predict(x_test)
30   mse = metrics.mean_squared_error(y_test,y_pred)
31   print('Root mean squared error:',sqrt(mse))
32   print('R-squared:',metrics.r2_score(y_test, y_pred))
```

- Line 29: Using the Bayesian Regression model, predict what y-value we would obtain from each x-value in the data.
- Line 30: Calculate the mean squared error between the Bayesian model and predicted y-values.
- Line 31-32: Print the root mean squared error and the R-squared value of Bayesian trend. Output shown below.



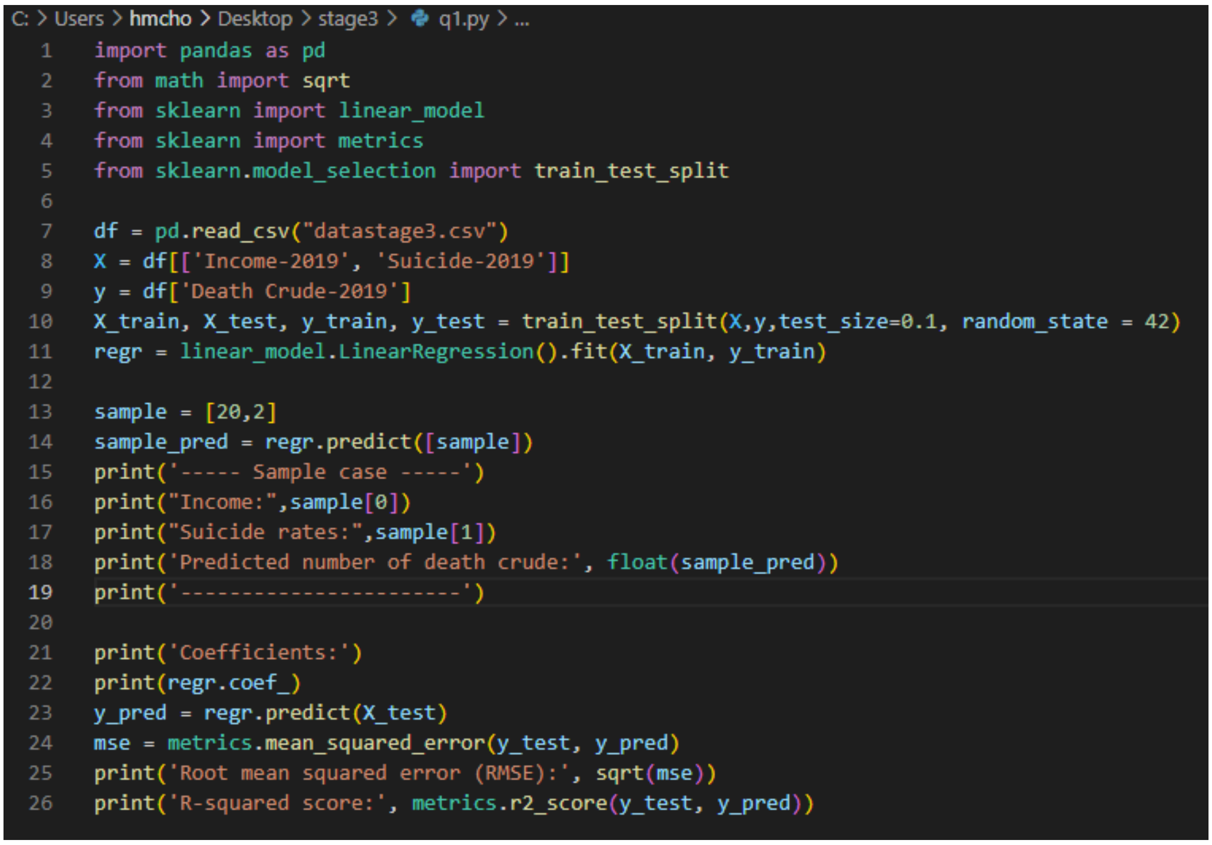
```
Root mean squared error: 1.3535551030484048
R-squared: 0.7678733661325408
```

We have shown the coefficient of our Bayesian Linear Regression model is 0.77 with a y-intercept of 1.34. The R-squared value of 0.78 (2 sig.fig) indicates that the model fits pretty well with the test data we had and the predicted data. However, the RMSE value of 1.4 (2 sig.fig) may be a bit higher than what would be ideal, which slightly decreases the predictive power of our model. From the coefficient we can determine that there is a positive correlation between a country's crude death rate in 2010 & 2020, indicating that a higher crude death rate in 2010 will result in a higher crude death rate in 2020. That said, the coefficient is less than 0, which means that a country's crude death rate still decreases with time. This approach has proven to be fairly reasonable, producing a linear model that conforms reasonably well with the data provided and has a clear trend.

