Table of Contents

Data description and research question

“Life expectancy declining in many English communities even before pandemic” (Head, 2021). The article published on Imperial College London’s blog grabbed our attention while researching for the dataset. After doing further research on this topic, we found that this is one of the alarming issues in developed countries like the UK and the US (Head, 2021). So, as a team, we decided to implement the skills that we had learned during our course to identify and understand the factors that can affect the life expectancy of human beings.

Based on the above problem statement, we designed the following research question.

Research question: “**Can we predict the life expectancy at birth based on the world development indicator such as Unemployment rate, Infant Mortality Rate, GDP, GNI, Clean fuels and cooking technologies, etc.,**”

After identifying the problem statement and formulating the research question, we started looking for the dataset. While searching, we found plenty of data related to life expectancy on the Kaggle and UCI machine learning repository. However, most of the data on those sites were not suitable for our coursework because those data were already cleaned and did not have enough attributes that fit our research question.

So, we decided to download the “World Development Indicator” dataset from the world bank. It was raw and had enough attributes for our prediction. It contains 1444 development indicators for 266 countries and country groups between the years 1960 to 2020. Further, we added a few datasets related to health which can help us get more insight into our research question.

The description of the data and its link to the source is in the following table.

Dataset	Description
World Development Indicators	This dataset contains the data of 1444 development indicators for 2666 countries and country groups between the years 1960 to 2020. This dataset was downloaded from the world bank’s data hub.
Health workforce	This dataset contains the health workforce information such as medical doctors (per 10000 population), number of medical doctors, number of Generalist medical practitioners, etc.
Mortality from CVD, cancer, diabetes or CRD between exact ages 30 and 70 (%)	This dataset contains information on mortality caused by various non-communicable diseases such as cardiovascular disease (CVD), cancer, diabetes etc. We have used two files for this dataset. Separately for both males and females. This dataset was downloaded from the world bank’s databank.
Suicide morality rate (per 100,000 population)	This data set contains information on the suicide mortality rate per 100,000 population. We have used two files for this dataset. Separately for both males and females. This dataset was downloaded from the world bank’s databank.

Data preparation and cleaning

Since we have four datasets, we decided to deal with each in sequential order. In this step, we have used *pandas* for reading CSV files and managing the data in tabular format, *matplotlib* and *missingno* to visualise the missing value in the graph. The data preparation and cleaning process are grouped into two phases before and after merging the dataset.

Before merging dataset

World development Indicator dataset

The raw WDI dataset had 383838 rows and 66 columns of data. In this data, columns were arranged according to years and the rows by indicators. So, to restructure the dataset, we use converted wide data into long data using the melt function available in the pandas library. It reduced multiple-year columns into a single year column and its value into a value column. As a result, the dataset has 23414118 rows and six columns. Still, our data is not in a suitable format for computation, so we need to convert the long dataset into a wide one. We used the pivot function to organise columns on indicators. After this transformation, we got data with 16165 rows and 1445 columns.

Since we have more than a thousand features in our WID dataset, it can complicate our data cleaning and EDA process because we have limited computing resources. So, we decided to use only the most essential features from the dataset based on our research and domain knowledge. So, we first created a subset of the life expectancy column and merged it with other features step by step. Most of the features in our dataset had a separate column for males and females. Therefore, we merged the features by creating a new column called gender, which is used as a foreign key to merge with the life expectancy subset created at the beginning. After merging all the attributes, we got 16165 rows and 18 columns.

Health Workforce Dataset

The raw dataset of the health workforce contains 3583 rows and seven columns. We will use only the “Medical doctors (per 10,000 population)” column from this dataset, so we will drop other columns except for country and year, which we will use as a foreign key while merging the dataset. After dropping the unwanted column, we visualised the missing value. As a result, we found 739 missing values out of 3583 rows. So, we imputed this missing value with the mean of each country. After this imputation, we found that there were still three missing values, so we dropped those rows.

Mortality from Non communicable disease dataset

This dataset has two files separated by gender, i.e., one for males and one for females. Each raw file has 266 rows based on the country and 66 columns based on the year. The columns and rows on this dataset were organised similar to the WDI dataset, i.e., columns by year and rows by country and mortality rate. So, to transform this data into a suitable format, we use melt to rotate year and value to a single column. After transformation, we rename the indicator name to Male and Female and merge them. After merging these datasets, we create a new column called gender and merge those Male and Female columns into one column.

Suicide mortality rate dataset

This dataset also has two files separated by gender. Same as the previous dataset, it has 266 rows based on the country and 66 columns based on years. First, we transform the dataset into a suitable structure for our computation. Merge both datasets and merge these columns into one based on gender.

We merged all the subsets to the life_exp_data subset using the left join technique with the merge function.

After merging dataset

After merging all the subsets into a single dataset, we continue the data cleaning process by trying to solve the issue of missing values. We printed the sum of null rows in each column to find the exact number of missing values. While counting, we found that a few columns had a high volume of missing data. Similarly, we plot this missing value in a bar graph using the missingno library to visualise it graphically. Similarly, to validate data validity according to the domain knowledge we acquired from research. We have created a function that checks the range of data in a dataset based on the parameter we have supplied to check data validity. The idea for this function was taken from validate package available in R.

Since we realised that our dataset had tons of missing values, we decided to use a subset of data after 2000 AD. Because we had already noticed this while running wdi_data.info() command at the beginning on the raw WDI dataset. After subsetting the dataset, we found that the missing values had decreased significantly. (See Appendix A)

Since our data were missing at random (MAR), we decided to impute it with the mean. To calculate the mean, we grouped the rows of every country for a particular column. After imputation, we again passed our dataset through missingno. From the visualization, we found still a few columns had high missing values. The malaria Incidence column had 4288, the Safe drinking water column had 4126, and Medical doctors (per 10,000 population) had 4126 missing values out of 9928. To maintain the dataset's quality, we decided to drop these columns.

After dropping these columns, we imputed the remaining missing value from the mean of each whole column. After imputation, we checked the missing value visually once more. Currently, there are no more missing values in our dataset.

In the next step, we checked the duplicate rows. Luckily, there were not any duplicate rows. So, we moved towards the exploratory data analysis technique (EDA).

In conclusion, we can say that data cleaning and preparation is one of the most challenging tasks in data science. In our case extracting the correct information from a big chunk of raw data and dealing with the high volume of missing values was very tough.

Exploratory data analysis

“Exploratory data analysis is the process of performing initial investigations on data to discover the patterns, spot anomalies, test hypothesis, and check assumptions with the help of summary statistics and graphical representations.” (Patil, 2018) We performed various univariate and multivariate analysis techniques on the dataset to investigate the data in detail.

Before starting graphical analysis, we tried to understand the data numerically by doing a numerical summary through the head() and describe() function. From head() function. We understand that there are 22 columns, out of which there is one categorical variable and one character, and the rest of them are numerical. With describe(), we can easily visualise the statistical summary such as count, mean, stander deviation, percentiles, minimum value and maximum value. However, we did not find any interesting facts about data through numerical summaries. Similarly, we tried to understand the skewness of each attribute using the skew() function of pandas. As a result, we found that GDP and GNI data are extremely skewed.

Univariate Analysis

We plotted each numerical variable on a histogram and boxplot to understand data distribution and outliers. While observing the distributions and outliers, we found some outliers and extreme skewness in the GDP and GNI columns. There were few aggerated values from a group of countries in GDP and GIN columns. For example, the world's GDP in 2019 was 87 trillion USD, which is the actual value, but this value was the sum of all GDP and is way higher than that of the highest economy, i.e., 20 trillion US. Since it can create bias in our prediction, we decided to drop these rows. Similarly, we did a log transformation to remove the extreme skewness on these columns. (See Appendix B)

Also, there were outliers in a few columns, but those were the natural variation in data. Sometimes it is best to keep outliers in data. They can capture valuable information that could be part of the study (Frost, 2021). So, we decided to keep these outliers in our dataset.

Multivariate Analysis

We plotted life expectancy against other numerical variables in a scatter plot to know the relationship between the variables. From these plots, we learned that GDP, GNI, Clean fuels and cooking technologies, Immunizations, Basic sanitation services, and urban population have a strong positive relation with Life expectancy. Similarly, Infant Mortality, Mortality caused by road traffic injury, Tuberculosis Incidence, Rural population and Morality caused by non-communicable diseases has a strong negative relationship with life expectancy.

