# 

# 

# DATA1002 Project Stage 3 Report

Group : **L19-G4.**

Members :

(aguo4791, 520644847)

(rcha0546, 520392823)

(slpa9090, 500545654)

(npai5889, 520617829)

# Part A

Note to markers: We wish for this report to be **individually marked**. Thank you!

## 1. Introduction

#### Dataset Description

The dataset chosen is the same as in stage 2. The public dataset, ‘SDG\_goal3\_clean.csv’ was collected from the United Nations Statistics Division (UNSD) which lists health related data by country and year between 2000 and 2015. The long schema dataset came precleaned, and as such no cleaning was performed.

## 2. Producing predicted models

This group analysed our variables using the following models: Linear Regression, KNN, Neural Network and Decision Tree.

The following section details the identifier in the dataset chosen and the members responsible for handling the data analysis of this feature.

### 2.1 Linear Regression

520392823 was responsible for the completion of this section.

1. Linear regression was used to make predictions about Maternal Mortality Ratio(MMR).
2. The algorithm was chosen as a reference point due to its simplicity. As it is the simplest form of regression there is, comparing this to other models would allow us to see the improvements made with regards to this dataset.
3. It is a capable model applicable to many scenarios.

#### 2.1.2 Model explanation with code

* The pre-processing that was done involved removing the following columns: MMR and all other columns which were not our target variable.
* The columns which were kept were

*Fig.1. Independent variables*

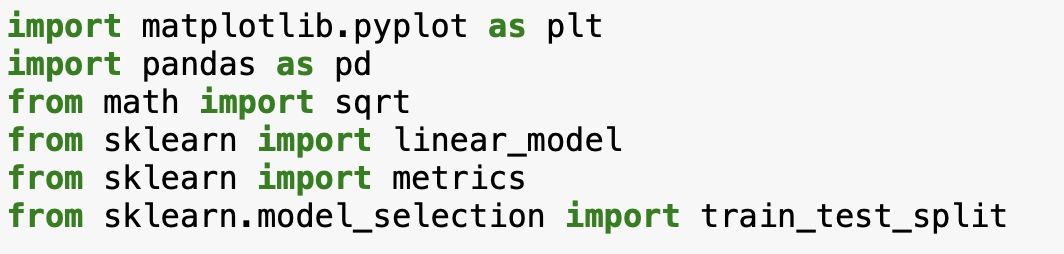
| **Independent variables** |
| --- |
| * Proportion of births attended by skilled health personnel (%) * Adolescent birth rate (per 1000 women aged 15-19 years) * Universal health coverage (UHC) service coverage index * Physician worker density (per 10000 population) * Nurse Midwife worker density (per 10000 population) * Pharmacist worker density (per 10000 population) |
|
|
|
|
|
|

* Unlike 520644847’s choice to include Region I did not feel the region column to be necessary, as such I did not include it in my model.
* For hyper parameters I did not make any changes as there weren’t any changes to be made in simple linear regression. This model already exists in sklearn and was simply applied in this case.

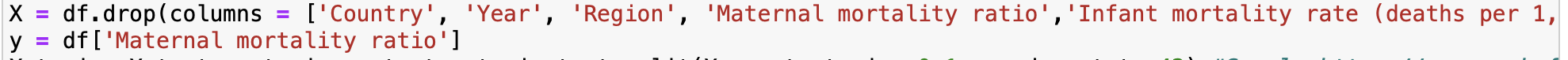
The code I used to produce this model is as follows:

* First I imported the relevant libraries

*Fig.2. Libraries imported*

* 
* Next I read the csv using pandas, stored it in a dataframe and removed irrelevant columns including MMR. I stored this dataframe as X, and I also stored the MMR values from the original data frame in a variable called y.

*Fig.2.5. Dataframes and storing variables*

* 
* Then I produced the actual model itself

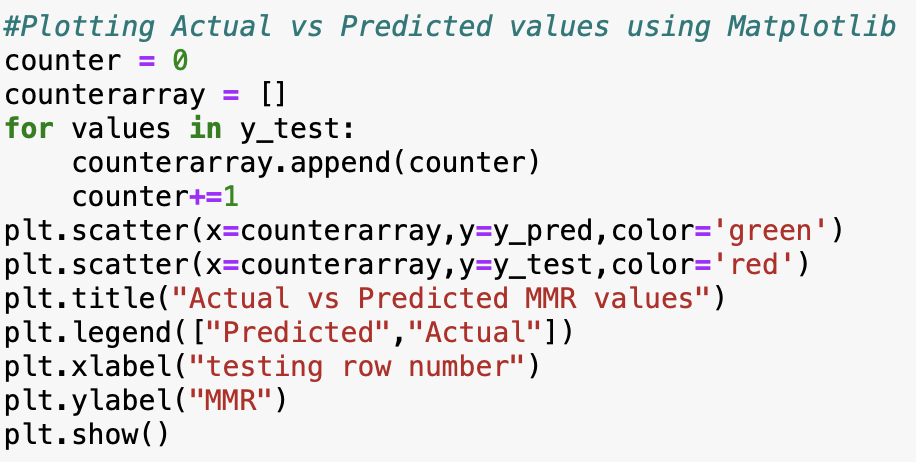
*Fig.3. Model Production (Scikit-Learn)*

* 
* Then I was able to pass these values to the metrics functions to get the following metrics
  + The metrics calculations except for r2 were built into scikit. (Scikit-Learn *Metrics and scoring)*
* The calculation for adjusted r2 score was not a ready function in scikit and as such had to be done manually through code. (Aritesh, 2018)