Member 4 - hcho8530

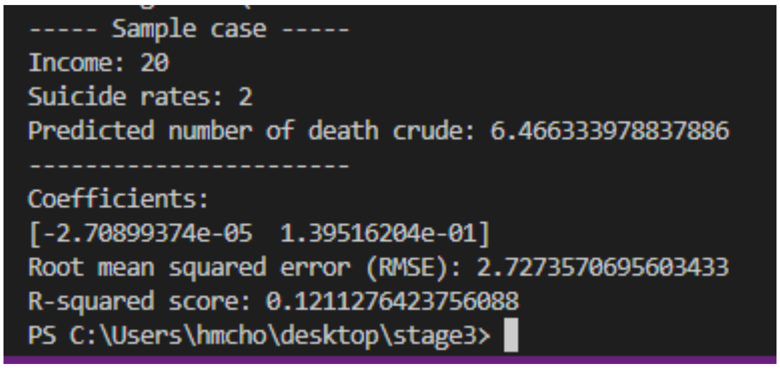
Linear regression - predicting death crude with suicide rate and income

The main goal of machine learning is to create a model based on a real data and predict the output that will be given when another input is given. In this situation, the most baisic and intuitive model is a line. Therefore, linear regression is the most commonly used, and basic model when predicting an output. For the linear regression of dataset “datastage3.csv”, “Income-2019” and “Suicide-2019” was used to predict a value for Death Crude. Below is the python code that was written for the analysis.



```
C: > Users > hmcho > Desktop > stage3 > q1.py > ...
1   import pandas as pd
2   from math import sqrt
3   from sklearn import linear_model
4   from sklearn import metrics
5   from sklearn.model_selection import train_test_split
6
7   df = pd.read_csv("datastage3.csv")
8   X = df[['Income-2019', 'Suicide-2019']]
9   y = df['Death Crude-2019']
10  X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.1, random_state = 42)
11  regr = linear_model.LinearRegression().fit(X_train, y_train)
12
13  sample = [20,2]
14  sample_pred = regr.predict([sample])
15  print('----- Sample case -----')
16  print("Income:",sample[0])
17  print("Suicide rates:",sample[1])
18  print('Predicted number of death crude:', float(sample_pred))
19  print('-----------------------')
20
21  print('Coefficients:')
22  print(regr.coef_)
23  y_pred = regr.predict(X_test)
24  mse = metrics.mean_squared_error(y_test, y_pred)
25  print('Root mean squared error (RMSE):', sqrt(mse))
26  print('R-squared score:', metrics.r2_score(y_test, y_pred))
```

Line 1-5: importing libraries for the code
Line 7: reading csv file “datastage3.csv” that is going to be used for analyize
Line 8-9: slicing dataframe for target/input variables
Line 10-11: setting test size and random states
Line 13-14: creating a sample value to test on
Line 15-19: print out the results of the prediction
Line 21-26: print out the details of RMSE and R^2 scores



```
----- Sample case -----
Income: 20
Suicide rates: 2
Predicted number of death crude: 6.466333978837886
-----------------------
Coefficients:
[-2.70899374e-05  1.39516204e-01]
Root mean squared error (RMSE): 2.7273570695603433
R-squared score: 0.1211276423756088
PS C:\Users\hmcho\desktop\stage3>
```

As can be seen on the results, RMSE value returns 2.72 and R^2 score returns 0.121. Low R^2 value indicates that Income-2019 and Suicide-2019 value does not explain much about Death Crude rate. Also, RMSE value is quite high which tells that this predictive model does not function greatly on given datasets. My evaluation regarding a result is that the variables itself might not have strong correlations since the code produced better accuracy when tested within the variables with proven strong correlations.

