Application of Data Mining in Healthcare with R

Abstract

In the USA, depression as a disease has always been a subject of researchers' interest. Over the last 50-60 years, a significant number of studies from the USA have been reported exploring different aspects of this prevalent condition. In addition to the effectiveness and tolerability of various antidepressants, the different factors examined evaluation, diagnosis, depression effects, treatment-related difficulties, and depression prevention. Here, information from the USA on different facets of depression is reviewed.

In this document, we are presenting our project report for Depression Detection analysis.

Introduction

Depression is a condition that induces a constant sense of lack of motivation. This affects how you act, think, and respond and can lead to many mental and physical issues. You may have difficulty completing regular day-to-day things, and you may feel like life is not worth living sometimes. Depression is not a flaw, rather than just about the blues, and you cannot easily snap out of it. Long-term therapy could be needed for depression. With medicine, psychotherapy or both, most persons with depression can feel better. The proposed models can be used to detect if the patient is suffering from depression. The value of the feature variables will be predicting the outcome.

Dataset

The depressed dataset is taken from Kaggle. It can be found on:

https://www.kaggle.com/diegobabativa/depression

n (number of observations):

In the below-mentioned screenshot, we are trying to display the number of rows and features for the data set. colnames(my data) is used to display the feature name of the dataset. The description of the dataset:

Rows = 1429; Features = 23.

Refer to below data explanation for each feature in our dataset to better understand the dataset.

Sex: This feature contains the sex of the person whether the person is male or female.
Age: This feature determines the age of the person.

Married: This feature determines the martial status of the person.

Number children: This feature contains data related to how many children the person who is getting surveyed for this analysis have.

Education level: The feature contains the data related to the education level of the person whether he is literate or illiterate.

total members (in the family): The features contain the number of family members of the person.

Gained asset: This feature refers to the data which individual has gained or earned in his life span

durable asset: This feature contains the assets which capable of generating flows of goods and services

Save asset: This feature contains data related to the saving of every individual.

living expenses: This feature contains the data which will have the expenses which is used to run the house smoothly.

Other expenses: This feature refers to data related to individuals' monthly expenses which can be house rent, foodexpensesetc.

incomingsalary: The features contain the data which will have the salary data of the person.

Incomingownfarm: The feature determines the data for the person who has his own farm or not.

Incomingbusiness: This feature contains only that type of data if the individual who is getting surveyed over here has any type of business.

Incomingnobusiness: This feature contains only that type of data if the individual who is getting surveyed over here has no business.

incomingagricultural : The feature determines that if the person is doing the agricultural work or not.

Farmexpenses: The feature determine that the expenses related to the farms for the person.

Laborprimary: The feature contain that the person is working as a labor or not.

Lastinginvestment: This feature refers to total number of savings individuals owns.

nolastinginvestmen: This feature determines whether the person has a loan or not.

depressed: This feature is our output feature and result would be [Zero: No depressed] or [One: depressed]

The problem statement:

Predicting and classifying the patients as suffering from depression or not suffering from depression based on the factor variables given in the dataset. Using this data set we can predict the age group which has a higher risk for severe depression. These results can be used by NGOs, governments to spread awareness about depression and will also help them to keep a check on people by giving them therapies to cure depression. This highlights the need for psychiatrists / Consultants in hospitals / schools / Universities / Offices, which can help to detect an individual with depression.

Proposed Solution

To address the problem mentioned in the problem statement supervised learning models are used, the predictive model and the classification models. In any supervised learning model, the built model must go through training and testing phase.

The predictive model can be used to predict the values of depressed variable with changing values of the x variables (attribute values). To build a predictive model we initially need to have a regression model of the data set selected. For building this model we need to preprocess the data variables. Build a multiple regression model, validate the model. To get the best model we need to select those features which have the maximum effect on the depressed variable. For this predictive model we have one dependent variable which will be predicting if the patient is depressed or not. *This variable is the depressed variable.*

The independent variables for this predictive model will be the attributes or the features used by one to determine the verdict of depressed or not. The independent variables present in the dataset are – *Age, Married, Sex, Number_children, education_level, Total members, income_salary, living_expense.*

The classification models are used to classify the depressed variable into labels**; 1 (Depressed) or 0 (not Depressed)**. There are three models built. To build a classification model there is some basic preprocessing of the data we must do to make the data ideal for each model. Every model has a different way of data preprocessing. The data set is then divided into the training and testing data set (a part of supervised learning). The data set id split into 80% as training data and 20% testing data. The model is then built based on the training set.

