WDI-WorldBank

WDI -World Bank Data

```
wdi_data <- read.csv("~/Downloads/WDI_csv/WDIData.csv")#has all the data we n
eed
indicator_names <- wdi_data[, c("Indicator.Name","Indicator.Code")]#will need
this df to understand the indicator codes
```

Select only OECD Countries

```
#features:
length(unique(wdi_data$Indicator.Name))#1443 different features

## [1] 1443

length(unique(wdi_data$Country.Name))#266 different countries

## [1] 266

OECD_countries <- read.csv("~/Downloads/csvData.csv")
dim(OECD_countries)# as of 2020 from OECD website (stats.oecd.org)

## [1] 36  3

new_countries <- c("Costa Rica", "Colombia")#new members added in 2021 (https
://www.oecd.org/newsroom/oecd-welcomes-costa-rica-as-its-38th-member.htm#:~:t
ext=The%20OECD's%2038%20members%20are,Norway%2C%20Poland%2C%20Portugal%2C%20S
lovak)
OECD_country_all <- c(OECD_countries$country, new_countries)
all_OECD_data <- wdi_data[wdi_data$Country.Name %in% OECD_country_all,]

dim(all_OECD_data)

## [1] 51948    67

a <- unique(all_OECD_data$Country.Name)
b <- sort(OECD_country_all)

b[!b%in%a]

## [1] "Slovakia"    "South Korea"

wdi_data[wdi_data$Country.Name=="Slovakia",]

##  [1] Country.Name   Country.Code   Indicator.Name Indicator.Code X1960
##  [6] X1961          X1962          X1963          X1964          X1965
## [11] X1966          X1967          X1968          X1969          X1970
## [16] X1971          X1972          X1973          X1974          X1975
## [21] X1976          X1977          X1978          X1979          X1980
## [26] X1981          X1982          X1983          X1984          X1985
```

```
## [31] X1986          X1987          X1988          X1989          X1990
## [36] X1991          X1992          X1993          X1994          X1995
## [41] X1996          X1997          X1998          X1999          X2000
## [46] X2001          X2002          X2003          X2004          X2005
## [51] X2006          X2007          X2008          X2009          X2010
## [56] X2011          X2012          X2013          X2014          X2015
## [61] X2016          X2017          X2018          X2019          X2020
## [66] X2021          X
## <0 rows> (or 0-length row.names)
```

```
wdi_data[wdi_data$Country.Name=="South Korea",]
```

```
##  [1] Country.Name   Country.Code   Indicator.Name Indicator.Code X1960
##  [6] X1961          X1962          X1963          X1964          X1965
## [11] X1966          X1967          X1968          X1969          X1970
## [16] X1971          X1972          X1973          X1974          X1975
## [21] X1976          X1977          X1978          X1979          X1980
## [26] X1981          X1982          X1983          X1984          X1985
## [31] X1986          X1987          X1988          X1989          X1990
## [36] X1991          X1992          X1993          X1994          X1995
## [41] X1996          X1997          X1998          X1999          X2000
## [46] X2001          X2002          X2003          X2004          X2005
## [51] X2006          X2007          X2008          X2009          X2010
## [56] X2011          X2012          X2013          X2014          X2015
## [61] X2016          X2017          X2018          X2019          X2020
## [66] X2021          X
## <0 rows> (or 0-length row.names)
```

```
#although South Korea and Slovakia are OECD countries the World Bank dataset
doesn't have any data for these countries
```

Select only Years Between 2000 - 2021 to account for technological and world-wide development since there has been many incidents over history and if we include all years we would also have to account for historical effects. This data will only entail the 21st century and OECD countries since this specific group has a higher chance of fitting the algorithm accurately and extracting information logically.

Also I will pivot the data table so that all indicators designated by the World Bank and the years which are going to be features will be explanatory variables in my models. All indicators will be columns and year will also be a column.

```
class(names(all_OECD_data))
```

```
## [1] "character"
```

```
delete_cols <- c(as.character(paste0("X", 1960:1999)), "X", "Indicator.Name",
"Country.Code")
OECD_21c_data <- all_OECD_data[,!names(all_OECD_data)%in%delete_cols]
names(OECD_21c_data)[3:24] <- as.numeric(2000:2021)

library(tidyr)
```

```
new_1 <- pivot_longer(data=OECD_21c_data,  cols = c(as.character(2000:2021)),
names_to="Year")
OECD_21c <- pivot_wider(data=new_1, names_from = Indicator.Code, values_from
= value)
dim(OECD_21c)

## [1]  792 1445

dim(na.omit(OECD_21c))#I can't exclude na values so I will generate artificia
l data points with knn clustering after splitting into test and train data

## [1]    0 1445
```

Working with Years Data

For this data set I selected the time frame to be the 21st century and I am first going to use the year variable as a feature to evaluate whether years have a certain effect on a country's development since with each new year there are new inventions, policies, technological developments, diseases, etc. After analyzing year's effect as a feature, I will try to fit the best model on to designated world-wide financial crisis years and designated world-wide prosperous years. I will do this analysis to observe whether my algorithm is better at predicting during crisis years or prosperous years which would provide a better understanding of the machine learning process and would enable my audience to use this algorithm for cases that this algorithm proves to be most accurate.

Specify Developed and Developing Countries

The data that specifies which OECD countries are developed and developing are from the UN classification document "World Economic Situation and Prospects 2020" (UNITED NATIONS DEPARTMENT FOR ECONOMIC AND SOCIAL AFFAIRS. World economic situation and prospects 2020. UN, 2020.) Greece and Turkey are categorized as economies in transition so for this project I am going to assume that it is a developing economy. For this part I am hard coding the explanatory variable of the project. Developing countries=0, developed countries=1

```
unique(OECD_21c$Country.Name)

##  [1] "Australia"      "Austria"        "Belgium"        "Canada"        
##  [5] "Chile"          "Colombia"       "Costa Rica"     "Czech Republic"
##  [9] "Denmark"        "Estonia"        "Finland"        "France"        
## [13] "Germany"        "Greece"         "Hungary"        "Iceland"       
## [17] "Ireland"        "Israel"         "Italy"          "Japan"         
## [21] "Latvia"         "Lithuania"      "Luxembourg"     "Mexico"        
## [25] "Netherlands"    "New Zealand"    "Norway"         "Poland"        
## [29] "Portugal"       "Slovenia"       "Spain"          "Sweden"        
## [33] "Switzerland"    "Turkey"         "United Kingdom" "United States"

y <- c(1,1,1,1,0,0,0,1,1,1,1,1,1,0,1,1,1,0,0,1,1,1,1,0,1,1,1,1,1,1,1,1,1,0,1,
1)
y_df <- data.frame(y=y, Country.Name=unique(OECD_21c$Country.Name))
```

```
OECD_21c <- merge(y_df, OECD_21c, by="Country.Name",all=T)
#since the data of developed and developing countries are not proportional an
d developing countries have less data I will use upsampling before running my
models
```

Data Exploration and Wrangling

```
set.seed(1000)
unique(apply(OECD_21c, 2, class))

## [1] "character"

OECD_21c[1:10,1:10]

##    Country.Name y Year EG.CFT.ACCS.ZS EG.ELC.ACCS.ZS EG.ELC.ACCS.RU.ZS
## 1     Australia 1 2000            100            100               100
## 2     Australia 1 2001            100            100               100
## 3     Australia 1 2002            100            100               100
## 4     Australia 1 2003            100            100               100
## 5     Australia 1 2004            100            100               100
## 6     Australia 1 2005            100            100               100
## 7     Australia 1 2006            100            100               100
## 8     Australia 1 2007            100            100               100
## 9     Australia 1 2008            100            100               100
## 10    Australia 1 2009            100            100               100
##    EG.ELC.ACCS.UR.ZS FX.OWN.TOTL.ZS FX.OWN.TOTL.FE.ZS FX.OWN.TOTL.MA.ZS
## 1                100             NA                NA                NA
## 2                100             NA                NA                NA
## 3                100             NA                NA                NA
## 4                100             NA                NA                NA
## 5                100             NA                NA                NA
## 6                100             NA                NA                NA
## 7                100             NA                NA                NA
## 8                100             NA                NA                NA
## 9                100             NA                NA                NA
## 10               100             NA                NA                NA

#it is also important to get rid of all columns that have mostly na values be
cause those metrics wouldn't provide any insight for us and the artificial da
ta generation with knn means wouldn't suffice
#for this na cleaning part I will delete the columns that have na values more
than or equal to half of the total data for each indicator/feature, I am omit
ting na values more than half but while regenerating this project other resea
rchers can use other numbers such as 3/4 or 1/5 depending on the accuracy the
y want to get for the knn mean data generation for missing values
#In this project I want to have less na values so that knn means could impute
missing data points better before fitting my models

drop_col <- c()
keep_col <- c()
for (i in 1:ncol(OECD_21c)) {
```

```
  if (sum(is.na(OECD_21c[, i])) >= (nrow(OECD_21c)*0.5)){
  drop_col <- c(i, drop_col)
  }else{
  keep_col <- c(i, keep_col)
}
}

drop_cols_names <- names(OECD_21c)[drop_col]

indicator_names <- indicator_names[!indicator_names$Indicator.Code %in% drop_
cols_names,]
OECD_21c <- OECD_21c[,-drop_col]
dim(OECD_21c)#I dropped all columns that had na values more than or equal to
half the data for each indicator I also dropped them from the indicator name
df that provides name and explanation info for each indicator

## [1] 792 914

#getting data summary on randomly selected variables:
library(skimr)
data_summary_df <- skim_to_wide(sample(OECD_21c,100))

## Warning: 'skim_to_wide' is deprecated.
## Use 'skim()' instead.
## See help("Deprecated")

data_summary_df_names <- unique(indicator_names[indicator_names$Indicator.Cod
e %in% data_summary_df$skim_variable,])

random_variables_summary_final <- merge(data_summary_df_names, data_summary_d
f, by.x="Indicator.Code", by.y="skim_variable", all=T)
head(random_variables_summary_final)

##      Indicator.Code
## 1 BN.CAB.XOKA.GD.ZS
## 2    BX.TRF.CURR.CD
## 3 BX.TRF.PWKR.CD.DT
## 4 DC.ODA.TLDC.GN.ZS
## 5 DC.ODA.TOTL.GN.ZS
## 6 EG.USE.PCAP.KG.OE
##                                                  Indicator.Name skim_type
## 1                             Current account balance (% of GDP)   numeric
## 2                    Secondary income receipts (BoP, current US$)   numeric
## 3                    Personal remittances, received (current US$)   numeric
## 4 Net ODA provided to the least developed countries (% of GNI)   numeric
## 5                             Net ODA provided, total (% of GNI)   numeric
## 6                    Energy use (kg of oil equivalent per capita)   numeric
##   n_missing complete_rate  numeric.mean   numeric.sd    numeric.p0
## 1        48     0.9393939 -1.904116e-01 5.506023e+00 -2.293929e+01
## 2        48     0.9393939  1.192490e+10 2.084852e+10  1.292499e+07
```

```
## 3        36     0.9545455  3.506664e+09 5.652443e+09  0.000000e+00
## 4       328     0.5858586  8.070841e-02 7.681777e-02  2.528189e-05
## 5       325     0.5896465  3.917152e-01 2.771223e-01  1.300000e-02
## 6       220     0.7222222  4.084034e+03 2.606890e+03  6.166066e+02
##     numeric.p25   numeric.p50  numeric.p75 numeric.p100 numeric.hist
## 1 -3.414430e+00 -6.693974e-01 3.190387e+00 1.617942e+01        ▁▁▇▆▁
## 2  1.909513e+09  5.422592e+09 1.184678e+10 1.663440e+11        ▇▁▁▁▁
## 3  5.530306e+08  1.371789e+09 3.940457e+09 4.287827e+10        ▇▁▁▁▁
## 4  2.223353e-02  5.825147e-02 1.144098e-01 4.827813e-01        ▇▂▁▁▁
## 5  1.860000e-01  3.040000e-01 5.130000e-01 1.405000e+00        ▇▃▂▂▁
## 6  2.517422e+03  3.575134e+03 4.972194e+03 1.817814e+04        ▇▃▁▁▁
```

```
tail(random_variables_summary_final)
```

```
##        Indicator.Code
## 95  TM.VAL.OTHR.ZS.WT
## 96  TX.VAL.MRCH.R5.ZS
## 97  TX.VAL.MRCH.R6.ZS
## 98  VC.IHR.PSRC.FE.P5
## 99  VC.IHR.PSRC.MA.P5
## 100    VC.IHR.PSRC.P5
##
Indicator.Name
## 95                                   Computer, communications and other se
rvices (% of commercial service imports)
## 96          Merchandise exports to low- and middle-income economies in Sou
th Asia (% of total merchandise exports)
## 97  Merchandise exports to low- and middle-income economies in Sub-Saharan
Africa (% of total merchandise exports)
## 98                                                              Intentiona
l homicides, female (per 100,000 female)
## 99                                                                  Intent
ional homicides, male (per 100,000 male)
## 100                                                                     In
tentional homicides (per 100,000 people)
##     skim_type n_missing complete_rate numeric.mean numeric.sd numeric.p0
## 95    numeric        72     0.9090909    41.660950  15.565452 6.79200740
## 96    numeric        72     0.9090909     1.052985   1.273720 0.01160267
## 97    numeric        72     0.9090909     1.148137   1.129610 0.01285524
## 98    numeric       188     0.7626263     1.358466   1.414966 0.00000000
## 99    numeric       188     0.7626263     6.244258  14.910468 0.00000000
## 100   numeric       124     0.8434343     3.619242   7.598932 0.00000000
##     numeric.p25 numeric.p50 numeric.p75 numeric.p100 numeric.hist
## 95   32.1661831  40.5514675   49.580141     89.42277        ▂▇▇▂▁
## 96    0.3148055   0.5988889    1.282205     10.04352        ▇▁▁▁▁
## 97    0.4853821   0.9287194    1.425336      9.34222        ▇▁▁▁▁
## 98    0.5907285   0.8717591    1.451069     10.78835        ▇▁▁▁▁
## 99    1.1075554   1.7062509    3.783771    130.51205        ▇▁▁▁▁
## 100   0.8908856   1.2683419    2.538270     69.44770        ▇▁▁▁▁
```

```
#Looking Into Data Types - What are the values?
table(indicator_names$Indicator.Name)[1:10]#the data are stored as percentage
s, dollars or scales/indexes

##
##     Access to clean fuels and technologies for cooking (% of population)
##                                                                      266
##                                    Access to electricity (% of population)
##                                                                      266
##                       Access to electricity, rural (% of rural population)
##                                                                      266
##                       Access to electricity, urban (% of urban population)
##                                                                      266
## Adjusted net enrollment rate, primary (% of primary school age children)
##                                                                      266
##                              Adjusted net national income (annual % growth)
##                                                                      266
##                            Adjusted net national income (constant 2015 US$)
##                                                                      266
##                                Adjusted net national income (current US$)
##                                                                      266
##                  Adjusted net national income per capita (annual % growth)
##                                                                      266
##                Adjusted net national income per capita (constant 2015 US$)
##                                                                      266

all_data_percentage <- grepl("%", indicator_names$Indicator.Name)
percentage_cleaned <- unique(indicator_names$Indicator.Name[!all_data_percent
age])#finding variables that are not percentage data

perc_and_dollar_cleaned <- percentage_cleaned[!grepl("\\$", percentage_cleane
d)]#variables that are not dollar not percentage

#From this data exploration I realized that some data are in LCU which means
in local currency, I will use local currency vs US$ rate data to convert all
LCU data to US$
#I will get the exchange rates from the World Bank data in solumn "PA.NUS.FCR
F" which is the "Official exchange rate (LCU per US$, period average)" period
average is yearly average in this case and I will fill in the missing exchang
e rate values from OECD yearly exchange rate data and exclude that column sin
ce we are going to account for it with all the data we are converting to USD

OECD_21c <- subset(OECD_21c, select = -c(PA.NUS.FCRF) )

exchange_rates_to_US <- read.csv("~/Downloads/DP_LIVE_29042022010707251.csv")
#OECD (2022), Exchange rates (indicator). doi: 10.1787/037ed317-en (Accessed
on 28 April 2022)

get_country_names <- read.csv("~/Downloads/SNA_TABLE1_22042022071317139.csv")
get_country_names <- unique(get_country_names[,c("LOCATION", "Country")])
```

