MEF UNIVERSITY

MORTALITY PREDICTION OF COUNTRIES

Capstone Project

Elif Efser Üşenmez

İSTANBUL, 2018

MEF UNIVERSITY

MORTALITY PREDICTION OF COUNTRIES

Capstone Project

Elif Efser Üşenmez

Advisor: Asst. Prof. Dr. Utku Koç

İSTANBUL, 2018

MEF UNIVERSITY

Name of the project: Mortality Prediction of Countries
Name/Last Name of the Student: Elif Efser Üşenmez
Date of Thesis Defense: .. / .. /20..

I hereby state that the graduation project prepared by Elif Efser Üşenmez has been completed under my supervision. I accept this work as a "Graduation Project".

.. / .. / 20..
Asst. Prof. Dr. Utku Koç

I hereby state that I have examined this graduation project by Elif Efser Üşenmez which is accepted by his supervisor. This work is acceptable as a graduation project and the student is eligible to take the graduation project examination.

.. / .. / 20..
Prof. Dr. Özgür Özlük

Director
Of
Big Data Analytics Program

We hereby state that we have held the graduation examination of _____ and agree that the student has satisfied all requirements.

THE EXAMINATION COMMITTEE

Committee Member	Signature
1. Asst. Prof. Dr. Utku Koç	……..
2. Prof. Dr. Özgür Özlük	……..

Academic Honesty Pledge

I promise not to collaborate with anyone, not to seek or accept any outside help, and not to give any help to others.

I understand that all resources in print or on the web must be explicitly cited.

In keeping with MEF University's ideals, I pledge that this work is my own and that I have neither given nor received inappropriate assistance in preparing it.

Name	Date	Signature

EXECUTIVE SUMMARY

MORTALITY PREDICTION OF COUNTRIES

Elif Efser Üşenmez

Advisor: Asst. Prof. Dr. Utku Koç

SEPTEMBER, 2018, 79 pages

In this study mortality reasons of countries detailed by sex and age-group is analyzed and different forecasting models are developed by using different machine learning algorithms. The dataset is obtained from the World Health Organization(WHO) Mortality Database. In WHO database there are different datasets for countries mortality reason number. The study used the dataset that used ICD-10 for classifying mortality reasons.ICD-10 is the 10 revision of International Statistical Classification of Diseases and Related Health Problems published by the World Health Organization. In addition to main mortality reason datasets, we add different independent variables and try to find the best features to fit models without biasing and overfitting and reaching high R^2 and Mean Square Errors. To find the best model for forecasting mortality reasons by age-groups and sex different machine learning algorithms are fitted and results of these algorithms are analyzed.

Key Words: Exploratory data analysis, statistical regression model, machine learning regression algorithms, mortality prediction.

ÖZET

ÜLKELERİN ÖLÜM TAMİNLEMESİ

Elif Efser Üşenmez

Advisor: Asst. Prof. Dr. Utku Koç

EYLÜL, 2018, 79 sayfa

Bu çalışmanın amacı ülkelerin cinsiyet ve yaş bazlı ölüm nedenlerini analiz ederek farklı makine öğrenmesi algoritmaları ile modeller oluşturarak en iyi modeli bulmaktır. Çalışmada kullanılan veri seti Dünya Sağlık Örgütünün Ölüm nedeni veri ambarından elde edilmiştir. İlgili veri ambarında ülkelerin ölüm nedeni rakamları ile ilgili birden fazla veri seti vardır. Bu çalışmada ölüm nedenleri ICD-10'a göre sınıflandırılan veri seti kullanılmıştır. ICD-10 Dünya Sağlık Örgütü tarafından yayımlanan hastalıkların ve ilgili sağlık sorunlarının uluslararası istatistiksel sınıflandırılması için kullanılan ICD(International Classification of Diseases) serisinin 10 uncu sürümüdür Ana ölüm nedeni verisine ek olarak farklı bağımsız değişkenler ekleyerek modeller en iyi değişkenler bulunmaya çalışılmıştır. Modeller için en iyi değişkenler seçilirken modellerde yanlılık ve ve aşırı öğrenmeye yol açmamasına ve modellerde en uygun R^2 ve ortalama karesel hatalara ulaşılmasına bakılmıştır. Farklı tahminleme modellerinin sonuçları incelenerek cinsiyet, yaş grubu ve neden bazında ölüm tahminlemesi için en uygun model bulunmaya çalışılmıştır.

Anahtar Kelimeler: Keşif Veri Analizi, İstatistiksel regresyon modelleri, makine öğrenmesi regresyon algoritmaları. Ölüm tahminleme

TABLE OF CONTENTS

1. INTRODUC TION

Nowadays the life expectancy of human beings continue to increase; as a result of this the forecasting of mortality reason become more important. There are too many different mortality reasons, in this study, the 10th revision of the ICD for mortality reason classification issued by WHO will be used. The number of deaths for all mortality reason changes through the years and according to countries. There are too many features that affect the different mortality reasons like smoke air and water pollution, obesity and hunger. The aim of this study is to create a model to predict mortality reason of countries by using regression algorithms.

1.1. Regression Machine Learning Algorithms

Regression is a supervised machine learning methodology that was used to predict continuous variables. There are different regression algorithms. The main and popular ones of these algorithms are linear, ridge, lasso and random forest algorithms. In general, the best method to find the most efficient algorithm is trying all of them and compares their results. Because these algorithms are widely used in machine learning world there are already coded in libraries of languages such as PYTHON, and R.

1.2. World Health Organization Mortality Database

The WHO (World Health Organization) Mortality Database is a compilation of mortality data by age, sex and cause of death, as reported annually by the Member States from their civil registration systems. Dataset can be reached from the website of the World Health Organization. In the website, raw database files can be downloaded by institutions and organizations mainly for research purposes. In the website, there are also explanations for required information technology (IT) resources and how to use information. In the raw database files, there are different datasets for the different version of International Classification of Diseases (ICD). In this study, we used the dataset coded by International Classification of Diseases (ICD)10. The database files also including code reference, instruction tables. In World Health Organization site there is also a detailed documentation file that gives information about the table, file formats, available data, ICD codes definitions and other information that will be necessary to understand the data.

1.3 Other datasets predictors

The datasets are planned to use in models as predictors are gathered from www.ourworldindata.org website. The datasets are:

- GDP per capita according to world bank expressed in constant 2011international dollars
- Smoke rates of countries detailed by country, year and sex
- PM2.5 air pollution, mean annual exposure (micrograms per cubic meter) detailed by country and years
- Share of total population with improved water sources of countries detailed by years
- Share of adults defined as obese from 2000-2015 detailed by country, year and sex
- Total Healthcare Expenditure as Share of National GDP by country and year
- Human Development Index by country and year

The planed predictor and datasets may be changed while we are studying on our models. New dataset and predictors may be added or some may be extracted from the model.

1.4 Literature Survey

Many mortality forecasting modeling have been developed since 1825-Gompertz publish mortality law. After Gompertz study the main and popular publication was done by Lopez and Hakama(1986), Polland(1987), Olshansky(1988), Willekens(1990), Keyfitz(1991), Gomez de Lean and Texmon(1992), Benjamin and Soliman(1993), Lee, Carter and Tuljapukar(1995), Caselli(1996), Murray and Lopez(1996). All of these methods use different concepts for forecasting but according to Gomez de Leon and Texman, the main division to classify these methods is if they are causes of death methods or not. In this study, we will discuss the causes of death methods.

The studies for mortality reason rates prediction can be divided into two groups, historical base and structural. The historical base studies are make forecasting for the future according to previous results. The main problem in these models is changing in the trends of independent variables may lead misprediction. Also stable and trustable historical data can only be obtained for high -income countries. An example in these type of study is Lee and Carter Poison log bilinear model and the extended version of it, based on the

Bayesian model by Czado et al.Girosi and King also develop a model based on calculating the risk of death.

In structural models, the forecasting is done according to different independent variables. The main problem with these models is the lack of data for independent variables and unexpected variables that may affect the mortality rates. One of the main studies in the type of structural model for mortality cause forecasting is Global Burden of Disease (GBD) that was funding by The World Bank and the World Health Organization. The first GBD study is published by Murray and Lopez in 1996. The study was including 138 types of disease injuries and 8 regions. Since 1996 improvements in GBD study and new concepts are studied. The main concept in GBD study is that it developed 3 different models (baseline, pessimistic, optimistic) for all age groups and mortality reason groups. The GBD model depends on DALY(Disability adjusted life years) variables. Before a model based on GBD is being started the DALY for all type of mortality reason, age group must be calculated. The problem in DALY calculation is, it assumes the expected lifetime for all countries as the same and not changing through the years.

This study also proposes a structural model. Unlike many other mortality reason forecasting models, our model is in country level and has much more independent variables such as air and water pollution, human development index. The models mentioned above are generally used death risk factors and make the assumption about life expectancy like DALY factor in GBD model. In this study, the risk factors for death is not included in the model therefore not need to make any assumptions. Like the GBD model, our model is a regression model. However, we use machine learning algorithms for building regression models.

2. PROJECT DEFINITION

2.1. Project Objectives

The objective of the project is to develop a model to predict the mortality reasons of countries. To make a good model the possible attributes that affect the mortality must be found. Therefore, the first aim of the project is analyzing the possible predictor data. Furthermore, the correlations between the predictors and mortality and each other are also an important issue in regression models. After relation analysis, it was aimed to find the best regression algorithm for our mortality prediction model. At last, it was aimed to develop a model to predict the mortality reason of a country according to inputs used as predictors in our model.

2.2. Project Scope

The scope of the project is predicting the mortality of countries according to predictors that will be put into the model. Of course, there are too many factors that affect the mortality in our model we will study only some of them. Furthermore, some unexpected occurrence such as wars, terrorism-related attacks, and natural disasters may have a huge effect on mortality At this project, we omit those effects. Also, the project aim is to make the prediction by the assumption that the predictor values are known. In other words, while predicting the next year mortality the project scope only make the prediction for the mortality rate not also make predictions for the predictors in the model.

3. METHODOLOGY

3.1. Prepare Raw Dataset File

The main mortality data gathered from the WHO mortality database is in a format that can be imported into any Data Base Management System (DBMS). SYBASE IQ is used as the DBMS. Moreover in the model data from different sources will be used. Therefore these different data sources and WHO mortality database have to be combined. The combination of these different data sources will also be done in SYBASE IQ database. The combined data prepared in a database will be unloaded as CVS data file format.

3.1.1 WHO Mortality Database

4 file from Who Mortality Database is loaded for the project. Two file for mortality reason records, one file for population and one file for country codes mapping. Total mortality reason records in two file are 3,611,339. Population file has 9,349 records. To load these three files three different table is created named MORTALITY, POPULATION and COUNTRY. While loading mortality reason files to MORTALITY table some format changes were done.

- Some columns (Admin1, Subdiv) include information that is not necessary for the model not loaded.
- In files, there are different columns for each age group. These columns transposed in MORTALITY table in the AGE column.

The same format changes were also done while loading the population file into POPULATION table.

A table for mortality reason mapping is created. The mortality reason mappings have 3 level hierarchy. (1st level is more detail-2nd level groups of 1st level, 3rd level groups of 2nd level). In the project, models are done according to 3rd level mortality reasons. However, the MORTALITY dataset includes records with mortality codes in 1st level mortality reason. Therefore two level mapping is needed to map the 1st level mortality code to 3rd level mortality codes. The Mortality Reason mappings were done according to the ICD 10 Mortality Tabulation List 1 which are found in documentation file in the WHO Mortality Database. The mortality reasons table in the documentation was not suitable for a relational database table format.

For the relation between 1st and 2nd level, the mapping in the documentation is converted to format in Table 1 and the MORTALITY_REASON_MAPPING table is loaded. After loading MORTALITY_REASON_MAPPING has 2064 row with unique in DETAILED MORTALITY REASON CODE.

MORTALIT REASON CODE	MORTALITY REASON DESC	DETAILED MORTALITY REASON CODE
1025	Remainder of certain infectious and parasitic diseases	A22
1025	Remainder of certain infectious and parasitic diseases	A23
1025	Remainder of certain infectious and parasitic diseases	A24
1025	Remainder of certain infectious and parasitic diseases	A25
…	…	…
1082	Diseases of the skin and subcutaneous tissue	L40
1082	Diseases of the skin and subcutaneous tissue	L41
1082	Diseases of the skin and subcutaneous tissue	L42
1082	Diseases of the skin and subcutaneous tissue	L43
1082	Diseases of the skin and subcutaneous tissue	L44
…	..	…

Table 1: 1st and 2nd level Mortality Reason Mapping Sample.

For the relation between 2nd and 3rd level, the ICD 10 Mortality Tabulation List 1 mapping in the WHO documentation is converted to format in Table 2 and the MORTALITY_REASON_PARENT_MAPPING table is loaded. After loading MORTALITY_REASON_PARENT_MAPPING has 91 row with unique in MORTALITY REASON CODE.

MORTALIT REASON PARENT CODE	MORTALITY REASON PARENT CODE DESC	MORTALITY REASON CODE
1001	Certain infectious and parasitic diseases	1003
1001	Certain infectious and parasitic diseases	1004
1001	Certain infectious and parasitic diseases	1005
1001	Certain infectious and parasitic diseases	1006
…	…	…
1026	Neoplasms	1027
1026	Neoplasms	1028
1026	Neoplasms	1029
1026	Neoplasms	1030
1026	Neoplasms	1031
…	..	…

Table 2: 2nd and 3th level Mortality Reason Mapping Sample.

In addition, a table that maps age group format with age-groups corresponding columns in Mortality and Population datasets. Before load, the table mapping between these is done as a format in Table 3 according to Annex Table 1 -Age group formats in WHO documentation. For different age-groups formats, age groups are reported at different intervals. The format code for each age group was put into age group mapping. For a stable age group data, the detail age groups were converted to non-detail groups. The last version of age group was shown in Table 3:

AGE GROUP	MORTALITY REASON COLUMN NAME	POPULATION COLUMN NAME
ALL	Deaths1	Pop1
0	Deaths2	Pop2
1-4	Deaths3	Pop3
1-4	Deaths4	Pop4
1-4	Deaths5	Pop5
1-4	Deaths6	Pop6
5-9	Deaths7	Pop7
10-14	Deaths8	Pop8
15-19	Deaths9	Pop9
20-24	Deaths10	Pop10
25-29	Deaths11	Pop11
30-34	Deaths12	Pop12
35-39	Deaths13	Pop13
40-44	Deaths14	Pop14
45-49	Deaths15	Pop15
50-54	Deaths16	Pop16
55-59	Deaths17	Pop17
60-64	Deaths18	Pop18
>65	Deaths19	Pop19
>65	Deaths20	Pop20
>65	Deaths21	Pop21
>65	Deaths22	Pop22
>65	Deaths23	Pop23
>65	Deaths24	Pop24
>65	Deaths25	Pop25
UNKNOWN	Deaths26	Pop26

Table 3: Age Group Mapping Sample.

After all format changes were done the records containing null or meaningless values in any column was deleted from the tables MORTALITY and POPULATION.

3.1.2 Other Datasets and Tables

For all other datasets, those will be used as predictors in the model, new tables were created.

If datasets detailed by year, country and sex the table models are as follow:

COLUMN NAME	COLUMN TYPE
COUNTRY	varchar
YEAR	integer
SEX	integer
VALUE	number

Table 4: Other Datasets Table Format_1.

If datasets detailed by year, country the table models are as follow:

COLUMN NAME	COLUMN TYPE
COUNTRY	varchar
YEAR	integer
VALUE	number

Table 5: Other Datasets Table Format_2.

In same datasets, because the associated measures are not reported for all years, there were missing years. For these datasets, the missing years are filled with Geometric Mean. The datasets that have missing years are.

- Smoke rates of countries detailed by country, year and sex
- PM2.5 air pollution, mean annual exposure (micrograms per cubic meter) detailed by country and years
- Human Development Index by country and year

3.1.3 Combine Datasets

The tables created in the previous steps joined with the main MORTALITY table by year, country and sex (if detailed by sex). For the columns in all joined tables that are not used to join was put as a column in a new table named MORTALITY_REASON_DATAFRAME. The join method will be an inner join, so MORTALITY_REASON_DATAFRAME is loaded without any null values in any columns. MORTALITY_REASON__DATAFRAME table format and column descriptions are shown in Table 6

COLUMN NAME	COLUMN_TYPE	COLUMN DESCRIPTION
COUNTRY_CODE	integer	Codes assigned to countries in WHO Mortality Database
COUNTRY_NAME	varchar	Names of countries
YEAR	integer	Year
MORTALITY_REASON_CODE	integer	Codes assigned to countries in WHO Mortality Database
MORTALITY_REASON_DESC	varchar	Descriptions of mortality reason codes
SEX	integer	1 male, 2 female
AGE_GROUP CODE	varchar	Age groups code
MORTALITY	number	Mortality value
POPULATION	number	Population value
SMOKING	number	Smok rates
AIR_POLLUTION	number	PM2.5 air pollution
IMPROVED_DRINKINGWATER	number	Share of total population with improved water sources
BMI	number	Body Mass Index
HEALTHCARE_EXPENDITURE	number	Total Healthcare Expenditure as Share of National GDP
HDI	number	Human Development Index
GDP	number	Gross Domestic Product.

Table 6: Mortality Reason Data frame Table Format.

After MORTALITY_REASON_MODEL table was loaded and the data produced was unloaded as csv file format.

3.1.4 MORTALITY_REASON_MODEL Table Analysis

Analysis on the MORTALITY_REASON_MODEL table is done before unloading it as CSV file.

- The countries and the number of records for each country are: United Kingdom (9044) , Spain (9044) , Netherlands (9044) , Japan (9044) , Germany (9044) , France (9044) , Egypt (9044) , Israel (8942) , Belgium (8874) , Sweden (8840) , Denmark (8602) , Poland (8568) , Norway (8568) , Latvia (8568) , Hungary (8568) , Romania (8534) , Lithuania (8534) , Czech Republic (8466) , Finland (8432) , Kyrgyzstan (8398) , Croatia (8398) , Australia (8398) , Switzerland (8296) , Serbia (8296) , Estonia (8262) , Slovenia (8228) , Austria (8058) , Luxembourg (7854) , Malta (7752) , Italy (7752) , Iceland (7616) , Slovakia (7412) , Georgia (6392) , Bulgaria (6358) , Portugal (6290) , Bahrain (6256) , Thailand (5746) , Mauritius (5202) , South Africa (5168) , Malaysia (5134) , Turkey (3740) , Armenia (3672) , Ireland (3536) , Canada (3230) , Jordan (2924) , Singapore (1734) , Uzbekistan (1292) , Kazakhstan (1292) , Bosnia and Herzegovina (1190) , Azerbaijan (612) , Oman (578) , Greece (578) , Fiji (578). To sum up there are records for 53 distinct country.
- The Sex and the number of records for each sex are: Male (171513) Female (171513). As it was seen the numbers of observations are same.
- For each age groups , (0, 1-4, 10-14, 15-19, 20-24, 25-29, 30-34, 35-39, 40-44, 45-49, 5-9, 50-54, 55-59, 60-64, >65) the number of records are 20178.
- The Mortality Reasons and number of records for each mortality reasons are shown in Table 7.