Predictions are checked on the test data to calculate the accuracy. There are different metrics used by different models to calculate the accuracy. There is one dependent variable which will be predicting if the patient is depressed or not. This variable is the depressed variable. The independent variables for this predictive model will be the attributes or the features used by one to determine the verdict of depressed or not. The independent variables present in the dataset are – Age, Married, Sex, Number_children, education_level, Total members, income_salary, living_expense.

About the Dataset

Exploratory Data Analysis

Exploratory data analysis is used to find conspicuous patterns, spot anomalies, provides context for further research and helps in checking assumptions regarding the data set. Below are the observations we found after reading the requirements mentioned in the Deliverable Document D2.

```
> nrow(mydata)
[1] 1429
> ncol(mydata)
[1] 23
> head(mydata$depressed)
[1] 0 1 0 0 0 0
```

Libraries: Different R Libraries like cart, lattice, reshape 2 , GG plot etc. were installed to build our model.

Missing Values:
Once we started working on our data set, we found out that our data set has 20 missing values as shown in the given image below. These missing values are found in just one feature so we took the mean of that feature and replaced all the null values with the mean value of the feature.

```
Console   Terminal ×   Jobs ×
C:/Users/dharm/Desktop/Fall 20 SEM3/IS 777/Project/
[1] FALSE
> introduce(mydata)
  rows columns discrete_columns continuous_columns all_missing_columns total_missing_values complete_rows
1 1429      21                8                 13                   0                   20          1409
  total_observations memory_usage
1              30009       129528
>
```

```
>
> mean_nli <- mean(mydata$no_lasting_investmen)
> mean_nli
[1] 33133538
>
> #replacing na with mean
> for (i in 1:nrow(mydata)) {
+   if(mydata$no_lasting_investmen[i] == 0){
+     mydata$no_lasting_investmen[i] <- mean_nli
+   }
+ }
> introduce(mydata)
  rows columns discrete_columns continuous_columns all_missing_columns total_missing_values complete_rows
1 1429      21                0                 21                   0                    0          1429
  total_observations memory_usage
1              30009       131144
> |
```

Dataset after the missing values were handled can be seen in the image below.

```
C:/Users/dharm/Desktop/Fall 20 SEM3/IS 777/Project/
> # Checking if null values are removed and replaced
> introduce(mydata)
  rows columns discrete_columns continuous_columns all_missing_columns total_missing_values complete_rows
1 1429      21                8                 13                   0                    0          1429
  total_observations memory_usage
1              30009       135240
> |
```

Dependent Variable (y):

The Dependent variable for this dataset id the "Depressed" variable. This is a binary variable with values: 0 and 1, where 1 is suffering from depression and 0 being not suffering from depression.

```
> head(mydata$depressed)
[1] 0 1 0 0 0 0
> (mydata$depressed)
   [1] 0 1 0 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 1 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 1 1 1
  [51] 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 1 0 0 1 1 1 0 0 1 0 0 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 0 1 0 0
 [101] 0 0 1 0 0 0 1 1 0 0 0 0 0 0 0 0 1 1 1 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 1 0
 [151] 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [201] 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 1 0 0 0
 [251] 1 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0
 [301] 0 1 0 0 0 0 1 1 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [351] 1 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 1 0 0 0 0 0 0 1 1 0
 [401] 0 0 0 0 0 1 0 0 1 0 0 0 0 1 0 1 0 0 0 0 0 0 1 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0
 [451] 0 0 0 0 0 0 0 0 1 0 1 0 0 0 1 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 1
 [501] 0 0 0 1 0 0 1 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0 1 0
 [551] 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
 [601] 0 0 0 0 1 0 1 0 0 1 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
 [651] 0 1 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 1 0 1 1 0 1 0 0 0 0 1 0 1 0 0 0 0 0 0 1 1 0 0 1 0 0 0 0 0 0 0 0
 [701] 0 0 0 0 1 0 0 0 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1
 [751] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0
 [801] 0 0 0 1 0 0 1 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0
 [851] 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0
 [901] 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0
 [951] 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
```

Independent Variables (x):