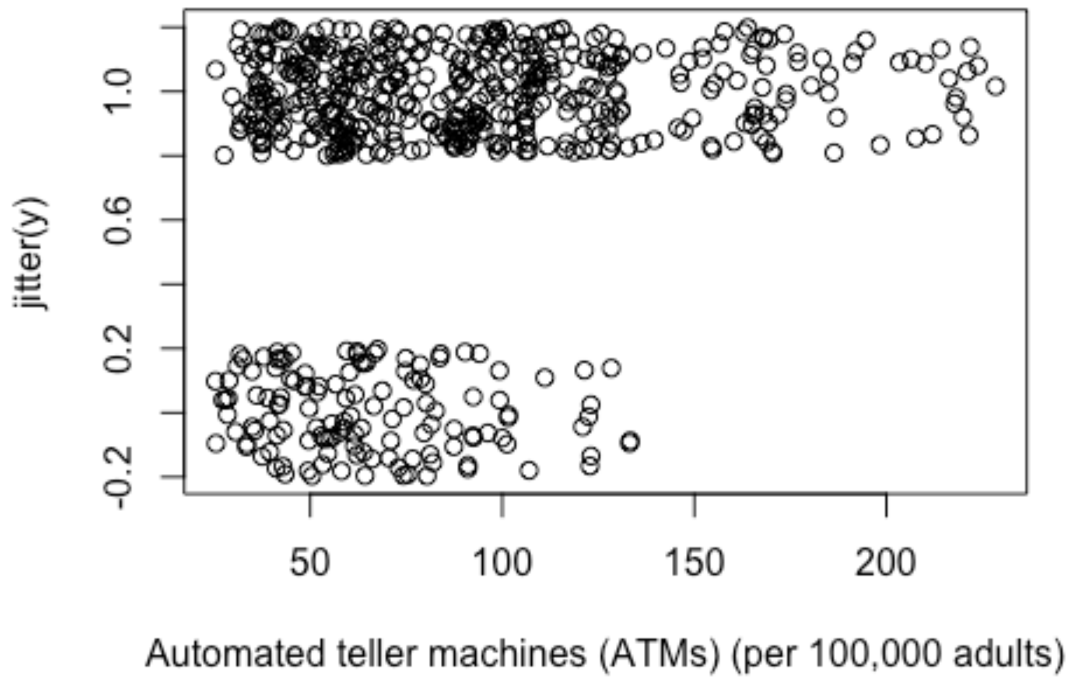
```
country_names_exchange_rates <- merge(exchange_rates_to_US, get_country_names
, by="LOCATION")
country_names_exchange_rates <- country_names_exchange_rates[,c("Country", "T
IME", "Value")]

table(country_names_exchange_rates$Country)[table(country_names_exchange_rate
s$Country)!=22]#2021 rate is missing for Turkey, so I get the 2021 yearly avg
from

## Turkey
##     21

tr_2021_rate <- data.frame(Country="Turkey", TIME=2021, Value=8.8922)

country_names_exchange_rates <- rbind(country_names_exchange_rates, tr_2021_r
ate)

#Now I will  multiply all LCU data with the corresponding exchange rates to b
e able to have all data in US$:

lcu_codes <- indicator_names$Indicator.Code[grepl("LCU", indicator_names$Indi
cator.Name)]


OECD_21c_with_exchange <- merge(OECD_21c, country_names_exchange_rates, by.x=
c("Country.Name", "Year"), by.y = c("Country", "TIME"), all.x=T)


dim(OECD_21c_with_exchange)

## [1] 792 914

in_dollars <- OECD_21c_with_exchange[,which(names(OECD_21c_with_exchange) %in
% lcu_codes)] * OECD_21c_with_exchange$Value

OECD_21c[,which(names(OECD_21c) %in% lcu_codes)] <- in_dollars

#Now all data is in US$ which enables comparison for the models
```

Visualizing Random Data That is Not Collected in Percentage and Dollars

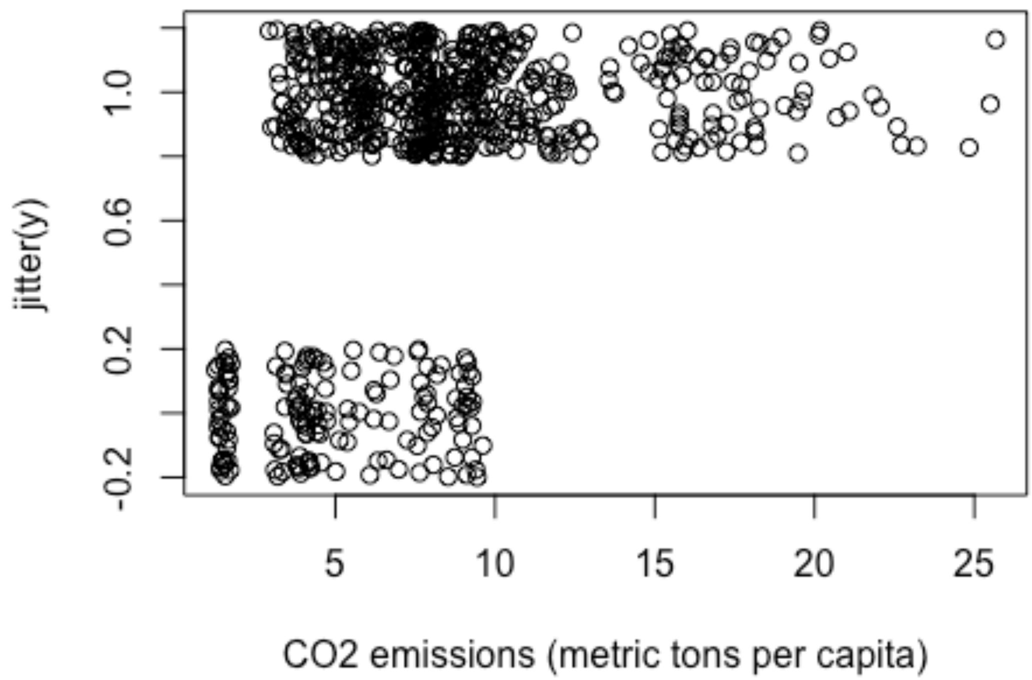
```
set.seed(1000)
random_variables <- sample(perc_and_dollar_cleaned, 10)

random_vars_df <- unique(indicator_names[indicator_names$Indicator.Name %in%
random_variables,])

quick_visualization_data <- OECD_21c[,c("y", random_vars_df$Indicator.Code)]
class(quick_visualization_data$FB.ATM.TOTL.P5)
```

```
## [1] "numeric"
```

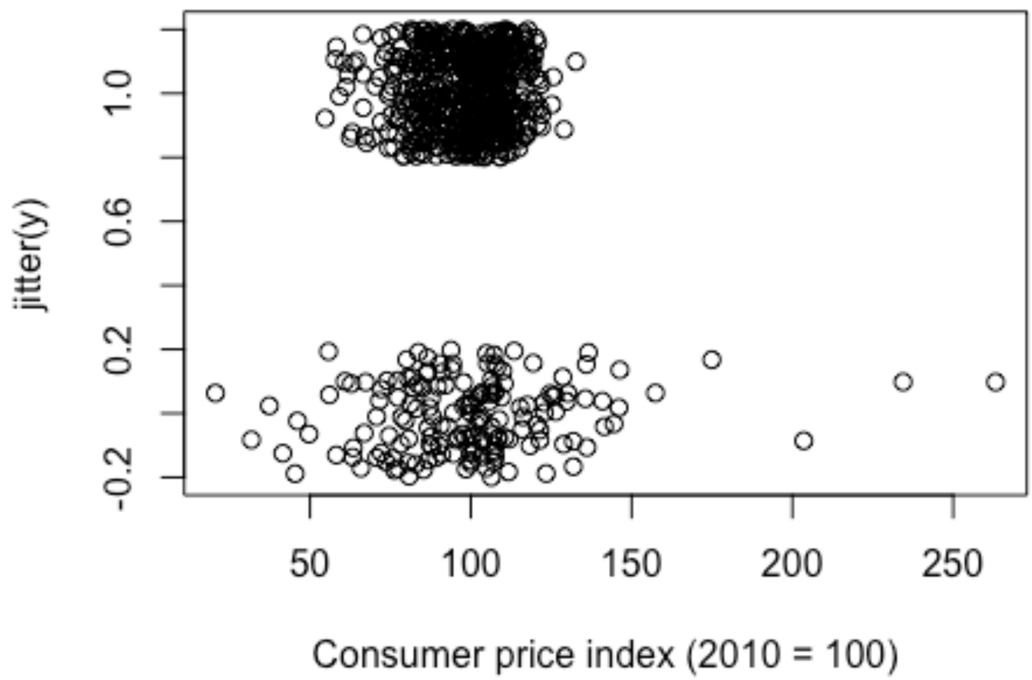
```
plot(jitter(y)~jitter(FB.ATM.TOTL.P5), data=quick_visualization_data, xlab=ra
ndom_vars_df$Indicator.Name[1])
```



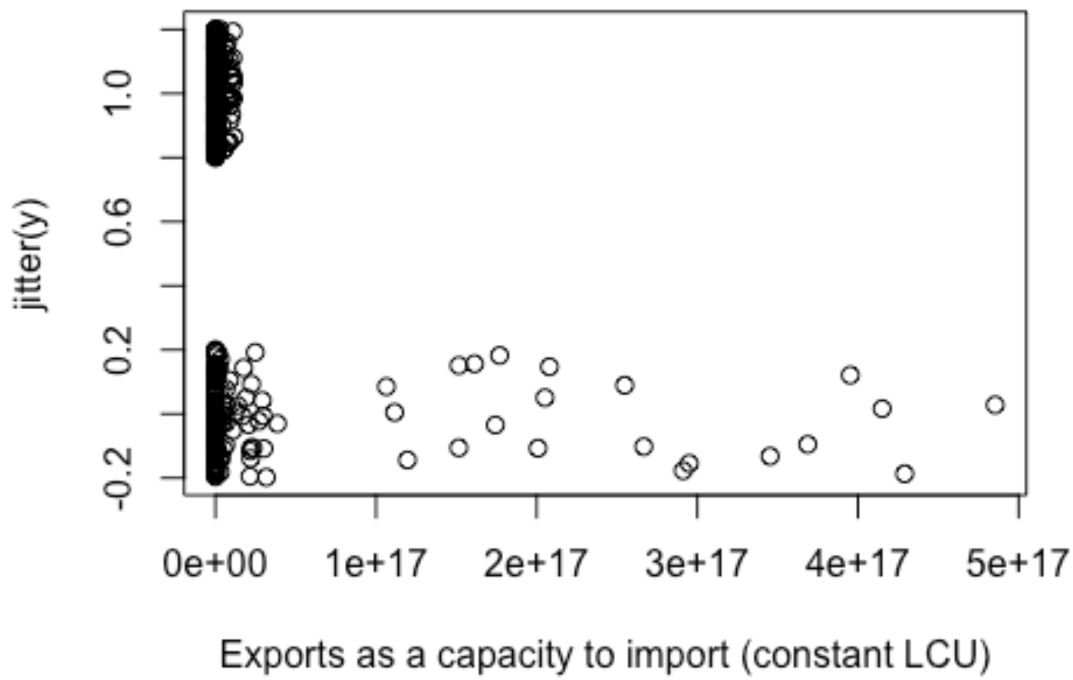

```
plot(jitter(y)~jitter(EN.ATM.CO2E.PC), data=quick_visualization_data, xlab=ra
ndom_vars_df$Indicator.Name[2])
```



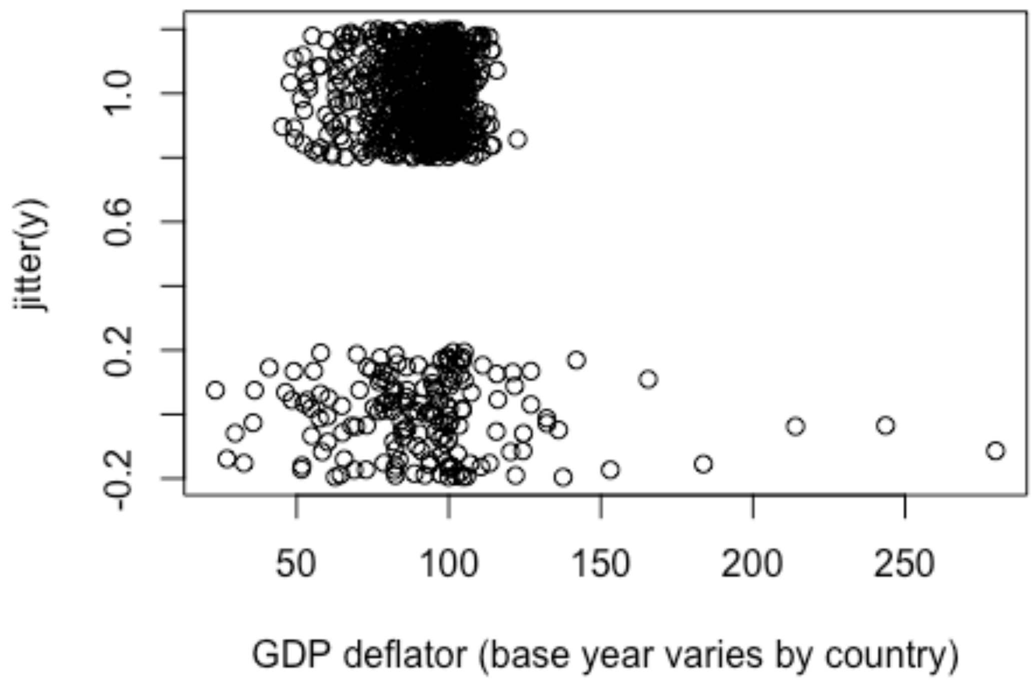

```
plot(jitter(y)~jitter(FP.CPI.TOTL), data=quick_visualization_data, xlab=rando
m_vars_df$Indicator.Name[3])
```



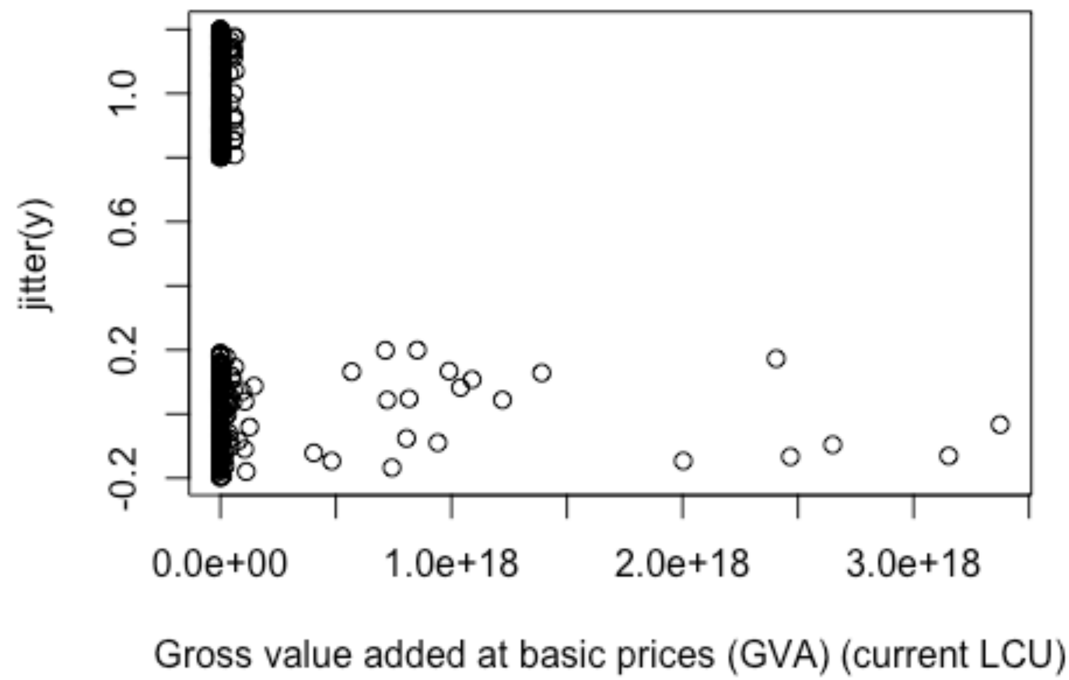

```
plot(jitter(y)~jitter(NY.EXP.CAPM.KN), data=quick_visualization_data, xlab=ra
ndom_vars_df$Indicator.Name[4])
```