MORTALITY_REASON_DESCRIPTION	number of observations
Certain conditions originating in the perinatal period	19006
Certain infectious and parasitic diseases	19006
Congenital malformations, deformations and chromosomal abnormalities	19006
Diseases of the blood and blood-forming organs and certain disorders involving the immune mechanism	18938
Diseases of the circulatory system	19006
Diseases of the digestive system	19006
Diseases of the ear and mastoid process	13600
Diseases of the eye and adnexa	8806
Diseases of the genitourinary system	19006
Diseases of the musculoskeletal system and connective tissue	18972
Diseases of the nervous system	19006
Diseases of the respiratory system	19006
Diseases of the skin and subcutaneous tissue	18564
Endocrine, nutritional and metabolic diseases	19006
External causes of morbidity and mortality	19006
Mental and behavioural disorders	18972
Neoplasms	19006
Pregnancy, childbirth and the puerperium	17102
Symptoms, signs and abnormal clinical and laboratory findings, not elsewhere classified	19006

Table 7: Number of observation for each Mortality Reason

3.2. Exploratory Data Analysis

To make exploratory data analysis the file CSV; produced after database process; is loaded to a data frame in PYTHON platform. As a first analysis, the number of observation and feature is checked. After all cleaning and combining steps, in Database 343026 observation remains with 16 features. The steps exploratory data analysis and processes according to results are:

- Check the types of features. The SEX, COUNTRY_CODE, MORTALITY_REASON_CODE features are converted to categorical values from numeric values.
- Check Na values. Only “number_of_population” feature has 20918 NAN values. We remove that 20918 observation from data frame.
- Check the unique values for categorical features, AGE_GROUP, MORTALITY_REASON_DESC, COUNTRY_CODE. All results can be seen in appendix. For age_group feature there were values ‘ALL’ and ‘UNKNOWN’. We get rid of from these observations.
- Control the correlations between numeric features. The features that have correlations higher than 0.6, ordered by correlation values, are HDI & GDP, IMPROVED_DRINKINGWATER & HDI, HEALTHCARE_EXPENDITURE & HDI. The correlation matrix can be seen in Figure 1:

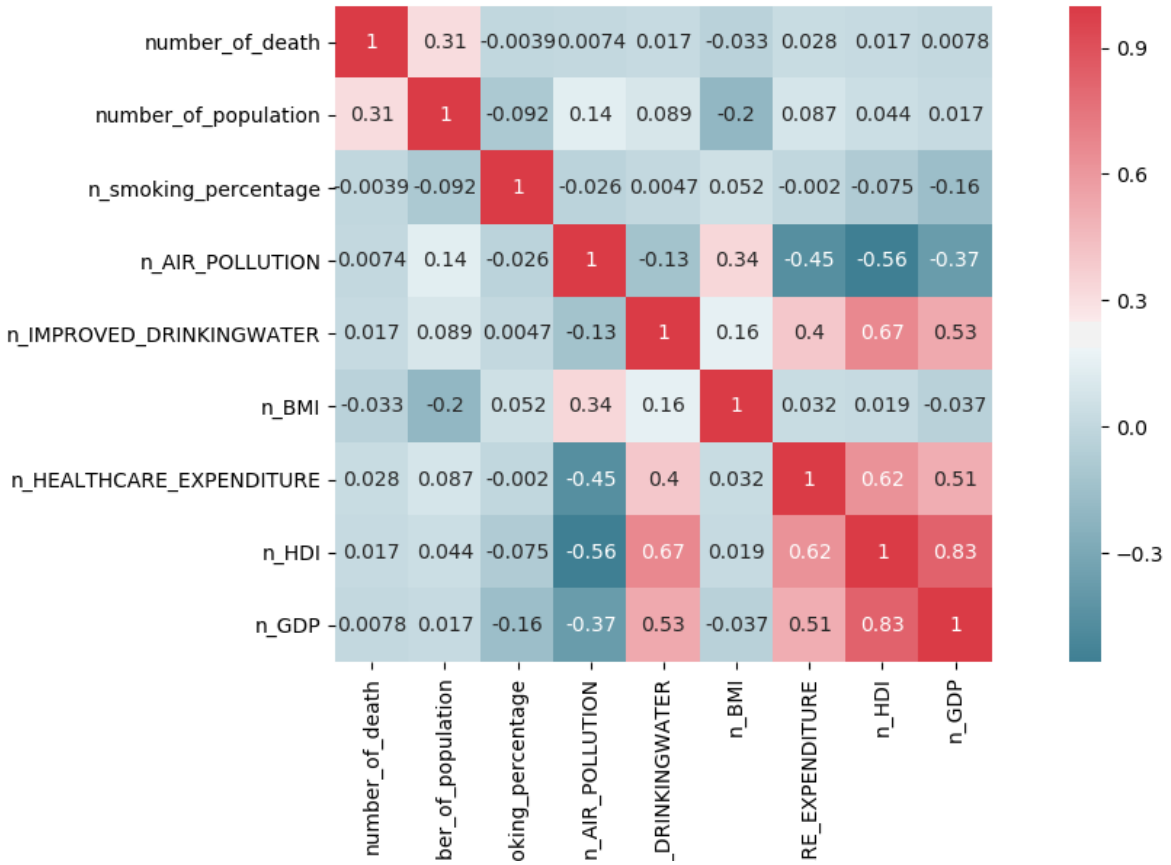

Figure 1: Correlation Matrix for numeric values

- Check the skewness of numeric features and plot KDE. The skewness values of all features can be seen in Table 8.KDE plots can be seen in Figure 2.

number_of_death	22.343613
number_of_population	4.327895
smoking_percentage	-0.045823
AIR_POLLUTION	3.360961
IMPROVED_DRINKINGWATER	-3.103840
BMI	-0.807375
HEALTHCARE_EXPENDITURE	-0.292766
HDI	-0.888111
GDP	0.760095

Table 8: Skewness of numeric features.

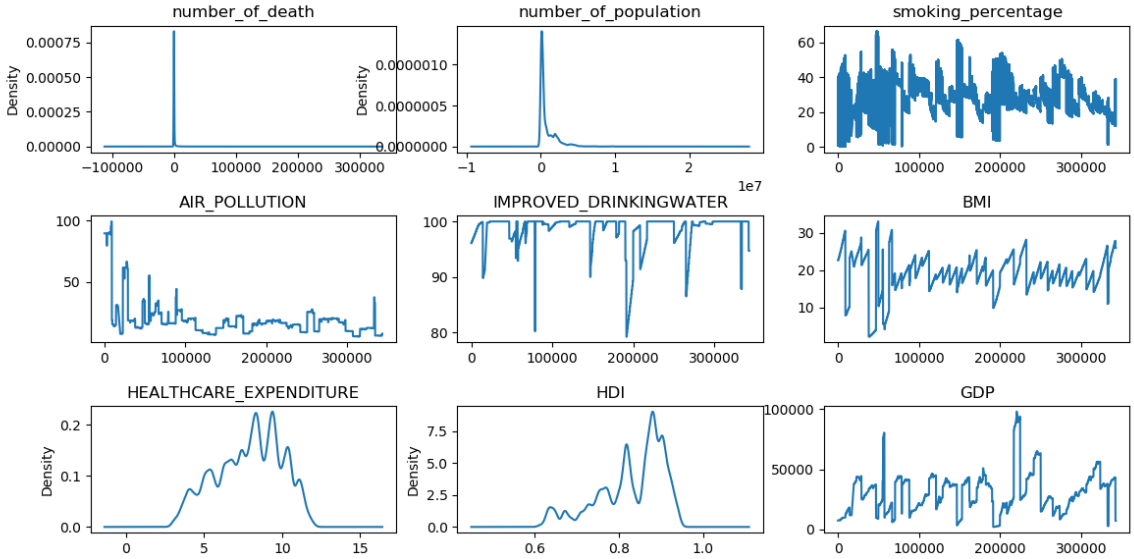

Figure 2: KDE plots for numeric features. The x-axis of the graph shows the values, according to the name of the graph, number of death, number of population, smoking percentages, air pollution, improved drinking water, BMI, healthcare expenditure, HDI,GDP. Y-axis shows the density of the corresponding x values.

- In the previous step, it was seen that the skewness of all numeric features is not good enough and not are normally distributed so the values of all features are replaced by logs of their values. The new KDE plot can be seen in Figure 3.

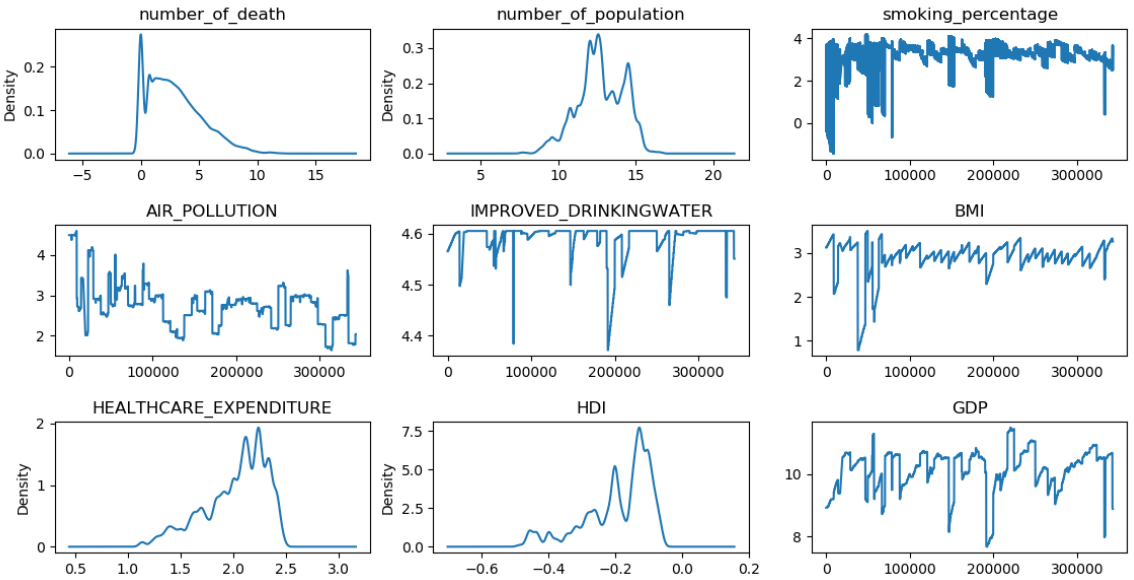

Figure 3: KDE plots after log transformation for numeric features. The x and y axises explanation are the same as the previous figure 2.

- For regression modeling, DUMMY columns are created for the categorical variables AGE_GROUP_CODE and MORTALITY_REASON_DESC and SEX.

3.3. Regression Model

Five Different regression models Linear Regression, Ridge Regression, Lasso Regression, Random Forest Regression and Gradient Boosting are fitted in PYTHON platform using sklearn and statsmodels packages. Before fitting the models to find the best parameters GRIDSEARCH with 10-fold cross validation is applied for all type of regression models except the linear regression model. For grid search python GridSearchCV library from sklearn.model_selection package is used. According to grid search results, best parameters found for regression models. Models are fitted with parameters results of grid search. For Linear Regression models two different Python libraries are used. The sklearn packages are used GRIDSEARCH library uses these package functions. On the other hand, the functions in statistic package libraries produced Multi Linear Statistical outcomes that sklearn package not produced such as p-values of all features, F score; MSE and so on. These statistical outcomes help to interpret the statistical significance of the model and features. To be sure if these two different python packages' libraries produced the same result, the r^2 scores and coefficients produced for all regression models by these packages are compared. For interpreting the Ridge Regression, Lasso Regression r^2 values, coefficients are used. Because these models are regularized linear regression p-values of features are not produced, the coefficients of features are shown if the features are significant or not. For interpreting the Random Forest Regression and Gradient Boosting models and their features r^2 values and importance values of the features are used. These indexes are also produced by sklearn packages so there is no necessity for using statistic package. After the models are fitted the predictions that have the biggest error is examined and tried to understand if a misprediction is occurred because of an abnormality such as war, natural disasters and so on. In addition, separated models for all type of the mortality reason are also built to analyze the significance of features according to all type of mortality reason.

4. RESULTS

For all type of regression models, before fitting them Grid Search with 10 fold cross validation is applied to find the best parameters. For grid search algorithm GridSearchCV function of sklearn.model_selection is used. In next steps of the study these parameters are used when fitting the model. The parameters grids and results for grid search:

- Linear Regression:

  Search Grid => fit_intercept:[True,False]=calculate intercept
  normalize:[True,False]=normalize X before fitting.
  copy_X : [True, False] : Copy X or overwrite.

  Best Parameters => fit_intercept : True
  normalize: False
  copy_X : True

  Best R^2 Score => 0.721

- Ridge Regression:

  Search Grid => alpha : [100,50,10,1.0,0.1,0.01,70,60,40,30]=Regularization strength

  Best Parameters => alpha : 50

  Best R^2 Score => 0.721

- Lasso Regression:

  Search Grid => alpha : [0.1,0.01,0.001,0.009,0.0005]=Regularization strength

  Best Parameters => alpha : 0.0005

  Best R^2 Score => 0.7204

- Random Forest Regression:

  Search Grid => n_estimators : [1,10,50,70] = number of trees

  Best Parameters => alpha : 50

  Best R^2 Score => 0.86

- Gradient Boosting Regressor:

  Search Grid => learning_rate: [0.1,0.01]=shrinks the contribution of each tree
  max_depth: [4,6] = maximum depth of each tree
  n_estimators : [10,50,100,200] = number of trees

  Best Parameters => learning_rate: 0.1
  max_depth: 6
  n_estimators : 200

  Best R^2 Score => 0.874

4.1 Models for Overall Mortality Reason Results

Before fitting the model data frame is split into train-test with 0.7-0.3 rate. The model's scores in these simple splitting are higher than the cross-validation results. However, the cross-validation results are more reliable than simple train-test splitting.

4.1.1 R^2 scores and Prediction Errors

The models fitted with parameters found in grid search and compared the according to their r^2 scores. Also, for each model train and test prediction scores are compared to understand if there are overfittings in the models.

The r^2 scores of Linear, Ridge and Lasso regression are close to each other. This shows us the correlations between the features are not high enough to affect the model scores. The similarity of these models' prediction can also be seen in their Prediction Error Plots. In plots x-axis (named as y) show the actual number of deaths and y-axis (named as y^) shows predicted number of deaths. According to the plots, the error margins are decreasing when the number of deaths increasing.

The summary of r^2 scores with 10-fold cross validation method and train-test split with 0.7-0.3 rate method can be seen in the Table 9.

	10-fold R^2	**Train-Test Split R^2**
Linear Regression	0.721	0.747
Ridge Regression	0.721	0.746
Lasso Regression	0.7204	0.747
Random Forest Regression	0.86	0.900
Gradient Boosting Regressor	0.874	0.900

Table 9: R^2 scores

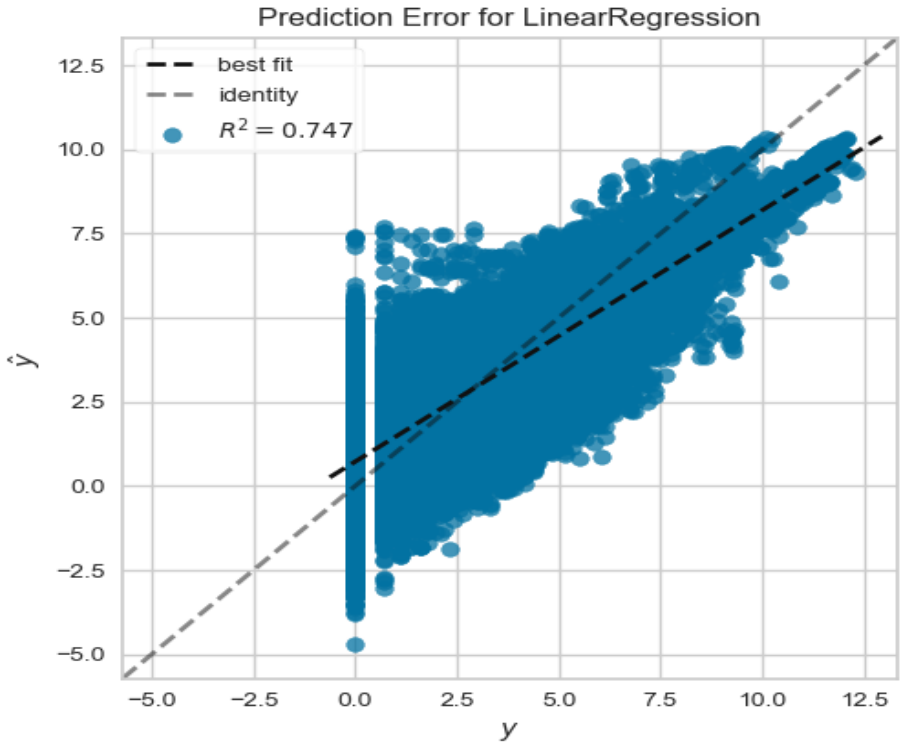

Figure 4 : Prediction Error Plot for Linear Regression

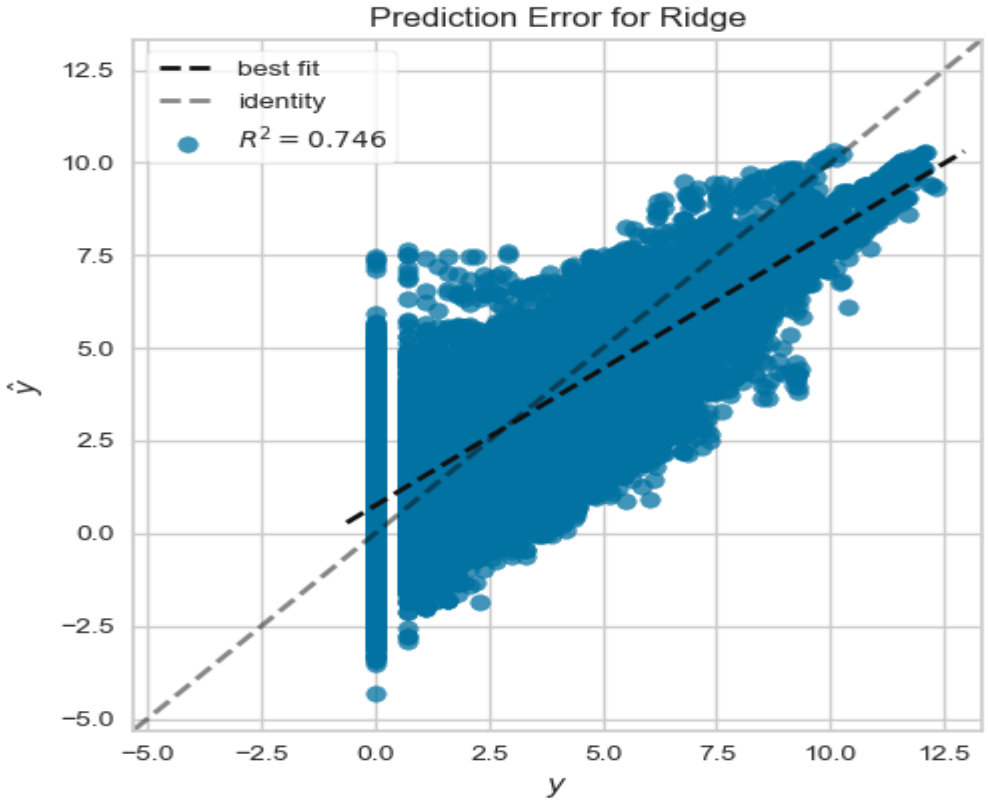

Figure 5 : Prediction Error Plot for Ridge Regression

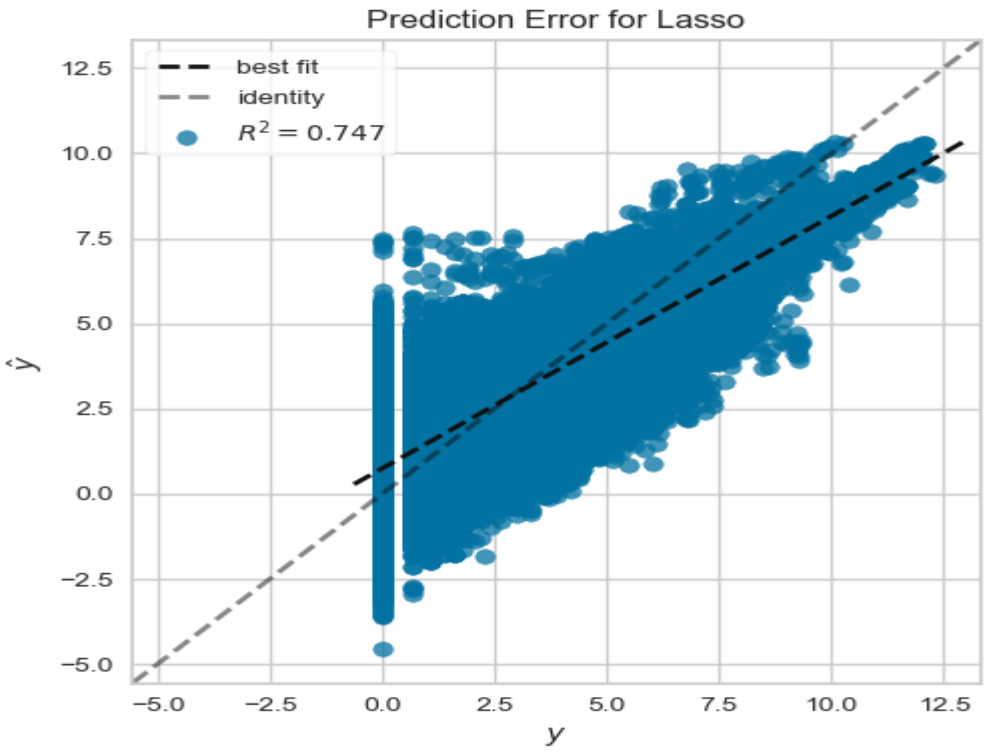

Figure 6: Prediction Error Plot for Lasso Regression

The R^2 scores for Random Forest Regressor and Gradient Boosting Regressor are better than the Linear, Ridge and Lasso regressions. On the other hand, the r^2 scores of these regressor models close to each other. They are both ensemble methods based on the decision tree. Unlike Random Forest in Gradient Boosting the observations are weighted according to their errors. The plots show that error weighting for observation does not affect the prediction results much. The similarity of Random Forest Regressor and Gradient Boosting Regressor predictions can also be seen in their Prediction Error Plots. In plots x-axis (named as y) show the actual number of deaths and y-axis (named as y^) shows predicted number of deaths. According to plots, the error margins are decreasing when the number of deaths increasing.