For this dataset we have 23 feature variables which will decide the value of the target variable. There are two variables Survey and Ville_ID, these are not considered as the predictor variable as they are just a count variable. Below image depicts the predictor variables which are part of the dataset. All the feature variables are of int datatype.

```
> str(mydata) # Variable Types of Col
'data.frame':   1429 obs. of  23 variables:
 $ Survey_id             : int  926 747 1190 1065 806 483 849 1386 930 390 ...
 $ Ville_id              : int  91 57 115 97 42 25 130 72 195 33 ...
 $ sex                   : int  1 1 1 1 0 1 0 1 1 1 ...
 $ Age                   : int  28 23 22 27 59 35 34 21 32 29 ...
 $ Married               : int  1 1 1 1 0 1 0 1 1 1 ...
 $ Number_children       : int  4 3 3 2 4 6 1 2 7 4 ...
 $ education_level       : int  10 8 9 10 10 10 9 10 9 10 ...
 $ total_members         : int  5 5 5 4 6 8 3 4 9 5 ...
 $ gained_asset          : int  28912201 28912201 28912201 52667108 82606287 35937466 41303144 12013633 110875
68 28912201 ...
 $ durable_asset         : int  22861940 22861940 22861940 19698904 17352654 736707 21925041 20323505 25224208
 22861940 ...
 $ save_asset            : int  23399979 23399979 23399979 49647648 23399979 23399979 23399979 48046108 800768
51 23399979 ...
 $ living_expenses       : int  26692283 26692283 26692283 397715 80877619 30696127 66730708 80076849 30162281
 26692283 ...
 $ other_expenses        : int  28203066 28203066 28203066 44042267 74503502 11531066 10890451 58456101 671844
79 28203066 ...
 $ incoming_salary       : int  0 0 0 0 1 0 0 0 1 0 ...
 $ incoming_own_farm     : int  0 0 0 1 0 1 0 0 0 0 ...
 $ incoming_business     : int  0 0 0 0 0 0 0 1 0 0 ...
 $ incoming_no_business  : int  0 0 0 1 0 1 0 0 0 0 ...
 $ incoming_agricultural: int  30028818 30028818 30028818 22288055 53384566 22688441 26692283 9275569 3256458
7 30028818 ...
 $ farm_expenses         : int  31363432 31363432 31363432 18751329 20731006 18907036 22243569 36979933 287386
91 31363432 ...
 $ labor_primary         : int  0 0 0 0 1 0 0 0 1 0 ...
 $ lasting_investment    : int  28411718 28411718 28411718 7781123 20100562 4442561 22562288 33922659 14018381
 28411718 ...
 $ no_lasting_investmen : int  28292707 28292707 28292707 69219765 43419447 76629095 55608922 54600174 151176
19 28292707 ...
 $ depressed             : int  0 1 0 0 0 0 1 0 0 0 ...
```

Representing qualitative variables as factor variables:

All the variables of Binary Qualitative Variables are converted into label using as.factor() function. Factors represent the efficient way to store character values because each unique character value is stored only one and the data itself is stored as the vector of integers.

```
> str(mydata)
'data.frame':   1429 obs. of  21 variables:
 $ sex                 : Factor w/ 2 levels "0","1": 2 2 2 2 1 2 1 2 2 2 ...
 $ Age                 : int  28 23 22 27 59 35 34 21 32 29 ...
 $ Married             : Factor w/ 2 levels "0","1": 2 2 2 2 1 2 1 2 2 2 ...
 $ Number_children     : int  4 3 3 2 4 6 1 2 7 4 ...
 $ education_level     : int  10 8 9 10 10 10 9 10 9 10 ...
 $ total_members       : int  5 5 5 4 6 8 3 4 9 5 ...
 $ gained_asset        : int  28912201 28912201 28912201 52667108 82606287 35937466 41303144 12013633 110875
68 28912201 ...
 $ durable_asset       : int  22861940 22861940 22861940 19698904 17352654 736707 21925041 20323505 25224208
 22861940 ...
 $ save_asset          : int  23399979 23399979 23399979 49647648 23399979 23399979 23399979 48046108 800768
51 23399979 ...
 $ living_expenses     : int  26692283 26692283 26692283 397715 80877619 30696127 66730708 80076849 30162281
 26692283 ...
 $ other_expenses      : int  28203066 28203066 28203066 44042267 74503502 11531066 10890451 58456101 671844
79 28203066 ...
 $ incoming_salary     : Factor w/ 2 levels "0","1": 1 1 1 1 2 1 1 1 2 1 ...
 $ incoming_own_farm   : Factor w/ 2 levels "0","1": 1 1 1 2 1 2 1 1 1 1 ...
 $ incoming_business   : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 2 1 1 ...
 $ incoming_no_business : Factor w/ 2 levels "0","1": 1 1 1 2 1 2 1 1 1 1 ...
 $ incoming_agricultural: int  30028818 30028818 30028818 22288055 53384566 22688441 26692283 9275569 3256458
7 30028818 ...
 $ farm_expenses       : int  31363432 31363432 31363432 18751329 20731006 18907036 22243569 36979933 287386
91 31363432 ...
 $ labor_primary       : Factor w/ 2 levels "0","1": 1 1 1 1 2 1 1 1 2 1 ...
 $ lasting_investment  : int  28411718 28411718 28411718 7781123 20100562 4442561 22562288 33922659 14018381
 28411718 ...
 $ no_lasting_investmen : num  28292707 28292707 28292707 69219765 43419447 ...
 $ depressed           : Factor w/ 2 levels "0","1": 1 2 1 1 1 1 2 1 1 1 ...
> |
```