We plotted the correlation between variables on a heatmap to get more insight into the relationship among variables. We got a helpful insight into multicollinearity among variables from this correlation plot. We found that “Urban Population” and “Rural Population” were highly correlated.

Since there is multicollinearity, we decided to implement principal component analysis to gain depth insight into the relationship between variables and reduce the dimension of the dataset. “PCA (Principal Component Analysis) takes advantage of multicollinearity and combines the highly correlated variables into a set of uncorrelated variables.” (Pramoditha, 2021)

Unsupervised Learning Method

“PCA is a linear dimensionality reduction technique (algorithm) that transforms a set of correlated variables (p) into a smaller k (k<p) several uncorrelated variables called *principal components* while retaining as much of the variation in the original dataset as possible” (Pramoditha, 2021).

Since PCA is highly affected by the scale of data, so we standardised the data by using StandardScaler class from sklearn and we removed the gender column i.e., categorical variable. It standardises the features by removing the mean and scaling to unit variance. The standard score of a sample x is calculated as $z = (x - u)/s$. Where u is the mean and s is the standard deviation of the training sample. (Pedregosa et al., 2011)

To do PCA, we have to use 90% variance thresholds. We got nine principal components. The first principal component (PC1) explains 39% of the variance, the highest variance among the nine components. While analysing the PC1, we can say that most attributes strongly correlate with it except unemployment, tuberculosis treatment and suicide rate. However, PC2 is highly correlated to decreasing GDP and GNI. (See Appendix B) So, we decided to use the transformed value after PCA in prediction for the prediction.

Machine learning prediction

For machine learning prediction, the random forest algorithm has been implemented. It is a supervised learning algorithm that uses Bootstrap Aggregation (Bagging) as an ensemble learning method for classification and regression.

I decided to implement a machine learning algorithm in both datasets before PCA and after PCA to train the model. So, we can evaluate which dataset performs very well in random forest regressors. With these datasets, I have implemented a random forest with default parameters and then tried to improve the performance by tuning the hyperparameter using RandomizedSearchCV.

Result With default parameters

By default, the number of trees used in the forest is set to 100, and the number of features allowed to make in each tree is “auto”. It will take all the features which make sense in every tree. Here it does not restrict the individual tree (Tavish, 2015).

Performance Matrix	**Before PCA**	**After PCA**
Mean Absolute Error	0.7765995157459821	1.2460337617306556
Mean Squared Error	1.588502781276441	3.1478674973967933
Root Mean Squared Error	1.260358195623943	1.7742230686688731
Execution Time	16.4 s	11.1 s

Result with hyperparameter

Various techniques such as Manual Search, Grid Search, Random Search, Bayes Search etc., are available to tune the hyperparameter in sklearn. The RandomizedSearchCV method from sklearn has been used, which searches for random combinations of the hyperparameters from the given criteria. (See Appendix C) Unlike the grid search technique, not all parameter values are tried out, but a fixed number of parameter settings is sampled from the specified distributions.

Performance Matrix	**Before PCA**	**After PCA**
Mean Absolute Error	0.7645556955128701	1.2404484927348916
Mean Squared Error	1.413915824757689	3.150570395446739
Root Mean Squared Error	1.1890819251665081	1.7749846183690547
Execution Time	5.99 s	6.32 s

We can easily conclude from the above validation metrics that the model performs best after tuning the parameter with the dataset before reducing the dimension. So, the tuned parameter are as follows:

Following are the best performing parameters we have found so far are:

- Number of trees in forest (n_estimators): 100

- Max number of features to be considered when looking for the best split (max_features): sqrt.
- Minimum number of samples required to split in an internal node (min_samples_split): 6
- Minimum number of data points allowed in leaf node (min_samples_leaf): 3
- The maximum depth of the tree (max_depth): 20
- Bootstrap is set to false so, the whole dataset is used to build each tree.

The above performance matrices show that the model with a tuned hyperparameter performs more accurately and has low time complexity. For example, the MSE value before the tuning parameter is 1.26035 for the dataset without PCA and 3.14786 after PCA. Similarly, after tuning the parameter, the MSE value before PCA is 1.41391 and 3.15057 after PCA. However, from the above evaluation, we can find that the model's accuracy decreased significantly after the PCA. Similarly, while experimenting with various parameters, we found that the increase in the number of trees can reduce the performance speed of the algorithm. So, it means if our dataset contains a high number of dimensions/features, it can impact the performance of algorithm speed.

Explain the features

To identify the impact of dependent variables on the best performing model, the features_importances_ property of RandomForestRegressor class was implemented. The importance of a feature was calculated from the criterion's total reduction brought.

It shows that life expectancy is highly affected by infant mortality rate, basic sanitation services and mortality caused by non-communicable diseases. Whereas unemployment, GDP, DNI, HepB3 Immunization and hospital bed are negligible. (See Appendix C)

Deep learning prediction

A deep neural network was used as a deep learning method to predict the life expectancy on our dataset. It was implemented in four different scenarios. We have used Keras for model building and training and keras_tuner to tune the hyperparameters and shap to understand the impact of each input attribute in the model.

Result with random parameter

To train the model using the deep learning technique, I have picked two datasets without reducing dimensions and after reducing dimensions using PCA. At first, the model was trained using one input layer, two hidden layers with 64 outputs and a ReLU activation function. Since our prediction is a regression problem, there is one output in the output layer and a linear activation function. The model was trained in 200 epochs, and the training dataset was divided into 80:20 ratio into training and validation data for cross-validation. Mean Squared Error (MSE) was used as a loss function.

	Before PCA	After PCA
Mean squared error	1.9988572597503662	2.857572078704834
Execution time	1min 41s	1min 42s
Total parameters	9601	9025
No of hidden layer	2	2

We can easily conclude from the above data that the dataset without dimensionality reduction performs better than the data after reducing dimension.

Result after tuning hyperparameter

To tune the hyperparameter, I have used the keras_tuner library. There are four methods - RandomSearch, Hyperband, BayesianOptimization and Sklearn available in keras_tuner. Among them, RandomSearch was used to reduce the search time. We tried to optimize the number of hidden layers between 1 to 5, the numbers of output in each layer between 32 to 512, and the learning rate from 1e-2, 1e-4 and 1e-4. From the calculation, the tuner had suggested neural network architecture with three hidden layers having 320, 400 and 64 output units and the learning rate of 1e-2 for the best performing model (i.e., on dataset before PCA).

Now let us compare the performance matrices of the optimized algorithm.

	Before PCA	After PCA
Mean squared error	1.9281240701675415	2.7040529251098633
Execution time	2min 44s	1min 42s
Total parameters	273777	18977
No of hidden layer	3	3

The above table shows that the model performs well with the dataset before PCA.

Overall, we can conclude that the PCA helps reduce the dimensions of a complex dataset, but it comes with some cost, i.e., loss of information. For example, with a random parameter, the MSE value before PCA is 1.99885 and 2.85757 after PCA. Similarly, after tuning the parameter, the MSE value before PCA is 1.92812 and 2.70405 after PCA. From the above table, we can conclude that the increase in hidden layers creates the model more complex. It is the major drawback of neural networks and makes it the Blackbox.

Evaluation of impact of features on model

Though the deep learning technique is considered a black-box technique, some research suggests that there are some methods to identify the impact of an attribute on the deep learning model.

To understand the impact of features on best performing model, I have implemented KarnelExplainer from the shap library. It uses the special weighted local linear regression to compute the importance of each feature. The computed importance values are Shapley values from game theory and coefficients from local linear regression. The Shapley kernel that recovers SHAP values is given by:

$$\pi_{x'}(z') = \frac{(M-1)}{(M\ choose\ |z'|)|z'|(M-|z'|)'}$$

Where M is the number of simplified input features and |z'| is the number of non-zero elements in z'.