Figure 7: Prediction Error Plot for Random Forest Regressor

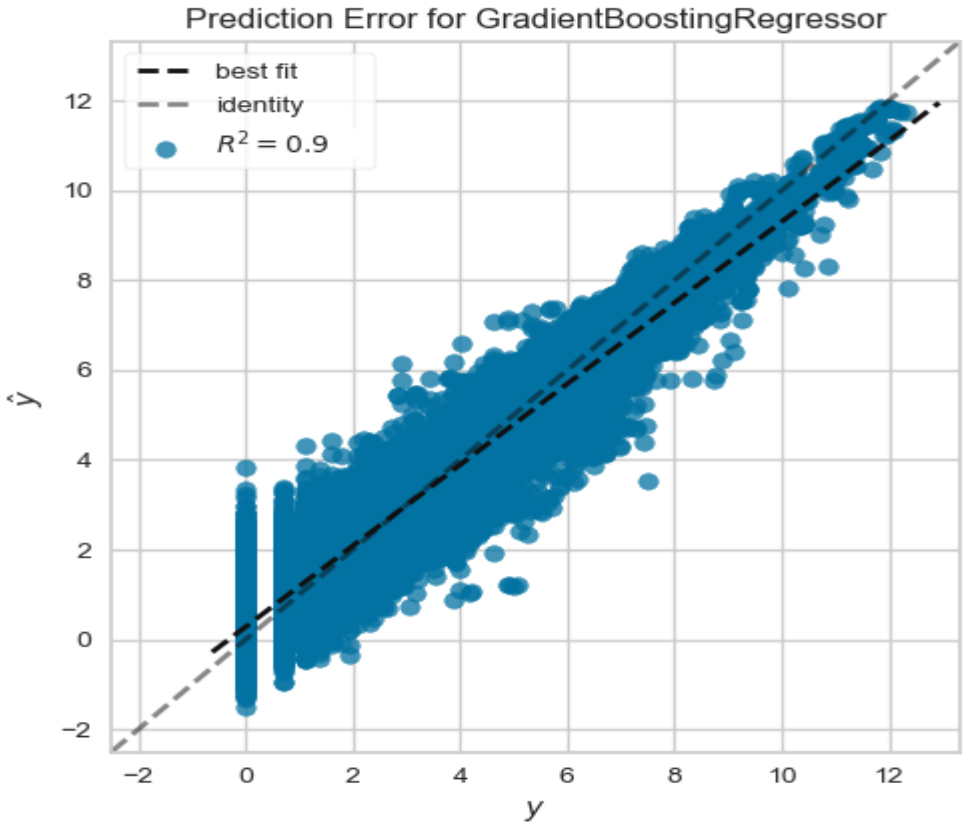

Figure 8: Prediction Error Plot for Gradient Boosting Regressor

4.1.2 Feature Analysis

To understand the effect of each feature in models, different indicators are used according to the type of regression model. For the linear model, the p-value scored are compared. If the p-value of a feature is less than 0.05, it means that the feature is statistically significant. For regularized linear models, ridge and lasso, the values of coefficients of features are used as the indicator. For ridge, if a feature coefficient is too small means the feature is not significant. For lasso, if a feature coefficient equals to 0 means the feature is not significant and not have any effect on predicting the number of deaths. For Random Forest Regressor and Gradient Boosting Regressor, importance values of the features are used as the indicator. As it was known these two models based on decision tree instead of coefficients or p-values importance of the features are calculated for showing the effects of the features in the model. The combination of results for non-dummy features can be seen in Table 10. The full list can be seen in APPENDIX-A.

	Linear (p-value)	Ridge (Coefficent)	Lasso (Coefficent)	RandomForest (importance)	GradientBoosting (importance)
number_of_population_LOG	**0**	**0,832730709**	**0,833782629**	**0,225738601**	**0,117158119**
n_smoking_percentage_LOG	7,48859E-05	-0,028284474	-0,021697431	**0,017406579**	0,018105463
n_AIR_POLLUTION_LOG	1,27413E-14	-0,031014904	-0,055449877	0,011736519	0,034567638
n_IMPROVED_DRINKINGWATER_LOG	1,25799E-28	**-1,059280589**	**-0,745710401**	0,009721158	0,017557328
n_BMI_LOG	5,27709E-27	0,077575269	0,076002078	0,008873718	0,018748745
n_HEALTHCARE_EXPENDITURE_LOG	6,66345E-25	-0,201118006	-0,167084287	0,013273588	**0,038638463**
n_HDI_LOG	**0**	**-2,97464631**	**-3,70165276**	**0,023956458**	**0,042235237**
n_GDP_LOG	**9,87384E-89**	0,113991697	0,176996973	0,011249181	0,026856176

Table 10: Feature analysis of models. The bold values show first three important features for all models.

The order of each non-dummy features according to their importance can be seen in Table11.

	Linear	Ridge	Lasso	RandomForest	GradientBoosting
number_of_population_LOG	**1**	**3**	**2**	**1**	**1**
n_smoking_percentage_LOG	7	8	8	**3**	7
n_AIR_POLLUTION_LOG	6	7	7	5	4
n_IMPROVED_DRINKINGWATER_LOG	**3**	**2**	**3**	7	8
n_BMI_LOG	4	6	6	8	6
n_HEALTHCARE_EXPENDITURE_LOG	5	4	5	4	**3**
n_HDI_LOG	**1**	**1**	**1**	**2**	**2**
n_GDP_LOG	**2**	5	4	6	5

Table 11: The order each non-dummy features according to their importance. The bold values show first three important features for all models.

The thresholds for the analysis of feature significance are 0.05 for p-values in linear regression, 0.001 for absolute of coefficient in Ridge and Lasso and 0.001 for importance values of Random Forest and Gradient Boosting. According to these thresholds, all of the continuous predictors have effect on forecasting mortality. According to the table 11 number of population and HDI are one of the most important features for all models. The effect of population on the number of death is an expected result. The higher number of population leads the higher of death. The effect of HDI shows the importance of human development level in life span of human being. To increase the lifetime of human being governments have to decide strategies to increase the Human Development Indexes of their countries. From table 11 GDP has also has high effect in Linear Model,however not have much effect on the other models. This is most probably because it has high correlation with HDI. The importance of IMPROVED_DRINKINGWATER is high in Linear,Ridge and Lasso models. On the other hand, it is one of the worst two feature in Random Forest and Gradient Boosting models. This is most probably because of the random chosen of predictors' orders in decision trees nodes build in these models. In this models local optimum results are found. Increasing the number of iteration and number of random initialization the results for importance of features can be changed. However, there are not many differences between the importance of IMPROVED_DRINKINGWATER and the other features there is no necessity to make such a change in the models. Because these changes increase the run time of models exponentially.

4.1.3 Top 1000 Errors Analysis

To understand if there are apparent errors for some situations, if there are cases model can't make the prediction or to find errors that are occurred as a result of abnormalities such as war, natural disasters. Top 1000 errors investigated. Top 1000 errors are found according to absolute (actual number of deaths – predicted number of death) formula.

In Linear, Ridge, Lasso regression models, nearly 0.35 of the top 1000 errors belongs to observations that mortality reason is 'Certain conditions originating in the perinatal period'. 0.35 of the top 1000 errors belongs to observations that mortality reason is 'Congenital malformations, deformations, and chromosomal abnormalities'. The worst predictions are done for ages after >50 and when the number of deaths equals 1. On the other hand, there is not an explicable description for the real reason of misprediction. Most of the times the deaths for that mortality reason is not recorded correctly because diagnosing it are difficult. Therefore; models can't create a stable linear function between those mortality reason number and the predictors. Moreover, nearly most of the mispredictions for 'Certain infectious and parasitic diseases' and 'Diseases of the blood and blood-forming organs and certain disorders involving the immune mechanism' belongs to South Africa between the years 2007-2009. The observations in the dataset are examined. It was seen that the number of deaths for that records that coherent with the distribution of the other records for that country. Therefore the model cannot make a correct prediction. To sum up, nearly %85 of the top 1000 errors can be explained by the noises in the dataset. This shows the reason for mispredictions is not the weakness of the model. On the other hand, for Random Forest and Gradient Boosting Regression, there is not an absolute reason that has more errors according to other mortality reason. This is most probably a result of the underlyıng mechanism in Random Forest and Gradient Boosting algorithms. These algorithms based on decision tree create different decision trees using different splitting and predictors and get average of these different trees to result so they are good at predicting the values that have high variance, as it was in Certain conditions originating in the perinatal period'. 0.35 of the top 1000 errors belongs to observations that mortality reason is 'Congenital malformations, deformations, and chromosomal abnormalities'. Errors rates of age groups and country distributed uniformly. Only for South Africa misprediction rate is a little higher than the other countries, most

probably because the observations are not recorded well enough. The uniform distribution of errors shows that these models handle the most of the noises in the dataset and they are stable.

4.2 Separated Models for All Mortality Reason Results

Besides creating models including all type of mortality reason, different models are developed for all type of 19 mortality reason. For fitting the models Linear Regression, Ridge Regression, Lasso Regression, Random Forest Regression and Gradient Boosting Regression algorithms are used for each type of mortality reason. The parameters that are found for the models of all type of mortality reason are also used in the models mentioned in this part of the study.

4.2.1 R^2 scores, Train and Test Accuracy Scores

The r^2 scores of models in test dataset can be seen in table 12.

MORTALITY_REASON_DESC	Linear	Ridge	Lasso	Random Forest	GradientBoosting
Certain infectious and parasitic diseases	0,8280303	0,7981051	0,8283151	0,949963459	0,95174719
Neoplasms	0,9688569	0,9664999	0,9687643	0,983782123	0,984737902
Diseases of the blood and blood-forming organs and certain disorders involving the immune mechanism	0,7773128	0,7546299	0,77797	0,905882911	0,909341202
Endocrine, nutritional and metabolic diseases	0,8849334	0,8770885	0,8848745	0,954587126	0,957189602
Mental and behavioural disorders	0,7193651	0,7113934	0,7195685	0,923551426	0,936168132
Diseases of the nervous system	0,886907	0,8802096	0,8868286	0,942969822	0,946816874
Diseases of the eye and adnexa	**0,3206499**	**0,3296177**	**0,3213662**	0,700476917	0,641621553
Diseases of the ear and mastoid process	**0,4636142**	**0,3920646**	**0,4554675**	0,623289679	0,553946512
Diseases of the circulatory system	0,9527592	0,9501338	0,9527758	0,982409278	0,983881263
Diseases of the respiratory system	0,90312	0,8936522	0,9030537	0,965765114	0,965978192
Diseases of the digestive system	0,8973789	0,8929922	0,8976093	0,967020861	0,970655958
Diseases of the skin and subcutaneous tissue	0,6973226	0,676019	0,6976215	0,899921617	0,910205472
Diseases of the musculoskeletal system and connective tissue	0,7795527	0,7723968	0,7794137	0,902908663	0,913337503
Diseases of the genitourinary system	0,8963782	0,8853051	0,8963338	0,953724635	0,957394813
Pregnancy, childbirth and the puerperium	0,7370152	0,6550321	0,7363921	0,83184997	0,831962201
Certain conditions originating in the perinatal period	0,9057104	0,8681221	0,9055205	0,966952757	0,966151164
Congenital malformations, deformations and chromosomal abnormalities	0,7551049	0,7523712	0,7551533	0,886158153	0,88925807
Symptoms, signs and abnormal clinical and laboratory findings, not elsewhere classified	0,77432	0,7632534	0,7739225	0,948194479	0,955658348
External causes of morbidity and mortality	0,9055822	0,902892	0,9056535	0,970370163	0,972052006

Table 12: R^2 scores of test dataset for all mortality reason and regression algorithm

According to Table 12 “Diseases of the eye and adnexa” and “Diseases of the ear and mastoid process” have absolute lower scores than the scores of general models. In Table 13 the worst scores for test and train dataset are shown to better understanding. In APPENDIX B the full table for r^2 scores and train test accuracy scores can be seen.

In the dataset the number of these mortality reason observations is low; “Diseases of the ear and mastoid process” have 13600 “Diseases of the eye and adnexa” has 8806. As a result of these linear, ridge and lasso regression can’t develop stable learning process. Also again as a result of not having enough example in Random Forest and Gradient

Boosting there is a big difference between the training score and test score which means there is overfitting.

REGRESSION_TYPE	MORTALITY_REASON_DESC	R^2_SCORE	TRAINING_SCORE	TEST_SCORE
Linear	Diseases of the eye and adnexa	0,3206499	0,420233945	0,320649946
Lasso	Diseases of the eye and adnexa	0,3213662	0,419903282	0,321366247
Ridge	Diseases of the eye and adnexa	0,3296177	0,379262892	0,329617659
Ridge	Diseases of the ear and mastoid process	0,3920646	0,375690975	0,392064605
Lasso	Diseases of the ear and mastoid process	0,4554675	0,42007805	0,455467478
Linear	Diseases of the ear and mastoid process	0,4636142	0,419327007	0,463614209
GradientBoosting	Diseases of the ear and mastoid process	0,5539465	0,980928757	0,553946512
Random Forest	Diseases of the ear and mastoid process	0,6232897	0,940369852	0,623289679
GradientBoosting	Diseases of the eye and adnexa	0,6416216	0,998174579	0,641621553

Table 13: The scores for Diseases of the eye and adnexa" and "Diseases of the ear and mastoid process" models.

4.2.2 Feature Analysis

To understand the effect of each feature in each model separated according to mortality reasons, different indicators are used according to the type of regression model. As in overall model feature analysis step for linear regression p-values of features, for ridge and lasso regressions coefficients of features and for random forest and gradient boosting the importance of the features are used as the indicator to measure the effect of the predictors. Because the tables created for the feature analysis of all models is too big it is not added in this paper, only the important results are showed and mentioned.

According the linear regression p-values results:

- Sex is not statistically significant for the 'Certain infectious and parasitic diseases', 'Mental and behavioral disorders', 'Diseases of the nervous system',' Diseases of the eye and adnexa', 'Diseases of the digestive system',' Diseases of the musculoskeletal system and connective tissue',' Diseases of the genitourinary system',' Pregnancy, childbirth and the puerperium',' Symptoms, signs and abnormal clinical and laboratory findings, not elsewhere classified', 'External causes of morbidity and mortality' reasons. That means there is no difference in the

number of death between Male and Female for that reasons. Pregnancy, childbirth and the puerperium is also on the list. Although this mortality reason is special for females sex is not a significant feature. Because all the observation of datasets belongs to females so the model cannot make a separation between male and female.

- Smoking is not statistically significant for the 'Certain infectious and parasitic diseases',' Diseases of the eye and adnexa',' Diseases of the ear and mastoid process',' Diseases of the circulatory system',' Certain conditions originating in the perinatal period'. The result for Certain infectious and parasitic diseases',' Diseases of the eye and adnexa',' Diseases of the ear and mastoid process' is reasonable. Smoking not effects on these diseases. On the other hand, it is interesting that smoking is not significant on 'Diseases of the circulatory system' and 'Certain conditions originating in the perinatal period'.The result for 'Diseases of the circulatory system' may be explained by smoking has more effect on 'Diseases of the respiratory system' and 'Diseases of the blood and blood-forming organs and certain disorders involving the immune mechanism'.So the deaths affected by smoking are mostly related to these two diseases. The result for 'Certain conditions originating in the perinatal period' is most probably because the smoking of mother affect the child and leads certain abnormalities and can not related to the death of child directly.
- Age is not statistically significant for 'Mental and behavioural disorders',' Diseases of the nervous system',' Diseases of the eye and adnexa', 'Diseases of the digestive system',' Diseases of the musculoskeletal system and connective tissue',' Diseases of the genitourinary system', 'Symptoms, signs and abnormal clinical and laboratory findings, not else where classified',' External causes of morbidity and mortality'. On the other hand, the result for Diseases of the eye and adnexa' is less trustable because the r^2 scores of these models are very low as mentioned before. It is reasonable that age has no effect on 'Mental and behavioral disorders', 'Symptoms, signs and abnormal clinical and laboratory findings, not else where classified',' External causes of morbidity and mortality'. The mortality reasons contain the different type of mortalities. Some of the mortalities classified in those labels may have relation with age. But the effects of age is may be balanced when

the overall of those these are summed. On the other hand, it is interesting that age has no effect on ‘Diseases of the nervous system’,‘Diseases of the digestive system’,’ Diseases of the musculoskeletal system and connective tissue’,’ Diseases of the genitourinary system’. The number of death related to those reasons for age group 60-65 and >65 is much higher than the other age groups. Analyzing the distribution of age-groups in the dataset may help to find a reasonable explanation.

- Air pollution is not statistically significant for the ‘Mental and behavioural disorders’,’ Diseases of the ear and mastoid process’,’ Pregnancy, childbirth and the puerperium’,’ Congenital malformations, deformations and chromosomal abnormalities ’. As mentioned above, the result for Diseases of the ear and mastoid process’ is less trustable because the low r^2 scores of these models. It is reasonable that air pollution has no effect on ‘Mental and behavioral disorders’,’ Congenital malformations, deformations and chromosomal abnormalities’, ’Pregnancy, childbirth and the puerperium’. Air pollution may affect those diseases. However, its effect is so little and cannot measure directly. These diseases are more related to genetical issues or social issues like HDI, GDP.
- HDI and GDP is not statistically significant for the ‘Certain conditions originating in the perinatal period’, ’Congenital malformations, deformations and chromosomal abnormalities’. It is reasonable that air pollution has no effect on ‘’Congenital malformations, deformations and chromosomal abnormalities’. This mortality reason is mostly related to genetic or environmental issues. On the other hand, it is interesting that it is interesting that HDI and GDP have no effect on ‘Certain conditions originating in the perinatal period’. The mothers in countries with higher HDI and GDP scores are more aware of the risks in pregnancy period. This result shows that genetic and environmental issues are more effective in the ‘Certain conditions originating in the perinatal period’.

For ridge and lasso regression, we take the threshold for the absolute value of coefficients as 0.001 and assume that features that have coefficients with absolute values less than 0.001 are not significant. Nearly for all models is not significant. The reason for the differences between linear regression and ridge and lasso regression models results may be caused because of the alpha for the penalty scores and the threshold that was chosen for measuring the significance of the coefficient.

For Random Forest and Gradient Boosting, we take the threshold for the value of importance as 0.001 and assume that features that have importance with values less than 0.001 are not significant. Sex is not statistically significant for the ‘Pregnancy, childbirth and the puerperium’,’ Certain conditions originating in the perinatal period’,’ Neoplasms.’. Pregnancy, childbirth and the puerperium is a mortality reason special for females. However, sex is not a significant feature of the models. Because all the observation of datasets belongs to females so the model cannot make a separation between male and female. The result for ’ Certain conditions originating in the perinatal period’,’ Neoplasms’ shows that the death ratio of males and females on those reasons are close to each other. It is an interesting outcome but cannot be said unreasonable.

4.3 Conclusion

Although all algorithms' r^2 scores and prediction errors are in an acceptable range, Gradient Boosting or Random Forest algorithms are better than Linear Regression, Ridge and Lasso Regression algorithm in predicting mortality. This is a predictive result because of the variance in our dataset. Random Forest and Gradient Boosting are ensemble methods in machine learning so they are good when modeling in the dataset with high variance. Furthermore, the r^2 and prediction errors of Random Forest and Gradient Boosting are close to each other. However, the performance of Random Forest according to running time is better than Gradient Boosting. Random Forest is a bagging method and the trees are fitted in parallel. In contrary, Gradient Boosting is a boosting method and the trees fitted sequentially. Therefore, the performance of Random Forest according to running time is better than gradient boosting. The runtimes of models can be seen in Table 13. Gradient boosting gives better prediction in complex models. In our study, our models are not too complex and the prediction scores of Random Forest and Gradient Boosting are close. To sum up, between the regressions methods studied in these paper Random Forest is the best one.

	runtime(second)
Linear Regression	6.686
Ridge Regression	2.847
Lasso Regression	126.4
Random Forest	1099
Gradient Boosting	3074

Table 13: Runtimes of Models for Overall Mortality Reasons.

According to the feature analysis, all features are significant for the overall model. However, in detail models for each mortality reason, some features are not significant. Although some of these results are acceptable some are may not be trustable because the scores of models for some mortality reasons are not in an acceptable range. For further studies using any oversampling method, the observation of these mortality reasons can be increased. This study may increase r^2 scores of the models for "Diseases of the eye and adnexa" and "Diseases of the ear and mastoid process".

5. SOCIAL AND ETHICAL ASPECTS

The increase in life expectancy is increasing as the years are passing. Especially in developed countries as a result of the increase in aging of population brings the questions, if that increase stops in future, how it will affect the birth rates, what will the population in future, how society and cultures will be affected by being older and so on. The age of the population will affect the Social Security policies of countries, for example, the retirement age must be rearranged. Moreover the insurance and banking sectors will also be affected by the increase in lifespan, for example, as it was known age is a big factor in insurance pricing and credit confirming. To meet the expectations and find answers to these entire question, mortality forecasting become an important issue. The overall average lifetime in developed countries is much higher. However, for some typical mortality reasons, the developing countries have higher rates than the undeveloped countries. Moreover usually females live longer the males. These phenomena's shows not only forecasting mortality is enough, age-reason and country based forecasting must be developed to better understanding mortality and developed and planed health policies according to it. In this paper, the effects of some social and economic issues on mortality are analyzed based on sex and age. The mortality forecast model, developed in this paper helps to predict mortality and the results of the model show the effects of some social and economic issues effect on mortality.

6. VALUE DELIVERED

In this project mortality prediction detailed on reason, age and sex are produced by using five common machine learning regression algorithms, Linear Regression, Ridge Regression, Lasso Regression, Random Forest Regression, Gradient Boosting Regression.

The accuracy and running time performances of models are compared to choose the best model. The algorithm and the libraries that were used in modeling are the five of the most common machine learning algorithms. Therefore, the results of the models are trustable according to mathematical and statistical issues. The feature analysis of the results proved that all the features are relevant and useful for predicting mortality. The wrong predictions are also investigated to identify the cases the models do not work well so that they can be used for improving the models. Some of the mortality reason specifics models gave better results than the overall model. These separate models can also be used when trying to predict specific mortalities. Moreover, all the models are flexible and can be improved easily. Adding new features, new data can be easily applied to the models. This study shows how some social and environmental issues affect the number of death according to mortality reason, age group, and sex. By adding more observations time series analysis can be done to predict the future. With time series models the future values of predictor features can be predicted so the model can be run to predict the future death values. In this study as mentioned before social and environmental issues are concerned, but no genetical issue is concerned. Genetic of the human being has an important effect on diseases and mortality reasons. Races dimension can be added to the model especially for multinational countries so that what is the distribution of mortalities on different races can be analyzed. That analysis may help to understand which races are to immunize to which mortality reasons.