Descriptive Analysis:

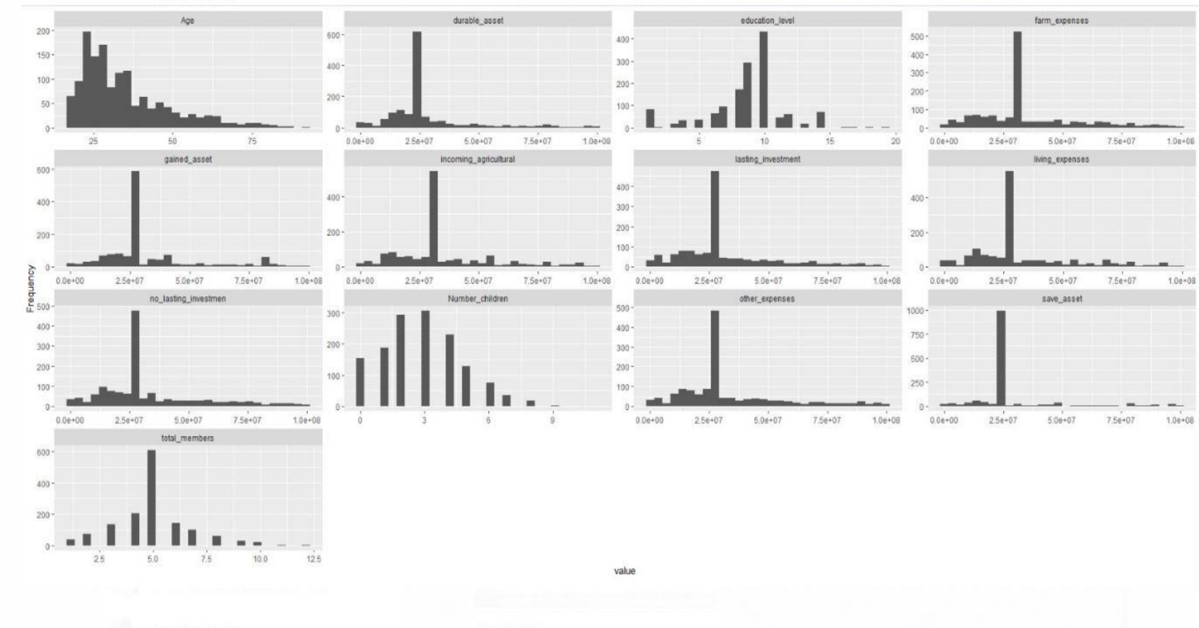
Descriptive statistics are used to describe the basic features of the data in a study. They provide simple summaries about the sample and the measures.

```
> summary(mydata)
      sex              Age           Married       Number_children  education_level  total_members
 Min.   :0.0000   Min.   :17.00   Min.   :0.0000   Min.   : 0.000   Min.   : 1.000   Min.   : 1.000
 1st Qu.:1.0000   1st Qu.:25.00   1st Qu.:1.0000   1st Qu.: 2.000   1st Qu.: 8.000   1st Qu.: 4.000
 Median :1.0000   Median :30.00   Median :1.0000   Median : 3.000   Median : 9.000   Median : 5.000
 Mean   :0.9181   Mean   :34.78   Mean   :0.7726   Mean   : 2.883   Mean   : 8.687   Mean   : 4.969
 3rd Qu.:1.0000   3rd Qu.:42.00   3rd Qu.:1.0000   3rd Qu.: 4.000   3rd Qu.:10.000   3rd Qu.: 6.000
 Max.   :1.0000   Max.   :91.00   Max.   :1.0000   Max.   :11.000   Max.   :19.000   Max.   :12.000
  gained_asset      durable_asset        save_asset       living_expenses     other_expenses
 Min.   :  325112   Min.   :  162556   Min.   :  172966   Min.   :  262919   Min.   :  172966
 1st Qu.:23269824   1st Qu.:19298521   1st Qu.:23399979   1st Qu.:20886711   1st Qu.:20980135
 Median :28912201   Median :22861940   Median :23399979   Median :26692283   Median :28203066
 Mean   :33634478   Mean   :27172957   Mean   :27424708   Mean   :32482566   Mean   :33666324
 3rd Qu.:37172832   3rd Qu.:26569498   3rd Qu.:23399979   3rd Qu.:38436887   3rd Qu.:40518887
 Max.   :99127548   Max.   :99615601   Max.   :99926758   Max.   :99295282   Max.   :99823799
 incoming_salary  incoming_own_farm incoming_business incoming_no_business incoming_agricultural
 Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000       Min.   :  325112
 1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000       1st Qu.:23222287