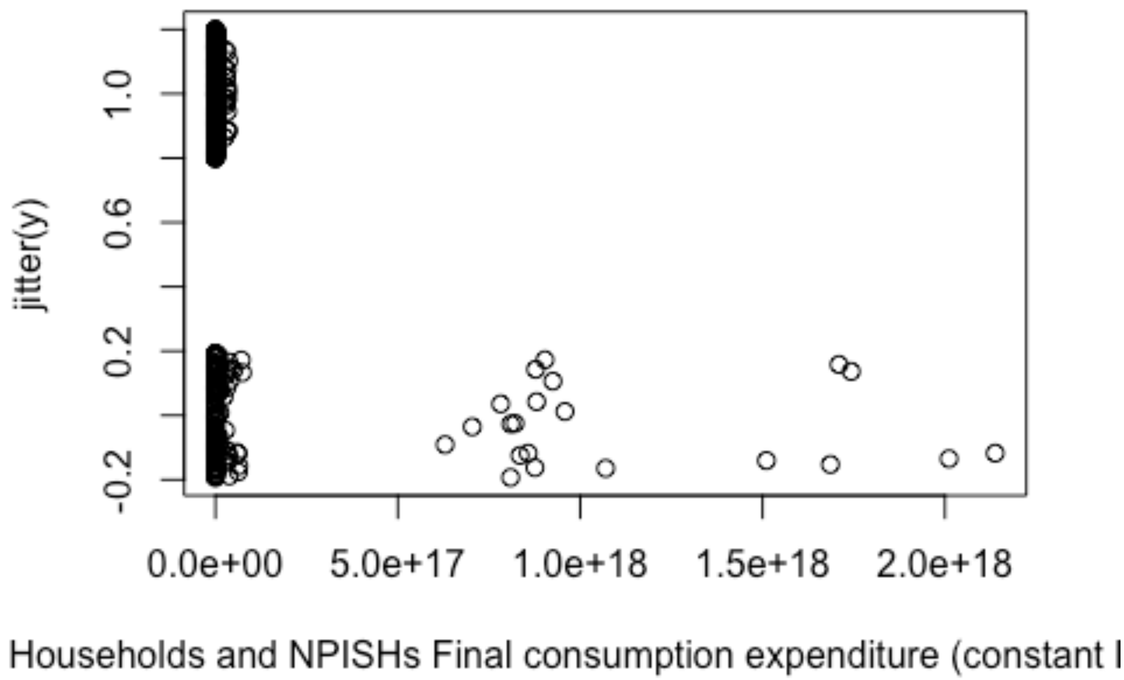

```
plot(jitter(y)~jitter(NY.GDP.DEFL.ZS), data=quick_visualization_data, xlab=ra
ndom_vars_df$Indicator.Name[5])
```



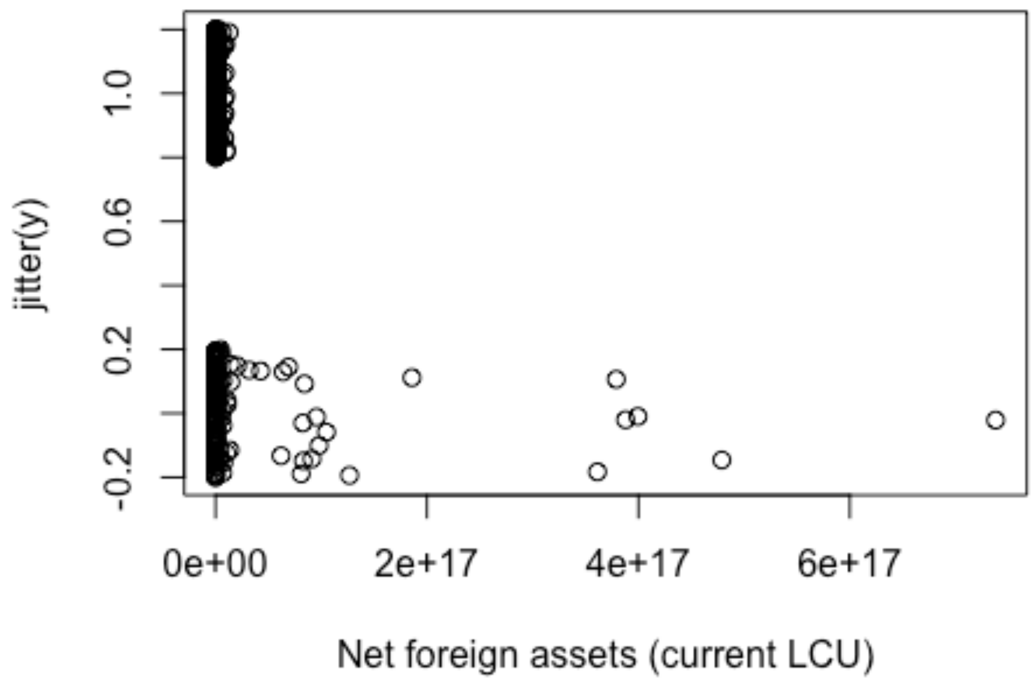

```
plot(jitter(y)~jitter(NY.GDP.FCST.CN), data=quick_visualization_data, xlab=ra
ndom_vars_df$Indicator.Name[6])
```



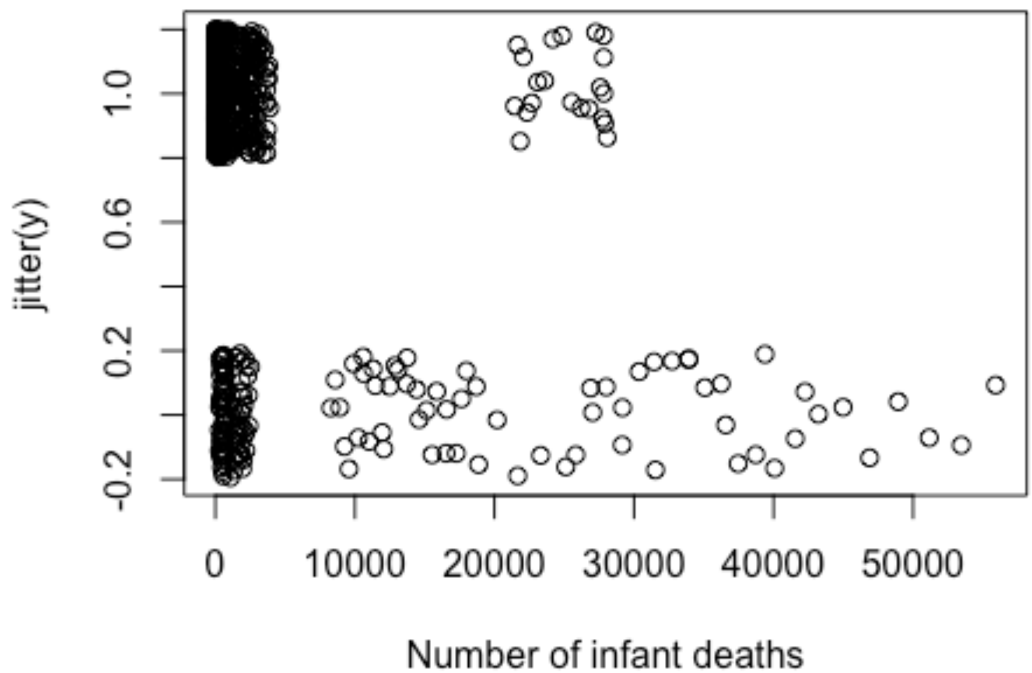

```
plot(jitter(y)~jitter(NE.CON.PRVT.KN), data=quick_visualization_data, xlab=ra
ndom_vars_df$Indicator.Name[7])
```



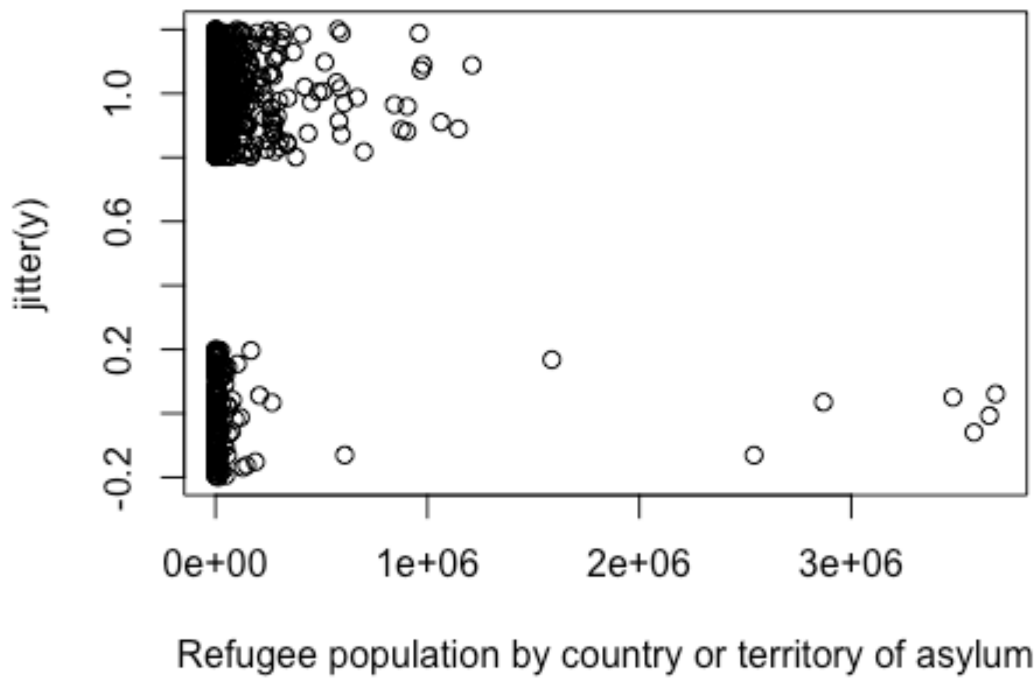

```
plot(jitter(y)~jitter(FM.AST.NFRG.CN), data=quick_visualization_data, xlab=ra
ndom_vars_df$Indicator.Name[8])
```




```
plot(jitter(y)~jitter(SH.DTH.IMRT), data=quick_visualization_data, xlab=rando
m_vars_df$Indicator.Name[9])
```




```
plot(jitter(y)~jitter(SM.POP.REFG), data=quick_visualization_data, xlab=rando
m_vars_df$Indicator.Name[10])
```

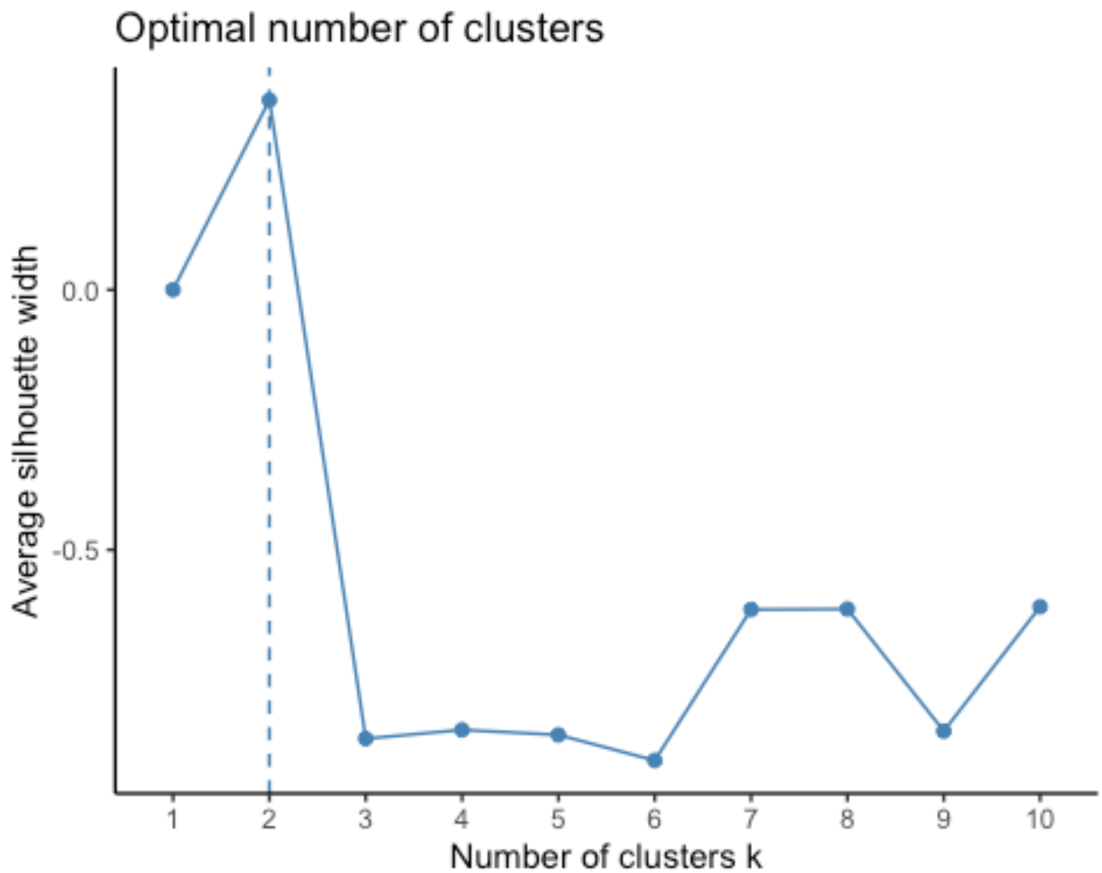


Filling NA Values with PreProcessing

After separating test and train and I will fill in th NA values and use preprocess() from caret package to impute the missing data. I can use three methods for imputation: knn, bag, median. I won't generate dummy variables for scale data because even though the dataset description has noted them as scaled some data points are float variables

```
library(cluster)
library(factoextra)

## Loading required package: ggplot2

## Welcome! Want to learn more? See two factoextra-related books at https://g
oo.gl/ve3WBa

library(caret)

## Loading required package: lattice

#Let's find k for knn impute:
OECD_21c$Year <- as.numeric(OECD_21c$Year)
fviz_nbclust(OECD_21c[,c(-1,-2)], clara, method = "silhouette", correct.d=TRU
E)+
```

```
  theme_classic() #according to clara clustering 2 is an optimal number for c
lusters, I used clara because there are many NA values in the dataset
```



Although k=2 clustering for knn or bag(like random forest but with smaller numbers of trees) imputation are optimal, the data set has a large number of NA values so for pre-processing, I am going to conduct a two sample t-test on the means of total na values in each developed and developing country entry to find out if the na values are randomly distributed amongst the two different groups of countries. If randomly distributed than I would be able to use a median impute method to impute the missing values with the sample medians since the assumption of median impute method relies on the NA values being distributed randomly in the data set in order to avoid biased sampling during my analysis.