PART B

The **first predictive model** within this report that was produced was a linear regression model. This supervised machine learning model represents the relationship between input values to estimate the value of a target value based on creating a line-of-best-fit. Because of its overall simplistic nature, this technique is often regarded as the most basic and common predictive analysis, yet being a powerful and important model. Some limitations of this model includes the constraints the all attributes have to be quantitative and the model is very fragile to outliers and errors in the dataset.

This predictive model was created through using a combination of the pandas and sklearn library. The pandas library was simply used to read in the database as a pandas object, which aided in splitting the dataset into an input variable set and a target variable set. The sklearn library was used to perform a majority of the function in creating the predictive model. Key steps in the production of the predictive model (following the beginning pandas section) includes the following:

- Splitting the dataset into a training and testing dataset, with a desired 90:10 respective split. This was performed using the *train_test_split()* function appart of the sklearn library
- Generating the linear regression predictive model. This was accomplished using the sklearn *linear_model.LinearRegression()* function, taking in the input and target training data set as parameters.

The evaluation of this predictive model can be investigated based on the coefficients, root mean squared error (RMSE), and R-Squared. The coefficient of the *Income-2021* is 0.00510949, while the coefficient of *Litearcy Value* is -0.054736. From this, it demonstrates that the *Income-2021* has a greater association with *Death Crude-2020* compared to *Literacy Value*. The RMSE score of this model is 1.192 which is quite high, as a lower score is better. This high RMSE shows that the data is not very well concentrated around the line of best fit. The value of the R-squared is -0.0712, which is very low, as a score closer to 1 is better. This shows that the independent variables of *Income* and *Literacy Value* isn't explaining much in the variation of the dependent variable *death crude rate*. From these three evaluations, this predictive model isn't very accurate in being able to predict the target attribute.

The **second predictive model** which is the K-Nearest Neighbor Model was created by using the pandas library and sklearn package from python. As for the plot, it was made with a pyplot. The K-Nearest Neighbor Model can be used both for regression and classification, in regression, mean of k nearest data points is calculated as the output, while for classification voting is applied over the nearest data points. Its hyperparameter includes the k value, the number of neighbors to participate in the algorithm and the distance function which is the Euclidean distance is the most used similarity function. Although this method is quite appealing, it can become impractical when the dimension of the dataset is increased or when there are many independent variables. The overall evaluation of death crude prediction through KNN model is not very satisfying as its error scores are pretty high meaning that the prediction of the death crude based on the income-2019 and suicide-2019 is somewhat inaccurate, this might also indicate that each variables doesn't affect each other completely, thus making it a limitation for this approach.

The **third predictive model** was produced by using the sklearn Python package, specifically the BayesianRidge function from the linear_model section of the package. What this function does is estimate the set of independent variables (what we classify as the x-axis on a graph) as a normally-distributed function (more colloquially known as a bell curve) and then calculates the probability of a y-value occurring as observed in the data set you are using. This means that the y-values are also expressed as a probability function, which means that there are multiple possible trendlines that could describe the relationship between x- and y-values. The Bayesian Regression function can display which of these functions is the most likely one to occur, thus giving us a model that can be used to determine the relationship between a country's crude death rate in 2010 versus its crude death rate in 2020. Looking at the evaluation of this model, we see that although it isn't perfect, it does give us a fairly robust model for this relationship which we can use to make some inferences about the relationship between the two variables, such as higher death rates in 2010 correlating to higher death rates in 2020 or the steady decline in death rates over time. This specific model would fail if the relationship was not linear, as it is a Linear Bayesian Regression, and so when using the model a person has to think if there are other, possibly non-linear, models that would more accurately describe the relationships between data points.