APPENDIX A

Feature analysis of models

	Linear(p-value)	Ridge(Coefficent)	Lasso(Coefficent)	RF(importance)	GB(importance)
const	6,82792E-05	-5,04651717	-7,161247929	NaN	NaN
number_of_population_LOG	0	0,832730709	0,833782629	0,225738601	0,117158119
n_smoking_percentage_LOG	7,48859E-05	-0,028284474	-0,021697431	0,017406579	0,018105463
n_AIR_POLLUTION_LOG	1,27413E-14	-0,031014904	-0,055449877	0,011736519	0,034567638
n_IMPROVED_DRINKINGWATER_LOG	1,25799E-28	-1,059280589	-0,745710401	0,009721158	0,017557328
n_BMI_LOG	5,27709E-27	0,077575269	0,076002078	0,008873718	0,018748745
n_HEALTHCARE_EXPENDITURE_LOG	6,66345E-25	-0,201118006	-0,167084287	0,013273588	0,038638463
n_HDI_LOG	0	-2,97464631	-3,70165276	0,023956458	0,042235237
n_GDP_LOG	9,87384E-89	0,113991697	0,176996973	0,011249181	0,026856176
n_sex_FEMALE	6,82792E-05	-0,216996288	-0,427775939	0,005739918	0,005735713
n_sex_MALE	6,82792E-05	0,216996288	5,63538E-14	0,005692109	0,005540346
MORTALITY_REASON_DESC_Certain conditions originating in the perinatal period	6,82792E-05	0,233094203	0	0,017580813	0,033589587
MORTALITY_REASON_DESC_Certain infectious and parasitic diseases	6,82792E-05	0,536280263	0,294044485	0,008299084	0,022175252
MORTALITY_REASON_DESC_Congenital malformations, deformations and chromosomal abnormalities	6,82792E-05	-0,153223273	-0,384464709	0,0272265	0,051053801
MORTALITY_REASON_DESC_Diseases of the blood and blood-forming organs and certain disorders involving the immune mechanism	6,82792E-05	-0,948376179	-1,184645403	0,016678739	0,024194975
MORTALITY_REASON_DESC_Diseases of the circulatory system	6,82792E-05	2,183908036	1,950576159	0,048922956	0,032074573
MORTALITY_REASON_DESC_Diseases of the digestive system	6,82792E-05	1,108506534	0,869396991	0,019992227	0,039621916
MORTALITY_REASON_DESC_Diseases of the ear and mastoid process	6,82792E-05	-3,434749836	-3,778833733	0,023575321	0,01553808
MORTALITY_REASON_DESC_Diseases of the eye and adnexa	6,82792E-05	-3,896017264	-4,395453806	0,014555704	0,013540978
MORTALITY_REASON_DESC_Diseases of the genitourinary system	6,82792E-05	-0,153370746	-0,383770147	0,008317891	0,01953172
MORTALITY_REASON_DESC_Diseases of the musculoskeletal system and connective tissue	6,82792E-05	-1,073190133	-1,310335566	0,013917193	0,017723455

MORTALITY_REASON_DESC_Diseases of the nervous system	6,82792E-05	0,884057483	0,64437624	0,007977659	0,009431727
MORTALITY_REASON_DESC_Diseases of the respiratory system	6,82792E-05	1,078411353	0,839377216	0,014765344	0,0203353
MORTALITY_REASON_DESC_Diseases of the skin and subcutaneous tissue	6,82792E-05	-1,972071636	-2,221744536	0,020441372	0,01795489
MORTALITY_REASON_DESC_Endocrine, nutritional and metabolic diseases	6,82792E-05	0,481400048	0,238921502	0,006753849	0,017742131
MORTALITY_REASON_DESC_External causes of morbidity and mortality	6,82792E-05	2,424373462	2,191492145	0,041249784	0,023851887
MORTALITY_REASON_DESC_Mental and behavioural disorders	6,82792E-05	-0,141219053	-0,370677385	0,006849966	0,016258551
MORTALITY_REASON_DESC_Neoplasms	6,82792E-05	2,452798665	2,220428473	0,051240908	0,02079173
MORTALITY_REASON_DESC_Pregnancy, childbirth and the puerperium	6,82792E-05	-0,628147795	-0,840458404	0,004501171	0,01006651
MORTALITY_REASON_DESC_Symptoms, signs and abnormal clinical and laboratory findings, not elsewhere classified	6,82792E-05	1,017535868	0,778470824	0,012045735	0,02003604
age_group_0	6,82792E-05	0,775939889	1,097906553	0,010588903	0,042645219
age_group_1-4	6,82792E-05	-0,941607777	-0,615571035	0,006886676	0,023578292
age_group_10-14	6,82792E-05	-1,746480858	-1,427978761	0,008687424	0,015254441
age_group_15-19	6,82792E-05	-1,318749716	-0,996933145	0,00307345	0,01623971
age_group_20-24	6,82792E-05	-1,13503159	-0,812138088	0,001988714	0,012231851
age_group_25-29	6,82792E-05	-0,919027458	-0,595347711	0,002252445	0,010582329
age_group_30-34	6,82792E-05	-0,638403157	-0,312758039	0,003870456	0,006915978
age_group_35-39	6,82792E-05	-0,321257452	0	0,00676171	0,00320957
age_group_40-44	6,82792E-05	0,04829985	0,363942855	0,012022914	0,005810354
age_group_45-49	6,82792E-05	0,420648963	0,739446365	0,018486624	0,012869808
age_group_5-9	6,82792E-05	-0,486395919	-0,152731888	0,00191707	0,008129131
age_group_50-54	6,82792E-05	0,774242212	1,094686003	0,023760906	0,014162733
age_group_55-59	6,82792E-05	1,124100267	1,447327017	0,029237949	0,01324378
age_group_60-64	6,82792E-05	1,456349859	1,781152478	0,032965564	0,01762386
age_group_>65	6,82792E-05	2,907372888	3,243880751	0,139219146	0,046846611

APPENDIX B

R^2 scores, train scores, test scores for all mortality reason and regression algorithm

REGRESSION_TYPE	MORTALITY_REASON_DESC	R^2_SCORE	TRAINING_SCORE	TEST_SCORE
Linear	Certain infectious and parasitic diseases	0,8280303	0,843194166	0,828030345
Ridge	Certain infectious and parasitic diseases	0,7981051	0,811471457	0,798105132
Lasso	Certain infectious and parasitic diseases	0,8283151	0,843022632	0,828315097
Random Forest	Certain infectious and parasitic diseases	0,9499635	0,992513883	0,949963459
GradientBoosting	Certain infectious and parasitic diseases	0,9517472	0,976650762	0,95174719
Linear	Neoplasms	0,9688569	0,968184276	0,968856895
Ridge	Neoplasms	0,9664999	0,965633556	0,966499854
Lasso	Neoplasms	0,9687643	0,96808224	0,968764296
Random Forest	Neoplasms	0,9837821	0,997558585	0,983782123
GradientBoosting	Neoplasms	0,9847379	0,993296024	0,984737902
Linear	Diseases of the blood and blood-forming organs and certain disorders involving the immune mechanism	0,7773128	0,784403056	0,777312774
Ridge	Diseases of the blood and blood-forming organs and certain disorders involving the immune mechanism	0,7546299	0,750132293	0,754629884
Lasso	Diseases of the blood and blood-forming organs and certain disorders involving the immune mechanism	0,77797	0,784078643	0,777970009
Random Forest	Diseases of the blood and blood-forming organs and certain disorders involving the immune mechanism	0,9058829	0,986387866	0,905882911
GradientBoosting	Diseases of the blood and blood-forming organs and certain disorders involving the immune mechanism	0,9093412	0,963471465	0,909341202
Linear	Endocrine, nutritional and metabolic diseases	0,8849334	0,885093955	0,884933431
Ridge	Endocrine, nutritional and metabolic diseases	0,8770885	0,876433396	0,877088481
Lasso	Endocrine, nutritional and metabolic diseases	0,8848745	0,884862248	0,884874472
Random Forest	Endocrine, nutritional and metabolic diseases	0,9545871	0,992968416	0,954587126
GradientBoosting	Endocrine, nutritional and metabolic diseases	0,9571896	0,978283046	0,957189602
Linear	Mental and behavioural disorders	0,7193651	0,698817733	0,719365125
Ridge	Mental and behavioural disorders	0,7113934	0,690278401	0,711393406
Lasso	Mental and behavioural disorders	0,7195685	0,698658406	0,719568518
Random Forest	Mental and behavioural disorders	0,9235514	0,987576359	0,923551426
GradientBoosting	Mental and behavioural disorders	0,9361681	0,971959014	0,936168132

Linear	Diseases of the nervous system	0,886907	0,889282448	0,886907011
Ridge	Diseases of the nervous system	0,8802096	0,882722027	0,880209648
Lasso	Diseases of the nervous system	0,8868286	0,889068326	0,886828552
Random Forest	Diseases of the nervous system	0,9429698	0,992448147	0,942969822
GradientBoosting	Diseases of the nervous system	0,9468169	0,977239933	0,946816874
Linear	Diseases of the eye and adnexa	0,3206499	0,420233945	0,320649946
Ridge	Diseases of the eye and adnexa	0,3296177	0,379262892	0,329617659
Lasso	Diseases of the eye and adnexa	0,3213662	0,419903282	0,321366247
Random Forest	Diseases of the eye and adnexa	0,7004769	0,949696627	0,700476917
GradientBoosting	Diseases of the eye and adnexa	0,6416216	0,998174579	0,641621553
Linear	Diseases of the ear and mastoid process	0,4636142	0,419327007	0,463614209
Ridge	Diseases of the ear and mastoid process	0,3920646	0,375690975	0,392064605
Lasso	Diseases of the ear and mastoid process	0,4554675	0,42007805	0,455467478
Random Forest	Diseases of the ear and mastoid process	0,6232897	0,940369852	0,623289679
GradientBoosting	Diseases of the ear and mastoid process	0,5539465	0,980928757	0,553946512
Linear	Diseases of the circulatory system	0,9527592	0,94848384	0,952759206
Ridge	Diseases of the circulatory system	0,9501338	0,945032628	0,950133808
Lasso	Diseases of the circulatory system	0,9527758	0,948387915	0,952775806
Random Forest	Diseases of the circulatory system	0,9824093	0,997152914	0,982409278
GradientBoosting	Diseases of the circulatory system	0,9838813	0,992177859	0,983881263
Linear	Diseases of the respiratory system	0,90312	0,90387575	0,903119952
Ridge	Diseases of the respiratory system	0,8936522	0,893768505	0,893652227
Lasso	Diseases of the respiratory system	0,9030537	0,903748688	0,903053703
Random Forest	Diseases of the respiratory system	0,9657651	0,994964331	0,965765114
GradientBoosting	Diseases of the respiratory system	0,9659782	0,985559444	0,965978192
Linear	Diseases of the digestive system	0,8973789	0,90210932	0,897378902
Ridge	Diseases of the digestive system	0,8929922	0,89534587	0,892992172
Lasso	Diseases of the digestive system	0,8976093	0,902023322	0,897609275
Random Forest	Diseases of the digestive system	0,9670209	0,995266469	0,967020861
GradientBoosting	Diseases of the digestive system	0,970656	0,987779791	0,970655958
Linear	Diseases of the skin and subcutaneous tissue	0,6973226	0,691653844	0,697322592
Ridge	Diseases of the skin and subcutaneous tissue	0,676019	0,662346404	0,676018959
Lasso	Diseases of the skin and subcutaneous tissue	0,6976215	0,69141263	0,697621477
Random Forest	Diseases of the skin and subcutaneous tissue	0,8999216	0,984732728	0,899921617
GradientBoosting	Diseases of the skin and subcutaneous tissue	0,9102055	0,96820477	0,910205472
Linear	Diseases of the musculoskeletal system and connective tissue	0,7795527	0,770867135	0,779552735

Ridge	Diseases of the musculoskeletal system and connective tissue	0,7723968	0,761040314	0,77239685
Lasso	Diseases of the musculoskeletal system and connective tissue	0,7794137	0,77059483	0,779413701
Random Forest	Diseases of the musculoskeletal system and connective tissue	0,9029087	0,986845547	0,902908663
GradientBoosting	Diseases of the musculoskeletal system and connective tissue	0,9133375	0,966463325	0,913337503
Linear	Diseases of the genitourinary system	0,8963782	0,901773011	0,896378193
Ridge	Diseases of the genitourinary system	0,8853051	0,889382797	0,88530515
Lasso	Diseases of the genitourinary system	0,8963338	0,901725821	0,896333816
Random Forest	Diseases of the genitourinary system	0,9537246	0,993562365	0,953724635
GradientBoosting	Diseases of the genitourinary system	0,9573948	0,983207149	0,957394813
Linear	Pregnancy, childbirth and the puerperium	0,7370152	0,738017303	0,737015224
Ridge	Pregnancy, childbirth and the puerperium	0,6550321	0,640412768	0,655032089
Lasso	Pregnancy, childbirth and the puerperium	0,7363921	0,737452667	0,736392097
Random Forest	Pregnancy, childbirth and the puerperium	0,83185	0,97578398	0,83184997
GradientBoosting	Pregnancy, childbirth and the puerperium	0,8319622	0,982791055	0,831962201
Linear	Certain conditions originating in the perinatal period	0,9057104	0,906015976	0,905710429
Ridge	Certain conditions originating in the perinatal period	0,8681221	0,87022283	0,868122072
Lasso	Certain conditions originating in the perinatal period	0,9055205	0,90587122	0,905520542
Random Forest	Certain conditions originating in the perinatal period	0,9669528	0,995466406	0,966952757
GradientBoosting	Certain conditions originating in the perinatal period	0,9661512	0,996541264	0,966151164
Linear	Congenital malformations, deformations and chromosomal abnormalities	0,7551049	0,772833423	0,755104851
Ridge	Congenital malformations, deformations and chromosomal abnormalities	0,7523712	0,768384728	0,752371185
Lasso	Congenital malformations, deformations and chromosomal abnormalities	0,7551533	0,772815432	0,755153314
Random Forest	Congenital malformations, deformations and chromosomal abnormalities	0,8861582	0,983964329	0,886158153
GradientBoosting	Congenital malformations, deformations and chromosomal abnormalities	0,8892581	0,947812208	0,88925807
Linear	Symptoms, signs and abnormal clinical and laboratory findings, not elsewhere classified	0,77432	0,783470861	0,77431999
Ridge	Symptoms, signs and abnormal clinical and laboratory findings, not elsewhere classified	0,7632534	0,772475732	0,763253421

Lasso	Symptoms, signs and abnormal clinical and laboratory findings, not elsewhere classified	0,7739225	0,783287015	0,773922521
Random Forest	Symptoms, signs and abnormal clinical and laboratory findings, not elsewhere classified	0,9481945	0,992171705	0,948194479
GradientBoosting	Symptoms, signs and abnormal clinical and laboratory findings, not elsewhere classified	0,9556583	0,979819623	0,955658348
Linear	External causes of morbidity and mortality	0,9055822	0,906159394	0,905582153
Ridge	External causes of morbidity and mortality	0,902892	0,90304814	0,902891978
Lasso	External causes of morbidity and mortality	0,9056535	0,906019861	0,905653502
Random Forest	External causes of morbidity and mortality	0,9703702	0,99568106	0,970370163
GradientBoosting	External causes of morbidity and mortality	0,972052	0,988128651	0,972052006

APPENDIX C

DBMS CODES

```
CREATE TABLE #TMP1
(
Country varchar(100), Admin1  varchar(100), SubDiv varchar(100), Year
varchar(100),List   varchar(100),Cause
varchar(100),Sex              varchar(100),Frmat       varchar(100),
IM_Frmat       varchar(100),
Deaths1
varchar(100),Deaths2       varchar(100),Deaths3       varchar(100),Deaths4       varchar(10
0),
Deaths5       varchar(100),Deaths6       varchar(100),Deaths7       varchar(100),Deaths8
  varchar(100),Deaths9       varchar(100),Deaths10 varchar(100),Deaths11
varchar(100),Deaths12 varchar(100), Deaths13 varchar(100),Deaths14
varchar(100),Deaths15 varchar(100),Deaths16 varchar(100), Deaths17
varchar(100),Deaths18 varchar(100),Deaths19 varchar(100),Deaths20 varchar(100),
Deaths21 varchar(100),Deaths22 varchar(100),Deaths23 varchar(100),Deaths24
varchar(100), Deaths25 varchar(100),Deaths26 varchar(100),
IM_Deaths1
varchar(100),IM_Deaths2       varchar(100),IM_Deaths3       varchar(100),IM_Deaths4
varchar(100)
);

truncate table #TMP1;
load Table #TMP1(
Country ',',Admin1  ',',SubDiv ',',Year   ',',List   ',',Cause
',',Sex              ',',Frmat       ',',IM_Frmat ',', Deaths1
',',Deaths2       ',',Deaths3       ',',Deaths4       ',',Deaths5       ',',Deaths6       ',',Deaths7
 ',', Deaths8       ',',Deaths9       ',',Deaths10 ',',Deaths11 ',',Deaths12 ',',Deaths13
',',Deaths14 ',', Deaths15 ',',Deaths16 ',',Deaths17 ',',Deaths18 ',',Deaths19 ',',Deaths20
',',Deaths21 ',',Deaths22 ',', Deaths23 ',',Deaths24 ',',Deaths25 ',',Deaths26 ',',
IM_Deaths1 ',',IM_Deaths2       ',',IM_Deaths3       ',',IM_Deaths4 '\n')
FROM '/iqbackup2/Morticd10_part1'
QUOTES OFF
ESCAPES OFF;
COMMIT;
delete from #TMP1 where country='Country';
COMMIT;
load Table #TMP1(
Country ',',Admin1  ',',SubDiv ',',Year   ',',List   ',',Cause
',',Sex              ',',Frmat       ',',IM_Frmat ',', Deaths1
',',Deaths2       ',',Deaths3       ',',Deaths4       ',',Deaths5       ',',Deaths6       ',',Deaths7
 ',', Deaths8       ',',Deaths9       ',',Deaths10 ',',Deaths11 ',',Deaths12 ',',Deaths13
',',Deaths14 ',', Deaths15 ',',Deaths16 ',',Deaths17 ',',Deaths18 ',',Deaths19 ',',Deaths20
',',Deaths21 ',',Deaths22 ',', Deaths23 ',',Deaths24 ',',Deaths25 ',',Deaths26 ',',
IM_Deaths1 ',',IM_Deaths2       ',',IM_Deaths3       ',',IM_Deaths4 '\n')
```

```
FROM '/iqbackup2/Morticd10_part2'
QUOTES OFF
ESCAPES OFF;
COMMIT;
delete from #TMP1 where country='Country';
COMMIT;

---------------------------------
CREATE TABLE #MAPPING
(
MAIN_Cause  varchar(100),DETAIL_Cause  varchar(100)
);
load Table #MAPPING(
MAIN_Cause ';',
DETAIL_Cause  '\n')
FROM '/iqbackup2/mortalityMapping.csv'
QUOTES OFF
ESCAPES OFF;
COMMIT;
----------------
CREATE TABLE #MORTALITY_REASON
(
MORTALITY_CODE  varchar(100),MORTALITY_DESC  varchar(100)
);
truncate table #MORTALITY_REASON;
load Table #MORTALITY_REASON(
MORTALITY_CODE ';',
MORTALITY_DESC  '\n')
FROM '/iqbackup2/mortalityReason.csv'
QUOTES OFF
ESCAPES OFF;
COMMIT;
--------------------------------------------------
 drop table #AGE;
CREATE TABLE #AGE
(
COLUMN_NAME  varchar(100),COLUMN_DESC
varchar(100),POPULATION_COLUMN_NAME varchar(100)
);
truncate table #AGE;
load Table #AGE(
COLUMN_NAME ';',
COLUMN_DESC  '\n')
FROM '/iqbackup2/mortalityAge.csv'
QUOTES OFF
ESCAPES OFF;
COMMIT;
update #AGE set column_desc =str_replace(column_desc, char(13), null);
```

```
commit;
update #AGE set
POPULATION_COLUMN_NAME=str_replace(COLUMN_NAME,'Deaths' , 'Pop')
where COLUMN_NAME not like 'IM_%';
------------
CREATE TABLE #AGE_FRMAT_MAPPING
(
Frmat       varchar(100),Frmat00_DESC varchar(100),Frmat_DESC varchar(100),RATIO
varchar(100)
);
truncate table #AGE_FRMAT_MAPPING;
load Table #AGE_FRMAT_MAPPING(
Frmat ';',Frmat00_DESC ';',Frmat_DESC ';',RATIO   '\n')
FROM '/iqbackup2/mortalityAgeFrmtMap.csv'
QUOTES OFF
ESCAPES OFF;
COMMIT;
update #AGE_FRMAT_MAPPING set Frmat00_DESC =str_replace(Frmat00_DESC,
char(13), null);
update #AGE_FRMAT_MAPPING set Frmat_DESC =str_replace(Frmat_DESC, char(13),
null);
update #AGE_FRMAT_MAPPING set Frmat =str_replace(Frmat, char(13), null);
Commit;
-----------------------------------------
CREATE TABLE #COUNTRY
(
COUNTRY_CODE varchar(100),
COUNTRY_NAME  varchar(100)
);
truncate table #COUNTRY;
load Table #COUNTRY(
COUNTRY_CODE ';',
COUNTRY_NAME '\n')
FROM '/iqbackup2/mortalityCountry.csv'
QUOTES OFF
ESCAPES OFF;
COMMIT;
update #country set country_name =str_replace(country_name, char(13), null);
commit;
-----------------------------

create table #sex (sex numeric(1));
insert into #sex values (1);
insert into #sex values (2);
```