##Two Sample T-Test for Mean Analysis of Random NA distribution

```
#H0=na values are randomly distributed across developing and developed countr
ies, x_bar1=x_bar2
#HA=na values are not randomly distributed across developing and developed co
untries, x_bar1!=x_bar2
sums <- c()
for (i in 1:nrow(OECD_21c)){
sums[i] <- sum(is.na(OECD_21c[i,]))
}
```

```
test_df <- data.frame(na_sums=sums, y=OECD_21c$y)
t.test(na_sums ~ y, data = test_df)#we do not reject the null hypothesis

##
##  Welch Two Sample t-test
##
## data:  na_sums by y
## t = -0.28272, df = 283.59, p-value = 0.7776
## alternative hypothesis: true difference in means between group 0 and group
1 is not equal to 0
## 95 percent confidence interval:
##  -33.27120  24.91405
## sample estimates:
## mean in group 0 mean in group 1
##        125.2500        129.4286
```

According to the hypothesis test NA values are randomly distributed as we do not reject the null hypothesis since the p-value is much larger than alpha=0.05. Although the median impute method is not the ideal pre processing strategy, for example knn or bag imputations would be much preferable for reasonable predictions, with my large amount of missing data I have to utilize this convenient and optimal method which I can satisfy its assumptions within my data. In the analysis of this data mining project I will account for the statistical imbalances or misinterpretations this median imputing method could cause.

```
preProcess_missingdata_model <- preProcess(OECD_21c, method='medianImpute')
preProcess_missingdata_model

## Created from 0 samples and 913 variables
##
## Pre-processing:
##   - ignored (1)
##   - median imputation (912)
```

According to the steps of the process of median imputation above; it ignored 1 variables and imputed data for all variables. I will now use this model to predict the missing values in OECD_21c:

```
library(RANN)  # required for knnInpute

OECD_21c <- predict(preProcess_missingdata_model, newdata = OECD_21c)
anyNA(OECD_21c)

## [1] FALSE
```

DATA MINING

In this part I will run models and look into veiled patterns within the data. My goal in this data mining project is to determine the most significant indicators that indicate the development level of OECD countries. I will be able to determine the most significant features or indicators by collecting the most frequently identified indicators by the models

I train my data on. I will first start with Lasso because it will assign a coefficient value of zero to insignificant indicators and I will be able to identify the number of explanatory/predictor variables necessary for my other models. After receiving the list of significant indicators and the amount of them, I will select the same number of indicators that are most significant in my other models which are logistics regression, pca+logistic regression, NaiveBayes, and RandomForest. In the end I will compare all models with the classification error, precision, recall, and sensitivity metrics of predicted data and choose the best model. After selecting the best model I will run the final model with the most frequently identified significant indicators(which I will get from comparing all my models) and run the model based on only one year which will be 2008(The Great Recession). I am using year first as an explanatory variable to see is the year indicator has a significant effect on the development level prediction, then I will use the data from only the crisis year, 2008, to assess whether my selected best fitting model works better when year and indicators are specified. Specifying year might have a better effect since each year the technology, international relations, and political events differ and affect the OECD countries in similar ways. I am trying to evaluate whether fitting the model per annum would yield better predictions. I think it is also important to determine how the model evaluates data during different periods because when this model is reproduced for country analysis world wide it would be imperative to account for a crisis yar which would have a negative effect on many indicators and a prosperous year which would pump up the economic indicators. These shifts, when year base data is not accounted for, could be misleading on determining a country's level.

Separating Train and Test Sets

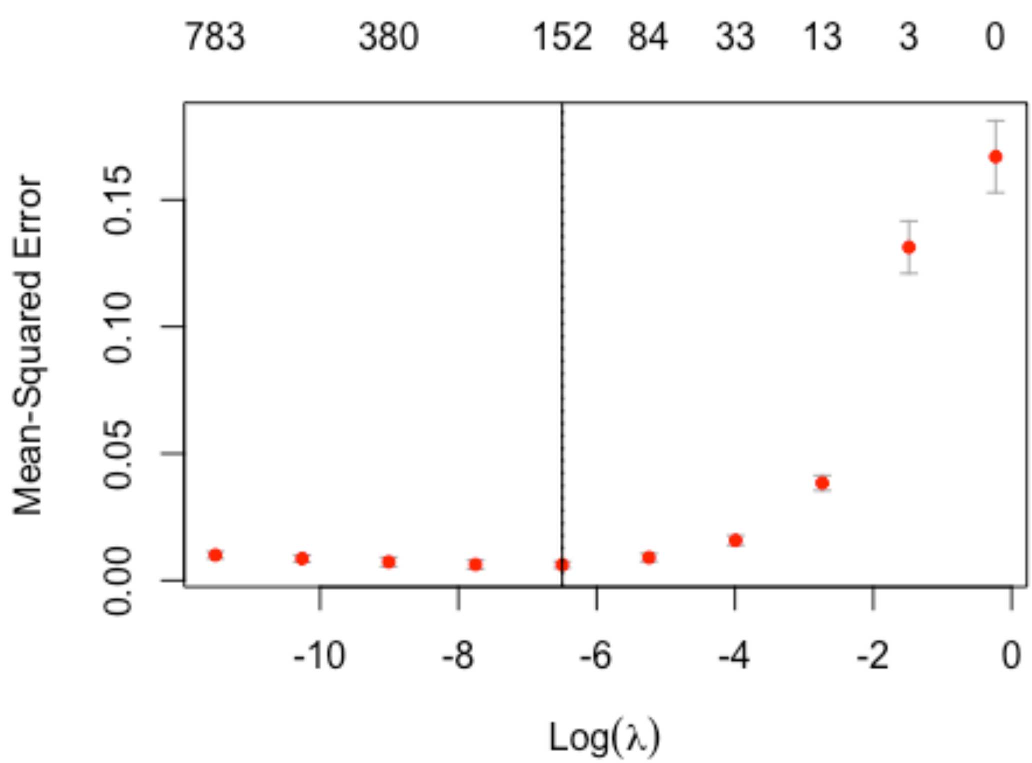
```
set.seed(1000)
#row numbers for the training data
trainRowNumbers <- createDataPartition(OECD_21c$y, p=0.8, list=FALSE)

#getting the training  dataset
trainData <- OECD_21c[trainRowNumbers,-1]

#getting test dataset
testData <- OECD_21c[-trainRowNumbers,-1]
```

Lasso(standardized)

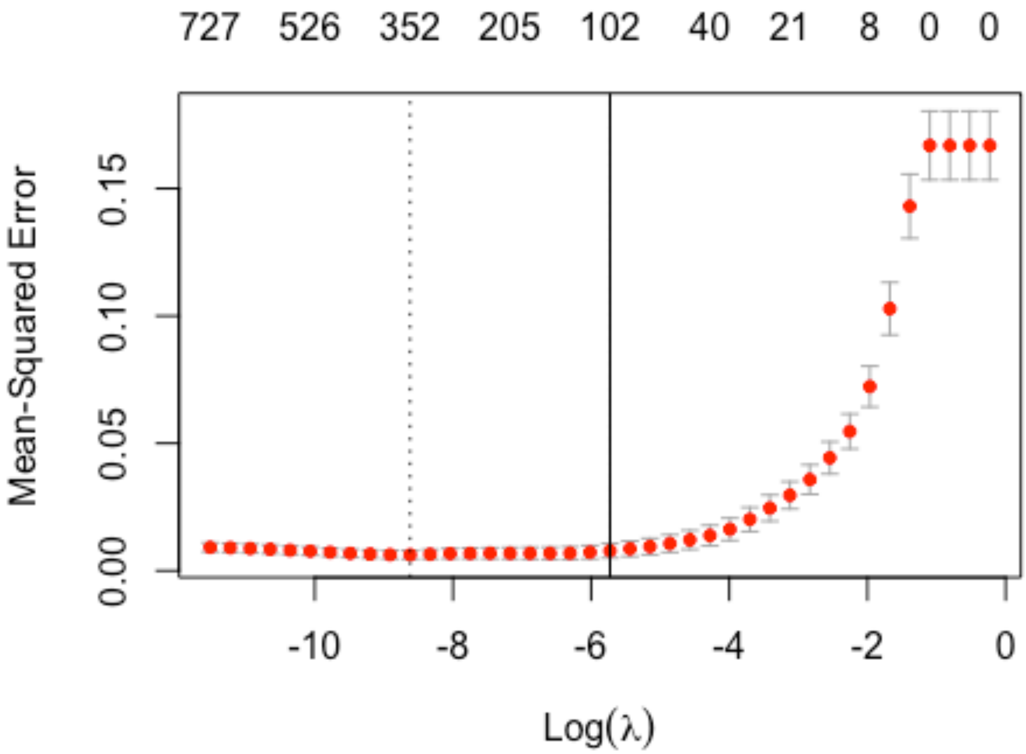
Sets coeff to absolute zero if not significant so a Lasso model is a good method for feature selection within the model.

```
library(glmnet)

## Loading required package: Matrix

##
## Attaching package: 'Matrix'
```

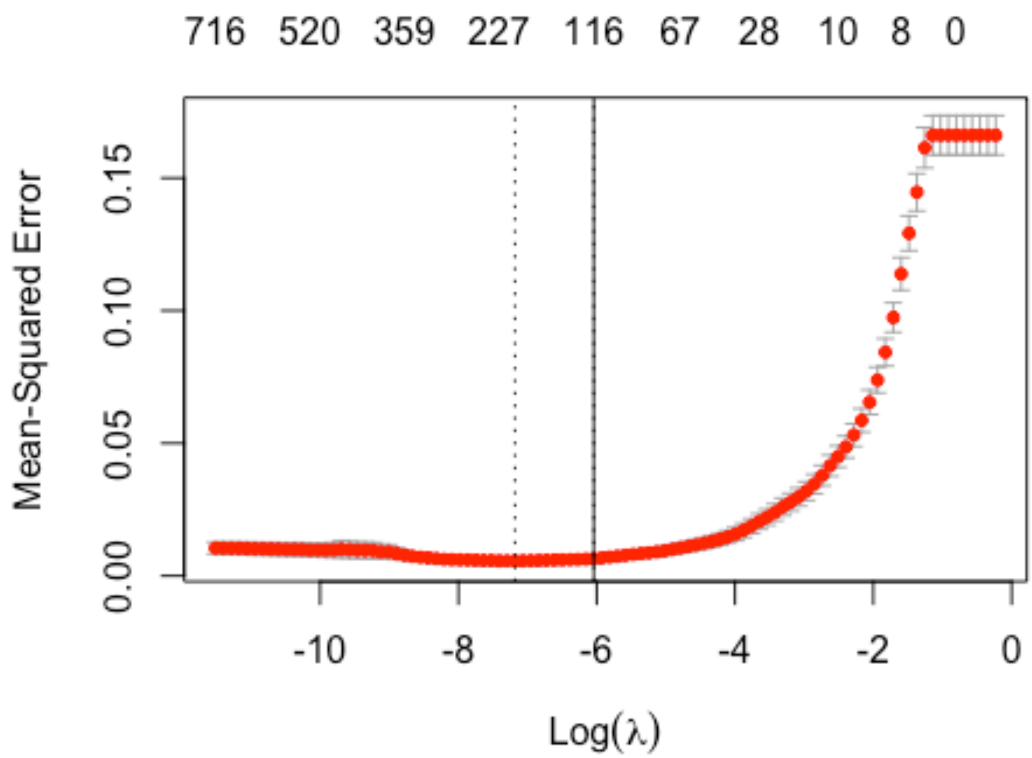
```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

## Loaded glmnet 4.1-4

set.seed(1000)

lasso.cv1 <- cv.glmnet(data.matrix(trainData[,-1]), trainData$y,
                       lambda = 10^seq(-5, -0.1, length.out = 10),
                       alpha=1, standardize=TRUE)
lasso.cv2 <- cv.glmnet(data.matrix(trainData[,-1]), trainData$y,
                       lambda = 10^seq(-5, -0.1, length.out = 40),
                       alpha=1, standardize=TRUE)
lasso.cv3 <- cv.glmnet(data.matrix(trainData[,-1]), trainData$y,
                       lambda = 10^seq(-5, -0.1, length.out = 100),
                       alpha=1, standardize=TRUE)

plot(lasso.cv1)
abline(v=log(lasso.cv1$lambda.1se))
```



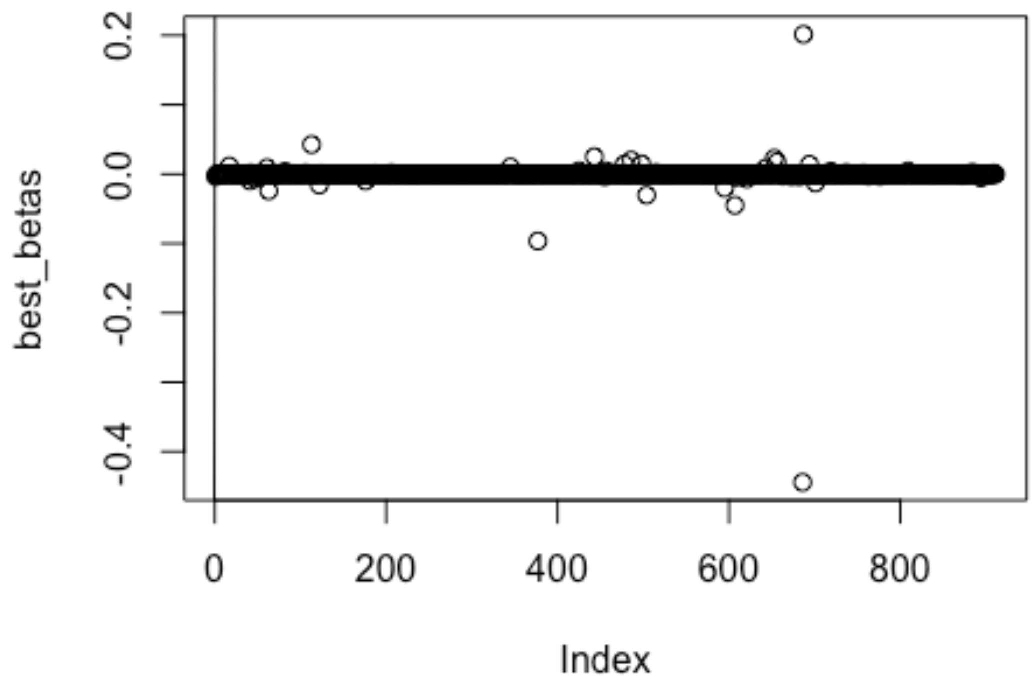

```
plot(lasso.cv2)
abline(v=log(lasso.cv2$lambda.1se))
```




```
plot(lasso.cv3)
abline(v=log(lasso.cv3$lambda.1se))
```




```
#best lambdas are in cv3 so we move forward with that model:
best_lambda <- which(lasso.cv3$lambda == lasso.cv3$lambda.1se)
best_betas <- lasso.cv3$glmnet.fit$beta[, best_lambda]
plot(best_betas)
abline(a=0,b=1)
```




```
mean(best_betas==0)

## [1] 0.8726674

lasso_model <- glmnet(data.matrix(trainData[,-1]), trainData$y, alpha = 1, la
mbda = lasso.cv3$lambda.1se)

lasso_prediction <- predict(lasso_model, s = lasso.cv3$lambda.1se, newx = dat
a.matrix(testData[,-1]), type="response")


lasso_test_table <-  table(predicted = round(lasso_prediction), actual = (tes
tData[,1]))
lasso_test_table

##          actual
## predicted   0   1
##         0  40   0
##         1   3 115

lasso_test_table_conf_mat <-  confusionMatrix(lasso_test_table, positive = "1
")
lasso_test_table_conf_mat
```

```
## Confusion Matrix and Statistics
##
##          actual
## predicted   0   1
##         0  40   0
##         1   3 115
##
##                Accuracy : 0.981
##                  95% CI : (0.9455, 0.9961)
##     No Information Rate : 0.7278
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.951
##
##  Mcnemar's Test P-Value : 0.2482
##
##             Sensitivity : 1.0000
##             Specificity : 0.9302
##          Pos Pred Value : 0.9746
##          Neg Pred Value : 1.0000
##              Prevalence : 0.7278
##          Detection Rate : 0.7278
##    Detection Prevalence : 0.7468
##       Balanced Accuracy : 0.9651
##
##        'Positive' Class : 1
##
```

```
lasso_classification <- 1-lasso_test_table_conf_mat$overall["Accuracy"][[1]]


lasso_precision <- lasso_test_table[2,2]/(lasso_test_table[2,1]+lasso_test_table[2,2])


lasso_recall <- lasso_test_table[2,2]/(lasso_test_table[1,2]+lasso_test_table[2,2])

lasso_sens<- data.frame(lasso_test_table_conf_mat[4])["Sensitivity",]


lasso_metrics <- data.frame(model="lasso", classification_error=lasso_classification, precision=lasso_precision, recall=lasso_recall, sensitivity=lasso_sens)
lasso_metrics
```

```
##   model classification_error precision recall sensitivity
## 1 lasso           0.01898734 0.9745763      1           1
```

Looks like a good fit model! But when we look at the confusion matrix data detection rate is low which means that measure ability to actually detect the diverse groups is low. Although the four metrics we use to evaluate the efficiency of the model are perfect, low detection rate does indicate that this model might not be the best fitting model. Yet, we can conclude that the top features selected by this model can be the most significant features or indicators on the development level of a country and the number of significant predictor variables selected is the ideal.