The basic concept behind the Kernel Shap is that Shapley values from game theory can be computed using weighted linear regression. (Lundberg and Lee, 2017; Molnar, 2022; Sukumar, 2020; Anonymous, 2018)

Kernel Shap shows that the model is negatively influenced by infant mortality and Tuberculosis Incidence. Similarly, the model is positively influenced by per capita income and GNI. (See Appendix D)

Performance evaluation and comparison of methods

The dataset was trained and tested using decision trees, random forest, support vector machine, k-nearest neighbour, and five neural network architectures.

To evaluate and report the performance of the regression model Mean Squared Error (MSE), Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) are used. MSE is the most popular and commonly used error metric for regression problems. MSE is calculated as the mean and average squared difference between the predicted and the actual target value. Mathematically, it is calculated as $\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$ where n is the number of data points, Y_i is expected values and $\hat{Y}_i$ is the predicted values. Similarly, RMSE is the extension of MSE and is square root of MSE. Likewise, MAE is calculated as the average of the absolute error values i.e., absolute value of both negative and positive error. Mathematically, it is calculated by $MAE = \frac{\sum_{i=1}^{n}|y_i - x_i|}{n}$ where y_i is predicted value, x_i is the actual value and n is the total number of data points. (Brownlee, 2021)

The performance of these algorithms are as follows:

Member Name	ML Method	MSE	RMSE	MAE	MAPE
Arshad (2152118)	Support Vector Machine	2.018321306	1.420676355	0.9270086	0.013781799
Kiran (2144420)	Random Forest	1.413915825	1.189081925	0.7645557	0.01121506
Vara (2154790)	Decision Tree	5.060867987	2.249637301	1.342292191	0.020792118
Rick (2155256)	Decision Tree	1.85912853	1.363498636	0.697122477	0.009961398
Maninder (2149011)	K-Nearest Neighbour	**0.984627395**	**0.992283929**	**0.55802319**	**0.008472745**

From the above table, we can conclude that the K-NN algorithm outperformed other algorithms. Similarly, Random Forest has a slightly higher MSE score compared to K-NN. However, the Decision tree algorithm implemented by Vara has the highest MSE score. The Decision Tree algorithm implemented by Vara and Rick has a significant difference in MSE scores. One of the reasons for this could be that the hyperparameter on Vara's implementation was not tuned.

While investigating the reason behind the best performance of K-NN, (Ray, 2019) and (Taunk, De, Verma and Swetapadma, 2019) highlighted that K-NN performs with high accuracy with low dimensional data.

Member Name	Hidden Layer	Activation Function	Optimizer	Epoch	MSE
Arshad (2152118)	0	ReLU	Nadam	500	3.551783323
Kiran (2144420)	3	ReLU Output layer: Linear	Adam	191	**1.92812407**
Vara (2154790)	5	ReLU	Adam	50	8.913784935
Rick (2155256)	1	Sigmoid Output layer: Linear	Adam	100	7.458803212

Maninder (2149011)	1	Sigmoid Output layer: Linear	Nadam	100	2.493233442

The deep learning model proposed by Kiran is performing better than the other proposed model. The best performing model has three hidden layers, ReLU activation function in the input and hidden layer and Linear in the output layer, Adam as an optimizer, and the epoch is 191. From the above observation, we can find that the high number of hidden layers and epochs does not guarantee better accuracy. This model was tuned using a random search method, so there is space to improve performance by tuning the hyperparameter.

Overall, machine learning is performing pretty well compared to deep learning. Mahapatra (2018) mentioned that if the training data size is large, deep learning outperforms machine learning and vice versa. So, in our case, the train data size might be small. Nevertheless, the ideal training data size was not mentioned in the blog. We had 5913 rows for training and 2535 rows for testing, split into a 70:30 ratio.

Discussion of the findings

While experimenting with machine learning and deep learning technique against various development indicators to predict the life expectancy, we found that machine learning methods can predict more accurately than deep learning methods. In particular, K-NN has the highest accuracy of 0.984627395 MSE score. Similarly, the Random Forest algorithm also has high accuracy than the deep learning method. In conclusion, we can conclude that a deep learning algorithm cannot always be the best solution. It depends on the size and nature of the dataset. Similarly, deep learning takes a long time to train the model compared to machine learning.

Similarly, we found that PCA helps reduce the dataset's dimension, which improves the algorithm's computing speed. Like, Lever, Krzywinski and Altman (2017) said, PCA helps find the patterns in data without reference to prior knowledge; we realised it while doing EDA. (Anonymous, 2022) mentioned that PCA is sensitive to the scale of the features, so we standardised the data using a standard scaler. This analysis found that the accuracy of machine learning and deep learning prediction has decreased after PCA. Therefore, the major limitation of PCA is the loss of information.

Likewise, we can derive some meaningful insights from EDA, Machine Learning and Deep Learning techniques. While exploring the data with EDA, we found an interesting fact that females have more life expectancy than males. Similarly, while trying to understand the impact of dependent variables on the target variable, we found that infant mortality has a high impact on life expectancy.

Though we have achieved pretty good accuracy, there are some limitations that we can overcome in further research to get a more accurate result. Within the boundary of the course, we have trained and tested a machine learning model using decision trees, a random forest, a support vector machine and a k-nearest neighbour. Further, we can use other machine learning algorithms like Linear Regression, Gaussian Regression, etc. and evaluate their performance. Similarly, we have used 20 development indicators out of hundreds of indicators available in the WID dataset to reduce the training and testing time. Therefore, other development indicators can be used to identify their impact on life expectancy. Uniformly, to tune the hyperparameter in both machine learning and deep learning methods, the random search technique has been used to reduce the time to search better hyperparameter. The hyperparameter suggested by this technique might not be the best, so the grid search technique can be used to find the best hyperparameter to tune the model.

It is an exciting journey where we got a hands-on experience with various machine learning techniques and deep learning techniques.

Data Management Plan and Author Contribution statement

Data Management Plan

The datasets that we have used for analysis were downloaded from the open data repository of the World Bank and World Health Organization. These data were collected once, and all the analyses were performed on those static data. The World Development Indicator dataset download from World Bank is our primary dataset that contains 254 MB of information. Similarly, other secondary datasets are relatively small compared to primary data. After cleaning and EDA, the overall data size, including the training and testing dataset, is 261 MB. To ensure we can recover the data and file in case of failure on our primary system, we had stored it in three different locations on the local laptop, OneDrive and GitHub. After analysing each file, the local storage and GitHub were manually updated, and OneDrive was set up to sync automatically. Since there is no ethical, legal or commercial restriction that might affect the data shared in this research, it can be shared with researchers interested in it.

The detailed data management plan is uploaded to appendix.

Author Contribution Statement

Each member played a vital role in achieving the best result. In the first few days, we decided to search and find two datasets by each member on their area of interest. I came up with an idea of predicting life expectancy at birth, and every team member agreed with it. After finalizing the dataset, we divided the task among team members and worked on it. Every member is involved in both data preparation and EDA tasks.

I was responsible for cleaning the World Development Indicator dataset. It was in a long format and contained more than 1400 development indicators per country, so I removed the unwanted rows and re-format the data based on our research question. In the EDA section, I plotted all the numerical variables in the bar graph to understand the distribution and the boxplot to understand the outlier. Also, I worked together with Vara to understand the relationship between variables, particularly correlation plots using a seaborn heatmap. Arshad was responsible for cleaning mortality from non-communicable disease datasets and transforming them using PCA. Similarly, Rick was responsible for cleaning the health workforce dataset and checking the missing values using the missingno library. Vara was responsible for merging all the datasets and visualising the correlation between variables with Kiran. Similarly, Maninder was responsible for identifying the relationship among variables using an unsupervised learning technique. Since PCA is affected by scale, I gave feedback on using StandardScaler to scale the data.

Apart from this group work, each member was responsible for implementing the machine learning and deep learning method. Arshad used a support vector machine, Kiran used Random Forest, and Maninder used K-NN for machine learning. Similarly, Vara and Rick implemented a decision tree in Machine learning. Similarly, every member used different parameters in Neural networks.

GitHub was used for team collaboration, and the GitHub task board was used to plan and manage the task. (See Appendix E) Similarly, we have created a group chat in teams and WhatsApp to communicate with team members.

References

Head, E., 2021. *Life expectancy declining in many English communities even before pandemic | Imperial News | Imperial College London*. [online] Imperial News. Available at: https://www.imperial.ac.uk/news/231119/life-expectancy-declining-many-english-communities/ [Accessed 23 March 2022].