#MORTALITY_PARENT_X_CHILD

```
CREATE TABLE #MORTALITY_PARENT_X_CHILD
(
MORTALITY_PARENT_CODE  varchar(100),
MORTALITY_CHILD_CODE  varchar(100)
);
insert into #MORTALITY_PARENT_X_CHILD values ('1001','1002');
insert into #MORTALITY_PARENT_X_CHILD values ('1001','1003');
insert into #MORTALITY_PARENT_X_CHILD values ('1001','1004');
insert into #MORTALITY_PARENT_X_CHILD values ('1001','1005');
insert into #MORTALITY_PARENT_X_CHILD values ('1001','1006');
insert into #MORTALITY_PARENT_X_CHILD values ('1001','1007');
insert into #MORTALITY_PARENT_X_CHILD values ('1001','1008');
insert into #MORTALITY_PARENT_X_CHILD values ('1001','1009');
insert into #MORTALITY_PARENT_X_CHILD values ('1001','1010');
insert into #MORTALITY_PARENT_X_CHILD values ('1001','1011');
insert into #MORTALITY_PARENT_X_CHILD values ('1001','1012');
insert into #MORTALITY_PARENT_X_CHILD values ('1001','1013');
insert into #MORTALITY_PARENT_X_CHILD values ('1001','1014');
insert into #MORTALITY_PARENT_X_CHILD values ('1001','1015');
insert into #MORTALITY_PARENT_X_CHILD values ('1001','1016');
insert into #MORTALITY_PARENT_X_CHILD values ('1001','1017');
insert into #MORTALITY_PARENT_X_CHILD values ('1001','1018');
insert into #MORTALITY_PARENT_X_CHILD values ('1001','1019');
insert into #MORTALITY_PARENT_X_CHILD values ('1001','1020');
insert into #MORTALITY_PARENT_X_CHILD values ('1001','1021');
insert into #MORTALITY_PARENT_X_CHILD values ('1001','1022');
insert into #MORTALITY_PARENT_X_CHILD values ('1001','1023');
insert into #MORTALITY_PARENT_X_CHILD values ('1001','1024');
insert into #MORTALITY_PARENT_X_CHILD values ('1001','1025');
insert into #MORTALITY_PARENT_X_CHILD values ('1026','1027');
insert into #MORTALITY_PARENT_X_CHILD values ('1026','1028');
insert into #MORTALITY_PARENT_X_CHILD values ('1026','1029');
insert into #MORTALITY_PARENT_X_CHILD values ('1026','1030');
insert into #MORTALITY_PARENT_X_CHILD values ('1026','1031');
insert into #MORTALITY_PARENT_X_CHILD values ('1026','1032');
insert into #MORTALITY_PARENT_X_CHILD values ('1026','1033');
insert into #MORTALITY_PARENT_X_CHILD values ('1026','1034');
insert into #MORTALITY_PARENT_X_CHILD values ('1026','1035');
insert into #MORTALITY_PARENT_X_CHILD values ('1026','1036');
insert into #MORTALITY_PARENT_X_CHILD values ('1026','1037');
insert into #MORTALITY_PARENT_X_CHILD values ('1026','1038');
insert into #MORTALITY_PARENT_X_CHILD values ('1026','1039');
insert into #MORTALITY_PARENT_X_CHILD values ('1026','1040');
insert into #MORTALITY_PARENT_X_CHILD values ('1026','1041');
insert into #MORTALITY_PARENT_X_CHILD values ('1026','1042');
insert into #MORTALITY_PARENT_X_CHILD values ('1026','1043');
insert into #MORTALITY_PARENT_X_CHILD values ('1026','1044');
insert into #MORTALITY_PARENT_X_CHILD values ('1026','1045');
```

```
insert into #MORTALITY_PARENT_X_CHILD values ('1026','1046');
insert into #MORTALITY_PARENT_X_CHILD values ('1026','1047');
insert into #MORTALITY_PARENT_X_CHILD values ('1048','1049');
insert into #MORTALITY_PARENT_X_CHILD values ('1048','1050');
insert into #MORTALITY_PARENT_X_CHILD values ('1051','1052');
insert into #MORTALITY_PARENT_X_CHILD values ('1051','1053');
insert into #MORTALITY_PARENT_X_CHILD values ('1051','1054');
insert into #MORTALITY_PARENT_X_CHILD values ('1055','1056');
insert into #MORTALITY_PARENT_X_CHILD values ('1055','1057');
insert into #MORTALITY_PARENT_X_CHILD values ('1058','1059');
insert into #MORTALITY_PARENT_X_CHILD values ('1058','1060');
insert into #MORTALITY_PARENT_X_CHILD values ('1058','1061');
insert into #MORTALITY_PARENT_X_CHILD values ('1062','1062');
insert into #MORTALITY_PARENT_X_CHILD values ('1063','1063');
insert into #MORTALITY_PARENT_X_CHILD values ('1064','1065');
insert into #MORTALITY_PARENT_X_CHILD values ('1064','1066');
insert into #MORTALITY_PARENT_X_CHILD values ('1064','1067');
insert into #MORTALITY_PARENT_X_CHILD values ('1064','1068');
insert into #MORTALITY_PARENT_X_CHILD values ('1064','1069');
insert into #MORTALITY_PARENT_X_CHILD values ('1064','1070');
insert into #MORTALITY_PARENT_X_CHILD values ('1064','1071');
insert into #MORTALITY_PARENT_X_CHILD values ('1072','1073');
insert into #MORTALITY_PARENT_X_CHILD values ('1072','1074');
insert into #MORTALITY_PARENT_X_CHILD values ('1072','1075');
insert into #MORTALITY_PARENT_X_CHILD values ('1072','1076');
insert into #MORTALITY_PARENT_X_CHILD values ('1072','1077');
insert into #MORTALITY_PARENT_X_CHILD values ('1078','1079');
insert into #MORTALITY_PARENT_X_CHILD values ('1078','1080');
insert into #MORTALITY_PARENT_X_CHILD values ('1078','1081');
insert into #MORTALITY_PARENT_X_CHILD values ('1082','1082');
insert into #MORTALITY_PARENT_X_CHILD values ('1083','1083');
insert into #MORTALITY_PARENT_X_CHILD values ('1084','1085');
insert into #MORTALITY_PARENT_X_CHILD values ('1084','1086');
insert into #MORTALITY_PARENT_X_CHILD values ('1087','1088');
insert into #MORTALITY_PARENT_X_CHILD values ('1087','1089');
insert into #MORTALITY_PARENT_X_CHILD values ('1087','1090');
insert into #MORTALITY_PARENT_X_CHILD values ('1087','1091');
insert into #MORTALITY_PARENT_X_CHILD values ('1092','1092');
insert into #MORTALITY_PARENT_X_CHILD values ('1093','1093');
insert into #MORTALITY_PARENT_X_CHILD values ('1094','1094');
insert into #MORTALITY_PARENT_X_CHILD values ('1095','1096');
insert into #MORTALITY_PARENT_X_CHILD values ('1095','1097');
insert into #MORTALITY_PARENT_X_CHILD values ('1095','1098');
insert into #MORTALITY_PARENT_X_CHILD values ('1095','1099');
insert into #MORTALITY_PARENT_X_CHILD values ('1095','1100');
insert into #MORTALITY_PARENT_X_CHILD values ('1095','1101');
insert into #MORTALITY_PARENT_X_CHILD values ('1095','1102');
insert into #MORTALITY_PARENT_X_CHILD values ('1095','1103');
```

```
--------------
 drop table #tmp2;
select
Country,convert(numeric(4),Year) as n_year,
isnull(m.MORTALITY_CODE,Cause) as MORTALITY_REASON_CODE,
isnull(m.MORTALITY_DESC,'') as MORTALITY_REASON_DESC,
convert(numeric(1),Sex) as n_sex,
Frmat, IM_Frmat,
sum(convert(numeric(25),Deaths1)) as Deaths1 ,
sum(convert(numeric(25),Deaths2)) as Deaths2,
sum(convert(numeric(25),Deaths3)) as Deaths3,
sum(convert(numeric(25),Deaths4)) as Deaths4,
sum(convert(numeric(25),Deaths5)) as Deaths5,
sum(convert(numeric(25),Deaths6)) as Deaths6,
sum(convert(numeric(25),Deaths7)) as Deaths7,
sum(convert(numeric(25),Deaths8)) as Deaths8,
sum(convert(numeric(25),Deaths9)) as Deaths9,
sum(convert(numeric(25),Deaths10)) as Deaths10,
sum(convert(numeric(25),Deaths11)) as Deaths11,
sum(convert(numeric(25),Deaths12)) as Deaths12,
sum(convert(numeric(25),Deaths13)) as Deaths13,
sum(convert(numeric(25),Deaths14)) as Deaths14,
sum(convert(numeric(25),Deaths15)) as Deaths15,
sum(convert(numeric(25),Deaths16)) as Deaths16,
sum(convert(numeric(25),Deaths17)) as Deaths17,
sum(convert(numeric(25),Deaths18)) as Deaths18,
sum(convert(numeric(25),Deaths19)) as Deaths19,
sum(convert(numeric(25),Deaths20)) as Deaths20,
sum(convert(numeric(25),Deaths21)) as Deaths21,
sum(convert(numeric(25),Deaths22)) as Deaths22,
sum(convert(numeric(25),Deaths23)) as Deaths23,
sum(convert(numeric(25),Deaths24)) as Deaths24,
sum(convert(numeric(25),Deaths25)) as Deaths25,
sum(convert(numeric(25),Deaths26)) as Deaths26,
sum(convert(numeric(25),IM_Deaths1)) as IM_Deaths1,
sum(convert(numeric(25),IM_Deaths2)) as IM_Deaths2,
sum(convert(numeric(25),IM_Deaths3)) as IM_Deaths3,
sum(convert(numeric(25),IM_Deaths4)) as IM_Deaths4
into #tmp2
from  #TMP1 as t left outer join (select
r.MORTALITY_CODE,r.MORTALITY_DESC,x.DETAIL_Cause from #MAPPING as
x,#MORTALITY_REASON as r where x.MAIN_Cause=r.MORTALITY_CODE) as m
on (substr(t.cause,1,3)=substr(DETAIL_Cause,1,3))
group by
Country,n_year,MORTALITY_REASON_CODE,MORTALITY_REASON_DESC,n_sex
,Frmat,IM_Frmat;
```

```
drop table #tmp3;
select
COUNTRY_CODE,COUNTRY_NAME,n_year,MORTALITY_REASON_CODE,MORT
ALITY_REASON_DESC,n_sex,a.COLUMN_DESC as age_group,
sum(case when a.COLUMN_NAME= 'Deaths1' then t.Deaths1 *
convert(numeric(7,4),m.RATIO)
        when a.COLUMN_NAME= 'Deaths2' then t.Deaths2 *
        convert(numeric(7,4),m.RATIO)
        when a.COLUMN_NAME= 'Deaths3' then t.Deaths3 *
        convert(numeric(7,4),m.RATIO)
        when a.COLUMN_NAME= 'Deaths4' then t.Deaths4 *
        convert(numeric(7,4),m.RATIO)
        when a.COLUMN_NAME= 'Deaths5' then t.Deaths5 *
        convert(numeric(7,4),m.RATIO)
        when a.COLUMN_NAME= 'Deaths6' then t.Deaths6 *
        convert(numeric(7,4),m.RATIO)
        when a.COLUMN_NAME= 'Deaths7' then t.Deaths7 *
        convert(numeric(7,4),m.RATIO)
        when a.COLUMN_NAME= 'Deaths8' then t.Deaths8 *
        convert(numeric(7,4),m.RATIO)
        when a.COLUMN_NAME= 'Deaths9' then t.Deaths9 *
        convert(numeric(7,4),m.RATIO)
        when a.COLUMN_NAME= 'Deaths10' then t.Deaths10 *
        convert(numeric(7,4),m.RATIO)
        when a.COLUMN_NAME= 'Deaths11' then t.Deaths11 *
        convert(numeric(7,4),m.RATIO)
        when a.COLUMN_NAME= 'Deaths12' then t.Deaths12 *
        convert(numeric(7,4),m.RATIO)
        when a.COLUMN_NAME= 'Deaths13' then t.Deaths13 *
        convert(numeric(7,4),m.RATIO)
        when a.COLUMN_NAME= 'Deaths14' then t.Deaths14 *
        convert(numeric(7,4),m.RATIO)
        when a.COLUMN_NAME= 'Deaths15' then t.Deaths15 *
        convert(numeric(7,4),m.RATIO)
        when a.COLUMN_NAME= 'Deaths16' then t.Deaths16 *
        convert(numeric(7,4),m.RATIO)
        when a.COLUMN_NAME= 'Deaths17' then t.Deaths17 *
        convert(numeric(7,4),m.RATIO)
        when a.COLUMN_NAME= 'Deaths18' then t.Deaths18 *
        convert(numeric(7,4),m.RATIO)
        when a.COLUMN_NAME= 'Deaths19' then t.Deaths19 *
        convert(numeric(7,4),m.RATIO)
        when a.COLUMN_NAME= 'Deaths20' then t.Deaths20 *
        convert(numeric(7,4),m.RATIO)
        when a.COLUMN_NAME= 'Deaths21' then t.Deaths21 *
        convert(numeric(7,4),m.RATIO)
        when a.COLUMN_NAME= 'Deaths22' then t.Deaths22 *
        convert(numeric(7,4),m.RATIO)
```

```
                when a.COLUMN_NAME= 'Deaths23' then t.Deaths23 *
                convert(numeric(7,4),m.RATIO)
                when a.COLUMN_NAME= 'Deaths24' then t.Deaths24 *
                convert(numeric(7,4),m.RATIO)
                when a.COLUMN_NAME= 'Deaths25' then t.Deaths25 *
                convert(numeric(7,4),m.RATIO)
                when a.COLUMN_NAME= 'Deaths26' then t.Deaths26 *
                convert(numeric(7,4),m.RATIO)
--              when a.COLUMN_NAME= 'IM_deaths1' then t.IM_Deaths1 *
convert(numeric(7,4),m.RATIO)
--              when a.COLUMN_NAME= 'IM_deaths2' then t.IM_Deaths2 *
convert(numeric(7,4),m.RATIO)
--              when a.COLUMN_NAME= 'IM_deaths3' then t.IM_Deaths3 *
convert(numeric(7,4),m.RATIO)
--              when a.COLUMN_NAME= 'IM_deaths4' then t.IM_Deaths4 *
convert(numeric(7,4),m.RATIO)
else 0 end) as number_of_death into #tmp3
from #tmp2 as t,#AGE_FRMAT_MAPPING m,#age as a,#COUNTRY as c
where t.country=c.country_code
and m.Frmat_DESC= a.COLUMN_DESC
and t.Frmat='0' || m.Frmat
group by
COUNTRY_CODE,COUNTRY_NAME,n_year,MORTALITY_REASON_CODE,MORT
ALITY_REASON_DESC,n_sex,a.COLUMN_DESC;


----------------

drop table #t2_1;
drop table #t2;
select distinct
n_year,country_code,country_name,mortality_reason_code,MORTALITY_REASON_DE
SC into #t2_1 from #tmp3 where n_year>=1998;
select distinct COLUMN_DESC, sex, n_year, country_code, country_name,
mortality_reason_code, MORTALITY_REASON_DESC into #t2 from #age as a,#sex as
s,#t2_1 as t;
insert into #tmp3
select t1.COUNTRY_CODE, t1.COUNTRY_NAME, t1.n_year, t1.mortality_reason_code,
t1.MORTALITY_REASON_DESC,t1.sex, t1.COLUMN_DESC, 0 as number_of_death
from #t2 as t1 left outer join #tmp3 as t2
                on (t1.country_code=t2.country_code and t1.n_year=t2.n_year and
                t1.mortality_reason_code=t2.mortality_reason_code and
                t1.COLUMN_DESC=t2.age_group and t1.sex=t2.n_sex)
where t2.country_code is null;

population

CREATE TABLE #POPULATION
```

```
(
Country varchar(100),Admin1  varchar(100),SubDiv varchar(100),Year   varchar(100),
Sex              varchar(100),Frmat       varchar(100),Pop1
varchar(100),Pop2       varchar(100),
Pop3       varchar(100),Pop4       varchar(100),Pop5       varchar(100),Pop6       varchar(1
00), Pop7       varchar(100),Pop8       varchar(100),Pop9       varchar(100),Pop10
varchar(100),
Pop11 varchar(100),Pop12 varchar(100),Pop13 varchar(100),Pop14 varchar(100),Pop15
varchar(100), Pop16 varchar(100),Pop17 varchar(100),Pop18 varchar(100),Pop19
varchar(100),Pop20 varchar(100), Pop21 varchar(100),Pop22 varchar(100),Pop23
varchar(100),Pop24 varchar(100),Pop25 varchar(100), Pop26 varchar(100),Lb
varchar(100));

truncate table #POPULATION;
load Table #POPULATION(
Country ',',Admin1  ',',SubDiv ',',Year   ',',Sex              ',',Frmat       ',',Pop1
',',Pop2       ',',Pop3       ',',
Pop4       ',',Pop5       ',',Pop6       ',',Pop7       ',',Pop8       ',',Pop9       ',',Pop10 ',',Pop11
',',Pop12 ',', Pop13 ',',Pop14 ',',Pop15 ',',Pop16 ',',Pop17 ',',Pop18 ',',Pop19 ',',Pop20
',',Pop21 ',',Pop22 ',',Pop23 ',', Pop24 ',',Pop25 ',',Pop26 ',',Lb '\n')
FROM '/iqbackup2/mortalityPop'
QUOTES OFF
ESCAPES OFF;
COMMIT;
delete from #POPULATION where Country='Country';
commit;

CREATE TABLE #AGE_FRMAT_MAPPING
(
Frmat       varchar(100),
Frmat00_DESC varchar(100),
Frmat_DESC varchar(100),
RATIO varchar(100)
);
truncate table #AGE_FRMAT_MAPPING;
load Table #AGE_FRMAT_MAPPING(
Frmat ';',
Frmat00_DESC ';',
Frmat_DESC ';',
RATIO   '\n')
FROM '/iqbackup2/mortalityAgeFrmtMap.csv'
QUOTES OFF
ESCAPES OFF;
COMMIT;
update #AGE_FRMAT_MAPPING set Frmat00_DESC =str_replace(Frmat00_DESC,
char(13), null);
update #AGE_FRMAT_MAPPING set Frmat_DESC =str_replace(Frmat_DESC, char(13),
null);
```

```
update #AGE_FRMAT_MAPPING set Frmat =str_replace(Frmat, char(13), null);

 drop table #pop;
select
country,convert(numeric(4),Year) as n_year,convert(numeric(1),Sex) as n_sex,
a.COLUMN_DESC as age_group,
ceil(sum(
            case when a.POPULATION_COLUMN_NAME= 'Pop1' then
            convert(numeric(25),t.Pop1) * convert(numeric(7,4),m.RATIO)
                  when a.POPULATION_COLUMN_NAME= 'Pop2' then
                  convert(numeric(25),t.Pop2) * convert(numeric(7,4),m.RATIO)
                  when a.POPULATION_COLUMN_NAME= 'Pop3' then
                  convert(numeric(25),t.Pop3) * convert(numeric(7,4),m.RATIO)
                  when a.POPULATION_COLUMN_NAME= 'Pop4' then
                  convert(numeric(25),t.Pop4) * convert(numeric(7,4),m.RATIO)
                  when a.POPULATION_COLUMN_NAME= 'Pop5' then
                  convert(numeric(25),t.Pop5) * convert(numeric(7,4),m.RATIO)
                  when a.POPULATION_COLUMN_NAME= 'Pop7' then
                  convert(numeric(25),t.Pop6) * convert(numeric(7,4),m.RATIO)
                  when a.POPULATION_COLUMN_NAME= 'Pop8' then
                  convert(numeric(25),t.Pop7) * convert(numeric(7,4),m.RATIO)
                  when a.POPULATION_COLUMN_NAME= 'Pop9' then
                  convert(numeric(25),t.Pop8) * convert(numeric(7,4),m.RATIO)
                  when a.POPULATION_COLUMN_NAME= 'Pop10' then
                  convert(numeric(25),t.Pop10) * convert(numeric(7,4),m.RATIO)
                  when a.POPULATION_COLUMN_NAME= 'Pop11' then
                  convert(numeric(25),t.Pop11) * convert(numeric(7,4),m.RATIO)
                  when a.POPULATION_COLUMN_NAME= 'Pop12' then
                  convert(numeric(25),t.Pop12) * convert(numeric(7,4),m.RATIO)
                  when a.POPULATION_COLUMN_NAME= 'Pop13' then
                  convert(numeric(25),t.Pop13) * convert(numeric(7,4),m.RATIO)
                  when a.POPULATION_COLUMN_NAME= 'Pop14' then
                  convert(numeric(25),t.Pop14) * convert(numeric(7,4),m.RATIO)
                  when a.POPULATION_COLUMN_NAME= 'Pop15' then
                  convert(numeric(25),t.Pop15) * convert(numeric(7,4),m.RATIO)
                  when a.POPULATION_COLUMN_NAME= 'Pop16' then
                  convert(numeric(25),t.Pop16) * convert(numeric(7,4),m.RATIO)
                  when a.POPULATION_COLUMN_NAME= 'Pop17' then
                  convert(numeric(25),t.Pop17) * convert(numeric(7,4),m.RATIO)
                  when a.POPULATION_COLUMN_NAME= 'Pop18' then
                  convert(numeric(25),t.Pop18) * convert(numeric(7,4),m.RATIO)
                  when a.POPULATION_COLUMN_NAME= 'Pop19' then
                  convert(numeric(25),t.Pop19) * convert(numeric(7,4),m.RATIO)
                  when a.POPULATION_COLUMN_NAME= 'Pop20' then
                  convert(numeric(25),t.Pop20) * convert(numeric(7,4),m.RATIO)
                  when a.POPULATION_COLUMN_NAME= 'Pop21' then
                  convert(numeric(25),t.Pop21) * convert(numeric(7,4),m.RATIO)
```

```
                when a.POPULATION_COLUMN_NAME= 'Pop22' then
                convert(numeric(25),t.Pop22) * convert(numeric(7,4),m.RATIO)
                when a.POPULATION_COLUMN_NAME= 'Pop23' then
                convert(numeric(25),t.Pop23) * convert(numeric(7,4),m.RATIO)
                when a.POPULATION_COLUMN_NAME= 'Pop24' then
                convert(numeric(25),t.Pop24) * convert(numeric(7,4),m.RATIO)
                when a.POPULATION_COLUMN_NAME= 'Pop25' then
                convert(numeric(25),t.Pop25) * convert(numeric(7,4),m.RATIO)
                when a.POPULATION_COLUMN_NAME= 'Pop26' then
                convert(numeric(25),t.Pop26) * convert(numeric(7,4),m.RATIO)
            else 0 end)
    )
 as number_of_population into #pop
from #POPULATION as t,#AGE_FRMAT_MAPPING m,#age a
where m.Frmat_DESC=a.COLUMN_DESC
and t.Frmat='0' || m.Frmat
group by country,n_year,n_sex,a.COLUMN_DESC;

drop table #t_year;
create table #t_year
(
year numeric(4)
);
insert into #t_year values (2000);
insert into #t_year values (2001);
insert into #t_year values (2002);
insert into #t_year values (2003);
insert into #t_year values (2004);
insert into #t_year values (2005);
insert into #t_year values (2006);
insert into #t_year values (2007);
insert into #t_year values (2008);
insert into #t_year values (2009);
insert into #t_year values (2010);
insert into #t_year values (2011);
insert into #t_year values (2012);
insert into #t_year values (2013);
insert into #t_year values (2014);
insert into #t_year values (2015);
insert into #t_year values (2016);

select * from (
select country,sex,y.year,Pop1,
lead(Pop1,1) over (partition by country,sex order by y.year asc) as next_Pop1
from #t_year y left outer join #POPULATION p on (convert(numeric(4),p.year)=y.year)
) as h where next_Pop1 is null
order by country,sex,year
```

```
select *,(next_Pop1-n_Pop1)/(next_year-n_year) as growth_rate from (
select country,sex,convert(numeric(4),p.year) as n_year,convert(numeric(25),Pop1) as
n_Pop1,
lead(n_Pop1,1) over (partition by country,sex order by n_year asc) as next_Pop1,
lead(n_year,1) over (partition by country,sex order by n_year asc) as next_year
from #POPULATION p where n_year between 2000 and 2015 and n_Pop1 is not null
) as h where next_year is not null and next_Pop1 is not null
order by country,sex,n_year
```