 Median :0.0000   Median :0.0000   Median :0.0000   Median :0.0000       Median :30028818
 Mean   :0.1798   Mean   :0.2519   Mean   :0.1078   Mean   :0.2603       Mean   :34510389
 3rd Qu.:0.0000   3rd Qu.:1.0000   3rd Qu.:0.0000   3rd Qu.:1.0000       3rd Qu.:40038424
 Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000       Max.   :99789095
 farm_expenses      labor_primary    lasting_investment no_lasting_investmen   depressed
 Min.   :  271505   Min.   :0.0000   Min.   :   74292   Min.   :  126312     Min.   :0.0000
 1st Qu.:22799659   1st Qu.:0.0000   1st Qu.:20019113   1st Qu.:20747913     1st Qu.:0.0000
 Median :31363432   Median :0.0000   Median :28411718   Median :28292707     Median :0.0000
 Mean   :35491526   Mean   :0.2134   Mean   :32992215   Mean   :33597268     Mean   :0.1666
 3rd Qu.:43485844   3rd Qu.:0.0000   3rd Qu.:39826862   3rd Qu.:40883682     3rd Qu.:0.0000
 Max.   :99651194   Max.   :1.0000   Max.   :99446667   Max.   :99651194     Max.   :1.0000
>
```

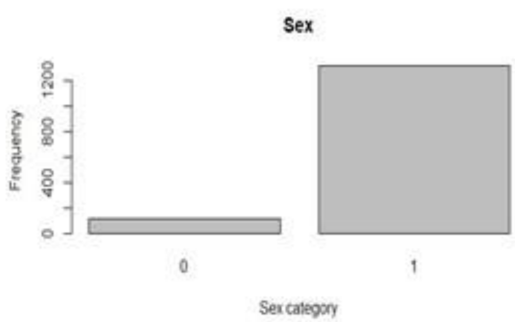
Summary statistics obtained from R for each variable. These include mean, median, and quartiles along with some other statistics. This analysis is used to display the mean, median, Quartile (used in Box plot) min, and max and this analysis is done on each feature in the dataset.

Histograms for quantitative variables and bar charts for the qualitative variables all produced in R. In our data set all the features have numerical data because of which we are reporting histograms for all the features given below.

We are displaying histograms and Bar plots for 20 features except "Survey_id" ,"Ville_id" as it has a wide variety of frequency and range so, also we are not planning to consume these features for predicting our final output and output feature "depressed".