*Fig.4. Metrics Production*

* 
* Lastly to produce the graph the code I used was

*Fig.5 Plotting actual vs predicted graph code*

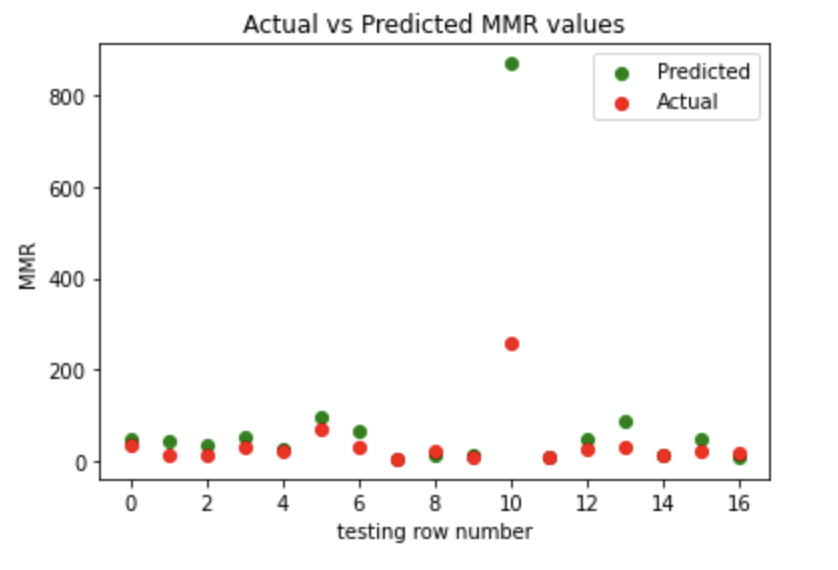
* 

#### 2.1.3 Output metrics

*Fig.6. Output metrics*

* 
* The r2 score is negative, while mathematically this is not possible, in sklearn it tells us that the model is suboptimal for this dataset simply terrible. However, it can still be explored as to why.
* To explore this further I looked at the individual regression coefficients.
* These are the slopes(m) in the y=mx+b of each individual X variable. What regression coefficients tell us is for 1 unit increase in the predictor, we expect MMR to increase by the slope. However, we see slopes of 1, -1, -0.6, 0.03, it tells us that MMR does not increase significantly. With many of the predictors barely having a significant relationship, it is a possible clue as to the model’s performance. Across these we can see one strong negative slope at -9, while most of them are around -1 to 3, with only one reaching 3.
* In addition, we can also look at a scatter plot of the predicted values vs the actual values for the testing data.

*Fig.7. Model Prediction vs Actual values Graph*

* 
* As we can see here, at testing row number 10 the predicted MMR is way higher than the actual MMR. I suspect this may be leading to the high mean absolute error value as the mean in general is susceptible to large outliers.
* I also kept using random\_state = 42. Changing random state leads to selecting a different set of testing data each time resulting in different metrics (cs95, 2018)

### 2.2 Neural Network

520644847 was responsible for the completion of this section.

#### 2.2.1 Model explanation with code

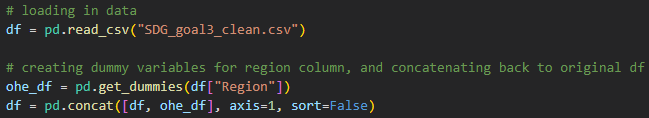
A Multilayer Perceptron was used to create an artificial Neural Network model, to predict the Maternal Mortality ratio (target feature of our data). This was used as it is a numeric regression model which works perfectly to predict our target feature and provides a different way in predicting values than other models used previously.

Though the data came precleaned, there was still preprocessing to be done on the data. The first step was to one hot encode the ‘Region’ column, showing which region an observation came from (Africa, America, Asia, Europe and Oceania). No more data transformations were performed as the rest of the features to be used were already numeric. Lastly, all the non features were dropped and it was decided that we would keep the following features.

| **Features** |
| --- |
| * Proportion of births attended by skilled health personnel (%) * Adolescent birth rate (per 1000 women aged 15-19 years) * Universal health coverage (UHC) service coverage index * Physician worker density (per 10000 population) * Nurse Midwife worker density (per 10000 population) * Pharmacist worker density (per 10000 population) * Region (One hot encoded) |
|
|
|
|
|
|

*Figure 8 : Features chosen for model*

The code used to preprocess the data can be seen below :

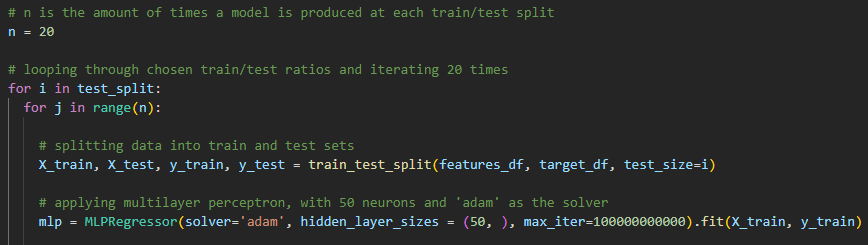


*Figure 9 : Python code used to one hot encoding the ‘Region’ data*



*Figure 10 : Python code used to drop all non-features of the data*

The machine learning model was generated using ‘MLPRegressor’ in the ‘sklearn.metrics’ library in Python. Using ‘train\_test\_split()’, the data was split into training and test splits, and fed into the ‘MLPRegressor()’ ML builder. A hidden layer of 50 neurons was chosen and the solver used to converge the data was the ‘adam’ solver. This was iterated through for 5 different train/test splits (0.1, 0.15, 0.2, 0.25, 0.3) and each train/test split was repeated 20 times, to average out the end statistics. The code used to produce this can be seen in the figure below.



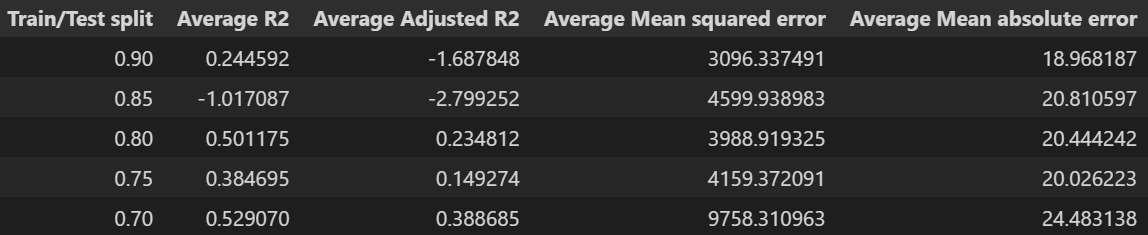
*Figure 11 : Python code used to form Neural Network model*

Note : Neural networks are black boxes, which mean that it is impossible to know how the neurons interact to predict a variable. Thus it is also impossible to know how much each variable influences the predicted variable. This is why no details about the neural network model are provided.

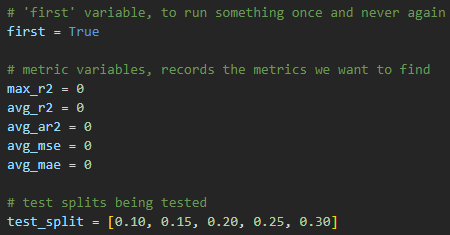
#### 2.2.2 Output metrics

The Neural network underperformed at lower train/test splits but at higher splits the model did average. From the summary table below, we can see that at higher test splits the model has a higher R-squared and adjusted R-squared than the lower test splits. The highest average R-squared achieved was for the 0.7 train/test split, where the R-squared was 0.53. This would suggest a medium correlation, where the splits suggest a weak correlation.