Patil, P., 2018. *What is Exploratory Data Analysis?*. [online] Towards Data Science. Available at: https://towardsdatascience.com/exploratory-data-analysis-8fc1cb20fd15 [Accessed 15 March 2022].

Tavish, T., 2015. *Random Forest Parameter Tuning | Tuning Random Forest*. [online] Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2015/06/tuning-random-forest-model/ [Accessed 18 March 2022].

Lundberg, S. and Lee, S., 2017. *Advances in Neural Information Processing Systems*. 30th ed. Curran Associates, Inc., pp.4765--4774.

Molnar, C., 2022. *9.6 SHAP (SHapley Additive exPlanations) | Interpretable Machine Learning*. [online] Christophm.github.io. Available at: https://christophm.github.io/interpretable-ml-book/shap.html [Accessed 23 March 2022].

Sukumar, R., 2020. *SHAP Part 2: Kernel SHAP*. [online] Medium. Available at: https://medium.com/analytics-vidhya/shap-part-2-kernel-shap-3c11e7a971b1 [Accessed 23 March 2022].

Anonymous, 2018. *shap.KernelExplainer — SHAP latest documentation*. [online] Shap-lrjball.readthedocs.io. Available at: https://shap-lrjball.readthedocs.io/en/latest/generated/shap.KernelExplainer.html [Accessed 23 March 2022].

Frost, J., 2021. *Guidelines for Removing and Handling Outliers in Data - Statistics By Jim*. [online] Statisticsbyjim.com. Available at: https://statisticsbyjim.com/basics/remove-outliers/ [Accessed 25 March 2022].

Pramoditha, R., 2021. *How do you apply PCA to Logistic Regression to remove Multicollinearity?*. [online] Towards Data Science. Available at: https://towardsdatascience.com/how-do-you-apply-pca-to-logistic-regression-to-remove-multicollinearity-10b7f8e89f9b [Accessed 25 March 2022].

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, É., 2011. Scikit-learn: Machine Learning in {P}ython. *Journal of Machine Learning Research*, 12, pp.2825--2830.

Brownlee, J., 2021. *Regression Metrics for Machine Learning*. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/regression-metrics-for-machine-learning/ [Accessed 08 April 2022].

Taunk, K., De, S., Verma, S. and Swetapadma, A., 2019. A Brief Review of Nearest Neighbor Algorithm for Learning and Classification. *2019 International Conference on Intelligent Computing and Control Systems (ICCS)*,.

Ray, S., 2019. A Quick Review of Machine Learning Algorithms. *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*,.

Mahapatra, S., 2018. *Why Deep Learning over Traditional Machine Learning?*. [online] Towards Data Science. Available at: https://towardsdatascience.com/why-deep-learning-is-needed-over-traditional-machine-learning-1b6a99177063 [Accessed 08 April 2022].

Anonymous. 2022. *A Guide to Principal Component Analysis (PCA) for Machine Learning*. [online] Available at: <https://www.keboola.com/blog/pca-machine-learning> [Accessed 19 April 2022].

Appendix

Appendix A: Data cleaning

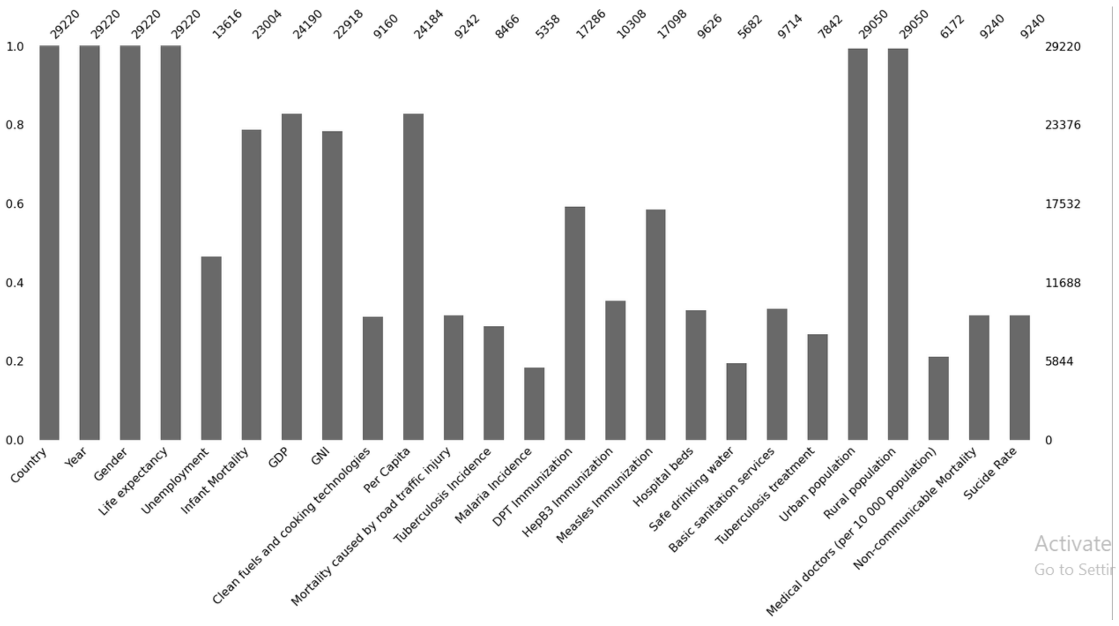


Figure 1: Missing values in raw Life Expectancy dataset

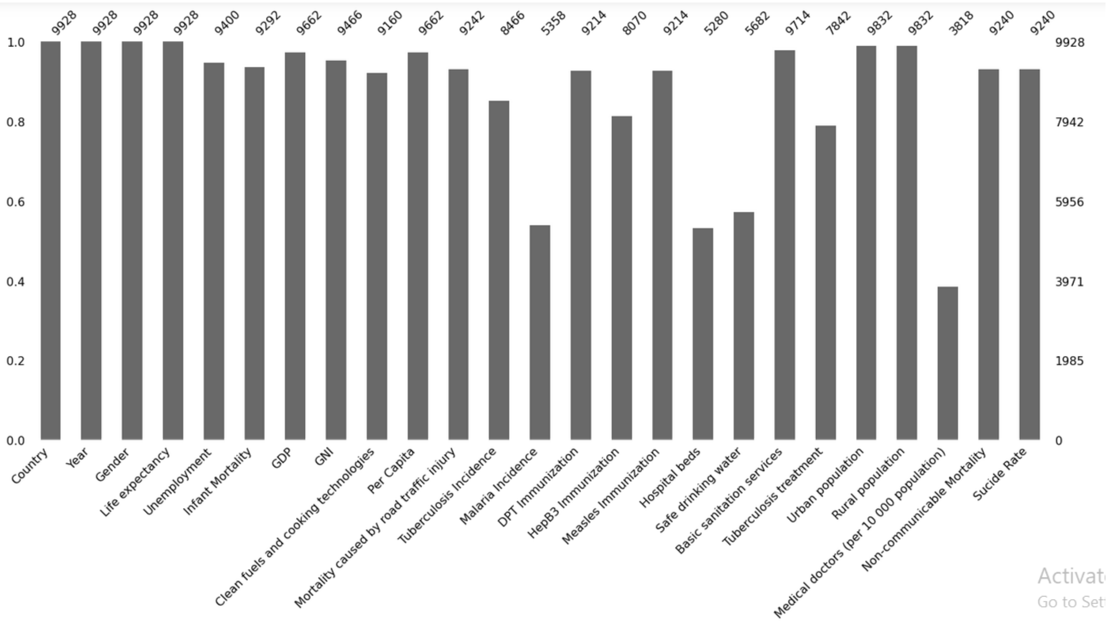


Figure 2: Missing values in Life expectancy dataset after droping rows before 2000 AD