smoking

```
CREATE TABLE #SMOKING
(
COUNTRY  varchar(100),
COUNTRY_CODE varchar(100),
YEAR   varchar(100),
SEX varchar(100),
PERCENTAGE   varchar(100)
);
truncate table #SMOKING;
load Table #SMOKING(
COUNTRY ',',
COUNTRY_CODE ',',
YEAR   ',',
PERCENTAGE '\n')
FROM '/iqbackup2/share-of-women-who-are-smoking.csv'
QUOTES OFF
ESCAPES OFF;
COMMIT;
------------------
update #SMOKING set sex='2';
commit;
load Table #SMOKING(
COUNTRY ',',
COUNTRY_CODE ',',
YEAR   ',',
PERCENTAGE '\n')
FROM '/iqbackup2/share-of-men-who-are-smoking.csv'
QUOTES OFF
ESCAPES OFF;
COMMIT;
update #SMOKING set sex='1' where sex is null;
Commit;

select COUNTRY, COUNTRY_CODE, convert(numeric(4),YEAR) as n_year,
convert(numeric(2),SEX) as n_sex, convert(numeric(25,4),PERCENTAGE) as
n_percentage into #smokingNUm from #smoking;
```

```
commit;


drop table #smokingYEars;
create table #smokingYEars
(
YEAR numeric(4),
NEXT_YEAR numeric(4),
DIFF_YEAR numeric(2),
COEF numeric(2),
NEW_YEAR numeric(4)
);
truncate table #smokingYEars;
insert into #smokingYEars values(2000,2005,5,1,2001);
insert into #smokingYEars values(2000,2005,5,2,2002);
insert into #smokingYEars values(2000,2005,5,3,2003);
insert into #smokingYEars values(2000,2005,5,4,2004);
insert into #smokingYEars values(2005,2010,5,1,2006);
insert into #smokingYEars values(2005,2010,5,2,2007);
insert into #smokingYEars values(2005,2010,5,3,2008);
insert into #smokingYEars values(2005,2010,5,4,2009);
insert into #smokingYEars values(2010,2012,2,1,2011);
insert into #smokingYEars values(2012,2015,3,1,2013);
insert into #smokingYEars values(2012,2015,3,2,2014);
commit;
insert into #smokingNUm
select s1.country,s1.country_code,new_year,s1.n_sex,s2.n_percentage +  (y.coef *
(s2.n_percentage-s1.n_percentage) / DIFF_YEAR )
from #smokingNUm s1,#smokingNUm s2,#smokingYEars y
where s1.COUNTRY=s2.country and s1.n_sex=s2.n_sex
and s1.n_year=y.year and y.next_year=s2.n_year;
commit;
```

Air pollution

```
CREATE TABLE #AIR_POLLUTION
(
COUNTRY  varchar(100),
COUNTRY_CODE varchar(100),
YEAR   varchar(100),
AIR_POLLUTION   varchar(100)
);
truncate table #AIR_POLLUTION;
load Table #AIR_POLLUTION(
COUNTRY ',',
COUNTRY_CODE ',',
YEAR   ',',
AIR_POLLUTION  '\n')
```

```
FROM '/iqbackup2/PM25-air-pollution.csv'
QUOTES OFF
ESCAPES OFF;
COMMIT;
-----------------
update #AIR_POLLUTION set AIR_POLLUTION=str_replace(AIR_POLLUTION,
char(13), null);
drop table #AIR_POLLUTION_NUM;
select COUNTRY, COUNTRY_CODE, convert(numeric(4),YEAR) as n_year,
convert(numeric(25,15),AIR_POLLUTION) as n_AIR_POLLUTION into
#AIR_POLLUTION_NUM from #AIR_POLLUTION;
commit;

--geometric mean
--drop table #geoMean;
--drop table #geoMeanY;
select country,country_code,EXP(sum(ln(n_AIR_POLLUTION))/count(*)) as gMean into
#geoMean from  #AIR_POLLUTION_NUM where n_year>=2000 group by
country,country_code;
select distinct yil as n_year into #years from dw2.takvim where yil between 2000 and
2015;
select country,country_code,gMean,n_year into #geoMeanY from #geoMean,#years;

select g.COUNTRY, g.COUNTRY_CODE, g.n_year,
isnull(n_AIR_POLLUTION,gMean) as n_AIR_POLLUTION
into #AIR_POLLUTION_FULL
from #geoMeanY as g left outer join #AIR_POLLUTION_NUM as n
 on (n.country=g.country  AND n.country_code=g.country_code and n.n_year=g.n_year)

IMPROVED_DRINKINGWATER

CREATE TABLE #IMPROVED_DRINKINGWATER
(
COUNTRY  varchar(100),
COUNTRY_CODE varchar(100),
YEAR   varchar(100),
IMPROVED_DRINKINGWATER   varchar(100)
);
truncate table #IMPROVED_DRINKINGWATER;
load Table #IMPROVED_DRINKINGWATER(
COUNTRY ',',
COUNTRY_CODE ',',
YEAR   ',',
IMPROVED_DRINKINGWATER  '\n')
FROM '/iqbackup2/share-of-the-population-with-access-to-improved-drinking-water.csv'
QUOTES OFF
ESCAPES OFF;
COMMIT;
```

```
------------------
update #IMPROVED_DRINKINGWATER set
IMPROVED_DRINKINGWATER=str_replace(IMPROVED_DRINKINGWATER,
char(13), null);
drop table #IMPROVED_DRINKINGWATER_NUM;
select COUNTRY, COUNTRY_CODE, convert(numeric(4),YEAR) as n_year,
convert(numeric(25,15),IMPROVED_DRINKINGWATER) as
n_IMPROVED_DRINKINGWATER into #IMPROVED_DRINKINGWATER_NUM
from #IMPROVED_DRINKINGWATER;
commit;
-------------------------------------------------------------------------------------------------------------
------------------------  BMI
CREATE TABLE #BMI
(
COUNTRY  varchar(100),
COUNTRY_CODE varchar(100),
YEAR   varchar(100),
BMI   varchar(100)
);
truncate table #BMI;
load Table #BMI(
COUNTRY ',',
COUNTRY_CODE ',',
YEAR   ',',
BMI  '\n')
FROM '/iqbackup2/share-of-adults-defined-as-obese.csv'
QUOTES OFF
ESCAPES OFF;
COMMIT;
------------------
update #BMI set BMI=str_replace(BMI, char(13), null);
drop table #BMI_NUM;
select COUNTRY, COUNTRY_CODE, convert(numeric(4),YEAR) as n_year,
convert(numeric(25,15),BMI) as n_BMI into #BMI_NUM from #BMI;
commit;
 -------------------------------------------------------------------------------------------------------------
------------------------
HEALTHCARE_EXPENDITURE
CREATE TABLE #HEALTHCARE_EXPENDITURE
(
COUNTRY  varchar(100),
COUNTRY_CODE varchar(100),
YEAR   varchar(100),
HEALTHCARE_EXPENDITURE   varchar(100)
);
truncate table #HEALTHCARE_EXPENDITURE;
load Table #HEALTHCARE_EXPENDITURE(
COUNTRY ',',
```

```
COUNTRY_CODE ',',
YEAR   ',',
HEALTHCARE_EXPENDITURE  '\n')
FROM '/iqbackup2/total-healthcare-expenditure-as-share-of-national-gdp-by-country.csv'
QUOTES OFF
ESCAPES OFF;
COMMIT;
-----------------
update #HEALTHCARE_EXPENDITURE set
HEALTHCARE_EXPENDITURE=str_replace(HEALTHCARE_EXPENDITURE,
char(13), null);
drop table #HEALTHCARE_EXPENDITURE_NUM;
select COUNTRY, COUNTRY_CODE, convert(numeric(4),YEAR) as n_year,
convert(numeric(25,15),HEALTHCARE_EXPENDITURE) as
n_HEALTHCARE_EXPENDITURE into #HEALTHCARE_EXPENDITURE_NUM
from #HEALTHCARE_EXPENDITURE;
commit;
-----------------------------------------------------------------------------------------------------------
-----------------------
Human development index
CREATE TABLE #HDI
(
COUNTRY  varchar(100),
COUNTRY_CODE varchar(100),
YEAR   varchar(100),
HDI   varchar(100)
);
truncate table #HDI;
load Table #HDI(
COUNTRY ',',
COUNTRY_CODE ',',
YEAR   ',',
HDI  '\n')
FROM '/iqbackup2/human-development-index.csv'
QUOTES OFF
ESCAPES OFF;
COMMIT;
-----------------
update #HDI set HDI=str_replace(HDI, char(13), null);
drop table #HDI_NUM;
select COUNTRY, COUNTRY_CODE, convert(numeric(4),YEAR) as n_year,
convert(numeric(25,15),HDI) as n_HDI into #HDI_NUM from #HDI;
commit;

select country,country_code,EXP(sum(ln(n_HDI))/count(*)) as gMean into #geoMean
from  #HDI_NUM where n_year>=2000 group by country,country_code;
select distinct yil as n_year into #years from dw2.takvim where yil between 2000 and
2015;
```

```
select country,country_code,gMean,n_year into #geoMeanY from #geoMean,#years;
select g.COUNTRY, g.COUNTRY_CODE, g.n_year, isnull(n_HDI,gMean) as n_HDI
into #HDI_FULL
from #geoMeanY as g left outer join #HDI_NUM as n
 on (n.country=g.country  AND n.country_code=g.country_code and n.n_year=g.n_year);
 ----------------------------------------------------------------------------------------------------------
----------------------
GDP
CREATE TABLE #GDP
(
COUNTRY  varchar(100),
COUNTRY_CODE varchar(100),
YEAR   varchar(100),
GDP   varchar(100)
);
truncate table #GDP;
load Table #GDP(
COUNTRY ',',
COUNTRY_CODE ',',
YEAR   ',',
GDP  '\n')
FROM '/iqbackup2/gdp-per-capita-worldbank.csv'
QUOTES OFF
ESCAPES OFF;
COMMIT;
-----------------
update #GDP set GDP=str_replace(GDP, char(13), null);
drop table #GDP_NUM;
select COUNTRY, COUNTRY_CODE, convert(numeric(4),YEAR) as n_year,
convert(numeric(25,15),GDP) as n_GDP into #GDP_NUM from #GDP;
commit;
----------------------------------------------------------------------------------------------------------
----------------------
join
drop table  #mortality_pop;
select m.*,p.number_of_population into #mortality_pop
from #tmp3 as m inner join #pop as p
        on (p.n_year>2000 and m.COUNTRY_CODE=p.COUNTRY and
        m.n_year=p.n_year and m.n_sex=p.n_sex and m.age_group=p.age_group)
where
m.n_year>=2000 and isnull(m.MORTALITY_REASON_DESC,")<>";
----------------
drop table MORTALITY_REASON_FRAMEWORK;
select p.COUNTRY_CODE, p.COUNTRY_NAME, p.n_year,
p.MORTALITY_REASON_CODE, p.MORTALITY_REASON_DESC, p.n_sex,
p.age_group, p.number_of_death, p.number_of_population,
```

```
          n_percentage as n_smoking_percentage,
        convert(numeric(25,15),n_AIR_POLLUTION) AS n_AIR_POLLUTION,
        n_IMPROVED_DRINKINGWATER, n_BMI,
          --n_HUNGER_INDEX,
          n_HEALTHCARE_EXPENDITURE,  convert(numeric(25,15),n_HDI) AS n_HDI,
          --n_SCHOOLING,
          n_GDP into MORTALITY_REASON_FRAMEWORK
from #mortality_pop as p,
          #smokingNUm as s, #AIR_POLLUTION_FULL as
        air,#IMPROVED_DRINKINGWATER_NUM as water,
          #BMI_NUM as bmi,--#HUNGER_INDEX_FULL as hinx
          #HEALTHCARE_EXPENDITURE_NUM as HEALTHCARE,
          #HDI_FULL as hdi,
         --  #SCHOOLING_NUM as school,
         #GDP_NUM as gdp
where p.COUNTRY_NAME=s.country
and p.n_year=s.n_year
and p.n_sex=s.n_sex
and p.n_year=air.n_year
and p.COUNTRY_NAME=air.COUNTRY
and p.n_year=water.n_year
and p.COUNTRY_NAME=water.COUNTRY
and p.n_year=bmi.n_year
and p.COUNTRY_NAME=bmi.COUNTRY
--and p.n_year=hinx.n_year
--and p.COUNTRY_NAME=hinx.COUNTRY
and p.n_year=HEALTHCARE.n_year
and p.COUNTRY_NAME=HEALTHCARE.COUNTRY
and p.n_year=hdi.n_year
and p.COUNTRY_NAME=hdi.COUNTRY
--and p.n_year=school.n_year
--and p.COUNTRY_NAME=school.COUNTRY
and p.n_year=gdp.n_year
and p.COUNTRY_NAME=gdp.COUNTRY;
update MORTALITY_REASON_FRAMEWORK set MORTALITY_REASON_DESC
=str_replace(MORTALITY_REASON_DESC, char(13), null);
Commit;
 ------------------------------------------------------------------------------------------------------------
------------------------
Unload MORTALITY_REASON_FRAMEWORK_SUMMARY
select COUNTRY_CODE, COUNTRY_NAME, n_year,
          MORTALITY_PARENT_CODE as MORTALITY_REASON_CODE,
        r.MORTALITY_DESC as MORTALITY_REASON_DESC,
          n_sex, age_group, sum(number_of_death) as number_of_death ,
          number_of_population, n_smoking_percentage, n_AIR_POLLUTION,
        n_IMPROVED_DRINKINGWATER, n_BMI,
        n_HEALTHCARE_EXPENDITURE, n_HDI, n_GDP
          into MORTALITY_REASON_FRAMEWORK_SUMMARY
```

```
from MORTALITY_REASON_FRAMEWORK as
f,#MORTALITY_PARENT_X_CHILD as p,#MORTALITY_REASON as r
where f.MORTALITY_REASON_CODE= p.MORTALITY_CHILD_CODE
and  r.MORTALITY_CODE= p.MORTALITY_PARENT_CODE
group by COUNTRY_CODE, COUNTRY_NAME, n_year,
          MORTALITY_PARENT_CODE, r.MORTALITY_DESC,
          n_sex, age_group,number_of_population, n_smoking_percentage,
        n_AIR_POLLUTION, n_IMPROVED_DRINKINGWATER, n_BMI,
        n_HEALTHCARE_EXPENDITURE, n_HDI, n_GDP;
update MORTALITY_REASON_FRAMEWORK_SUMMARY set
MORTALITY_REASON_DESC =str_replace(MORTALITY_REASON_DESC, char(13),
null);
Commit;

drop table #t1;
  create table #t1(
      COUNTRY_CODE varchar(1000) null,
     COUNTRY_NAME varchar(1000) null,
      n_year varchar(1000) null,
      MORTALITY_REASON_CODE varchar(1000) null,
      MORTALITY_REASON_DESC varchar(1000) null,
      n_sex varchar(1000) null,
      age_group varchar(1000) null,
      number_of_death varchar(1000) null,
      number_of_population varchar(1000) null,
      n_smoking_percentage varchar(1000) null,
      n_AIR_POLLUTION varchar(1000) null,
      n_IMPROVED_DRINKINGWATER varchar(1000) null,
      n_BMI varchar(1000) null,
      n_HEALTHCARE_EXPENDITURE varchar(1000) null,
      n_HDI varchar(1000) null,
      n_GDP varchar(1000) null,
      );
    insert into #t1 values(
    'COUNTRY_CODE', 'COUNTRY_NAME', 'n_year',
'MORTALITY_REASON_CODE', 'MORTALITY_REASON_DESC', 'n_sex',
'age_group', 'number_of_death',
    'number_of_population', 'n_smoking_percentage', 'n_AIR_POLLUTION',
'n_IMPROVED_DRINKINGWATER', 'n_BMI', 'n_HEALTHCARE_EXPENDITURE',
'n_HDI', 'n_GDP'
     ) ;
    insert into #t1
      select  COUNTRY_CODE,
COUNTRY_NAME,convert(varchar(1000),n_year),MORTALITY_REASON_CODE,
MORTALITY_REASON_DESC,
      convert(varchar(1000),n_sex), age_group, convert(varchar(1000),number_of_death),
convert(varchar(1000),number_of_population),
```

```
      convert(varchar(1000),n_smoking_percentage),
convert(varchar(1000),n_AIR_POLLUTION),
convert(varchar(1000),n_IMPROVED_DRINKINGWATER),
convert(varchar(1000),n_BMI),
      convert(varchar(1000),n_HEALTHCARE_EXPENDITURE),
convert(varchar(1000),n_HDI), convert(varchar(1000),n_GDP)
        from MORTALITY_REASON_FRAMEWORK_SUMMARY;
    set temporary option Temp_Extract_Null_As_Empty = 'ON';
    set temporary option Temp_Extract_Column_Delimiter = ';';
    set temporary option Timestamp_format = 'DD.MM.YYYY hh:nn:ss';
    set temporary option Date_format = 'YYYY-MM-DD';
    set temporary option Temp_Extract_Name1 =
'/iqbackup2/MORTALITY_REASON_FRAMEWORK_SUMMARY.csv';
    select * from #t1;
    set temporary option Temp_Extract_Name1 = '';
    commit work;
```

APPENDIX D

PYTHON CODES

1. loadData.py

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn import linear_model
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
from sklearn.ensemble import GradientBoostingRegressor
import statsmodels.api as sm
####################FUNCTIONS#################
#apply preprocessing.LabelEncoder to given categorical values for a given
dataframe and add new labels.And store the encodings in LblsCode.
def EncodingTheLabels(datafrm,LblName,LblsCode):
    for l in LblName:
        LblsCode[l]=preprocessing.LabelEncoder()
        LblsCode[l].fit(datafrm[l])
        datafrm.loc[:,l+ '_ENCODED'] =
pd.Series(LblsCode[l].transform(datafrm[l]), index=datafrm.index)

#print encoding classes that were produced by EncodingTheLabels funtion.
def PrintLabelEncodingsClass(LblsCode,LblName):
    for l in LblName:
        print(l + ':')
        print(LblsCode[l].classes_)
#convert categorical features to dummy features accordig to their values

def Categorical_to_Dummies(datafrm1, datafrm2, LblName):
    for l in LblName:
        dummies = pd.get_dummies(datafrm1[l])
        LblNameD = dummies.columns.tolist()
        for l2 in LblNameD:
            LblNameD[LblNameD.index(l2)] = l + '_' + l2
            dummies.columns = LblNameD
            # print(dummies.columns)
        datafrm2 = pd.concat([datafrm2, dummies], axis=1)
    return datafrm2

#apply preprocessing.LabelEncoder to given categorical values for a given
dataframe.And store the encodings in LblsCode.
# The difference of this funtion from EncodingTheLabels is add encoding
columns to given dataframe EncodingTheLabels_out add encoding columns to
a new dataframe.
def EncodingTheLabels_out(datafrm, LblName, LblsCode, datafrmOUT):
    for l in LblName:
        LblsCode[l] = preprocessing.LabelEncoder()
        LblsCode[l].fit(datafrm[l])
        datafrmOUT.loc[:, l + '_ENCODED'] =
pd.Series(LblsCode[l].transform(datafrm[l]), index=datafrm.index)
```