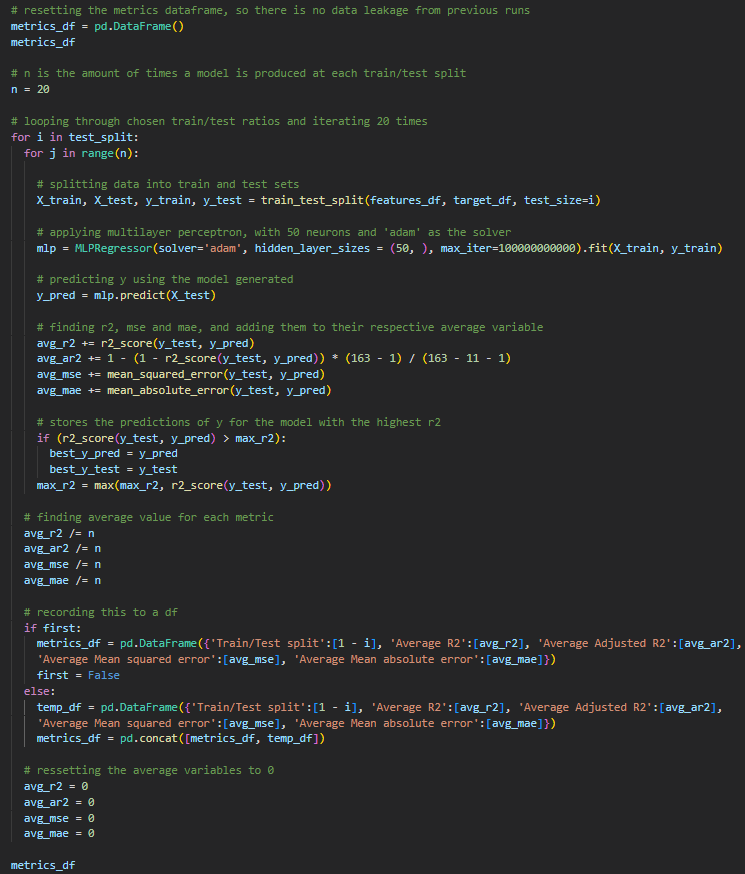
The statistics were calculated for the Neural Network using the loops used to fit the models in the first place. Initially, average metric variables were initialised so that they could be used in the loop. Then the statistics themselves were calculated and added to the initialised variable, then at the end of every 20 iterations, it was averaged and recorded to a dataframe. At the same time, the predictions that had the best R-squared value were recorded and used to plot a scatter graph showing the true values versus the predicted values of Maternal Mortality ratio later on. Once all the metrics were recorded to the dataframe, they were reset to 0 so they could average over the next train/test split. The results table of statistics along with the Python code can be seen below.



*Figure 12 : Table of average statistics at each train/test split, repeated 20 times*

**

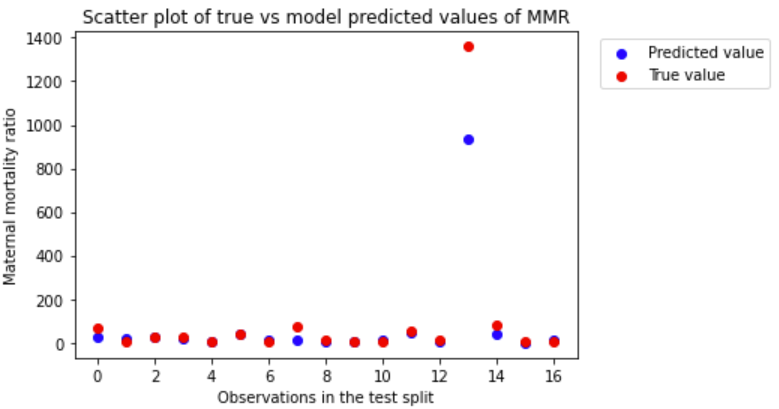
*Figure 13 : Python code of initialising statistic variables*

**

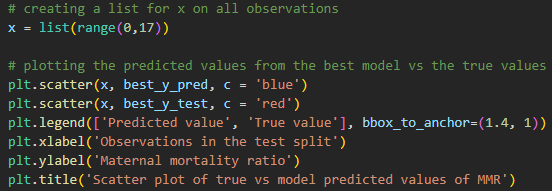
*Figure 14 : Python code of all statistics records*

As seen in the code above, three main choices were made for the hyper-parameters. The solver was chosen to be the ‘adam’ solver, which is the default solver but unlike the other options e.g ‘lbfgs’, the ‘adam’ solver takes more time to converge but is generally more accurate. The second hyper-parameter was hidden\_layer\_sizes, which designates how many neurons should be in the hidden layer. From previous testing, it seemed that at the same split, 50 neurons outperformed 100 neurons (the default) and 20 neurons. At 20, the model was definitely underfitting and had the lowest R-squared values. This can be attributed to the network not having enough neurons to learn properly. However, at 100 we had the highest R-squared values, indicative of overfitting as the network had learnt too much from the training data. This was why 50 neurons was chosen, as this number of neurons made the regressor generalise better. Lastly, the train/test splits were used and had a great impact on the end statistics of the model. As discussed before, it seemed that as the test split grew in size, the model performed better on average.

Lastly, using ‘matplotlib.pyplot’, the best predictions recorded previously were plotted against the actual values on a scatter plot. This is more to get a visualisation of what the machine learning model is predicting, and can be seen below along with the Python code used.

**

*Figure 15 : Python code used to plot the scatter plot of true vs predicted values of MMR*



*Figure 16 : Python code used to plot the scatter plot of true vs predicted values of MMR*

### 2.3 K-Nearest Neighbours

520617829 was responsible for the completion of this section.

1. K- nearest neighbours was used to make predictions about the attribute Maternal Mortality Ratio.
2. This algorithm was chosen because it produces fairly accurate predictions while being simple.
3. It is also versatile and makes no assumptions about the data which gives it a quick running time(Aggarwal, 2020).
4. KNN has 2 hyperparameters K value and distance function(Aggarwal, 2020), it essentially takes an unknown datapoint and checks which point in the known data it is closest to, to then determine the value of the unknown point.

#### 2.3.1 Model explanation with code

1) Python was used to create the model, then scikit - learn,pandas and math libraries were imported.

2) The unnecessary/categorical columns were removed from the dataset and a variable was assigned to the column of maternal mortality ratio.

3) After this the training and testing parts of the dataset were split and set to 90:10.

4) Next an object was created to fit the data and build the model that predicts the value of maternal mortality ratio.

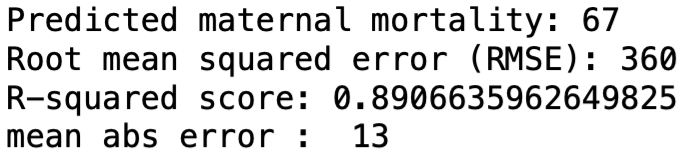
5) The scaling and hyperparameters were not changed.