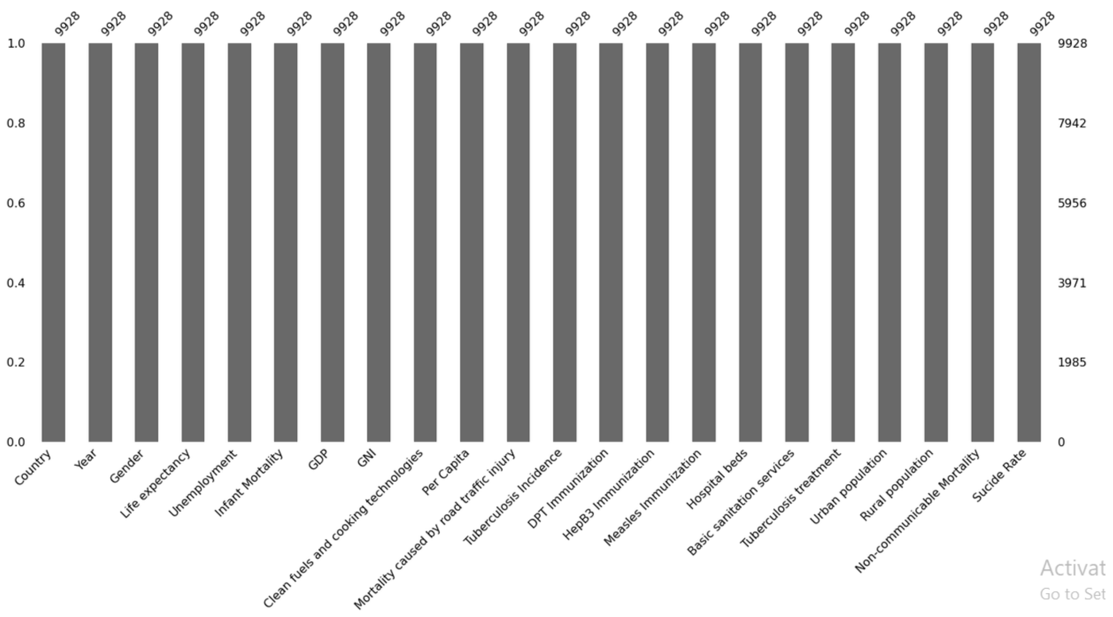
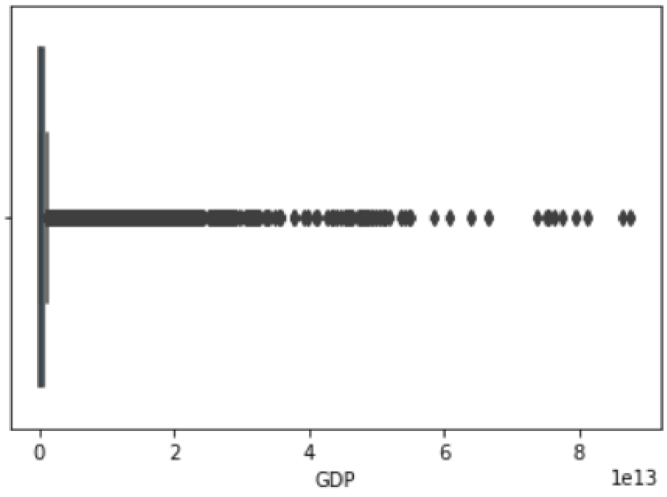


Figure 3: Visualisation after removing missing value

Appendix B: Exploratory Data Analysis



```
life_exep_data['GDP'].skew()
```

```
5.889286292623093
```

Figure 4: Highly skewed data on GDP column

```
life_exep_data['GDP'] = np.log(life_exep_data['GDP'])
```

```
sns.boxplot(data=life_exep_data, x=life_exep_data['GDP'])
```

```
<AxesSubplot:xlabel='GDP'>
```

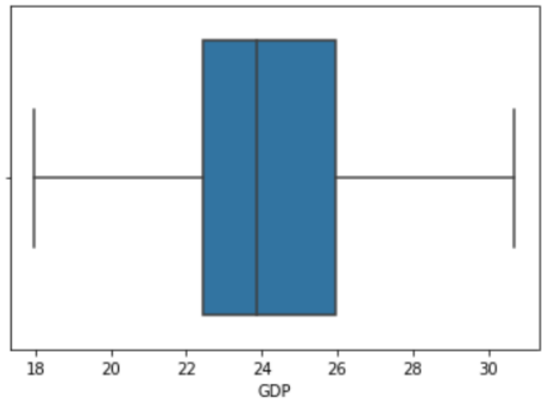


Figure 5: Skewness after log transformation

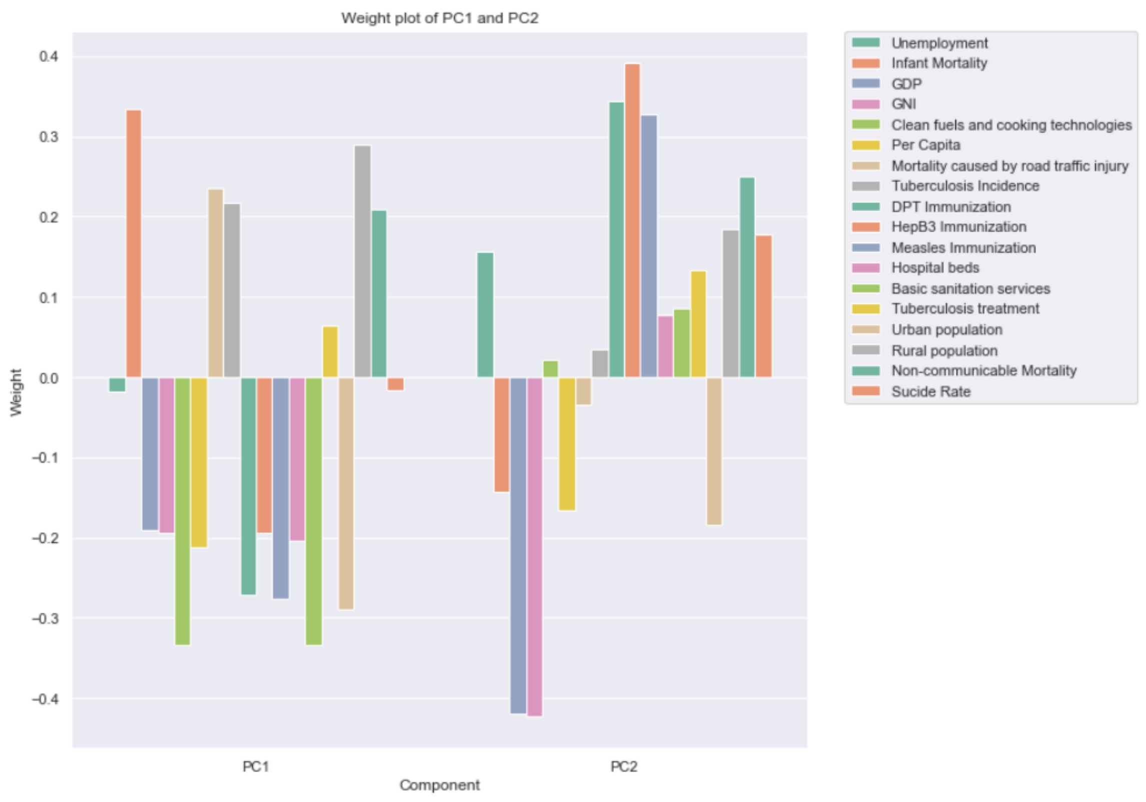
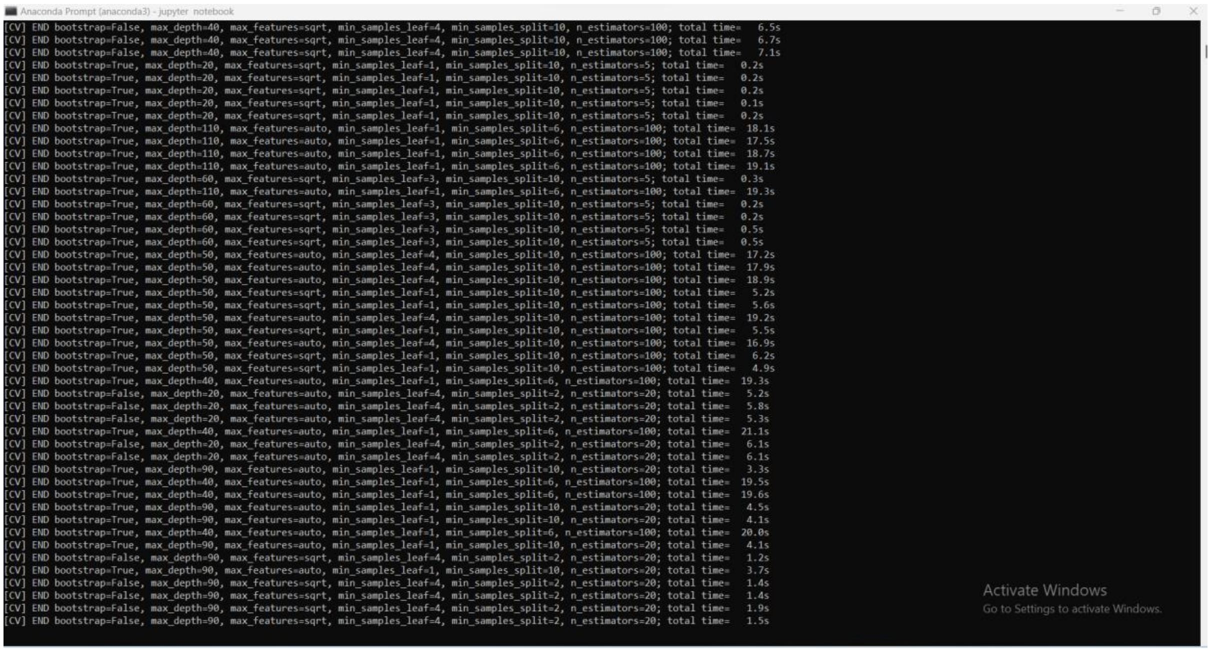