```
    return datafrmOUT

def Add_logTransormedColumns(datafrm,LblName):
    for l in LblName:
        datafrm.loc[:,l + '_LOG']=list(np.log(datafrm[l]))

def Add_ReverseLogTransformedColumns(datafrm):
        for l in datafrm.columns:
            if l.find('_LOG') != -1:
                datafrm.loc[:, l.replace("_LOG", "")] =
list(np.exp(datafrm[l]))

def Drop_LogColumns(datafrm):
        for l in datafrm.columns:
            if l.find('_LOG') != -1:
                datafrm.drop([l],axis='columns', inplace=True)
##################################

filename = 'MORTALITY_REASON_FRAMEWORK_SUMMARY.csv'
mortality_df = pd.read_csv(filename,sep=';',encoding='Latin1')
"""""""""""""""""""""""Initial analysis of data"""""""""""""""""""""""""
##list(mortality_df) ##column names
mortality_df.drop(['Unnamed: 16'],axis='columns', inplace=True)
"""

mortality_df.shape ## look the observation and variable count (1432250
observation,16 variable)
mortality_df.dtypes ## look at the data types of variables

#descriptive statistics for numeric data
mortality_df.describe()
#descriptive statistics for categorical data
mortality_df[mortality_df.dtypes[mortality_df.dtypes ==
"object"].index].describe()

"""

#convert sex columns to object type
mortality_df.n_sex=mortality_df.n_sex.astype(str)
index= np.where(mortality_df["n_sex"] == "1")
mortality_df["n_sex"].loc[index] = "MALE"
index= np.where(mortality_df["n_sex"] == "2")
mortality_df["n_sex"].loc[index] = "FEMALE"
######
##########################################
#drop NA values
#mortality_df.isnull().sum()
mortality_df=mortality_df.dropna(subset = ['number_of_population'])
#mortality_df.shape ## look the observation and variable count (1343959
observation,16 variable)
##########################################3
#print(mortality_df.age_group.unique())
mortality_df=mortality_df[mortality_df['age_group']!='ALL']
mortality_df=mortality_df[mortality_df['age_group']!='UNKNOWN']
mortality_df=mortality_df[mortality_df['number_of_death']!=0]
mortality_df=mortality_df[mortality_df['number_of_population']!=0]
mortality_df_org=mortality_df

####log transform######
```

```
labels=['number_of_death','number_of_population','n_smoking_percentage',
'n_AIR_POLLUTION', 'n_IMPROVED_DRINKINGWATER', 'n_BMI',
'n_HEALTHCARE_EXPENDITURE', 'n_HDI', 'n_GDP']
Add_logTransormedColumns(mortality_df,labels)

#mortality_df.shape #(186953, 25)
##################33prediction#############
####Label.encoding

RMortalityY=mortality_df['number_of_death_LOG']
##RMortalityY = np.log(RMortalityY)
RMortalityX=mortality_df[['number_of_population_LOG',
'n_smoking_percentage_LOG', 'n_AIR_POLLUTION_LOG',
'n_IMPROVED_DRINKINGWATER_LOG', 'n_BMI_LOG',
'n_HEALTHCARE_EXPENDITURE_LOG', 'n_HDI_LOG', 'n_GDP_LOG']]
labels = ['n_sex','MORTALITY_REASON_DESC','age_group']
#########create dummy columns#############3
RMortalityX=Categorical_to_Dummies(mortality_df,RMortalityX,labels)

np.random.seed(42)
X_train, X_test, y_train, y_test = train_test_split(RMortalityX,
RMortalityY, test_size=0.3, random_state=10)
```

2. dataAnalysis.py

```
#functions defination from loadData must run.
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn import preprocessing
import seaborn as sns


filename = 'MORTALITY_REASON_FRAMEWORK_SUMMARY.csv'
mortality_df = pd.read_csv(filename,sep=';',encoding='Latin1')
mortality_df.drop(['Unnamed: 16'],axis='columns', inplace=True)
mortality_df.shape #(343026, 16)
#convert sex columns to object type
mortality_df.n_sex=mortality_df.n_sex.astype(str)
index= np.where(mortality_df["n_sex"] == "1")
mortality_df["n_sex"].loc[index] = "MALE"
index= np.where(mortality_df["n_sex"] == "2")
mortality_df["n_sex"].loc[index] = "FEMALE"
#convert country code and mortality reason code to str
mortality_df.COUNTRY_CODE=mortality_df.COUNTRY_CODE.astype(str)
mortality_df.MORTALITY_REASON_CODE=mortality_df.MORTALITY_REASON_CODE.ast
ype(str)
##########################################
#drop NA values
mortality_df.isnull().sum()
mortality_df=mortality_df.dropna(subset = ['number_of_population'])

mortality_df.groupby(['age_group']).size()
mortality_df=mortality_df[mortality_df['age_group']!='ALL']
mortality_df=mortality_df[mortality_df['age_group']!='UNKNOWN']

len(mortality_df[mortality_df['number_of_death']==0]) #95764
len(mortality_df[mortality_df['number_of_population']==0]) #636
```

```
mortality_df=mortality_df[mortality_df['number_of_death']!=0]
mortality_df=mortality_df[mortality_df['number_of_population']!=0]

mortality_df.shape #(186953, 16)

mortality_numeric_df=mortality_df.select_dtypes([np.number])
mortality_numeric_df.columns
mortality_numeric_df.drop(['n_year'],axis='columns', inplace=True)

###correlation#########
mortality_numeric_df.corr().unstack().drop_duplicates()
plt.matshow(mortality_numeric_df.corr())
plt.show()
corr = mortality_numeric_df.corr()
sns.heatmap(corr,
            xticklabels=corr.columns.values,
            yticklabels=corr.columns.values)

f, ax = plt.subplots(figsize=(5,5))
corr = mortality_numeric_df.corr()
sns.heatmap(corr, mask=np.zeros_like(corr, dtype=np.bool),
cmap=sns.diverging_palette(220, 10, as_cmap=True),
            square=True, ax=ax,annot=True)

f, ax = plt.subplots(figsize=(5,5))
sns.pairplot(mortality_numeric_df,kind="reg")

mortality_numeric_df.skew()

def kdeSubPlots():
    fig, axes = plt.subplots(nrows=3, ncols=3)
    fig.subplots_adjust(hspace=0.5)
    mortality_numeric_df['number_of_death'].plot(ax=axes[0,0],kind='kde')
    axes[0,0].set_title('number_of_death')

mortality_numeric_df['number_of_population'].plot(ax=axes[0,1],kind='kde'
)
    axes[0,1].set_title('number_of_population')
    mortality_numeric_df['n_smoking_percentage'].plot(ax=axes[0,2])
    axes[0,2].set_title('smoking_percentage')
    mortality_numeric_df['n_AIR_POLLUTION'].plot(ax=axes[1,0])
    axes[1,0].set_title('AIR_POLLUTION')
    mortality_numeric_df['n_IMPROVED_DRINKINGWATER'].plot(ax=axes[1,1])
    axes[1,1].set_title('IMPROVED_DRINKINGWATER')
    mortality_numeric_df['n_BMI'].plot(ax=axes[1,2])
    axes[1,2].set_title('BMI')

mortality_numeric_df['n_HEALTHCARE_EXPENDITURE'].plot(ax=axes[2,0],kind='
kde')
    axes[2,0].set_title('HEALTHCARE_EXPENDITURE')
    mortality_numeric_df['n_HDI'].plot(ax=axes[2,1],kind='kde')
    axes[2,1].set_title('HDI')
    mortality_numeric_df['n_GDP'].plot(ax=axes[2,2])
    axes[2,2].set_title('GDP')

kdeSubPlots()
labels=['number_of_death','number_of_population','n_smoking_percentage',
'n_AIR_POLLUTION', 'n_IMPROVED_DRINKINGWATER', 'n_BMI',
'n_HEALTHCARE_EXPENDITURE', 'n_HDI', 'n_GDP']
```

```
logTransformed(mortality_numeric_df,labels)
kdeSubPlots()
```

3. gridSearchAndModels.py

```
#####################before run this code run loadData
file*******************************

from sklearn.model_selection import GridSearchCV
from yellowbrick.regressor import PredictionError
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestRegressor
from sklearn import linear_model
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.linear_model import Ridge


###############linear regression#########################
LR = linear_model.LinearRegression()
params = {'fit_intercept':[True,False], 'normalize':[True,False],
'copy_X':[True, False]}
model = linear_model.LinearRegression()
gridS = GridSearchCV(estimator=model,param_grid=params,cv=10,verbose = 1)
##, scoring = 'neg_mean_squared_error'
gridS.fit(RMortalityX, RMortalityY)
#gridS.cv_results_
gridS.best_params_  #{'copy_X': True, 'fit_intercept': True, 'normalize':
False}
gridS.best_score_  # 0.72097726951902619


LR.fit(X_train, y_train) #fit the model
RMortality_y_pred = LR.predict(X_test) ##predited values test data
print("Mean squared error: %.2f" % mean_squared_error(y_test,
RMortality_y_pred))
print('Variance score: %.2f' % r2_score(y_test, RMortality_y_pred))
print("Training set score: {:.2f}".format(LR.score(X_train, y_train)))
print("Test set score: {:.2f}".format(LR.score(X_test, y_test)))


visualizer = PredictionError(LR)
visualizer.fit(X_train, y_train)  # Fit the training data to the
visualizer
visualizer.score(X_test, y_test)  # Evaluate the model on the test data
g = visualizer.poof()

#####################RIDGE REGRESSION###############################
params={'alpha': [100,50,10,1.0,0.1,0.01]}
model = Ridge()
gridS = GridSearchCV(estimator=model , param_grid=params,cv=10,verbose =
1)
gridS.fit(RMortalityX, RMortalityY)
gridS.best_params_  #{'alpha': 50}
gridS.best_score_  # 0.72097726951902619

params={'alpha': [70,60,50,40,30]}
gridS = GridSearchCV(estimator=model , param grid=params,cv=10,verbose =
1) ##, scoring = 'neg_mean_squared_error'
```

```
gridS.fit(RMortalityX, RMortalityY)
#gridS.cv_results_
gridS.best_params_  #{'alpha': 50}
gridS.best_score_  # 0.72097726951902619

#we reach the best score when alpha=50

ridgeR50 = Ridge(alpha=50).fit(X_train, y_train)
RMortality_y_pred = ridgeR50.predict(X_test)
print('Variance score: %.2f' % r2_score(y_test, RMortality_y_pred))
print("Training set score: {:.2f}".format(ridgeR50.score(X_train,
y_train)))
print("Test set score: {:.2f}".format(ridgeR50.score(X_test, y_test)))

ridgeR50.intercept_   ##-5.0465171695666413
ridgeR50.coef_

visualizer = PredictionError(Ridge(alpha=50))
visualizer.fit(X_train, y_train)  # Fit the training data to the
visualizer
visualizer.score(X_test, y_test)  # Evaluate the model on the test data
g = visualizer.poof()

###################LASSO REGRESSION############################
model = Lasso()
params={'alpha': [0.1,0.01,0.001]}
gridS = GridSearchCV(estimator=model , param_grid=params,cv=10,verbose =
1) ##, scoring = 'neg_mean_squared_error'
gridS.fit(RMortalityX, RMortalityY)
#gridS.cv_results_
gridS.best_params_  #{'alpha': 0.001}
gridS.best_score_  # 0.71986322741314912

params={'alpha': [0.009,0.001,0.0005]}
gridS = GridSearchCV(estimator=model , param_grid=params,cv=10,verbose =
1)
gridS.fit(RMortalityX, RMortalityY)
#gridS.cv_results_
gridS.best_params_  #{'alpha': 0.0005}
gridS.best_score_  # 0.72044488915780713

model = Lasso(max_iter=10000)
params={'alpha': [0.0005,0.0001]}
gridS = GridSearchCV(estimator=model , param_grid=params,cv=10,verbose =
1)
gridS.fit(RMortalityX, RMortalityY)
#gridS.cv_results_
gridS.best_params_  # {'alpha': 0.0001}
gridS.best_score_  # 0.72058546890534869

#the best score gatterhed when alpha=0.0001 and max_iter=10000.
# However the run time increase a lot.On the other hand the score not
increase much according to when alpha=0.0005 and max iter=1000
#There for the model with alpha=0.0005 and and max_iter=1000 choosen.

Rlasso = Lasso(alpha=0.0005).fit(X_train, y_train)
RMortality_y_pred = Rlasso.predict(X_test)
print('Variance score: %.2f' % r2_score(y_test, RMortality_y_pred))
print("Training set score: {:.2f}".format(Rlasso.score(X_train,
```

```
y_train)))
print("Test set score: {:.2f}".format(Rlasso.score(X_test, y_test)))
print("Number of features used: {}".format(np.sum(Rlasso.coef_ != 0)))

Rlasso.intercept_   ##-7.1612479287354187
Rlasso.coef_

visualizer = PredictionError(Lasso(alpha=0.0005))
visualizer.fit(X_train, y_train)  # Fit the training data to the
visualizer
visualizer.score(X_test, y_test)  # Evaluate the model on the test data
g = visualizer.poof()

######################RANDOM FOREST############################
model = RandomForestRegressor()
params={'n_estimators': [1,10,50]}
gridS = GridSearchCV(estimator=model , param_grid=params,cv=10,verbose =
1)
gridS.fit(RMortalityX, RMortalityY)
#gridS.cv_results_
gridS.best_params_  #{'n_estimators': 50}
gridS.best_score_  # 0.85971162217654284

params={'n_estimators': [50,70]}
gridS = GridSearchCV(estimator=model , param_grid=params,cv=10,verbose =
1)
gridS.fit(RMortalityX, RMortalityY)
#gridS.cv_results_
gridS.best_params_  #{'n_estimators': 50}
gridS.best_score_  # 0.85971162217654284

Rforest= RandomForestRegressor(n_estimators = 50)
Rforest.fit(X_train, y_train)
RMortality_y_pred = Rforest.predict(X_test)
print('Variance score: %.2f' % r2_score(y_test, RMortality_y_pred))
print("Training set score: {:.2f}".format(Rforest.score(X_train,
y_train)))
print("Test set score: {:.2f}".format(Rforest.score(X_test, y_test)))

visualizer = PredictionError(RandomForestRegressor(n_estimators = 50))
visualizer.fit(X_train, y_train)  # Fit the training data to the
visualizer
visualizer.score(X_test, y_test)  # Evaluate the model on the test data
g = visualizer.poof()

###################GradientBoostingRegressor#################
model=GradientBoostingRegressor()
params= {'learning_rate': [0.1],
              'max_depth': [4,6,10],
               'n_estimators':[20,50,100,200]
              }
gridS = GridSearchCV(estimator=model, param_grid=params,cv=10,verbose =
1)
gridS.fit(RMortalityX, RMortalityY)
#gridS.cv_results_
gridS.best_params_  # {'learning_rate': 0.1, 'max_depth': 6,
'n_estimators': 200}
gridS.best_score   # 0.873604058849186
```

```
RGradientB =
GradientBoostingRegressor(n_estimators=200,max_depth=6,learning_rate=0.1)
RGradientB.fit(X_train, y_train)
RMortality_y_pred = RGradientB.predict(X_test)
print('Variance score: %.2f' % r2_score(y_test, RMortality_y_pred))
print("Training set score: {:.2f}".format(RGradientB.score(X_train,
y_train)))
print("Test set score: {:.2f}".format(RGradientB.score(X_test, y_test)))

visualizer =
PredictionError(GradientBoostingRegressor(n_estimators=200,max_depth=6,le
arning_rate=0.1))
visualizer.fit(X_train, y_train)  # Fit the training data to the
visualizer
visualizer.score(X_test, y_test)  # Evaluate the model on the test data
g = visualizer.poof()
```

4. ModelresultAnalysis.py

```
#################before run this code run loadData
file*******************************

X= sm.add_constant(RMortalityX)
model1=sm.OLS(RMortalityY,X)
results1 = model1.fit()
print(results1.summary())

#ridge
results1=sm.OLS(RMortalityY,X).fit_regularized(alpha=50,L1_wt=0)
print(results1.summary())
results1.coef

#lasso
model2=sm.OLS(RMortalityY,X).fit_regularized(alpha=0.0001,L1_wt=1)
results1 = model1.fit()
print(results1.summary())

#random forest
Rforest= RandomForestRegressor(n_estimators = 50)
Rforest.fit(X_train, y_train)
RF_feature_importances = pd.DataFrame(Rforest.feature_importances_,
                                      index = X_train.columns,

columns=['importance']).sort_values('importance',ascending=False)

########gradient boosting###############3

RGradientB =
GradientBoostingRegressor(n_estimators=200,max_depth=6,learning_rate=0.1)
RGradientB.fit(X_train, y_train)
RGradientB_feature_importances =
pd.DataFrame(RGradientB.feature_importances_,
                                      index = X_train.columns,

columns=['importance']).sort_values('importance',ascending=False)
```

```
##########################excel format##########################
cols=X_train.columns.tolist()
cols.append('const')
cols = cols[-1:] + cols[:-1]
########Linear#############
X= sm.add_constant(X_train)
results1=sm.OLS(y_train,X).fit()
modelPvalue= pd.DataFrame(columns=['Linear(p-
value)'],data=results1.pvalues)
ModelResultsAll=modelPvalue

########Ridge############
ridgeR50 = Ridge(alpha=50).fit(X_train,y_train)
modelCoef = pd.DataFrame(columns=['Ridge(Coefficent)'], index=cols)
index=0
for i in cols:
    if str(i)=='const':
        modelCoef.iloc[index][0]=ridgeR50.intercept_
    else :
        modelCoef.iloc[index][0] = ridgeR50.coef_[index-1]
    index=index+1
ModelResultsAll=pd.concat([ModelResultsAll,modelCoef], axis=1)

########Lasso############
RLasso = Lasso(alpha=0.0005).fit(X_train,y_train)
modelCoef = pd.DataFrame(columns=['Lasso(Coefficent)'], index=cols)
index=0
for i in cols:
    if str(i)=='const':
        modelCoef.iloc[index][0]=RLasso.intercept_
    else :
        modelCoef.iloc[index][0] = RLasso.coef_[index-1]
    index=index+1
ModelResultsAll=pd.concat([ModelResultsAll,modelCoef], axis=1)
###############RandomForest
Rforest= RandomForestRegressor(n_estimators = 50).fit(X_train, y_train)
#len(feature_importances_list)
feature_importances_list=list(Rforest.feature_importances_)
feature_importances_list.append('NaN')
feature_importances_list = feature_importances_list[-1:] +
feature_importances_list[:-1]
df_feature_importances = pd.DataFrame(feature_importances_list,
                                      index = cols,
                                      columns=['RandomForest(importance)'])
#.sort_values('importance',ascending=False)
ModelResultsAll=pd.concat([ModelResultsAll,df_feature_importances],
axis=1)
###############GradientBoosting
RGradientB =
GradientBoostingRegressor(n_estimators=200,max_depth=6,learning_rate=0.1)
.fit(X_train, y_train)
#len(feature_importances_list)
feature_importances_list=list(RGradientB.feature_importances_)
feature_importances_list.append('NaN')
feature_importances_list = feature_importances_list[-1:] +
feature_importances_list[:-1]
df_feature_importances = pd.DataFrame(feature_importances_list,
                                      index = cols,
```

```
columns=['GradientBoosting(importance)'])
#.sort_values('importance',ascending=False)
ModelResultsAll=pd.concat([ModelResultsAll,df_feature_importances],
axis=1)

#########################################################################
#######################
writer = pd.ExcelWriter('ModelResults_ALL_2.xlsx')
ModelResultsAll.to_excel(writer,'GeneralModels')
writer.save()
```

5. findTop1000errors.py

```
####################before run this code run loadData
file*********************************

from sklearn.ensemble import RandomForestRegressor
from sklearn import linear_model
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.linear_model import Ridge
from pandas import ExcelWriter
from openpyxl.workbook import Workbook


########linear
model= linear_model.LinearRegression() #model create
model.fit(X_train, y_train) #fit the model
RMortality_y_pred = model.predict(X_test) ##predited values test data
predictionFrame=pd.concat([X_test,y_test], axis=1)
predictionFrame['predicted_number_of_death_LOG']=list(RMortality_y_pred)
predictionFrame['error']=list(abs(y_test-RMortality_y_pred))
top1000ErrorL=predictionFrame.sort_values(by=['error'],ascending=False).h
ead(1000)
top1000ErrorL=top1000ErrorL[['error','number_of_death_LOG','predicted_num
ber_of_death_LOG']].join(mortality_df_org)
#######ridge
model= Ridge(alpha=50).fit(X_train, y_train)
RMortality_y_pred = model.predict(X_test)
predictionFrame=pd.concat([X_test,y_test], axis=1)
predictionFrame['predicted_number_of_death_LOG']=list(RMortality_y_pred)
predictionFrame['error']=list(abs(y_test-RMortality_y_pred))
top1000ErrorR=predictionFrame.sort_values(by=['error'],ascending=False).h
ead(1000)
top1000ErrorR=top1000ErrorR[['error','number_of_death_LOG','predicted_num
ber_of_death_LOG']].join(mortality_df_org)
##########lasso
model= Lasso(alpha=0.0005).fit(X_train, y_train)
RMortality_y_pred = model.predict(X_test)
predictionFrame=pd.concat([X_test,y_test], axis=1)
predictionFrame['predicted_number_of_death_LOG']=list(RMortality_y_pred)
predictionFrame['error']=list(abs(y_test-RMortality_y_pred))
top1000ErrorL=predictionFrame.sort_values(by=['error'],ascending=False).h
ead(1000)
top1000ErrorL=top1000ErrorL[['error','number_of_death_LOG','predicted_num
ber_of_death_LOG']].join(mortality_df_org)
#####random forest
```