The **predictive model 4** was produced by sklearn package based on pandas csv. Two variables Income-2019 and Suicide-2019 was used for linear regression to produce predictions on Death crude and the code itself calculates the RMSE value and R^2 score of the predictions. When tested with several sample values, it seemed to return legit values for death crude which was around 6-7 however high RMSE score and low R^2 value indicated that this model does not function accurately on given datasets. When building this model, linear regression method seemd quite fitting for the dataset however it did not produce satisfying results. After testing the model, I could not find a specific reason why this was failing so I made a hypothesis about the variables themselves do not correlate much. The backup reason for this is that when tested with variables those already are proven to have strong correlations, the model seem to produce accruate outcomes.

The Linear Regression model is the most common and basic predictive analysis. There are variations within this model, but in its simplest form, it produces a straight hyperplane of one's dataset. A leading strength of this predictive model is its straightforwardness to understand and comprehend. Additionally, new data can be added using stochastic gradient descent. A major weakness of this model occurs when the relationship is non-linear. Linear regression does not naturally capture non-linear patterns well, with alterations to capture correct relationships tricky and time-consuming. All attributes within this model have to be quantitative, so encoding would have to be used to convert any categorical or order attributes to quantitative. Furthermore, this model is quite fragile when the dataset contains errors and outliers, requiring time for pre-processing to ensure the dataset is clean.

When comparing Linear Regression to other models, the strengths of linear regression begin to fade. Other predictive models such as Elastic-net, Ridge, and LASSO often outclass linear regression because of its regularisation, which is a technique used to avoid overfitting. The predictive model KNN is sometimes favored over linear regression as it does not require a linear relationship, which linear regression does.

The KNN model is considered to be one of the simplest and easiest machine learning methods to implement as it has only a few hyperparameters to modify in order for the prediction model to function. Moreover, the KNN model requires no training before concluding the prediction, the algorithm's accuracy won't be affected even when new data is added. It came with some disadvantages as well. The KNN model is not suitable for predicting large datasets as it will degrade the accuracy of the algorithm. Aside from the large dataset, the algorithm also doesn't work well with high dimensional data as it becomes complicated for the algorithm to find the distance for each point. KNN algorithm has high sensitivity towards null data or any missing values, any noise in the data might trigger false prediction and high error value.

When comparing KNN to other machine-learning models, the main difference which can be spotted is the computation time needed, KNN in general needs larger real-time computation compared to most machine-learning techniques. KNN is much slower in terms of computing cost compared to the Bayes method, another difference between the two methods is that Bayes is parametric whereas KNN is non-parametric. Similarly to the Bayes model, the Linear model is a parametric model, unlike KNN. KNN is relatively preferable to be used when the data have high SNR

Bayesian Linear Regression is an effective machine learning statistical method for when a researcher has a small dataset where each data point appears to be in a normal probabilistic distribution and they believe their variables have some linear correlation. With normal linear regression methods small datasets can often lead to inaccurate models due to a lack of uncertainty. Bayesian regression propagates these uncertainties as they are an integral part of the process, the trends calculated are based on probability and the most likely trend can be chosen from these probabilities. This provides a clearer understanding of the uncertainty within the trend, providing a more complete image of the dataset. As datasets become larger, then the uncertainty within the Bayesian regression will decrease, making it suitable for a wide range of dataset sizes.

This method is weak in that if one's data does not have a normal distribution then getting a proper fit for the data will require modifying the hyper parameters of the Bayesian equation, which can be complicated or confusing for someone who is not well versed in the underlying mathematical theory. It would also be inappropriate to use Bayesian Linear

Regression when there’s a nonlinear relationship between variables, and so other Bayesian Regression methods would need to be used instead.