Since the Lasso model is very good at selecting the significant coefficients I will look into all of them and utilize them to compare with the other models' selected indicators/features:

```
best_betas_df <- data.frame(codes_betas=names(best_betas), best_betas)

sorted_best_betas_df <- best_betas_df[order(-best_betas),]

lasso_betas_df <- sorted_best_betas_df[!sorted_best_betas_df$best_betas==0,]
dim(lasso_betas_df)#there are only 116 significant indicator variables on dev
elopment level of country

## [1] 116   2

top_codes_lasso <- lasso_betas_df[,1]
top_codes_lasso[top_codes_lasso=="Year"]#year is significant for Lasso

## [1] "Year"

important_indicators_lasso <- unique(indicator_names[indicator_names$Indicato
r.Code %in% top_codes_lasso,])[,1]
```

Logistic Regression with All Features(no scaling nor tuning-raw model)

```
set.seed(100)

log_reg_model <- glm(y~., data=trainData, family=binomial(link="logit"))

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

log_reg_model_prediction <- predict(log_reg_model, newdata=testData, type="re
sponse")

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type
== :
## prediction from a rank-deficient fit may be misleading

#significant indicators:
model_summary <- summary(log_reg_model)
```

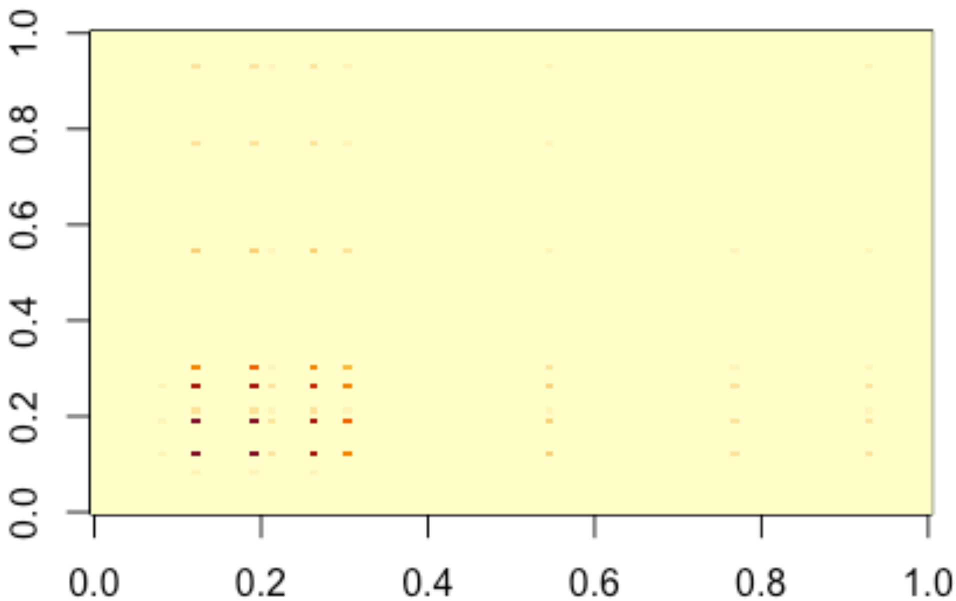
```
modl_summ_df <- data.frame(model_summary$coefficients)
sum((modl_summ_df$Pr...z..)<0.05)#in this model 634 of the beta values are si
gnificant as thy have a p-value less than alpha=0.05

## [1] 634

sorted_modl_summ_df <- modl_summ_df[order(modl_summ_df$Pr...z..),]
#I will select top 116 indicators to be able to compare the most significant
116 indicators with the other models and try to come up with a conclusion on
what indicators contribute mostly to the development of an economy, I picked
116 because Lasso determined 116 indicators to be significant
log_reg_top_indicators <- unique(indicator_names[indicator_names$Indicator.Co
de %in% rownames(sorted_modl_summ_df[1:116,],),])[,1]
log_reg_top_indicators <- c("Year",log_reg_top_indicators)#year is also signi
ficant when model summary is evaluated

#analysis of model:
log_reg_model_prediction_rounded <- round(log_reg_model_prediction)
table(log_reg_model_prediction_rounded)

## log_reg_model_prediction_rounded
##  0  1
## 83 75

log_reg_test_table <-  table(predicted = log_reg_model_prediction_rounded, ac
tual = testData$y)

log_reg_test_con_mat <-  confusionMatrix(log_reg_test_table, positive = "1")
log_reg_test_con_mat

## Confusion Matrix and Statistics
##
##          actual
## predicted  0  1
##         0 20 63
##         1 23 52
##
##                Accuracy : 0.4557
##                  95% CI : (0.3764, 0.5367)
##     No Information Rate : 0.7278
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : -0.0641
##
##  Mcnemar's Test P-Value : 2.605e-05
##
##             Sensitivity : 0.4522
##             Specificity : 0.4651
##          Pos Pred Value : 0.6933
##          Neg Pred Value : 0.2410
##              Prevalence : 0.7278
```

```
##          Detection Rate : 0.3291
##    Detection Prevalence : 0.4747
##       Balanced Accuracy : 0.4586
##
##        'Positive' Class : 1
##
```

```
log_reg_classification_error <- 1-log_reg_test_con_mat$overall["Accuracy"][[1
]]



log_reg_precision <- log_reg_test_table[2,2]/(log_reg_test_table[2,1]+log_reg
_test_table[2,2])



log_reg_recall <- log_reg_test_table[2,2]/(log_reg_test_table[1,2]+log_reg_te
st_table[2,2])

log_reg_sens <- data.frame(log_reg_test_con_mat[4])["Sensitivity",]


log_reg_metrics <- data.frame(model="logistic regression", classification_err
or=log_reg_classification_error, precision=log_reg_precision, recall=log_reg_
recall, sensitivity=log_reg_sens)
log_reg_metrics
```

```
##                 model classification_error precision    recall sensitivity
## 1 logistic regression            0.5443038 0.6933333 0.4521739   0.4521739
```

```
image(cov(sample(trainData, 100)))#it is hard to visualize collinearity among
all data so when we sample 100 we can see that there is collinearity in the d
ata and thus we should utilize PCA to overcome this handicap
```

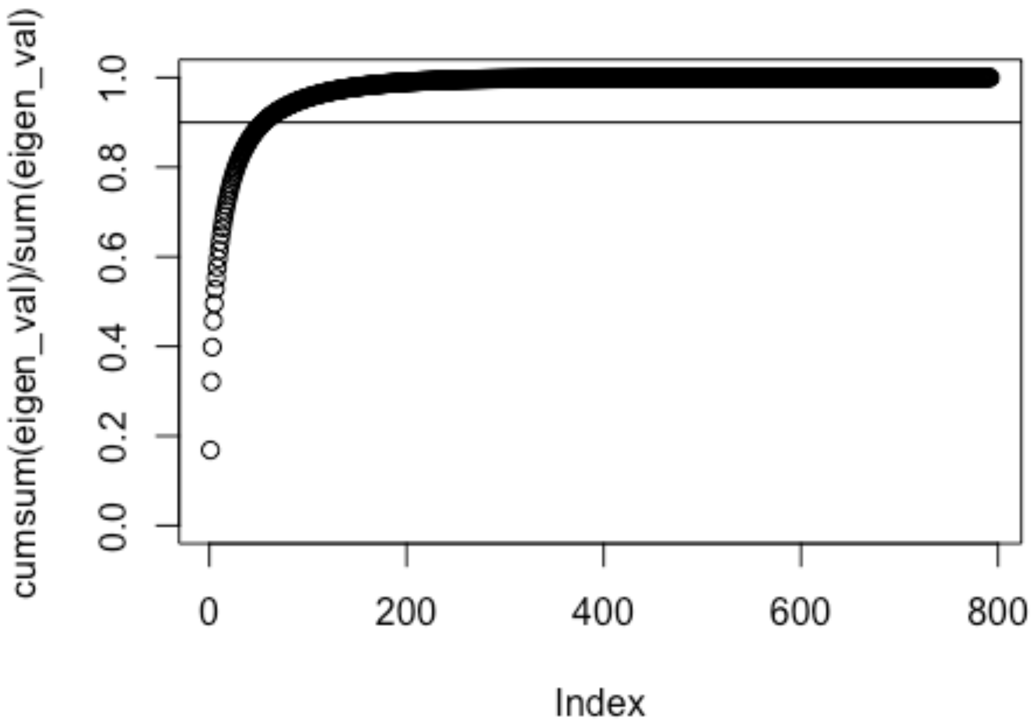


In this model of logistic regression where all features are included the model summary shows us that none of the beta values are significant. This might be because there are too many predictor variables that are collinear or interacting intrinsically with each other. The metrics suggest that the model fits poorly. The model is predicting less accurately and the predictors have less significance compared to previous model which can also be a repercussion of the median imputation method I used. For a better model, we need feature selection so I will use PCA and logistic regression together next to determine the significant features.

#PCA + Logistic Regression for Feature Selection and Model Fitting(scaled)

I will use PCA to move forward in selecting the most significant features.

```
set.seed(1000)
pca_out <- prcomp(OECD_21c[,-c(1, 2)], center=TRUE, scale=TRUE)
eigen_val <- pca_out$sdev^2
plot(cumsum(eigen_val) / sum(eigen_val),
     ylim=c(0, 1))
abline(h=.9)
```




```
#I pick the k to be 50 because the kink happens at that point and the variabi
lity is lower after 50:
k <- 50

W <- pca_out$x[, 1:k]
df_w <- data.frame(y=OECD_21c$y, W)
```

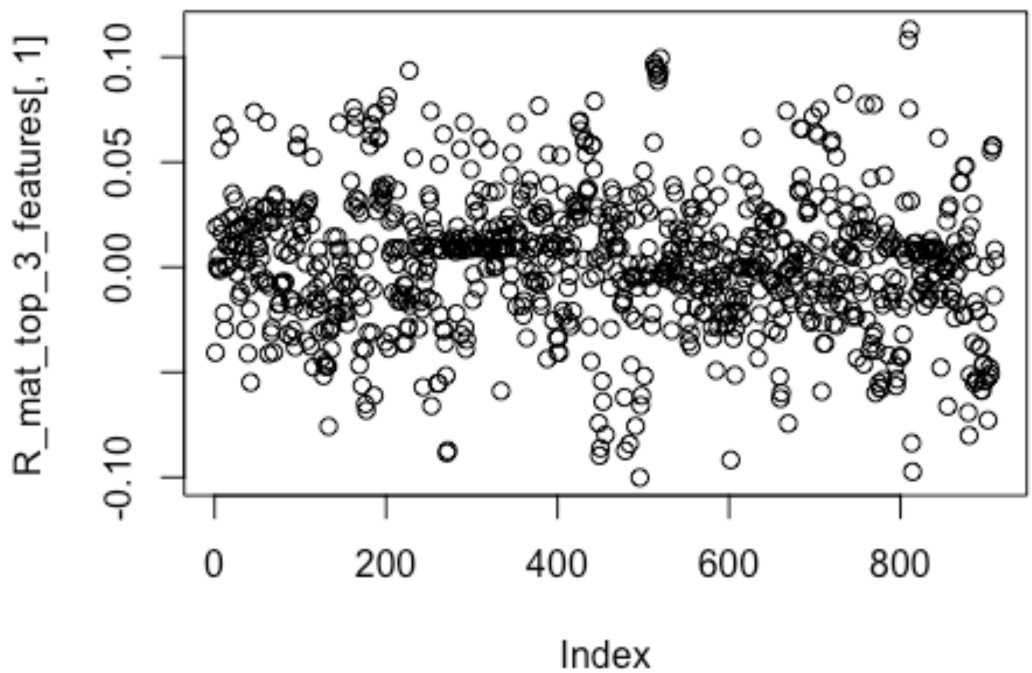
Separating Train and Test Data for PCA

For test data I will get the 20% of all data and the rest will be train data

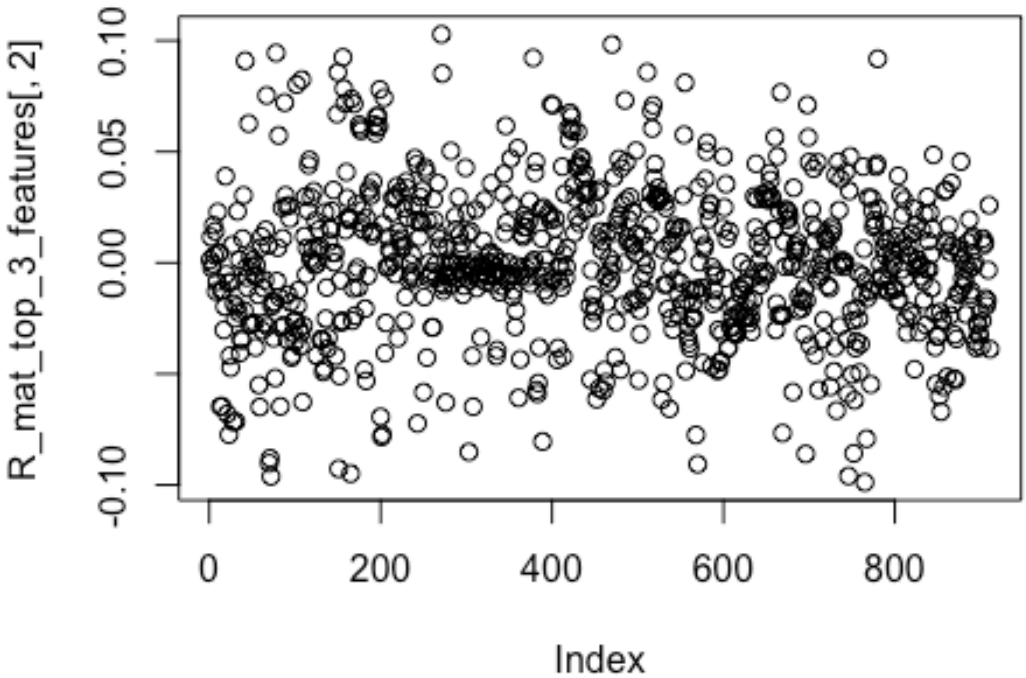
```
library(caret)

set.seed(1000)

#getting the training  dataset
trainData_W <- df_w[trainRowNumbers,]

#getting test dataset
testData_W <- OECD_21c[-trainRowNumbers,]
```