6) The code used is as follows:



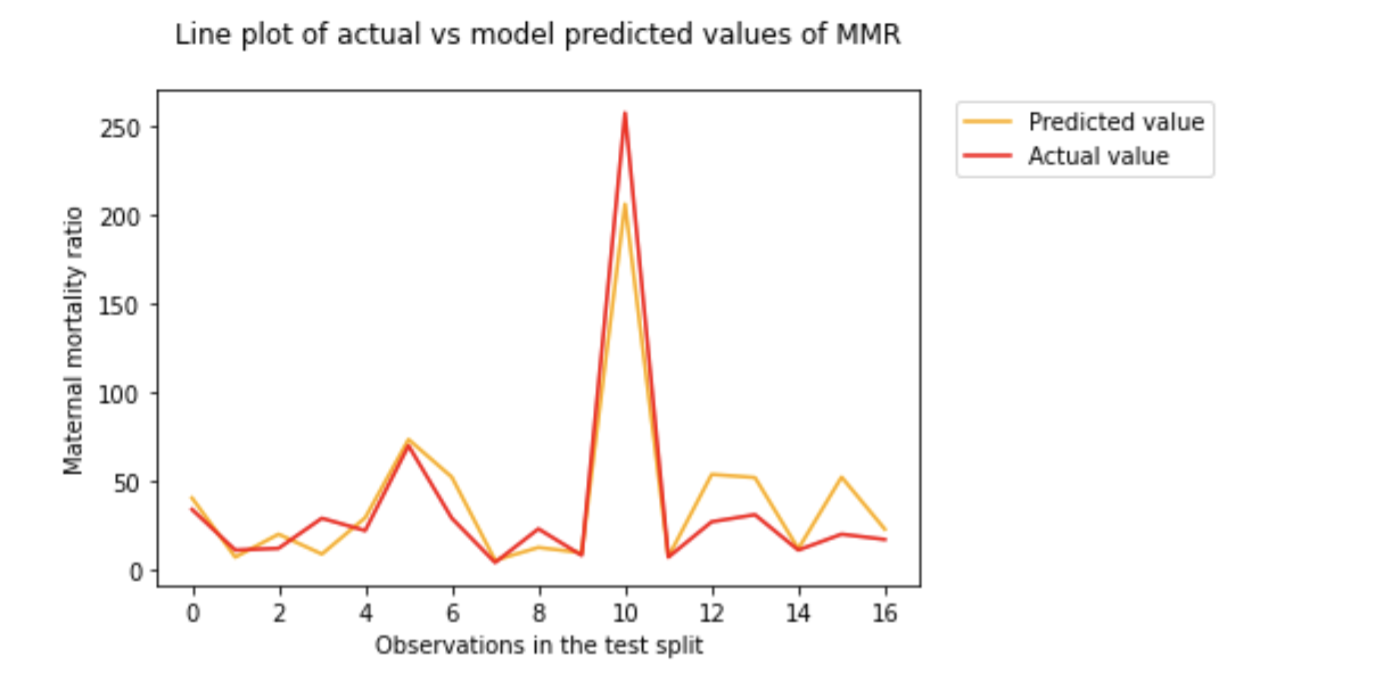
*Fig 17: Code used to produce model and metrics*

#### 2.3.2 Output metrics



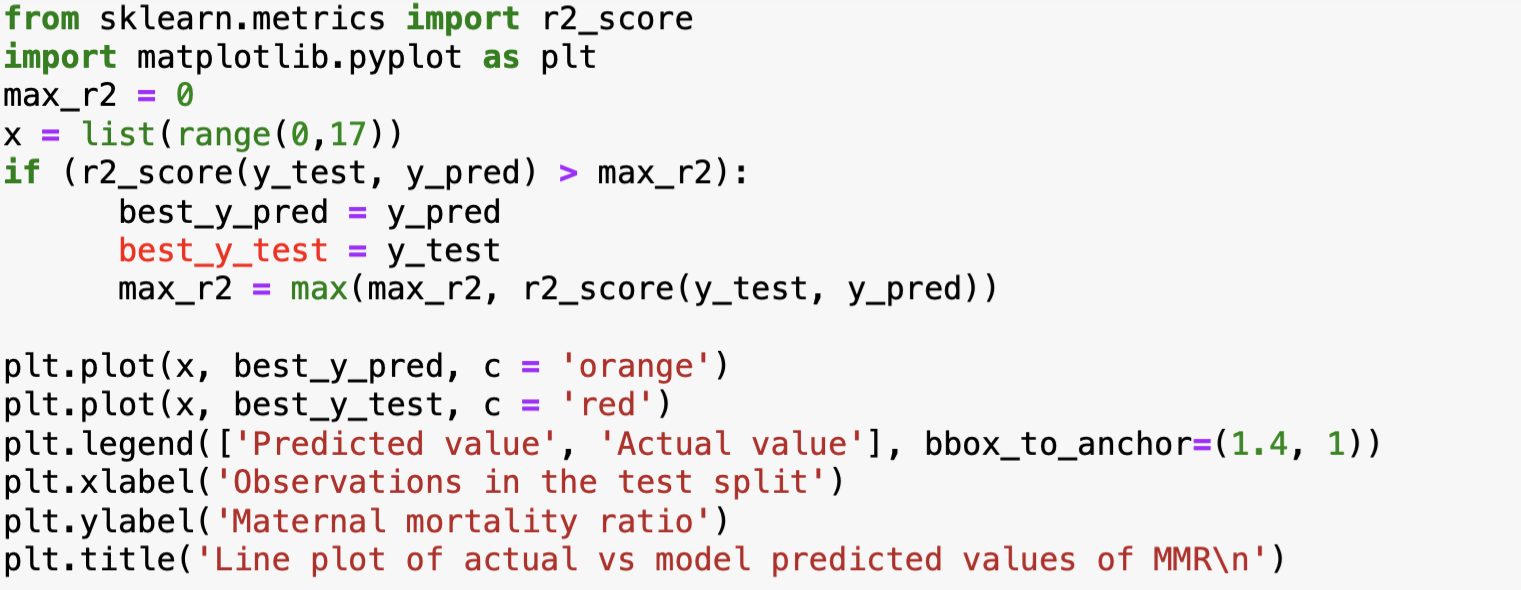
*Fig 18: Metrics calculated*

1. The predicted maternal mortality on entering other features while predicting a sample is 67 this value depends on the value of other features.
2. The r squared score is 0.8906 which means approximately eighty nine percent of the observed variation can be explained by the value of the model’s input.
3. The root mean squared error is 360 meaning the root of the average error of squares is 360.
4. The mean absolute error is 13 meaning the model’s predictions are off by a value of 13.



*Fig 19: Graph to show accuracy of the model*

1. A graph was plotted to show the accuracy of the predictions by the model and the actual values of the test part of the dataset,more specifically maternal mortality ratio.
2. It would be safe to say that this model predicts values to a fairly high level of accuracy.
3. The code used to produce this graph is as follows:



*Fig20 : Code used to produce the above graph*

### 2.4 Decision Tree

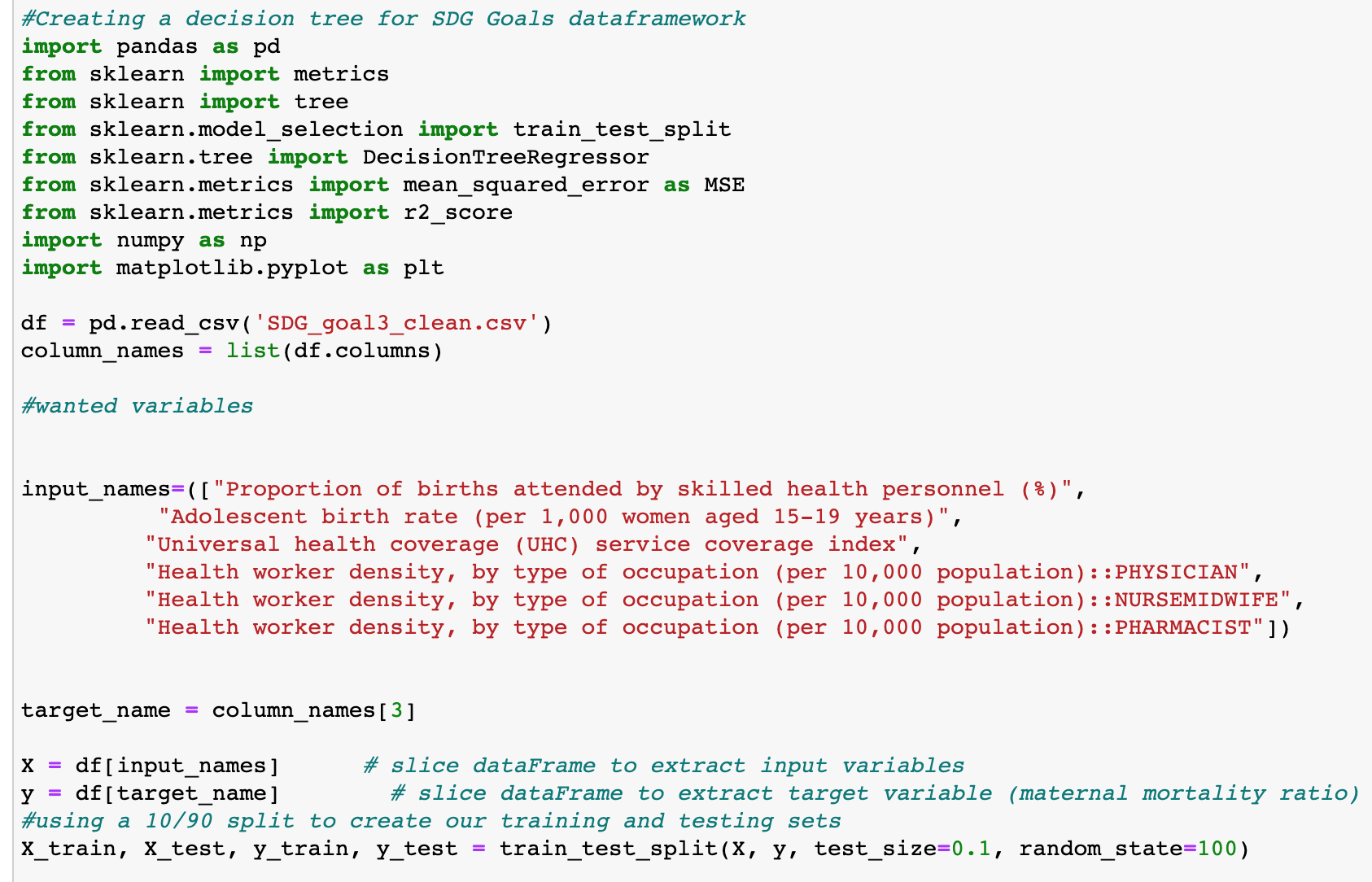
500545654 was responsible for the completion of this section.

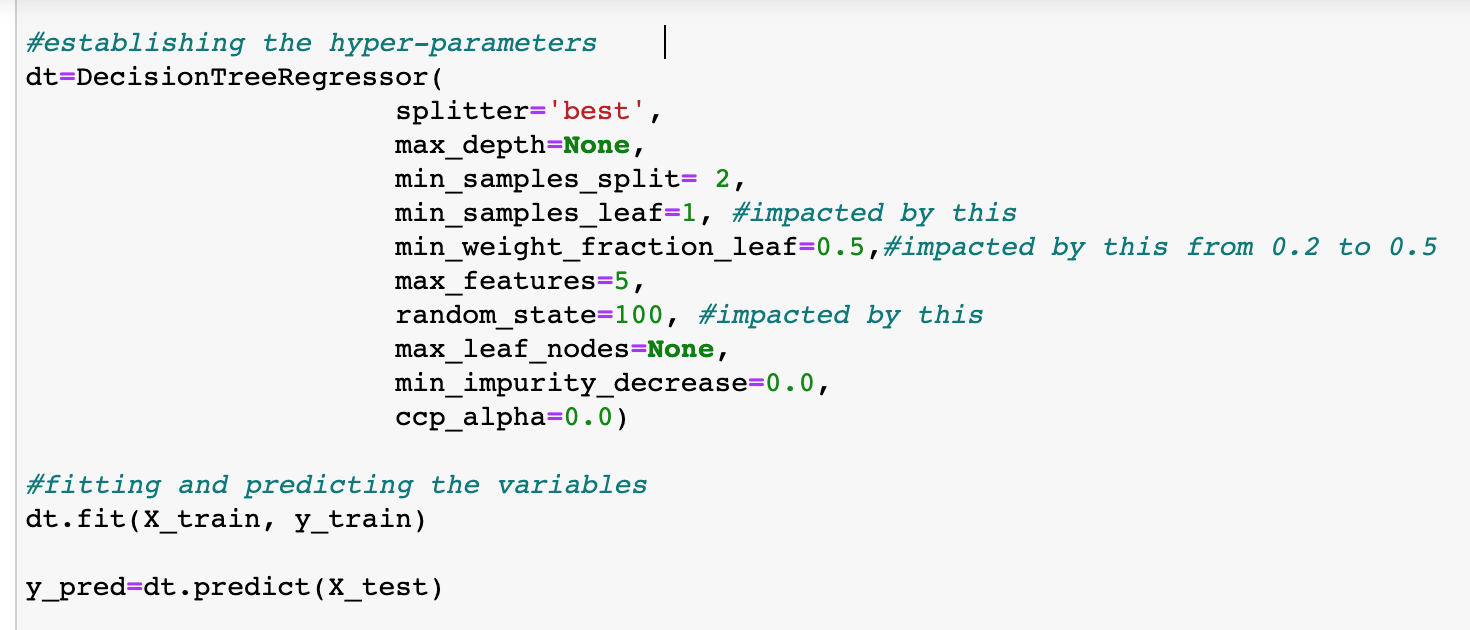
**Methods**

1. Pandas was imported to analyse the dataframe.
2. From sklearn, “DecisionTreeRegresor” was imported because our dependent variable is a quantitative variable (MMR).
3. The relevant dependent and independent variables were fed into the “train\_test\_split” function with a 10 % to 90% split of our data from test to train to create 4 separate items These were the X and y train arrays, followed by the X and y test arrays.
4. The hyper-paramters were established(i.e. Max\_depth etc.) with most except for “min\_weight\_fraction\_leaf” set to default (reasons will be explored in part B).
5. The newly tuned “DecisionTreeRegresor” function was applied and the training arrays were fitted and predicted to extract the “y\_pred” variable which holds the predicted variables for y.
6. From sk-learn metrics, the r2\_score and mean squared error function was imported to calculate the main evaluation metrics.