Figure 6: Weight of features on Principal Component 1 and 2

Appendix C: Machine Learning



```
[CV] END bootstrap=False, max_depth=40, max_features=sqrt, min_samples_leaf=4, min_samples_split=10, n_estimators=100; total time=   6.5s
[CV] END bootstrap=False, max_depth=40, max_features=sqrt, min_samples_leaf=4, min_samples_split=10, n_estimators=100; total time=   6.7s
[CV] END bootstrap=False, max_depth=40, max_features=sqrt, min_samples_leaf=4, min_samples_split=10, n_estimators=100; total time=   7.1s
[CV] END bootstrap=True, max_depth=20, max_features=sqrt, min_samples_leaf=1, min_samples_split=10, n_estimators=5; total time=   0.2s
[CV] END bootstrap=True, max_depth=20, max_features=sqrt, min_samples_leaf=1, min_samples_split=10, n_estimators=5; total time=   0.2s
[CV] END bootstrap=True, max_depth=20, max_features=sqrt, min_samples_leaf=1, min_samples_split=10, n_estimators=5; total time=   0.2s
[CV] END bootstrap=True, max_depth=20, max_features=sqrt, min_samples_leaf=1, min_samples_split=10, n_estimators=5; total time=   0.1s
[CV] END bootstrap=True, max_depth=20, max_features=sqrt, min_samples_leaf=1, min_samples_split=10, n_estimators=5; total time=   0.2s
[CV] END bootstrap=True, max_depth=110, max_features=auto, min_samples_leaf=1, min_samples_split=6, n_estimators=100; total time=  18.1s
[CV] END bootstrap=True, max_depth=110, max_features=auto, min_samples_leaf=1, min_samples_split=6, n_estimators=100; total time=  17.5s
[CV] END bootstrap=True, max_depth=110, max_features=auto, min_samples_leaf=1, min_samples_split=6, n_estimators=100; total time=  18.7s
[CV] END bootstrap=True, max_depth=110, max_features=auto, min_samples_leaf=1, min_samples_split=6, n_estimators=100; total time=  19.1s
[CV] END bootstrap=True, max_depth=60, max_features=sqrt, min_samples_leaf=3, min_samples_split=10, n_estimators=5; total time=   0.3s
[CV] END bootstrap=True, max_depth=110, max_features=auto, min_samples_leaf=1, min_samples_split=6, n_estimators=100; total time=  19.3s
[CV] END bootstrap=True, max_depth=60, max_features=sqrt, min_samples_leaf=3, min_samples_split=10, n_estimators=5; total time=   0.2s
[CV] END bootstrap=True, max_depth=60, max_features=sqrt, min_samples_leaf=3, min_samples_split=10, n_estimators=5; total time=   0.2s
[CV] END bootstrap=True, max_depth=60, max_features=sqrt, min_samples_leaf=3, min_samples_split=10, n_estimators=5; total time=   0.5s
[CV] END bootstrap=True, max_depth=60, max_features=sqrt, min_samples_leaf=3, min_samples_split=10, n_estimators=5; total time=   0.5s
[CV] END bootstrap=True, max_depth=50, max_features=auto, min_samples_leaf=4, min_samples_split=10, n_estimators=100; total time=  17.2s
[CV] END bootstrap=True, max_depth=50, max_features=auto, min_samples_leaf=4, min_samples_split=10, n_estimators=100; total time=  17.9s
[CV] END bootstrap=True, max_depth=50, max_features=auto, min_samples_leaf=4, min_samples_split=10, n_estimators=100; total time=  18.9s
[CV] END bootstrap=True, max_depth=50, max_features=sqrt, min_samples_leaf=1, min_samples_split=10, n_estimators=100; total time=   5.2s
[CV] END bootstrap=True, max_depth=50, max_features=sqrt, min_samples_leaf=1, min_samples_split=10, n_estimators=100; total time=   5.6s
[CV] END bootstrap=True, max_depth=50, max_features=auto, min_samples_leaf=4, min_samples_split=10, n_estimators=100; total time=  19.2s
[CV] END bootstrap=True, max_depth=50, max_features=sqrt, min_samples_leaf=1, min_samples_split=10, n_estimators=100; total time=   5.5s
[CV] END bootstrap=True, max_depth=50, max_features=auto, min_samples_leaf=4, min_samples_split=10, n_estimators=100; total time=  16.9s
[CV] END bootstrap=True, max_depth=50, max_features=sqrt, min_samples_leaf=1, min_samples_split=10, n_estimators=100; total time=   6.2s
[CV] END bootstrap=True, max_depth=50, max_features=sqrt, min_samples_leaf=1, min_samples_split=10, n_estimators=100; total time=   4.9s
[CV] END bootstrap=True, max_depth=40, max_features=auto, min_samples_leaf=1, min_samples_split=6, n_estimators=100; total time=  19.3s
[CV] END bootstrap=False, max_depth=20, max_features=auto, min_samples_leaf=4, min_samples_split=2, n_estimators=20; total time=   5.2s
[CV] END bootstrap=False, max_depth=20, max_features=auto, min_samples_leaf=4, min_samples_split=2, n_estimators=20; total time=   5.8s
[CV] END bootstrap=False, max_depth=20, max_features=auto, min_samples_leaf=4, min_samples_split=2, n_estimators=20; total time=   5.3s
[CV] END bootstrap=True, max_depth=40, max_features=auto, min_samples_leaf=1, min_samples_split=6, n_estimators=100; total time=  21.1s
[CV] END bootstrap=False, max_depth=20, max_features=auto, min_samples_leaf=4, min_samples_split=2, n_estimators=20; total time=   6.1s
[CV] END bootstrap=False, max_depth=20, max_features=auto, min_samples_leaf=4, min_samples_split=2, n_estimators=20; total time=   6.1s
[CV] END bootstrap=True, max_depth=90, max_features=auto, min_samples_leaf=1, min_samples_split=10, n_estimators=20; total time=   3.3s
[CV] END bootstrap=True, max_depth=40, max_features=auto, min_samples_leaf=1, min_samples_split=6, n_estimators=100; total time=  19.5s
[CV] END bootstrap=True, max_depth=40, max_features=auto, min_samples_leaf=1, min_samples_split=6, n_estimators=100; total time=  19.6s
[CV] END bootstrap=True, max_depth=90, max_features=auto, min_samples_leaf=1, min_samples_split=10, n_estimators=20; total time=   4.5s
[CV] END bootstrap=True, max_depth=90, max_features=auto, min_samples_leaf=1, min_samples_split=10, n_estimators=20; total time=   4.1s
[CV] END bootstrap=True, max_depth=40, max_features=auto, min_samples_leaf=1, min_samples_split=6, n_estimators=100; total time=  20.0s
[CV] END bootstrap=True, max_depth=90, max_features=auto, min_samples_leaf=1, min_samples_split=10, n_estimators=20; total time=   4.1s
[CV] END bootstrap=False, max_depth=90, max_features=sqrt, min_samples_leaf=4, min_samples_split=2, n_estimators=20; total time=   1.2s
[CV] END bootstrap=True, max_depth=90, max_features=auto, min_samples_leaf=1, min_samples_split=10, n_estimators=20; total time=   3.7s
[CV] END bootstrap=False, max_depth=90, max_features=sqrt, min_samples_leaf=4, min_samples_split=2, n_estimators=20; total time=   1.4s
[CV] END bootstrap=False, max_depth=90, max_features=sqrt, min_samples_leaf=4, min_samples_split=2, n_estimators=20; total time=   1.4s
[CV] END bootstrap=False, max_depth=90, max_features=sqrt, min_samples_leaf=4, min_samples_split=2, n_estimators=20; total time=   1.9s
[CV] END bootstrap=False, max_depth=90, max_features=sqrt, min_samples_leaf=4, min_samples_split=2, n_estimators=20; total time=   1.5s
```