```
model= RandomForestRegressor(n_estimators = 50)
model.fit(X_train, y_train)
RMortality_y_pred = model.predict(X_test)
predictionFrame=pd.concat([X_test,y_test], axis=1)
predictionFrame['predicted_number_of_death_LOG']=list(RMortality_y_pred)
predictionFrame['error']=list(abs(y_test-RMortality_y_pred))
top1000ErrorRF=predictionFrame.sort_values(by=['error'],ascending=False).
head(1000)
top1000ErrorRF=top1000ErrorRF[['error','number_of_death_LOG','predicted_n
umber_of_death_LOG']].join(mortality_df_org)
#######Gradient Boosting
model=GradientBoostingRegressor(n_estimators=200,max_depth=6,learning_rat
e=0.1)
model.fit(X_train, y_train)
RMortality_y_pred = model.predict(X_test)
predictionFrame=pd.concat([X_test,y_test], axis=1)
predictionFrame['predicted_number_of_death_LOG']=list(RMortality_y_pred)
predictionFrame['error']=list(abs(y_test-RMortality_y_pred))
top1000ErrorGB=predictionFrame.sort_values(by=['error'],ascending=False).
head(1000)
top1000ErrorGB=top1000ErrorGB[['error','number_of_death_LOG','predicted_n
umber_of_death_LOG']].join(mortality_df_org)

writer = pd.ExcelWriter('PredictionErrors.xlsx')
top1000ErrorL.to_excel(writer,'Linear')
top1000ErrorR.to_excel(writer,'Ridge')
top1000ErrorL.to_excel(writer,'Lasso')
top1000ErrorRF.to_excel(writer,'RandomForest')
top1000ErrorGB.to_excel(writer,'GradientBoosting')
writer.save()

top1000Error.groupby(['COUNTRY_NAME']).size().sort_values()
top1000Error.groupby(['MORTALITY_REASON_DESC']).size().sort_values()
top1000Error.groupby(['age_group']).size().sort_values()

top1000Error[top1000Error['age_group']=='>65'].groupby(['MORTALITY_REASON
_DESC']).size().sort_values()
top1000Error[top1000Error['age_group']=='0'].groupby(['MORTALITY_REASON_D
ESC']).size().sort_values()

mortality_df_org.groupby(['MORTALITY_REASON_DESC']).size().sort_values()

org=mortality_df_org[mortality_df_org['MORTALITY_REASON_DESC'] =='Certain
conditions originating in the perinatal
period'].groupby(['age_group']).size().sort_values()
err=top1000Error[top1000Error['MORTALITY_REASON_DESC'] =='Certain
conditions originating in the perinatal
period'].groupby(['age_group']).size().sort_values()
err_df=pd.DataFrame(err,columns=['ErrorCnt'])
org_df=pd.DataFrame(org,columns=['OrgCnt'])
err_df=pd.concat([err_df,org_df],axis=1)
err_df['rate']= err_df['ErrorCnt'] / err_df['OrgCnt']

err=top1000Error.groupby(['age_group']).size().sort_values()
org=mortality_df_org.groupby(['age_group']).size().sort_values()

err=top1000Error.groupby(['COUNTRY_NAME']).size().sort_values()
org=mortality_df_org.groupby(['COUNTRY_NAME']).size().sort_values()
```

6. seperatedMotels_forAll_Mortality.py

```
 ##################before run this code run loadData
file*********************************


import numpy
import statsmodels.api as sm
from sklearn.ensemble import GradientBoostingRegressor
##########create dummy columns#############
mortality_df_loopY=mortality_df[['MORTALITY_REASON_DESC','number_of_death
_LOG']]
mortality_df_loopX=mortality_df[['MORTALITY_REASON_DESC','number_of_popul
ation_LOG', 'n_smoking_percentage_LOG', 'n_AIR_POLLUTION_LOG',
'n_IMPROVED_DRINKINGWATER_LOG', 'n_BMI_LOG',
'n_HEALTHCARE_EXPENDITURE_LOG', 'n_HDI_LOG', 'n_GDP_LOG']]
labels = ['n_sex','age_group']
mortality_df_loopX=Categorical_to_Dummies(mortality_df,mortality_df_loopX
,labels)
catIniqueArray=mortality_df.MORTALITY_REASON_DESC.unique()
nbr_columns=mortality_df_loopX.shape[1]
scoresAll=pd.DataFrame(columns=['REGRESSION_TYPE','MORTALITY_REASON_DESC'
,'R2_SCORE','TRAINING_SCORE','TEST_SCORE'])

X= sm.add_constant(mortality_df_loopX.iloc[:,1:nbr_columns])
results1=sm.OLS(mortality_df_loopY.iloc[:,1],X).fit()
modelPvalue= pd.DataFrame(columns=['p-value'],data=results1.pvalues)
PValuesAll=modelPvalue
PValuesAll = PValuesAll.transpose()
PValuesAll["MortalityReason"] = "ALL"
PValuesAll["REGRESSION_TYPE"] = "LINEAR"
cols = PValuesAll.columns.tolist()
cols = cols[-1:] + cols[-2:-1] + cols[:-2]
PValuesAll= PValuesAll[cols]

for i in catIniqueArray:
    print(i)
    X =
sm.add_constant(mortality_df_loopX[mortality_df_loopX['MORTALITY_REASON_D
ESC'] == i].iloc[:, 1:nbr_columns])
    Y = mortality_df_loopY[mortality_df_loopY['MORTALITY_REASON_DESC'] ==
i].iloc[:, 1]
    ####
    results1 = sm.OLS(Y,X).fit()
    modelPvalue = pd.DataFrame(columns=['p-value'],
data=results1.pvalues)
    if 'const' not in list(results1.pvalues.index):
        const= pd.DataFrame(columns=['p-value'],
index=['const'],data=[0])
        PValues=pd.concat([const, modelPvalue ])
    else:
        PValues = modelPvalue
    PValues = PValues.transpose()
    PValues["MortalityReason"] = str(i)
    PValues["REGRESSION_TYPE"] = "LINEAR"
    cols = PValues.columns.tolist()
    cols = cols[-1:] + cols[-2:-1] + cols[:-2]
```

```
    PValues = PValues[cols]
    PValuesAll=pd.concat([PValuesAll, PValues])
############RIDGE#################

cols=['const','number_of_population_LOG', 'n_smoking_percentage_LOG',
       'n_AIR_POLLUTION_LOG', 'n_IMPROVED_DRINKINGWATER_LOG',
'n_BMI_LOG',
       'n_HEALTHCARE_EXPENDITURE_LOG', 'n_HDI_LOG', 'n_GDP_LOG',
       'n_sex_FEMALE', 'n_sex_MALE', 'age_group_0', 'age_group_1-4',
       'age_group_10-14', 'age_group_15-19', 'age_group_20-24',
       'age_group_25-29', 'age_group_30-34', 'age_group_35-39',
       'age_group_40-44', 'age_group_45-49', 'age_group_5-9',
       'age_group_50-54', 'age_group_55-59', 'age_group_60-64',
       'age_group_>65']
ridgeR50 = Ridge(alpha=50).fit(mortality_df_loopX.iloc[:,1:nbr_columns],
mortality_df_loopY.iloc[:,1])
modelCoef = pd.DataFrame(columns=['Coefficent'], index=cols)
index=0
for i in cols:
    print(i)
    print(index)
    if str(i)=='const':
        modelCoef.iloc[index][0]=ridgeR50.intercept_
    else :
        modelCoef.iloc[index][0] = ridgeR50.coef_[index-1]
    index=index+1
modelCoef = modelCoef.transpose()
modelCoef["MortalityReason"] = 'ALL'
modelCoef["REGRESSION_TYPE"] = "RIDGE"
cols2 = modelCoef.columns.tolist()
cols2 = cols2[-1:] + cols2[-2:-1] + cols2[:-2]
modelCoef= modelCoef[cols2]
ModelCoefAll=modelCoef
for i in catIniqueArray:
    print(i)
    X =
sm.add_constant(mortality_df_loopX[mortality_df_loopX['MORTALITY_REASON_D
ESC'] == i].iloc[:, 1:nbr_columns])
    Y = mortality_df_loopY[mortality_df_loopY['MORTALITY_REASON_DESC'] ==
i].iloc[:, 1]
    ####
    ridgeR50 = Ridge(alpha=50).fit(X, Y)
    modelCoef = pd.DataFrame(columns=['Coefficent'], index=cols)
    index = 0
    for i2 in cols:
        print(i2)
        print(index)
        if str(i2) == 'const':
            modelCoef.iloc[index][0] = ridgeR50.intercept_
        else:
            modelCoef.iloc[index][0] = ridgeR50.coef_[index - 1]
        index = index + 1
    modelCoef = modelCoef.transpose()
    modelCoef["MortalityReason"] = str(i)
    modelCoef["REGRESSION_TYPE"] = "RIDGE"
    cols2 = modelCoef.columns.tolist()
    cols2 = cols2[-1:] + cols2[-2:-1] + cols2[:-2]
    modelCoef = modelCoef[cols2]
    ModelCoefAll=pd.concat([ModelCoefAll , modelCoef])
```

```
########LASSO########
RLasso =
Lasso(alpha=0.0005).fit(mortality_df_loopX.iloc[:,1:nbr_columns],
mortality_df_loopY.iloc[:,1])
modelCoef = pd.DataFrame(columns=['Coefficent'], index=cols)
index=0
for i in cols:
    print(i)
    print(index)
    if str(i)=='const':
        modelCoef.iloc[index][0]=RLasso.intercept_
    else :
        modelCoef.iloc[index][0] = RLasso.coef_[index-1]
    index=index+1
modelCoef = modelCoef.transpose()
modelCoef["MortalityReason"] = 'ALL'
modelCoef["REGRESSION_TYPE"] = "LASSO"
cols2 = modelCoef.columns.tolist()
cols2 = cols2[-1:] + cols2[-2:-1] + cols2[:-2]
modelCoef= modelCoef[cols2]
ModelCoefAllLasso=modelCoef
for i in catIniqueArray:
    print(i)
    X =
sm.add_constant(mortality_df_loopX[mortality_df_loopX['MORTALITY_REASON_D
ESC'] == i].iloc[:, 1:nbr_columns])
    Y = mortality_df_loopY[mortality_df_loopY['MORTALITY_REASON_DESC'] ==
i].iloc[:, 1]
    ####
    RLasso = Lasso(alpha=0.0005).fit(X, Y)
    modelCoef = pd.DataFrame(columns=['Coefficent'], index=cols)
    index = 0
    for i2 in cols:
        print(i2)
        print(index)
        if str(i2) == 'const':
            modelCoef.iloc[index][0] = RLasso.intercept_
        else:
            modelCoef.iloc[index][0] = RLasso.coef_[index - 1]
        index = index + 1
    modelCoef = modelCoef.transpose()
    modelCoef["MortalityReason"] = str(i)
    modelCoef["REGRESSION_TYPE"] = "LASSO"
    cols2 = modelCoef.columns.tolist()
    cols2 = cols2[-1:] + cols2[-2:-1] + cols2[:-2]
    modelCoef = modelCoef[cols2]
    ModelCoefAllLasso=pd.concat([ModelCoefAllLasso, modelCoef])

#########random forest########
cols=mortality_df_loopX.iloc[:,1:nbr_columns].columns
Rforest= RandomForestRegressor(n_estimators =
50).fit(mortality_df_loopX.iloc[:,1:nbr_columns],
mortality_df_loopY.iloc[:,1])
RF_feature_importances = pd.DataFrame(Rforest.feature_importances_,
                                      index = cols,
                                      columns=['importance'])
#.sort_values('importance',ascending=False)
modelImportance = RF feature importances.transpose()
modelImportance["MortalityReason"] = 'ALL'
```

```
modelImportance["REGRESSION_TYPE"] = "RandomForest"
cols2 = modelImportance.columns.tolist()
cols2 = cols2[-1:] + cols2[-2:-1] + cols2[:-2]
modelImportance= modelImportance[cols2]
modelImportanceAll_RF=modelImportance

for i in catIniqueArray:
    print(i)
    X = mortality_df_loopX[mortality_df_loopX['MORTALITY_REASON_DESC'] ==
i].iloc[:, 1:nbr_columns]
    Y = mortality_df_loopY[mortality_df_loopY['MORTALITY_REASON_DESC'] ==
i].iloc[:, 1]
    Rforest = RandomForestRegressor(n_estimators=50).fit(X,Y)
    RF_feature_importances = pd.DataFrame(Rforest.feature_importances_,
                                          index=cols,
                                          columns=['importance'])
    modelImportance = RF_feature_importances.transpose()
    modelImportance["MortalityReason"] = str(i)
    modelImportance["REGRESSION_TYPE"] = "RandomForest"
    cols2 = modelImportance.columns.tolist()
    cols2 = cols2[-1:] + cols2[-2:-1] + cols2[:-2]
    modelImportance = modelImportance[cols2]
    modelImportanceAll_RF =
pd.concat([modelImportanceAll_RF,modelImportance])
################GRADIENT BOOSTING###########
cols=mortality_df_loopX.iloc[:,1:nbr_columns].columns
RGradientB =
GradientBoostingRegressor(n_estimators=200,max_depth=6,learning_rate=0.1)
.fit(mortality_df_loopX.iloc[:,1:nbr_columns],
mortality_df_loopY.iloc[:,1])
RF_feature_importances = pd.DataFrame(RGradientB.feature_importances_,
                                      index = cols,
                                       columns=['importance'])
#.sort_values('importance',ascending=False)
modelImportance = RF_feature_importances.transpose()
modelImportance["MortalityReason"] = 'ALL'
modelImportance["REGRESSION_TYPE"] = "Gradient  Boosting"
cols2 = modelImportance.columns.tolist()
cols2 = cols2[-1:] + cols2[-2:-1] + cols2[:-2]
modelImportance=modelImportance[cols2]
modelImportanceAll_GB=modelImportance
for i in catIniqueArray:
    print(i)
    X = mortality_df_loopX[mortality_df_loopX['MORTALITY_REASON_DESC'] ==
i].iloc[:, 1:nbr_columns]
    Y = mortality_df_loopY[mortality_df_loopY['MORTALITY_REASON_DESC'] ==
i].iloc[:, 1]
    RGradientB = GradientBoostingRegressor(n_estimators=200, max_depth=6,
learning_rate=0.1).fit(X,Y)
    RF_feature_importances =
pd.DataFrame(RGradientB.feature_importances_,
                                          index=cols,
                                          columns=['importance'])
#.sort_values('importance', ascending=False)
    modelImportance = RF_feature_importances.transpose()
    modelImportance["MortalityReason"] = str(i)
    modelImportance["REGRESSION_TYPE"] = "RandomForest"
    cols2 = modelImportance.columns.tolist()
    cols2 = cols2[-1:] + cols2[-2:-1] + cols2[:-2]
```

```
    modelImportance = modelImportance[cols2]
    modelImportanceAll_GB =
pd.concat([modelImportanceAll_GB,modelImportance])  # , ignore_index=True

writer = pd.ExcelWriter('ModelResults.xlsx')
PValuesAll.to_excel(writer,'Linear')
ModelCoefAll.to_excel(writer,'Ridge')
ModelCoefAllLasso.to_excel(writer,'Lasso')
modelImportanceAll_RF.to_excel(writer,'RandomForest')
modelImportanceAll_GB.to_excel(writer,'GradientBoosting')
writer.save()
#################R2 scores
mortality_df_Y=mortality_df[['MORTALITY_REASON_DESC','number_of_death_LOG
']]
mortality_df_X=mortality_df[['MORTALITY_REASON_DESC','number_of_populatio
n_LOG', 'n_smoking_percentage_LOG', 'n_AIR_POLLUTION_LOG',
'n_IMPROVED_DRINKINGWATER_LOG', 'n_BMI_LOG',
'n_HEALTHCARE_EXPENDITURE_LOG', 'n_HDI_LOG', 'n_GDP_LOG']]
labels = ['n_sex','age_group']
mortality_df_X=Categorical_to_Dummies(mortality_df,mortality_df_X,labels)
np.random.seed(42)
mortality_df_loopX, mortality_df_X_test, mortality_df_loopY,
mortality_df_y_test = train_test_split(mortality_df_X, mortality_df_Y,
test_size=0.3, random_state=10)
###################
catIniqueArray=mortality_df.MORTALITY_REASON_DESC.unique()
nbr_columns=mortality_df_loopX.shape[1]
scoresAll=pd.DataFrame(columns=['REGRESSION_TYPE','MORTALITY_REASON_DESC'
,'R2_SCORE','TRAINING_SCORE','TEST_SCORE'])

for i in catIniqueArray:
   # i = 'Neoplasms'
    print(i)
    X_train =
mortality_df_loopX[mortality_df_loopX['MORTALITY_REASON_DESC'] ==
i].iloc[:, 1:nbr_columns]
    y_train =
mortality_df_loopY[mortality_df_loopY['MORTALITY_REASON_DESC'] ==
i].iloc[:, 1]
    X_test =
mortality_df_X_test[mortality_df_X_test['MORTALITY_REASON_DESC'] ==
i].iloc[:, 1:nbr_columns]
    y_test =
mortality_df_y_test[mortality_df_y_test['MORTALITY_REASON_DESC'] ==
i].iloc[:, 1]
    result=linear_model.LinearRegression().fit(X_train,y_train)
    RMortality_y_pred = result.predict(X_test) ##predited values test
data
    r= r2_score(y_test, RMortality_y_pred)
    train_score =result.score(X_train, y_train)
    test_score=result.score(X_test, y_test)
    scoresAll.loc[len(scoresAll)] =
['Linear',str(i),r,train_score,test_score]
    ##########ridge########
    result = Ridge(alpha=50).fit(X_train, y_train)
    RMortality_y_pred = result.predict(X_test)  ##predited values test
data
    r = r2_score(y_test, RMortality_y_pred)
    train_score = result.score(X_train, y_train)
```

```
    test_score = result.score(X_test, y_test)
    scoresAll.loc[len(scoresAll)] = ['Ridge', str(i), r, train_score,
test_score]
    ###########Lasso############
    result = Lasso(alpha=0.0005).fit(X_train, y_train)
    RMortality_y_pred = result.predict(X_test)  ##predited values test
data
    r = r2_score(y_test, RMortality_y_pred)
    train_score = result.score(X_train, y_train)
    test_score = result.score(X_test, y_test)
    scoresAll.loc[len(scoresAll)] = ['Lasso', str(i), r, train_score,
test_score]
    ###########RandomForest############
    result = RandomForestRegressor(n_estimators = 50).fit(X_train,
y_train)
    RMortality_y_pred = result.predict(X_test)  ##predited values test
data
    r = r2_score(y_test, RMortality_y_pred)
    train_score = result.score(X_train, y_train)
    test_score = result.score(X_test, y_test)
    scoresAll.loc[len(scoresAll)] = ['Random Forest', str(i), r,
train_score, test_score]
    ###########GradientBoosting############
    result =
GradientBoostingRegressor(n_estimators=200,max_depth=6,learning_rate=0.1)
.fit(X_train, y_train)
    RMortality_y_pred = result.predict(X_test)  ##predited values test
data
    r = r2_score(y_test, RMortality_y_pred)
    train_score = result.score(X_train, y_train)
    test_score = result.score(X_test, y_test)
    scoresAll.loc[len(scoresAll)] = ['GradientBoosting', str(i), r,
train_score, test_score]

writer = pd.ExcelWriter('ModelResults_ALL.xlsx')
scoresAll.to_excel(writer,'Model_Accuracy')
writer.save()
```

REFERENCES

[1] Mathers CD, Loncar D (2006) Projections of global mortality and burden of disease from 2002 to 2030. PLoS Med 3(11): e442. doi:10. 1371/journal.pmed.0030442

[2] Department of Information, Evidence and Research WHO, Geneva(2017) .WHO methods and data sources for global burden of disease estimates 2000-2015. Global Health Estimates Technical Paper WHO/HIS/IER/GHE/2017.1

[3] Ewa Tabeau, Anneke Van Den Verg Jeths, Christopher Heatcote (2002).Forecasting Mortality in Developed Countries Insights from a Statistical, Demographic and Epidemiological Perspective. KLUWER ACADEMIC PUBLISHERS

[4] A. D. LOPEZ* and C. D. MATHERS (2006) . Measuring the global burden of disease and epidemiological transitions: 2002–2030 . The Liverpool School of Tropical Medicine DOI: 10.1179/136485906X97417

[5] CLAUDIA PEDROZA(2006). A Bayesian forecasting model: predicting U.S. male mortality. Biostatistics (2006), 7, 4, pp. 530–550 doi:10.1093/biostatistics/kxj024

[6] H. Booth and L. Tickle(2008). MORTALITY MODELLING AND FORECASTING:A REVIEW OF METHODS

[7] Jacques Vallin and France Mesle (2008) . Minimum Mortality: A Predictor of Future Progress? Population 2008/4 (Vol. 63)

[8] Samir Soneji & Gary King (2012). Statistical Security for Social Security. Population Association of America

[9] Federico Girosi and Gary King. 2008. Demographic Forecasting. Princeton: Princeton University Press. Copy at http://j.mp/2nxZqJD

[10] Federico Girosi and Gary King (2007). Understanding the Lee-Carter Mortality Forecasting Method

[11] Edouard Duchesnay, Tommy Löfstedt (2018). Statistics and Machine Learning in Python

[12] Skipper Seabold, Josef Perktold(2010). Statsmodels: Econometric and Statistical Modeling with Python. PROC. OF THE 9th PYTHON IN SCIENCE CONF. (SCIPY 2010)

[13] Fabian Pedregosa, Ga¨el Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel,Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher,Matthieu Perrot and ´Edouard Duchesnay(2011). Scikit-learn: Machine Learning in Python.Journal of Machine Learning Research 12 (2011) 2825-2830

RESOURCES

[1] http://www.who.int/healthinfo/mortality_data/en/

[2] https://data.worldbank.org/indicator/NY.GDP.MKTP.CD

[3] www.ourworldindata.org