#### 2.4.1 Model explanation with code

*Fig21 :*





*Figure 22: Code used to develop a decision tree regression model*

#### 2.4.2 Description and justification of chosen model

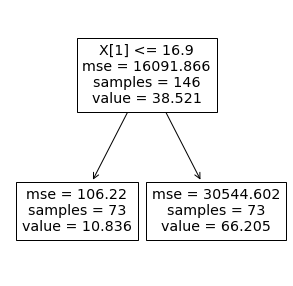
Decision trees are supervised predictive models that determine a target value based on a set of binary rules. A decision tree is used to reach an estimate based on performing a series of questions on the dataset and by narrowing down values based on true/ false questions.

The choice of hyperparameters are essential in ensuring an accurate model. For example, the splitting of the model and the number of nodes available in the tree is decided by evaluating the Mean Squared Error (MSE) for each stage. Hence, in the model above, the parameter: max\_depth was set to its default value of “None” where the nodes will be expanded until all leaf nodes are pure. This is important or otherwise limiting the leaf nodes/ stages may ensure that the model is stopped at a high MSE when more nodes could have been evaluated to further ensure accuracy. However, it is noted that a default max\_depth may lead to overfitting and vice versa. However, as min\_samples\_split and min\_smples\_leaf are directly related to max\_depth, we have chosen the default values of 2 and 1 respectively.

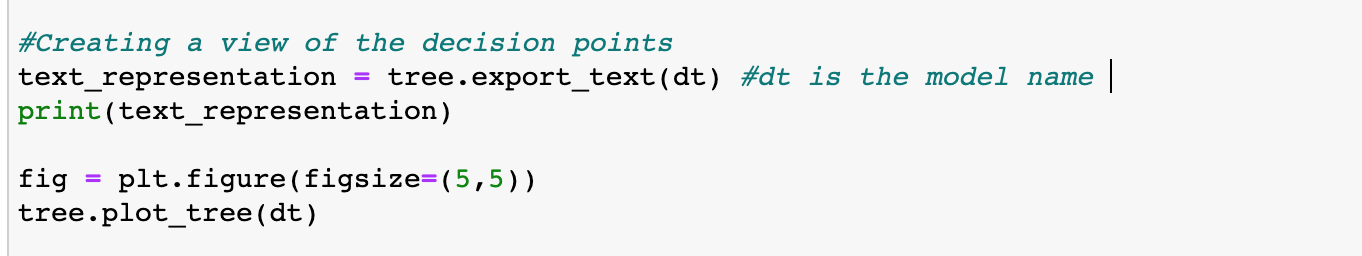
The choice to use decision tree models (not included in the grok modules) have multiple benefits:

1. Decision Tree models are effective for non-linear models which is beneficial for chosen variables as one unit increase in a significant factor like Proportion of births attended by skilled health personnel (%) (PHB%) may have a non-linear reduction in MMR.
2. Techniques are available to develop regression models (i.e. models with quantitative dependent variables).
3. The output of a decision tree can be easily understood by an audience.
4. It is generally believed that the creation of decision tree regression models are relatively easy. This is beneficial as for this particular task and other machine learning tasks , the decision tree regression model can be used as a supplementary technique to compare evaluation metrics. For example:
   1. no data preparation is required as missing values and outliers don’t largely influence model
   2. no normalisation of the data is required.

**Overview of the Decision Points**

An example of the decision points created by setting the hyper-parameter maximum depth to 2 for brevity:   
*Fig23 : Example of decision points with a maximum depth of 2*

Note how an MSE is calculated for each stage of the tree to determine the decision for the next stage.



*Figure 24: Code used to produce the decision points*

# Part B

This report has collectively decided upon (coefficient of determination figure), adjusted and root mean square error ( RMSE) to evaluate the 4 models discussed above. The value represents how much of the predicted variables are represented by the models with the most value. However this evaluation is limited as the addition of new variables (including those that are non-significant) can increase the value. To account for this disadvantage, the adjusted is used. The RMSE is the square root of the difference between the predicted and the observed value. It is useful as its units mirror that of the dependent variable being analysed.

These evaluation metrics have been developed with the sole purpose of benefiting an audience of healthcare data-analysts who use machine learning to inform key health care policies.

## 3. Evaluation

This part of the report is aimed at the following stakeholders.

* Health workers
* Individuals interested in Machine learning/data science
* Government
* UN Women
* WHO
* ProjectCure
* MedShare

### 3.1 Subsection 1

520392823 was responsible for the completion of this section.

A linear regression model is one of the simplest models in regression. It is a very basic model and has a set of assumptions that need to be met eg: Linearity, Independence of Errors, Normality of Errors, Equal variance or homoscedasticity. These must be analysed by plotting the residuals (difference in predictions) and the x variable. However, if any of these assumptions are violated, it means the model simply cannot be effective. Especially with multiple predictors, r^2 always goes up with more predictors added, even if the actual variables do not predict the target or dependent variable better. As such linear regression is quite a limited model, and cannot apply to any dataset.

This issue can also be explained to some extent by the random state. By removing the random state parameter and re-running the program it randomises the parameter each time. However, each time a different test and training set of data is chosen. As a result of this, all the metrics keep changing.

Re-running it with different test splits and random state at 42 shows us that the r2 score does improve. This improvement can be confirmed with the adjusted r2 score also going up. However, even with more testing data, the model’s performance improvements are negligible at best. Interestingly the Mean squared error (MSE) also increases upto test split 0.4 and then starts to decrease, which may be a sign of underfitting past the 0.4 test split. In general we can see that the MSE is quite high in comparison to figures 26, 27.

*Fig 25 : Table of different test splits and metrics*

| R2 score | Adjusted r2 score | Mean absolute error | MSE | Test split | Random state |
| --- | --- | --- | --- | --- | --- |
| -6.044153051107838 | -6.315082014611986 | 42.19695122531944 | 14201.335974038502 | 0.2 | 42 |
| -0.4437871480305349 | -0.4993174229547863 | 45.72075189460375 | 15284.380541965329 | 0.3 |
| 0.4312005609438587 | 0.40932365944169935 | 38.746079191845496 | 19407.906420463045 | 0.4 |
| 0.4330650717151775 | 0.41125988216576115 | 34.09349327227971 | 15781.610098003393 | 0.5 |
| 0.4346822151168066 | 0.4129392233905299 | 30.4242018559312 | 13280.58461286345 | 0.6 |

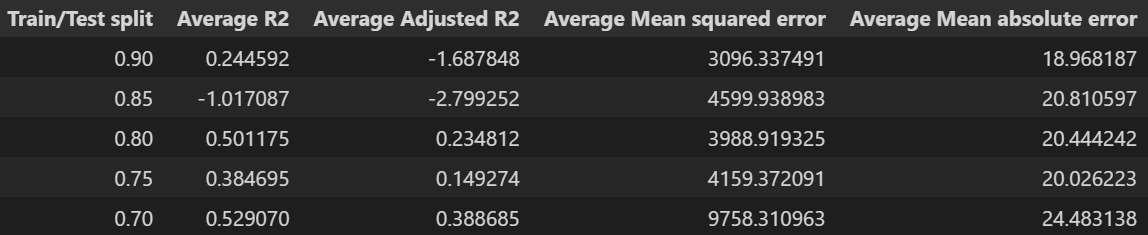
Additionally with the combination of practically infinite random state possibilities and upto 99 training test splits, there are far too many to experiment with for optimal results.