Figure 7: Searching best parameter using RandomizedSearchCV for model tuning.

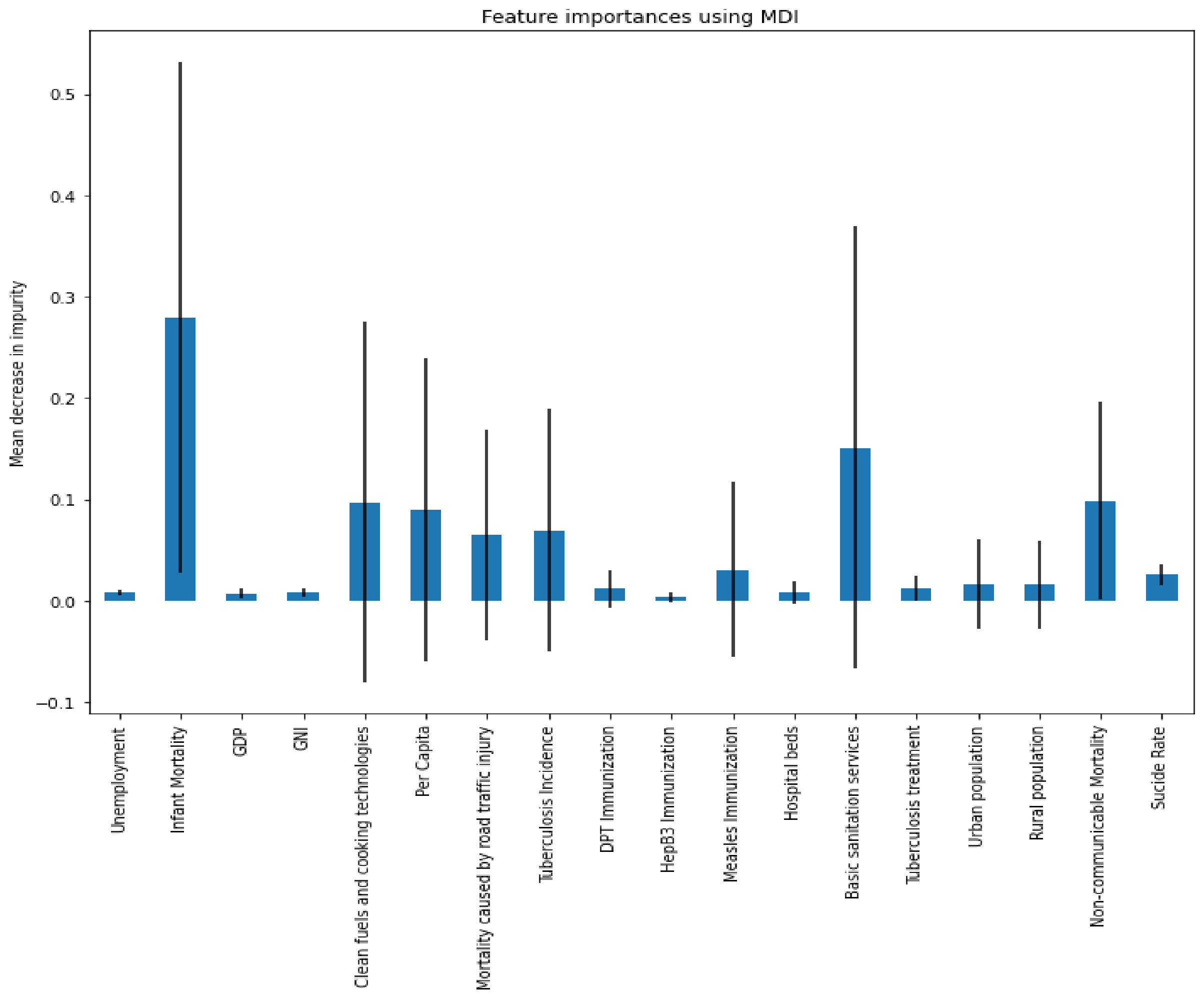
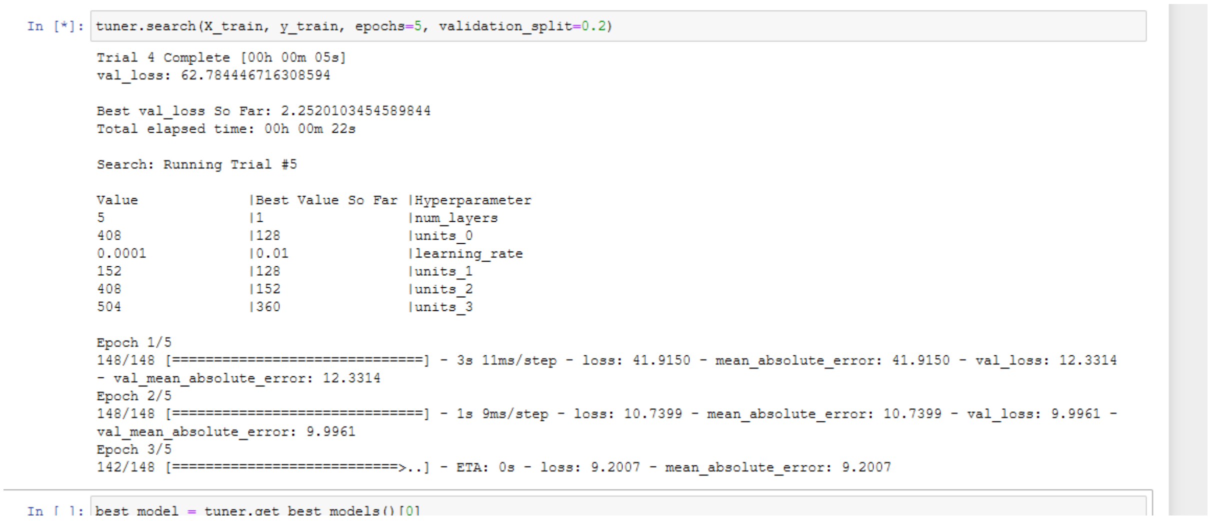


Figure 8: Feature importance using Gini importance method

Appendix D: Deep Learning



```
In [*]: tuner.search(X_train, y_train, epochs=5, validation_split=0.2)

        Trial 4 Complete [00h 00m 05s]
        val_loss: 62.784446716308594

        Best val_loss So Far: 2.2520103454589844
        Total elapsed time: 00h 00m 22s

        Search: Running Trial #5

        Value             |Best Value So Far |Hyperparameter
        5                 |1                 |num_layers
        408               |128               |units_0
        0.0001            |0.01              |learning_rate
        152               |128               |units_1
        408               |152               |units_2
        504               |360               |units_3

        Epoch 1/5
        148/148 [==============================] - 3s 11ms/step - loss: 41.9150 - mean_absolute_error: 41.9150 - val_loss: 12.3314
        - val_mean_absolute_error: 12.3314
        Epoch 2/5
        148/148 [==============================] - 1s 9ms/step - loss: 10.7399 - mean_absolute_error: 10.7399 - val_loss: 9.9961 -
        val_mean_absolute_error: 9.9961
        Epoch 3/5
        148/148 [===========================>..] - ETA: 0s - loss: 9.2007 - mean_absolute_error: 9.2007

In [ ]: best_model = tuner.get_best_models()[0]
```