##PCA+Logistic Regression

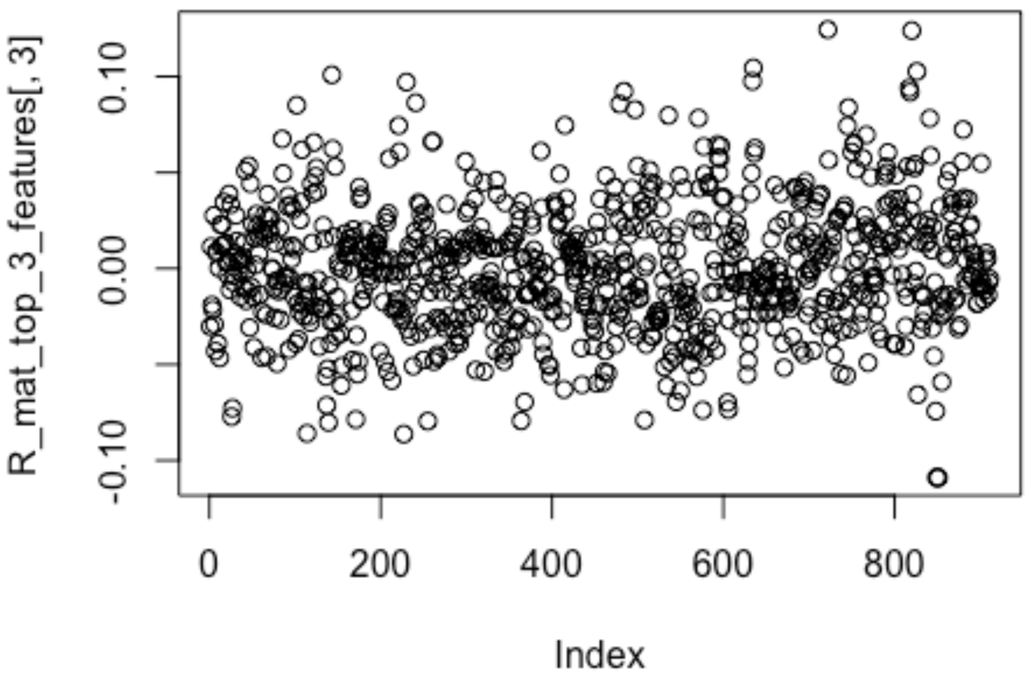
```r
set.seed(1000)

pca_model <- glm(y ~ ., data=trainData_W, family=binomial(link="logit"))

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

pca_model_prediction <- predict(pca_out, newdata=testData_W, type="response")

## Warning: In predict.prcomp(pca_out, newdata = testData_W, type = "response
") :
##  extra argument 'type' will be disregarded

pca_model_prediction_df <- as.data.frame(pca_model_prediction)

#select the first k components
pca_model_prediction_df_selected <- pca_model_prediction_df[,1:k]

#make prediction on test data
pca_prediction_final <- predict(pca_model, pca_model_prediction_df_selected,
type="response")

glm_summary <- summary(pca_model)$coefficients

glm_sum_df <- data.frame(glm_summary)[-1,]

sorted_glm_sum_df <- glm_sum_df[order(glm_sum_df$Pr...z..),]

top_3_features <- rownames(sorted_glm_sum_df[2:4,])
top_3_features

## [1] "PC5"  "PC15" "PC30"

R_mat_top_3_features <- pca_out$rotation[,top_3_features]
head(R_mat_top_3_features)

##                           PC5          PC15        PC30
## Year              -0.0406243562  0.0013104412 -0.03019231
## EG.CFT.ACCS.ZS     0.0192099087  0.0115919231  0.01136224
## EG.ELC.ACCS.ZS     0.0005341528 -0.0023968641 -0.01908885
## EG.ELC.ACCS.RU.ZS -0.0006473697 -0.0004963823 -0.02155625
## EG.ELC.ACCS.UR.ZS  0.0052664777  0.0168996609  0.02762565
## SE.PRM.TENR       -0.0014808721  0.0034279582 -0.02917892

plot(R_mat_top_3_features[,1])
```




```
plot(R_mat_top_3_features[,2])
```




```
plot(R_mat_top_3_features[,3])
```




```
#I will take loadings values that are above 0.02 based on the plots I visuali
zed and only get top 2 because the third plot loadings don't provide signific
ant information

pc1_names <- names(which(R_mat_top_3_features[,1]>0.02))
pc2_names <- names(which(R_mat_top_3_features[,2]>0.02))

pca_codes <- unique(pc1_names, pc2_names)
length(pca_codes)

## [1] 232

important_indicators_pca <- unique(indicator_names[indicator_names$Indicator.
Code %in% pca_codes,])[,1]


pca_model_prediction_rounded <- round(pca_prediction_final)
table(pca_model_prediction_rounded)

## pca_model_prediction_rounded
##   0   1
##  42 116
```

```
pca_test_table <-  table(predicted = pca_model_prediction_rounded, actual = t
estData$y)

pca_test_con_mat <-  confusionMatrix(pca_test_table, positive = "1")
pca_test_con_mat

## Confusion Matrix and Statistics
##
##          actual
## predicted   0   1
##         0  41   1
##         1   2 114
##
##                Accuracy : 0.981
##                  95% CI : (0.9455, 0.9961)
##     No Information Rate : 0.7278
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9517
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9913
##             Specificity : 0.9535
##          Pos Pred Value : 0.9828
##          Neg Pred Value : 0.9762
##              Prevalence : 0.7278
##          Detection Rate : 0.7215
##    Detection Prevalence : 0.7342
##       Balanced Accuracy : 0.9724
##
##        'Positive' Class : 1
##

pca_classification_error <- 1-pca_test_con_mat$overall["Accuracy"][[1]]


pca_precision <- pca_test_table[2,2]/(pca_test_table[2,1]+pca_test_table[2,2]
)

pca_recall <- pca_test_table[2,2]/(pca_test_table[1,2]+pca_test_table[2,2])

pca_sens <- data.frame(pca_test_con_mat[4])["Sensitivity",]

pca_log_reg_metrics <- data.frame(model="pca + logistic regression", classifi
cation_error=pca_classification_error, precision=pca_precision, recall=pca_re
call, sensitivity=pca_sens)
pca_log_reg_metrics
```

```
##                         model classification_error precision    recall
## 1 pca + logistic regression           0.01898734 0.9827586 0.9913043
##   sensitivity
## 1   0.9913043
```

The PCA + logistic regression can detect a signal between our "W" and Y and it has a lower classification error compared to logistic regression model. The model has selected 232 features to be indicative of development of a country which is more than the number of indicators lasso had selected. The model can't avoid random noise as we get the necessary features list to be very long. I will try out other algorithms to compare which is the best fitting.

NaiveBayes(tuned)

```
library(e1071)
library(nnet)
library(tidyverse)

## ── Attaching packages ─────────────────────────────────────── tidyverse 1.
3.1 ──

## ✓ tibble  3.1.6     ✓ dplyr   1.0.8
## ✓ readr   2.1.2     ✓ stringr 1.4.0
## ✓ purrr   0.3.4     ✓ forcats 0.5.1

## ── Conflicts ────────────────────────────────────────── tidyverse_conflict
s() ──
## x Matrix::expand() masks tidyr::expand()
## x dplyr::filter()  masks stats::filter()
## x dplyr::lag()     masks stats::lag()
## x purrr::lift()    masks caret::lift()
## x Matrix::pack()   masks tidyr::pack()
## x Matrix::unpack() masks tidyr::unpack()

library(caret)

set.seed(1000)
tune.control <- tune.control(random =F, nrepeat=1, repeat.aggregate=min,sampl
ing=c("cross"),sampling.aggregate=mean, cross=10, best.model=T, performances=
T)


NB_model <- naiveBayes(y ~ ., trainData, tune.control)

NB_model_pred <- predict(NB_model, testData)


conf_mat_NB <- table(actual=testData$y, prediction=NB_model_pred)
conf_mat_NB
```

```
##       prediction
## actual   0   1
##      0  34   9
##      1   0 115
```

```r
conf_mat_NB_results <- confusionMatrix(conf_mat_NB)
conf_mat_NB_results
```

```
## Confusion Matrix and Statistics
## 
##       prediction
## actual   0   1
##      0  34   9
##      1   0 115
## 
##                Accuracy : 0.943
##                  95% CI : (0.8946, 0.9736)
##     No Information Rate : 0.7848
##     P-Value [Acc > NIR] : 3.518e-08
## 
##                   Kappa : 0.8461
## 
##  Mcnemar's Test P-Value : 0.007661
## 
##             Sensitivity : 1.0000
##             Specificity : 0.9274
##          Pos Pred Value : 0.7907
##          Neg Pred Value : 1.0000
##              Prevalence : 0.2152
##          Detection Rate : 0.2152
##    Detection Prevalence : 0.2722
##       Balanced Accuracy : 0.9637
## 
##        'Positive' Class : 0
## 
```

```r
NB_classification_error <- 1-conf_mat_NB_results$overall["Accuracy"][[1]]

NB_precision <- conf_mat_NB[2,2]/(conf_mat_NB[2,1]+conf_mat_NB[2,2])

NB_recall <- conf_mat_NB[2,2]/(conf_mat_NB[1,2]+conf_mat_NB[2,2])

NB_sens <- data.frame(conf_mat_NB_results[4])["Sensitivity",]


NB_metrics <- data.frame(model="NaiveBayes", classification_error=NB_classifi
cation_error, precision=NB_precision, recall=NB_recall, sensitivity=NB_sens)
NB_metrics
```

```
##        model classification_error precision    recall sensitivity
## 1 NaiveBayes           0.05696203         1 0.9274194           1
```

RandomForest(tuned)

```
set.seed(1000)
trainData$y <- as.factor(trainData$y)
levels(trainData$y) <- c("zero", "one")

fitControl <- trainControl(method = 'cv', number = 5, savePredictions = 'fina
l', classProbs = T, summaryFunction=twoClassSummary)


model_rf <- train(y ~ ., data=trainData, method='rf', tuneLength=5, trControl
= fitControl)

## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy" wa
s not
## in the result set. ROC will be used instead.

model_rf

## Random Forest
##
## 634 samples
## 911 predictors
##   2 classes: 'zero', 'one'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 506, 507, 508, 508, 507
## Resampling results across tuning parameters:
##
##   mtry  ROC        Sens       Spec
##     2   0.9997037  0.9552707  1
##     9   0.9996296  0.9851852  1
##    42   0.9993704  0.9851852  1
##   197   0.9980000  0.9851852  1
##   911   0.9970370  0.9698006  1
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.

rf_pred <- predict(model_rf, testData)

conf_mat_rf <- table(actual=testData$y, prediction=ifelse(rf_pred=="one", 1,0
))
conf_mat_rf

##       prediction
## actual   0   1
```

```
##      0  38   5
##      1   0 115
```

```r
conf_mat_rf_results <- confusionMatrix(conf_mat_rf)
conf_mat_rf_results
```

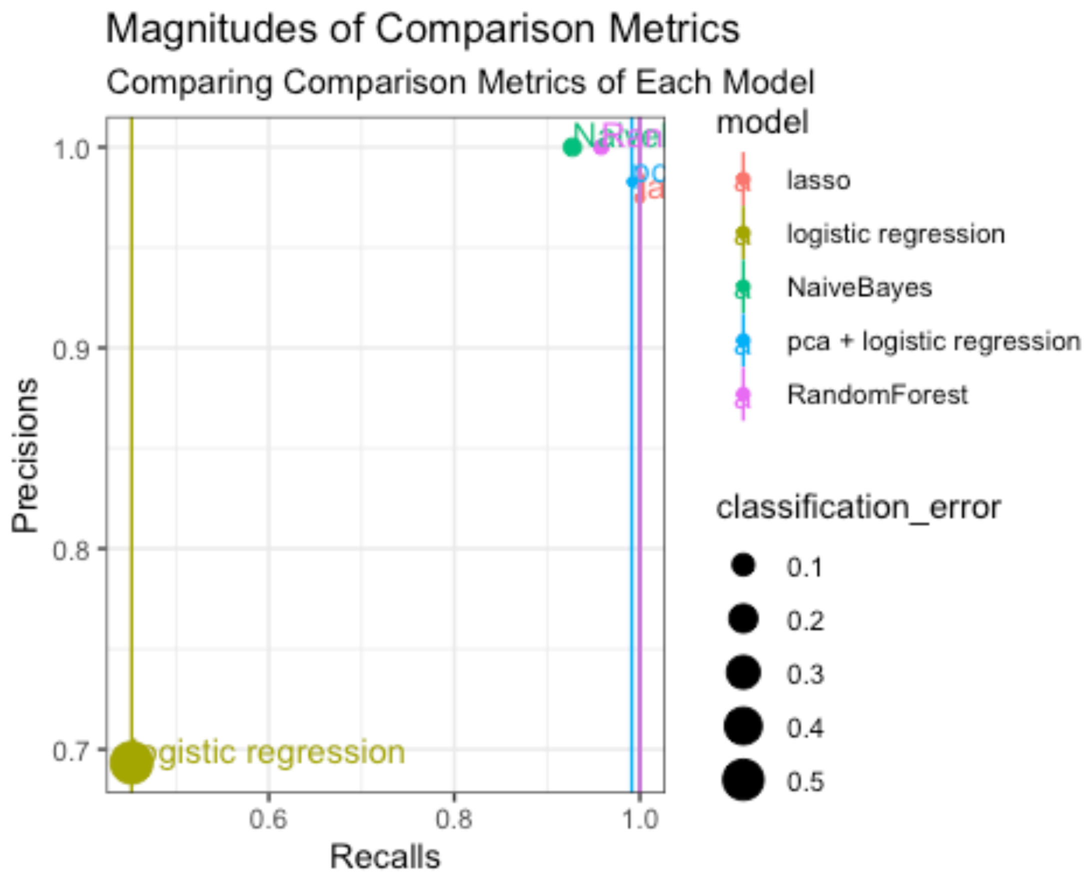
```
## Confusion Matrix and Statistics
##
##       prediction
## actual   0   1
##      0  38   5
##      1   0 115
##
##                Accuracy : 0.9684
##                  95% CI : (0.9277, 0.9896)
##     No Information Rate : 0.7595
##     P-Value [Acc > NIR] : 3.615e-13
##
##                   Kappa : 0.9171
##
##  Mcnemar's Test P-Value : 0.07364
##
##             Sensitivity : 1.0000
##             Specificity : 0.9583
##          Pos Pred Value : 0.8837
##          Neg Pred Value : 1.0000
##              Prevalence : 0.2405
##          Detection Rate : 0.2405
##    Detection Prevalence : 0.2722
##       Balanced Accuracy : 0.9792
##
##        'Positive' Class : 0
##
```

```r
rf_classification_error <- 1-conf_mat_rf_results$overall["Accuracy"][[1]]

rf_precision <- conf_mat_rf[2,2]/(conf_mat_rf[2,1]+conf_mat_rf[2,2])

rf_recall <- conf_mat_rf[2,2]/(conf_mat_rf[1,2]+conf_mat_rf[2,2])

rf_sens <- data.frame(conf_mat_rf_results[4])["Sensitivity",]


rf_metrics <- data.frame(model="RandomForest", classification_error=rf_classi
fication_error, precision=rf_precision, recall=rf_recall, sensitivity=rf_sens
)
rf_metrics
```

```
##          model classification_error precision    recall sensitivity
## 1 RandomForest           0.03164557         1 0.9583333           1
```

Comparing the Metrics

```
metrics_comparison_df <- rbind(lasso_metrics, log_reg_metrics, pca_log_reg_me
trics, NB_metrics, rf_metrics)
metrics_comparison_df

##                      model classification_error precision    recall
## 1                    lasso           0.01898734 0.9745763 1.0000000
## 2      logistic regression           0.54430380 0.6933333 0.4521739
## 3 pca + logistic regression          0.01898734 0.9827586 0.9913043
## 4               NaiveBayes           0.05696203 1.0000000 0.9274194
## 5             RandomForest           0.03164557 1.0000000 0.9583333
##   sensitivity
## 1   1.0000000
## 2   0.4521739
## 3   0.9913043
## 4   1.0000000
## 5   1.0000000

theme_set(theme_bw())
gg <- ggplot(metrics_comparison_df, aes(x=recall, y=precision, color=model, g
roup=model)) +
  geom_point(aes(size=classification_error)) +
  labs(subtitle="Comparing Comparison Metrics of Each Model",
       y="Precisions",
       x="Recalls",
       title="Magnitudes of Comparison Metrics") + geom_vline(aes(group=model
, color=model, xintercept = sensitivity)) + geom_text(aes(label=model ,hjust=
0,vjust=0))


plot(gg)
```



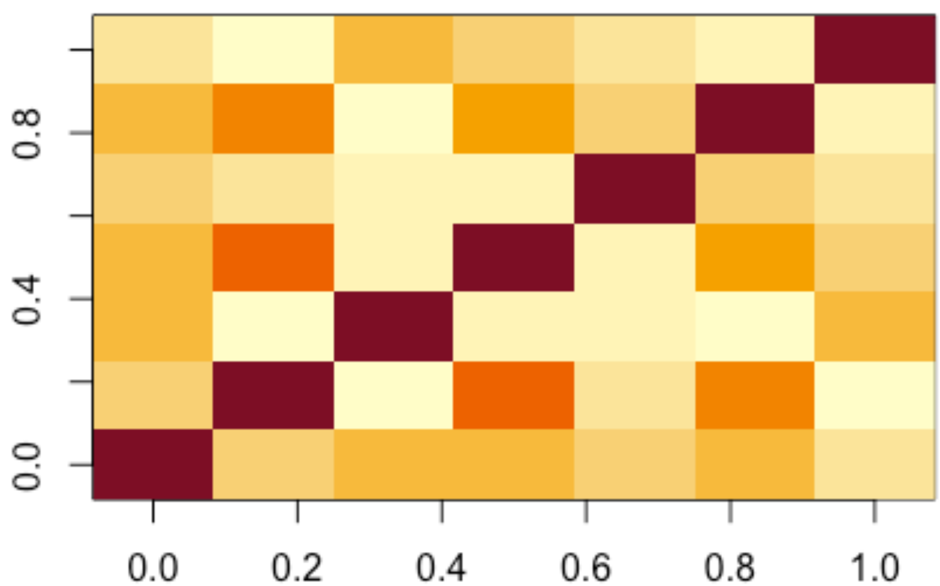
In this metrics comparison graph we see that scaled lasso and tuned random forest and naive bayes have the same sensitivity which is 1. So for creating the best fit model with significant indicators from the financial crisis year 2008 I will get the significant indicators determined by the lasso model and fit my model with random forest because lasso has highest recall and random forest has highest precision with less classification error. I will also engineer a new feature utilizing the significant indicators that were selected by all the models and include it in this final model. To asses the significance of my engineered feature I will conduct a chi-square test using anova() for it by fitting a logistic regression with the most significant indicators that were in all models.