Comparing linear regression to the other models we can see that it performs rather poorly. With the worst r2 score out of all of them at 0.1 training/test split, it can be confirmed that linear regression is not a good fit for this dataset. Perhaps Elastic net regression may be a better fit for this data, as it includes the penalization functions of Lasso and Ridge regression.

### 3.2 Subsection 2

520644847 was responsible for the completion of this section.

Neural Network modelling in Python is done using multilayer perceptron, a machine learning algorithm that simulates an artificial Neural Network. For the dataset, at 5 different train/test splits, a random sample of the data was used to train the model 20 times. This was chosen to eliminate the effects randomness could have on the statistics produced for the model and give us a greater understanding of what train/test splits might work best for Neural Network modelling.



*Figure 26 : Table of average statistics at each train/test split, repeated 20 times*

From the summary table above, we can see that around a 70% train/test split, we produce the best results. This can be for a lot of reasons, but the most glaring of them being that the higher the train/test split chosen, the more the model begins to overfit. The model is taking in far too much information at around 90% (even though 85% with a negative R-squared could be attributed to randomness), making it conform to the training dataset and worse at predicting values it has never seen before. Additionally, another reason for low R-squared might be attributed to the feature selection. With only 7 features, the variation may be extremely high and with more and more features the model may perform better.

On the other hand, the model performs reasonably well at around the 75% train/test split. It’s in the perfect balance of too much data and not enough data in the training set, so that it does not underfit/overfit the data. Additionally, the mean squared error and mean absolute error are reasonable enough, having a RMSE of ~65 in the units of Maternal Mortality ratio.

In summary, Neural Networking is very flexible (being able to perform regression and classification) and reliable when done correctly, and can have extremely promising results. As with all machine learning models, Neural Networks are highly sensitive to the train/test splits chosen (Racz, 2021) and it was found that around 75% was the sweet spot for this particular investigation. Additionally, in hindsight it would have been beneficial to initially test how the model performs with a variety of different features, finding out what features make the model work better and making a final model with those features. Overall, the model converged quickly at around 8 minutes per run, considering it was run at 5 train/test splits, 20 times and gave fairly correlated models at lower splits.

### 3.3 Subsection 3

520617829 was responsible for the completion of this section.

1. K-nearest neighbours is a simple machine learning algorithm as stated above in section 2.3, it is versatile, quick and highly accurate(Aggarwal, 2020).
2. The algorithm worked accurately for this dataset as seen in :
3. Graph in Fig19 of section 2.3: the predicted values are very close to the actual values.
4. absolute mean error which is 13,root mean squared error which is 360 and also from the r squared score which was 0.89, these metrics show that the model is efficient and works well for this dataset.
5. The limitations of this algorithm are that it depends on the accuracy of the data, requires high memory and does not work well with large datasets(Aggarwal, 2020).
6. The algorithm probably worked well with this data due to the high quality and size of the dataset. If the scaling was different or the dataset was large the model might not have been this accurate .
7. Another reason why the model worked efficiently is because it was used with python.(Chatterjee, 2022)

### 3.4 Subsection 4

500545654 was responsible for the completion of this section. Note that in general, the and adjusted figures were generally low compared to what is universally accepted for a decent model (Figure 27). For the decision tree regression model, the and RMSE values significantly varied when one specific hyper-paramter was varied- “min\_weight\_fraction\_leaf”.

| **“min\_weight\_fraction\_leaf”** | **0.2** | **0.1** | **0.3** | **0.4** | **0.5** |
| --- | --- | --- | --- | --- | --- |
|  | 0.21 | 0.20 | 0.16 | 0.13 | 0.10 |
| Adjusted | -0.15 | -0.16 | -0.22 | -0.26 | -0.31 |
| RMSE | 112.12 | 112.93 | 115.52 | 117.72 | 119.85 |

*Figure 27: The changes in evaluation metrics as* **“min\_weight\_fraction\_leaf”** *is varied.*

Interestingly, the model is very sensitive to the value of min\_weight\_fraction\_leaf compared to changes in other hyper-parameters. A figure of 0.5 provides an of 0.10 and 0.1 provides a value of 0.20. It is hypothesised that this is because min\_weight\_fraction\_leaf is defined as the fraction of the input samples required to be at the leaf nodes. Therefore, a higher for a lower fraction may indicate how adding in more data points has reduced the accuracy of our model as there may be something inherently wrong with it. This is further proven by our Adjusted significantly becoming more negative as **“**min\_weight\_fraction\_leaf**”** fraction increased as seen by figure 27. Note that the RMSE value kept in line with the worsening .

When compared to the linear regression model (with all relevant hyper-parameters kept constant), all evaluation metrics were significantly better (the other models had negative figures), suggesting that the existing dataset maybe requires models that are non-linear. The evaluation metrics were also better than the Neural Network method (Adjusted of -1.68 vs the decision model’s Adjusted which were less than < -1 and an RMSE >1000 compared to the decision models values). This could be because Neural Network models work on complicated interconnected processing nodes and require large datasets where as decision tree regression models don’t. However, the KNN model had significantly better metrics which could be because KNN models are non-parametric and doesn’t have any underlying assumption of the data compared to decision models, thus giving a better accuracy score as no hyper-parameters had to be adjusted.

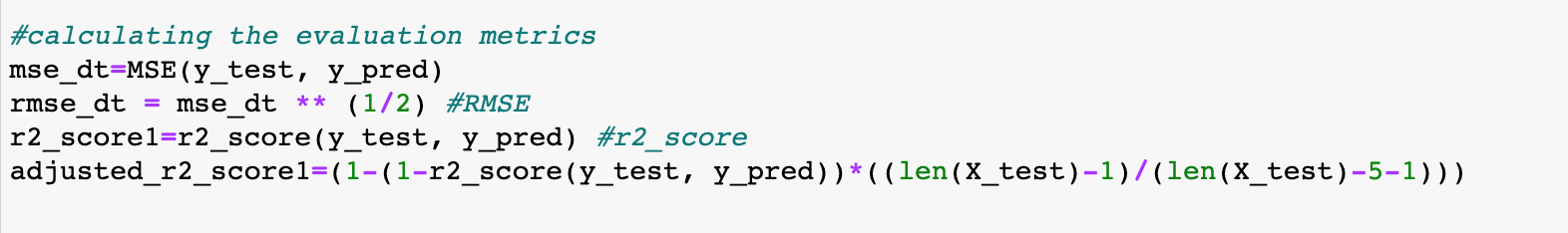
**Determining if model is overfitting or underfitting**

In general, if the training performance is high and the testing results are poor, there is a case of overfitting, and if the training performance is relatively low, there is a case of underfitting. By using the metric of RMSE (the error between observed and predicted values) to test the performance between training and testing results, the model has produced values that ranged between 112-120, approximately. Combined with the fact that average MMR is ~40 compared to the high RMSE, and the model has produced negative adjusted , it can be assumed that there is a significant case of underfitting. This could have been better viewed through a learning curve graph. A better model can be produced through manipulation of hyperparameters.