Figure 9: Tuning hyperparameter using keras_tuner

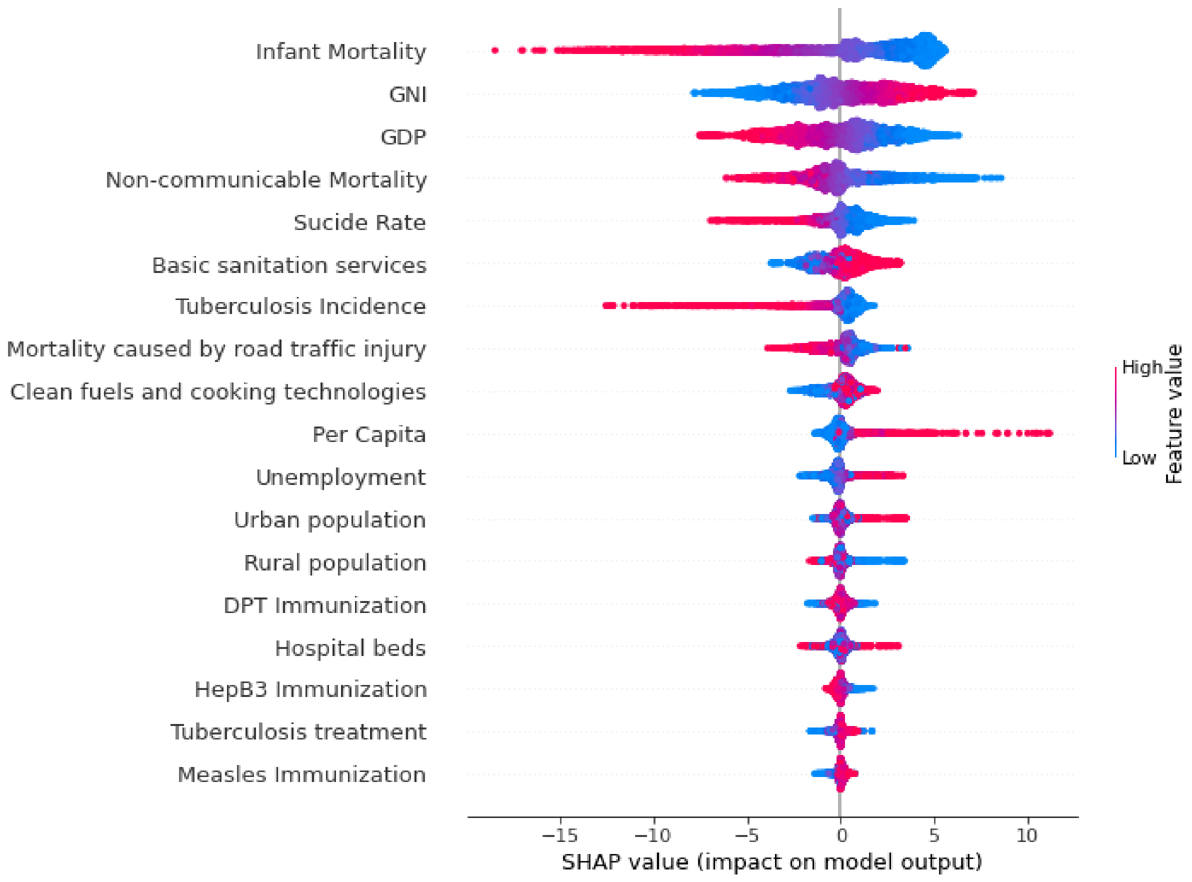
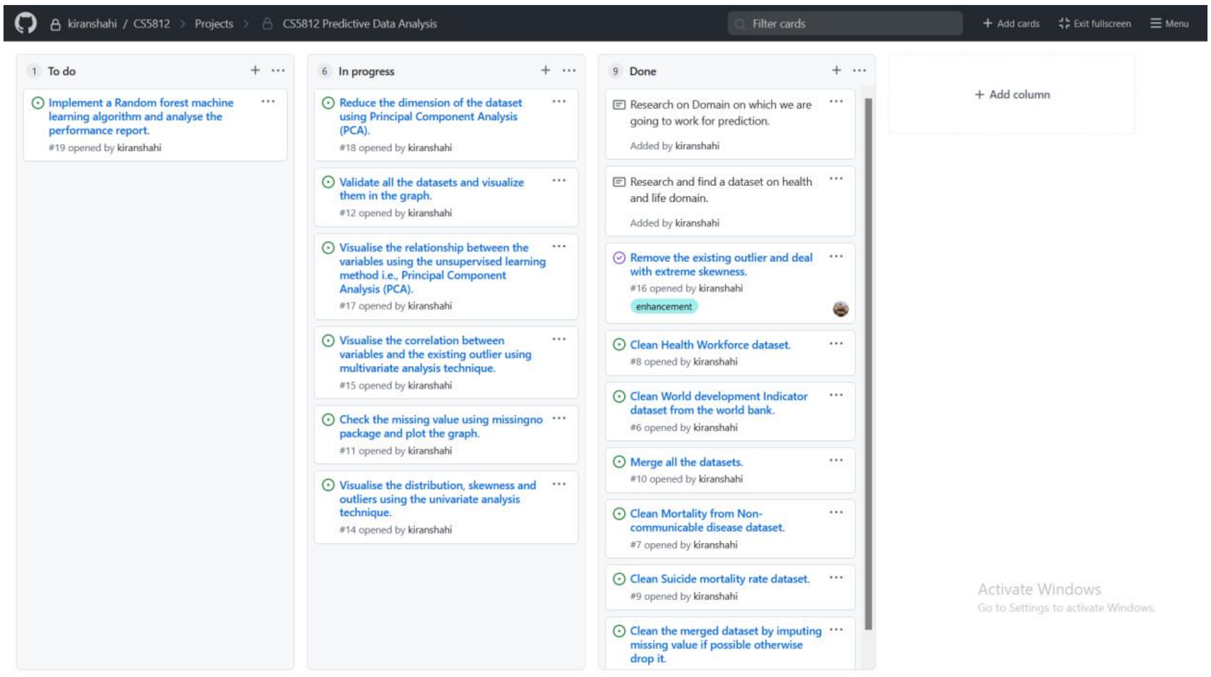


Figure 10: Feature importance by shap value in deep learning model

Appendix E: Task Management


kiranshahi / CS5812 > Projects > CS5812 Predictive Data Analysis

1 To do

- Implement a Random forest machine learning algorithm and analyse the performance report.
  #19 opened by kiranshahi

6 In progress

- Reduce the dimension of the dataset using Principal Component Analysis (PCA).
  #18 opened by kiranshahi
- Validate all the datasets and visualize them in the graph.
  #12 opened by kiranshahi
- Visualise the relationship between the variables using the unsupervised learning method i.e., Principal Component Analysis (PCA).
  #17 opened by kiranshahi
- Visualise the correlation between variables and the existing outlier using multivariate analysis technique.
  #15 opened by kiranshahi
- Check the missing value using missingno package and plot the graph.
  #11 opened by kiranshahi
- Visualise the distribution, skewness and outliers using the univariate analysis technique.
  #14 opened by kiranshahi

9 Done

- Research on Domain on which we are going to work for prediction.
  Added by kiranshahi
- Research and find a dataset on health and life domain.
  Added by kiranshahi
- Remove the existing outlier and deal with extreme skewness.
  #16 opened by kiranshahi
  enhancement
- Clean Health Workforce dataset.
  #8 opened by kiranshahi
- Clean World development Indicator dataset from the world bank.
  #6 opened by kiranshahi
- Merge all the datasets.
  #10 opened by kiranshahi
- Clean Mortality from Non-communicable disease dataset.
  #7 opened by kiranshahi
- Clean Suicide mortality rate dataset.
  #9 opened by kiranshahi
- Clean the merged dataset by imputing missing value if possible otherwise drop it.

Figure 10: GitHub task board