Feature Engineering

```
set.seed(1000)
library(corrplot)

## corrplot 0.92 loaded

joint_indicators <- important_indicators_lasso[important_indicators_lasso %in
% log_reg_top_indicators]

joint_indicators <- joint_indicators[joint_indicators %in% important_indicato
rs_pca]

joint_indicators
```

```
## [1] "Adjusted savings: carbon dioxide damage (% of GNI)"
## [2] "Agricultural land (sq. km)"
## [3] "Agricultural raw materials imports (% of merchandise imports)"
## [4] "Arable land (hectares per person)"
## [5] "Bank capital to assets ratio (%)"
## [6] "Coal rents (% of GDP)"
## [7] "Combustible renewables and waste (% of total energy)"
```

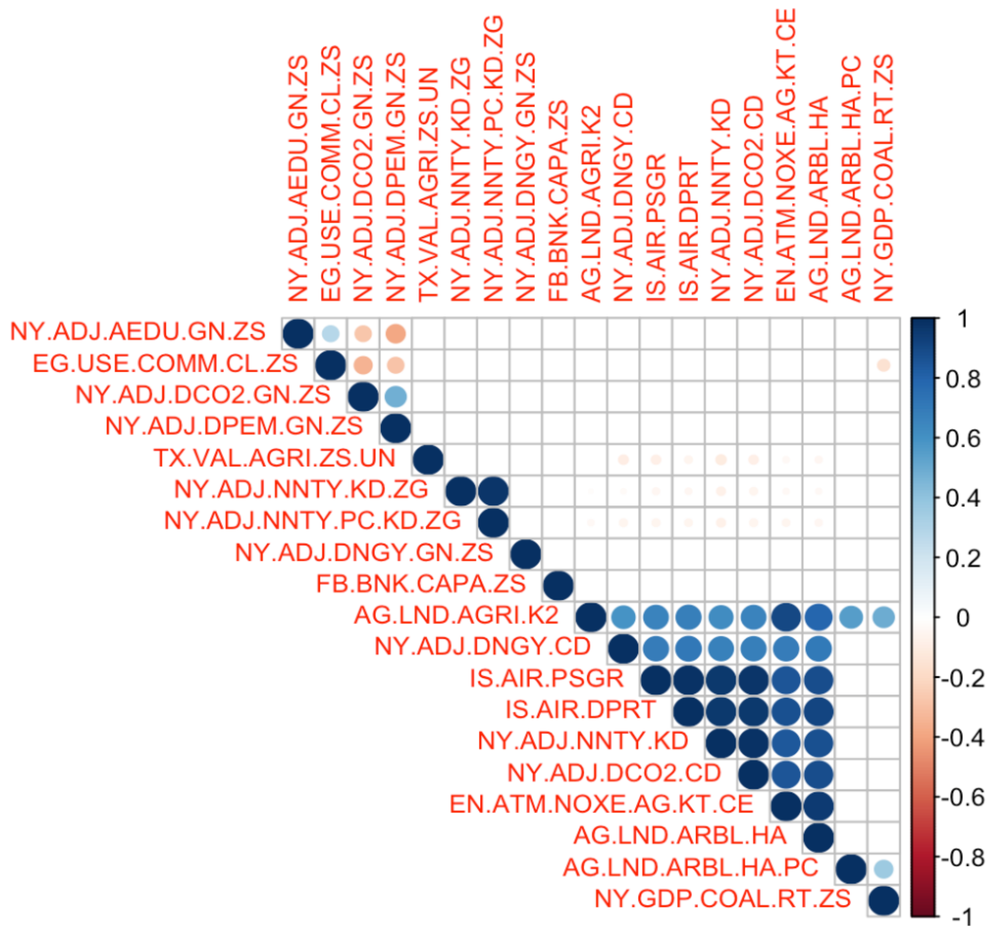
```
joint_ind_codes <- unique(indicator_names[indicator_names$Indicator.Name%in%
joint_indicators,])[,2]

joint_features_matrix <- data.matrix(OECD_21c[,joint_ind_codes])

joint_feature_corr_matrix <- cor(joint_features_matrix)
image(joint_feature_corr_matrix)#there is correlation
```




```
corrplot(joint_feature_corr_matrix, method="circle")
```

```
cor.mtest = function(mat, ...) {
  mat = as.matrix(mat)
  n = ncol(mat)
  p.mat = lowCI.mat = uppCI.mat = matrix(NA, n, n)
```

```r
  diag(p.mat) = 0
  diag(lowCI.mat) = diag(uppCI.mat) = 1
  for (i in 1:(n - 1)) {
    for (j in (i + 1):n) {

      tmp = cor.test(x = mat[, i], y = mat[, j], ...)
      p.mat[i, j] = p.mat[j, i] = tmp$p.value

      ## only 'pearson' method provides confidence intervals
      if (!is.null(tmp$conf.int)) {
        lowCI.mat[i, j] = lowCI.mat[j, i] = tmp$conf.int[1]
        uppCI.mat[i, j] = uppCI.mat[j, i] = tmp$conf.int[2]
      }
    }
  }

  colnames(p.mat) = rownames(p.mat) = colnames(mat)
  colnames(lowCI.mat) = rownames(lowCI.mat) = colnames(mat)
  colnames(uppCI.mat) = rownames(uppCI.mat) = colnames(mat)

  list(
    p = p.mat,
    lowCI = lowCI.mat,
    uppCI = uppCI.mat
  )
}
## matrix of the p-value of the correlation
significance_test1 <- cor.mtest(joint_feature_corr_matrix)

corrplot(joint_feature_corr_matrix, type="upper", p.mat = significance_test1$
p, insig='blank', sig.level = 0.05, order = 'hclust', tl.cex=0.8)
```




```
unique(indicator_names[indicator_names$Indicator.Code%in%joint_ind_codes,])

##                                                        Indicator.Name
## 31             Adjusted savings: carbon dioxide damage (% of GNI)
## 60                                        Agricultural land (sq. km)
## 68  Agricultural raw materials imports (% of merchandise imports)
## 91                                Arable land (hectares per person)
## 109                                Bank capital to assets ratio (%)
## 205                                           Coal rents (% of GDP)
## 206           Combustible renewables and waste (% of total energy)
##         Indicator.Code
## 31  NY.ADJ.DCO2.GN.ZS
## 60     AG.LND.AGRI.K2
## 68  TM.VAL.AGRI.ZS.UN
## 91  AG.LND.ARBL.HA.PC
## 109    FB.BNK.CAPA.ZS
## 205 NY.GDP.COAL.RT.ZS
## 206    EG.USE.CRNW.ZS

#unique(indicator_names[indicator_names$Indicator.Code%in%c("IS.AIR.PSGR", "I
S.AIR.DPRT", "NY.ADJ.NNTY.KD", "NY.ADJ.DCO2.CD", "NY.ADJ.NNTY.KD.ZG", "NY.ADJ
```

```
.NNTY.PC.KD.ZG","AG.LND.AGRI.K2","EN.ATM.NOXE.AG.KT.CE", "NY.ADJ.AEDU.GN.ZS",
"NY.ADJ.DPEM.GN.ZS", "NY.ADJ.DCO2.GN.ZS"),])
```

Based on the correlation plot "IS.AIR.PSGR"(Air transport, passengers carried), "IS.AIR.DPRT"(Air transport, registered carrier departures worldwide), "NY.ADJ.NNTY.KD"(Adjusted net national income (constant 2015 US)), and "NY.ADJ.DCO2.CD"(Adjusted savings: carbon dioxide damage (current US)) are all highly positively correlated.

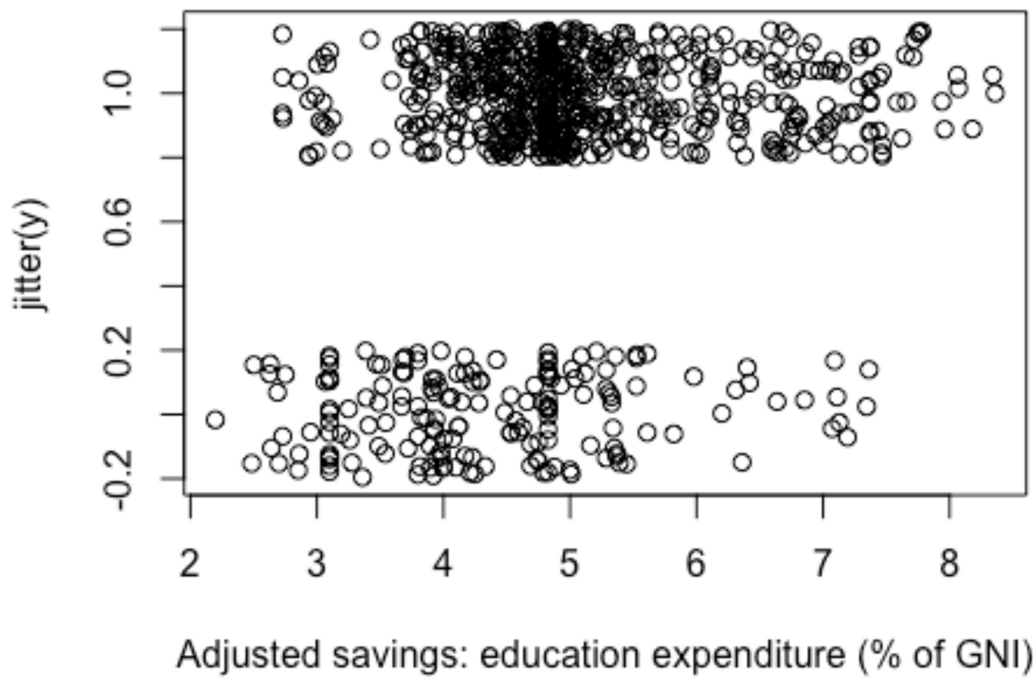
"NY.ADJ.NNTY.KD.ZG"(Adjusted net national income (annual % growth)) is highly positively correlated with "NY.ADJ.NNTY.PC.KD.ZG". (Adjusted net national income per capita (annual % growth))

"AG.LND.AGRI.K2"(Agricultural land (sq. km)) is highly positively correlated with "EN.ATM.NOXE.AG.KT.CE"(Agricultural nitrous oxide emissions (thousand metric tons of CO2 equivalent)).

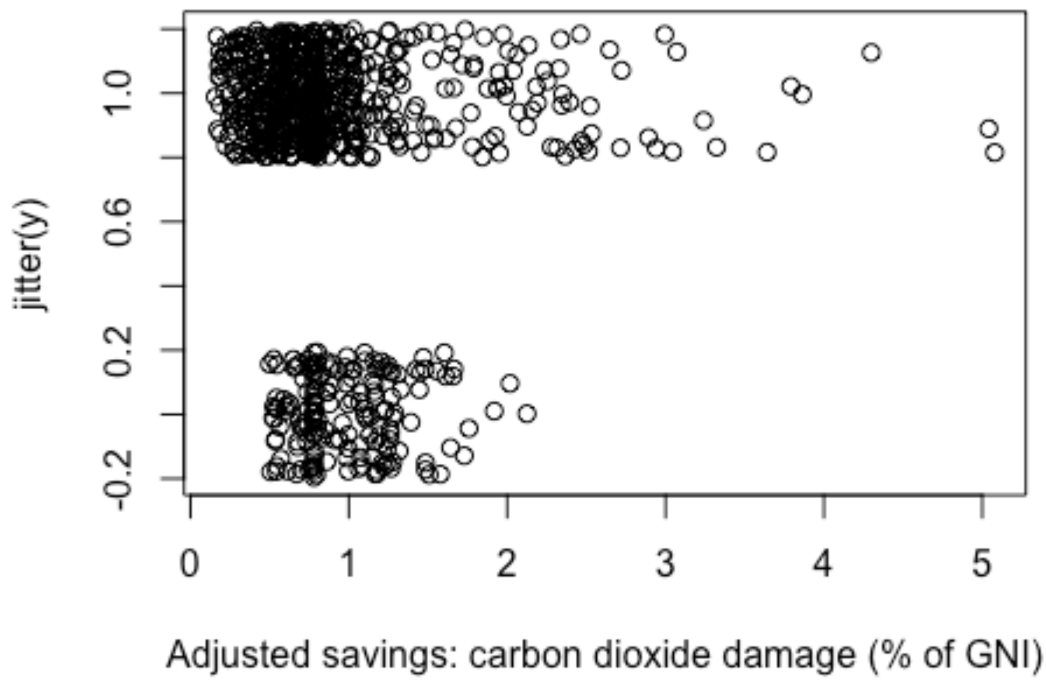
"NY.ADJ.AEDU.GN.ZS"(Adjusted savings: education expenditure (% of GNI)) is negatively significantly correlated with "NY.ADJ.DPEM.GN.ZS"(Adjusted savings: particulate emission damage (% of GNI)) and "NY.ADJ.DCO2.GN.ZS"(Adjusted savings: carbon dioxide damage (% of GNI))

One of the most interesting correlation among the most significant indicators, in my opinion, is the negative correlation betwee education expenditure and cost of carbon dioxide damage. Below are the visualizations:

```
plot(jitter(y)~jitter(NY.ADJ.AEDU.GN.ZS), data=OECD_21c, xlab="Adjusted savin
gs: education expenditure (% of GNI)")
```




```
plot(jitter(y)~jitter(NY.ADJ.DCO2.GN.ZS), data=OECD_21c, xlab="Adjusted savin
gs: carbon dioxide damage (% of GNI)")
```