**Code used to produce the metrics**

These figures were generated by importing the metrics function (mean squared error and score) from Scikit.

*Fig28 : Code used to produce the metrics*



## 

## 4. Discussion and Limitations

By comparing the and RMSE values of all models, the only model that could potentially be accepted is the K-nearest method - not only did it have the highest and lowest RMSE values, the is the one that is closest to .99 which is the one that is universally accepted of a good model (Nau, 2019).

**Linear (520392823)**

Limitations of Linear Regression are:

* The data needs to meet the assumptions layed out in 3.1 Subsection 1 which were linearity, homoscedasticity, equal variance of errors and normality of errors.
* Additionally, with multiple predictor variables, linear regression’s performance may not be reliable. The adjusted r2 score after adding each variable indicates whether that variable was helpful in explaining or not.
* In general the adjusted r2 score acts as a penalization measure, as it reduces when unhelpful variables are added. In figure 6 we can see that the adjusted r2 score is lower than the r2 score. This acts as an indicator in this scenario that not all the independent variables were helpful in explaining MMR, which can also be seen in the regression coefficients with many being close to 0.
* Additionally, linear regression is a simple model which is not flexible unlike Neural networks, KNN or decision trees. Given this simplicity, it means that it is also quite susceptible to different training test splits. As an example of this, keeping the training/test split at 0.1 but changing the random state to 41 results in an r2 score of 0.019329359843506255

**Neural Network (520644847)**

As stated previously, Neural Networks can be very reliable when the hyper-parameters are chosen correctly, but not so reliable when they’re not. One of the underlying biggest weaknesses of the model is its black box nature, which prevents us from seeing the weights of each feature. Additionally, as explored before when incorrect train/test splits and features are chosen, the model can perform suboptimally. Neural Networks also require a lot of computational power more so than other models, such as logistic regression (Tu, 1996), making it more demanding and time consuming to run repeatedly.

**KNN (520617829)**

The limitations of KNN as stated in Part A and B are as follows:

1. It depends on the quality and size of the dataset, both of which were not a problem for the dataset because it was clean and small.
2. While the model has a good R squared score it does not tell how good the fit is.
3. KNN works by finding the nearest cluster’s categorization to categorise an unknown point, meaning it does not work well with datasets that are large, which is fine because the dataset has 163 points.
4. For regressions KNN is worse than linear regression and also slower than a decision tree model (Tayo, 2020).
5. It is highly dependent on scaling(Aggarwal, 2020).
6. It requires a lot of memory.

**Decision Model (500545654)**

Whilst the advantages of this model were stated above in section A, the model has two significant limitations. **(1)** Whilst Scikit does allow for a regression function, in general decision tree models are more suited towards classification problems. **(2)** A small change in the data can result in a major change in the structure of the decision tree. The provided dataset is in long format with MMR that could have been classified according to year and region. As reported by stage 2, there were slight improvement in MMR for each discrete year from 2005 to 2015 and across different geographic regions- these changes in data could have reduced the accuracy of the overall model. If the decision tree was divided according to years and/or regions, and a classification model was created to understand the relationship between years and region, then maybe better and RMSE could have been extracted.

There were also limitations to the comparison of the performance of the decision model tree against the other models. The comparison between decision models and the neural network model was unfair considering that a neural network’s excellency in performance in comparison can only be realised when the dataset is large- our dataset only had 163 datapoints when usually datasets with points >1000 are used.

**Overall comments**

However, in general, all of the machine learning techniques had disadvantages and advantages that could benefit data-analysts in the health-care field that are involved in using machine learning models.

## 5. References

Chatterjee, M. (2022, August 29). *A quick introduction to KNN algorithm*. Great Learning Blog: Free Resources what Matters to shape your Career!. <https://www.mygreatlearning.com/blog/knn-algorithm-introduction/>

Tayo, B. O. (2020, May 11). *A comparative study of linear and KNN regression*. Medium.

<https://pub.towardsai.net/a-comparative-study-of-linear-and-knn-regression-a31955e6263d>

Aggarwal, S. (2020, June 8). *K-nearest neighbors*. Medium. <https://towardsdatascience.com/k-nearest-neighbors-94395f445221>

Racz, A (2021, February 19). *Effect of Dataset Size and Train/Test Split Ratios in QSAR/QSPR Multiclass Classification*.

<https://www.mdpi.com/1420-3049/26/4/1111/pdf>

Tu, JV (1996, Nov). *Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes*

<https://pubmed.ncbi.nlm.nih.gov/8892489/#:~:text=Disadvantages%20include%20its%20%22black%20box,this%20modeling%20technique%20are%20discussed.>

Nau, R. (2019). *What’s a good value for R-squared? Duke.edu.* <https://people.duke.edu/~rnau/rsquared.html>

World Health Organisation “SDG Target 3.1 Maternal Mortality.” *Www.who.int*, 2022, [www.who.int/data/gho/data/themes/topics/sdg-target-3-1-maternal-mortality](http://www.who.int/data/gho/data/themes/topics/sdg-target-3-1-maternal-mortality).

World Health Organisation. “Indicator Metadata Registry Details.” *Www.who.int*, 2022, [www.who.int/data/gho/indicator-metadata-registry/imr-details/25](http://www.who.int/data/gho/indicator-metadata-registry/imr-details/25).

How to Calculate Mean Absolute Error in Python? (2021, November 26). *GeeksforGeeks*.

<https://www.geeksforgeeks.org/how-to-calculate-mean-absolute-error-in-python/>

Python | Mean Squared Error. (2019, June 28). *GeeksforGeeks*.

<https://www.geeksforgeeks.org/python-mean-squared-error/>

Aritesh. (2018, March 20). *Answer to ‘How do I calculate the Adjusted R-squared score using scikit-learn?’* Stack Overflow.

<https://stackoverflow.com/a/49381947>

*Sklearn.linear\_model.LinearRegression*. (n.d.). Scikit-Learn. Retrieved 22 October 2022, from <https://scikit-learn/stable/modules/generated/sklearn.linear_model.LinearRegression.html>

*3.3. Metrics and scoring: Quantifying the quality of predictions*. (n.d.). Scikit-Learn. Retrieved 30 October 2022, from

<https://scikit-learn/stable/modules/model_evaluation.html>

cs95. (2018, March 7). *Answer to ‘What is “random-state” in sklearn.model\_selection.train\_test\_split example?’* Stack Overflow. <https://stackoverflow.com/a/49147859>