The graphs indicate that for developed countries the rate of education expenditure is high and cost of carbon dioxide damage is low while it is the opposite in developing countries. The visualizations don't provide explicit data on effect on education and especially female education so I am going to engineer a feature based on female education indicators to account for the effect of education among the most significant indicators. The importance of education is obvious from this analysis and could be represented as another variable.

```
education_indicators <- rbind(unique(indicator_names[grepl("school",indicator_names$Indicator.Name),]), unique(indicator_names[grepl("education",indicator_names$Indicator.Name),]))

#from the indicators that specify data about female education and schooling I am selecting the ones below:
education_ind_names <- c("Share of youth not in education, employment or training, female (% of female youth population)", "Primary education, pupils (% female)", "Labor force with advanced education, female (% of female working-age population with advanced education)", "Compulsory education, duration (years)")
#I will normalize between 0 and 1 each indicator(except compulsory education years) then I will multiply all indicator data to generate a females education score that will be added to the best fit model I will generate

education_female_cols <- unique(indicator_names[indicator_names$Indicator.Nam
```

```
e%in%education_ind_names,])[,2]

col1 <- OECD_21c[,education_female_cols[1]]
col2 <- (OECD_21c[,education_female_cols[2]] - min(OECD_21c[,education_female
_cols[2]])) / (max(OECD_21c[,education_female_cols[2]]) - min(OECD_21c[,educa
tion_female_cols[2]]))
col3 <- (OECD_21c[,education_female_cols[3]] - min(OECD_21c[,education_female
_cols[3]])) / (max(OECD_21c[,education_female_cols[3]]) - min(OECD_21c[,educa
tion_female_cols[3]]))
col4 <- (OECD_21c[,education_female_cols[4]] - min(OECD_21c[,education_female
_cols[4]])) / (max(OECD_21c[,education_female_cols[4]]) - min(OECD_21c[,educa
tion_female_cols[4]]))

female_education_index <- col1*col2*col3*col4

#Now lets test the significance of this index I engineered with anova():
#H0=the female education index I engineered is not significant--coefficient=0
#HA=the female education index I engineered is significant--coefficient!=0
female_education_index_test_df <- cbind(OECD_21c[,c("y",joint_ind_codes)],  f
emale_education_index=female_education_index)

m1 <- glm(y~., data=female_education_index_test_df, family=binomial(link="log
it"))
m0 <- glm(y~.-female_education_index, data=female_education_index_test_df, fa
mily=binomial(link="logit"))
anova(m0,m1,test="Chisq")

## Analysis of Deviance Table
##
## Model 1: y ~ (NY.ADJ.DCO2.GN.ZS + AG.LND.AGRI.K2 + TM.VAL.AGRI.ZS.UN +
##     AG.LND.ARBL.HA.PC + FB.BNK.CAPA.ZS + NY.GDP.COAL.RT.ZS +
##     EG.USE.CRNW.ZS + female_education_index) - female_education_index
## Model 2: y ~ NY.ADJ.DCO2.GN.ZS + AG.LND.AGRI.K2 + TM.VAL.AGRI.ZS.UN +
##     AG.LND.ARBL.HA.PC + FB.BNK.CAPA.ZS + NY.GDP.COAL.RT.ZS +
##     EG.USE.CRNW.ZS + female_education_index
##   Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
## 1       784     625.49
## 2       783     491.60  1    133.9 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Based on the hypothesis test the female education index is significant for the model that was created with the indicators that were selected to be significant by all models in this data mining project. The p-value of the chi-square test of significance is less than alpha=0.05 so we reject the null hypothesis. The female education index captures the accessibility and quality of female education as it entails data on females that are not taking part in any daily activity(an indicator of unfairness towards females), females access to primary education, educated females proportion in the labor force, and compulsory education duration which enables female students to attend school by law.

Final Best Fit Model

predictor variables = female education index and 116 significant features selected by lasso model only for year 2008

```
set.seed(1000)
trainData$y <- as.factor(trainData$y)
levels(trainData$y) <- c("zero", "one")

train_final <- cbind(trainData[,c("y", "Year", top_codes_lasso)], female_educ
ation_index=female_education_index[trainRowNumbers])
train_final <- train_final[train_final$Year==2008, -2]


test_final <- cbind(testData[,c("y", "Year", top_codes_lasso)], female_educat
ion_index=female_education_index[-trainRowNumbers])
test_final <- test_final[test_final$Year==2008, -2]

model_rf_final <- train(y ~ ., data=train_final, method='rf', tuneLength=5, t
rControl = fitControl)

## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy" wa
s not
## in the result set. ROC will be used instead.

model_rf_final

## Random Forest
##
##  28 samples
## 117 predictors
##   2 classes: 'zero', 'one'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 22, 21, 23, 23, 23
## Resampling results across tuning parameters:
##
##   mtry  ROC   Sens  Spec
##     2   1.00  0.7   1
##    30   0.98  0.7   1
##    59   0.94  0.7   1
##    88   0.98  0.7   1
##   117   0.92  0.7   1
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.

rf_pred_final <- predict(model_rf_final, test_final)
```

```r
conf_mat_rf_final <- table(actual=test_final$y, prediction=rf_pred_final)
colnames(conf_mat_rf_final) <- c(0,1)

conf_mat_rf_results_final <- confusionMatrix(conf_mat_rf_final)
conf_mat_rf_results_final
```

```
## Confusion Matrix and Statistics
##
##          prediction
## actual 0 1
##        0 0 2
##        1 0 6
##
##                Accuracy : 0.75
##                  95% CI : (0.3491, 0.9681)
##     No Information Rate : 1
##     P-Value [Acc > NIR] : 1.0000
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : 0.4795
##
##             Sensitivity :   NA
##             Specificity : 0.75
##          Pos Pred Value :   NA
##          Neg Pred Value :   NA
##              Prevalence : 0.00
##          Detection Rate : 0.00
##    Detection Prevalence : 0.25
##       Balanced Accuracy :   NA
##
##        'Positive' Class : 0
##
```

```r
rf_classification_error_final <- 1-conf_mat_rf_results_final$overall["Accurac
y"][[1]]

rf_precision_final <- conf_mat_rf_final[2,2]/(conf_mat_rf_final[2,1]+conf_mat
_rf_final[2,2])

rf_recall_final <- conf_mat_rf_final[2,2]/(conf_mat_rf_final[1,2]+conf_mat_rf
_final[2,2])

rf_sens_final <- data.frame(conf_mat_rf_results_final[4])["Sensitivity",]


rf_metrics_final <- data.frame(model="RandomForest_best_fit_2008", classifica
tion_error=rf_classification_error_final, precision=rf_precision_final, recal
l=rf_recall_final, sensitivity=rf_sens_final)
rf_metrics_final
```

```
##                          model classification_error precision recall sensiti
vity
## 1 RandomForest_best_fit_2008                 0.25         1   0.75
NA
```

Let's also try RandomForest without year specification since the classification error is high for only year 2008:

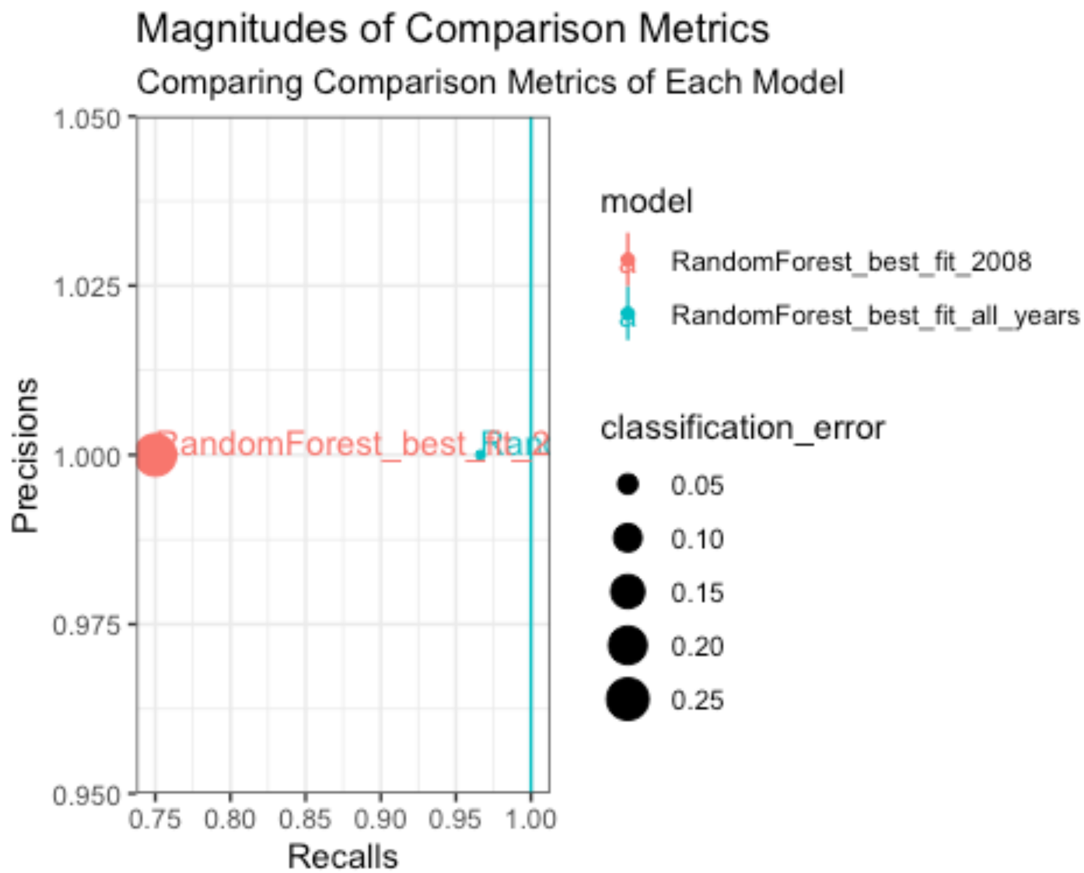
```
set.seed(1000)
trainData$y <- as.factor(trainData$y)
levels(trainData$y) <- c("zero", "one")

train_final_all_years <- cbind(trainData[,c("y", top_codes_lasso)], female_ed
ucation_index=female_education_index[trainRowNumbers])

test_final_all_years <- cbind(testData[,c("y", top_codes_lasso)], female_educ
ation_index=female_education_index[-trainRowNumbers])



model_rf_final_all_years <- train(y ~ ., data=train_final_all_years, method='
rf', tuneLength=5, trControl = fitControl)

## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy" wa
s not
## in the result set. ROC will be used instead.

model_rf_final_all_years

## Random Forest
##
## 634 samples
## 117 predictors
##   2 classes: 'zero', 'one'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 506, 507, 508, 508, 507
## Resampling results across tuning parameters:
##
##   mtry  ROC        Sens       Spec
##     2   1.0000000  0.9777778  1
##    30   0.9994815  0.9851852  1
##    59   0.9990370  0.9851852  1
##    88   0.9982593  0.9851852  1
##   117   0.9982194  0.9774929  1
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.

rf_pred_final_all_years <- predict(model_rf_final_all_years, test_final_all_y
ears)
```

```r
conf_mat_rf_final_all_years <- table(actual=test_final_all_years$y, predictio
n=ifelse(rf_pred_final_all_years=="one", 1,0))
conf_mat_rf_final_all_years

##       prediction
## actual   0   1
##      0  39   4
##      1   0 115

conf_mat_rf_results_final_all_years <- confusionMatrix(conf_mat_rf_final_all_
years)
conf_mat_rf_results_final_all_years

## Confusion Matrix and Statistics
## 
##       prediction
## actual   0   1
##      0  39   4
##      1   0 115
##                                           
##                Accuracy : 0.9747          
##                  95% CI : (0.9365, 0.9931)
##     No Information Rate : 0.7532          
##     P-Value [Acc > NIR] : 1.105e-14       
##                                           
##                   Kappa : 0.9342          
##                                           
##  Mcnemar's Test P-Value : 0.1336          
##                                           
##             Sensitivity : 1.0000          
##             Specificity : 0.9664          
##          Pos Pred Value : 0.9070          
##          Neg Pred Value : 1.0000          
##              Prevalence : 0.2468          
##          Detection Rate : 0.2468          
##    Detection Prevalence : 0.2722          
##       Balanced Accuracy : 0.9832          
##                                           
##        'Positive' Class : 0               
## 

rf_classification_error_final_all_years <- 1-conf_mat_rf_results_final_all_ye
ars$overall["Accuracy"][[1]]

rf_precision_final_all_years <- conf_mat_rf_final_all_years[2,2]/(conf_mat_rf
_final_all_years[2,1]+conf_mat_rf_final_all_years[2,2])

rf_recall_final_all_years <- conf_mat_rf_final_all_years[2,2]/(conf_mat_rf_fi
nal_all_years[1,2]+conf_mat_rf_final_all_years[2,2])
```

```
rf_sens_final_all_years <- data.frame(conf_mat_rf_results_final_all_years[4])
["Sensitivity",]


rf_metrics_final_all_years <- data.frame(model="RandomForest_best_fit_all_yea
rs", classification_error=rf_classification_error_final_all_years, precision=
rf_precision_final_all_years, recall=rf_recall_final_all_years, sensitivity=r
f_sens_final_all_years)
rf_metrics_final_all_years

##                              model classification_error precision    recall
## 1 RandomForest_best_fit_all_years           0.02531646         1 0.9663866
##   sensitivity
## 1           1
```

Comparing RandomForest Methods with Crisis Year vs All Years

```
rf_metrics_compare_df <- rbind(rf_metrics_final_all_years, rf_metrics_final)

theme_set(theme_bw())
gg_rf <- ggplot(rf_metrics_compare_df, aes(x=recall, y=precision, color=model
, group=model)) +
  geom_point(aes(size=classification_error)) +
  labs(subtitle="Comparing Comparison Metrics of Each Model",
       y="Precisions",
       x="Recalls",
       title="Magnitudes of Comparison Metrics") + geom_vline(aes(group=model
, color=model, xintercept = sensitivity)) + geom_text(aes(label=model ,hjust=
0,vjust=0))


plot(gg_rf)

## Warning: Removed 1 rows containing missing values (geom_vline).
```



I was expecting to get a better fit with specifying a crisis year because I thought the model would be a better fit if it is setup according to a specific year with specific attributes that minimize the unaccounted exogenous variables. In the two RandomForest models I fitted, I realized that the one that has data accounting for all years without a year explanatory variable is a better fitted model. I had initially thought that year had to be a significant variable that accounts for major historical, technological and political events yet after comparing the final models and the previous other models, I realized that year is not significant. Year wasn't selected by all models either since it doesn't come up when I filter all significant indicators from all models to find the overlapping indicators.

In conclusion, organizations that are interested in calculating indicators that significantly imply or address development levels of countries and the effect of each indicator on future development should focus primarily on the joint_indicators I got from all models which are:

[1] "Adjusted net national income (annual % growth)"
[2] "Adjusted net national income (constant 2015 US$)" [3] "Adjusted net national income per capita (annual % growth)" [4] "Adjusted savings: carbon dioxide damage (% of GNI)"
[5] "Adjusted savings: carbon dioxide damage (current US$)"
[6] "Adjusted savings: education expenditure (% of GNI)"
[7] "Adjusted savings: energy depletion (% of GNI)"
[8] "Adjusted savings: energy depletion (current US$)"
[9] "Adjusted savings: particulate emission damage (% of GNI)"

[10] "Agricultural land (sq. km)"
[11] "Agricultural nitrous oxide emissions (thousand metric tons of CO2 equivalent)" [12] "Agricultural raw materials exports (% of merchandise exports)"
[13] "Air transport, passengers carried"
[14] "Air transport, registered carrier departures worldwide"
[15] "Alternative and nuclear energy (% of total energy use)"
[16] "Arable land (hectares per person)"
[17] "Arable land (hectares)"
[18] "Bank capital to assets ratio (%)"
[19] "Coal rents (% of GDP)"

This list can be expanded to a total of 117 variables including the feature I generated (female education index) and the significant variables lasso model has detected. For countries that weren't included in this dataset, year or other politic indicators could be significant as these other countries won't be part of a group like OECD. The ideal dataset would be the data set where no missing values exist because in this data mining project I had to impute data which might have affected the overall performance of all models. We should also consider the positive effect the median imputation method could have caused because the metrics of the models I used were close to each other but with real data the metrics data could have been more diverse.