Age
durable_asset
education_level
farm_expenses
gained_asset
incoming_agricultural
lasting_investment
living_expenses
no_lasting_investmen
Number_children
other_expenses
save_asset
total_members
Frequency
value

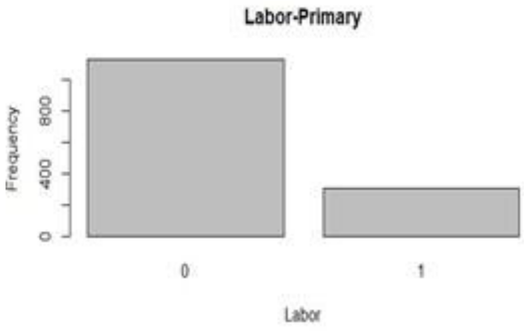
Plots:

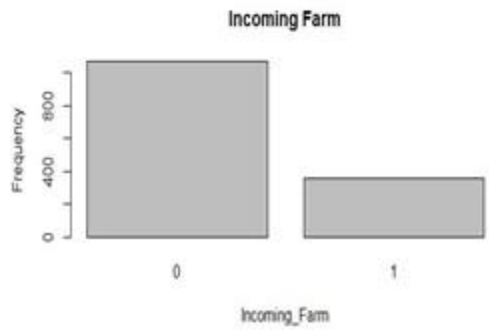

Sex
Frequency
Sex category


incoming Salary
Frequency
Incoming_Salary


Incoming Business
Frequency
Incoming_business


Incoming No Business
Frequency
Incoming_No_business





Correlation in the feature variables and feature selection:

As seen from below images and graph the correlation between different features in the dataset is explained. This matrix and the graph below help us to understand the relationship between the different features and their correlation and helps us to understand how important the relationship between different features is.

```
> str(mydata)
'data.frame':   1429 obs. of  11 variables:
 $ Age                 : int  28 23 22 27 59 35 34 21 32 29 ...
 $ Married             : int  1 1 1 1 0 1 0 1 1 1 ...
 $ Number_children     : int  4 3 3 2 4 6 1 2 7 4 ...
 $ education_level     : int  10 8 9 10 10 10 9 10 9 10 ...
 $ total_members       : int  5 5 5 4 6 8 3 4 9 5 ...
 $ incoming_salary     : int  0 0 0 0 1 0 0 0 1 0 ...
 $ incoming_own_farm   : int  0 0 0 1 0 1 0 0 0 0 ...
 $ incoming_business   : int  0 0 0 0 0 0 0 1 0 0 ...
 $ incoming_no_business: int  0 0 0 1 0 1 0 0 0 0 ...
 $ labor_primary       : int  0 0 0 0 1 0 0 0 1 0 ...
 $ depressed           : Factor w/ 2 levels "0","1": 1 2 1 1 1 1 2 1 1 1 ...
> 
```

The correlation among different feature variables were calculated. The values above the 0.4 values were considered as important for the model as there was low collinearity among the features in the dataset. So, based on the below values the total of 11 features were shortlisted after feature selection as they respond to the higher correlation amongst them and it gives better accuracy in the prediction of the model. The screenshot shows the features that were selected after feature selection.

The data set has many binary variables, with the predict variables having almost 75% of 0 class (not depressed). The feature variables also have small correlation. There are only a few variables showing strong correlation.

The features that were selected after the feature selection based on the correlation values are: Age, Married, No of Children, Education Level, Total Members, Incoming Salary, Incoming own farm, incoming business, Incoming business, Labor Primary.

	sex	Age	Married	Number_children	education_level
sex	1.000000000	-0.159376468	0.2824717731	0.2142969559	-0.072136566
Age	-0.159376468	1.000000000	-0.3969435383	-0.1384482522	-0.377146361
Married	0.282471773	-0.396943538	1.0000000000	0.2272057284	0.218406391
Number_children	0.214296956	-0.138448252	0.2272057284	1.0000000000	0.175164810
education_level	-0.072136566	-0.377146361	0.2184063911	0.1751648097	1.000000000
total_members	0.180664189	-0.073935572	0.2468082682	0.7817307685	0.130235801
gained_asset	0.022315573	0.008315619	-0.0153167730	0.0161945008	0.014533575
durable_asset	0.028602285	0.045368312	-0.0453561999	-0.0143579664	-0.011658119
save_asset	0.006636670	-0.026684583	0.0091639573	0.0278499323	0.046625981
living_expenses	-0.003492730	-0.036599266	0.0245176427	-0.0006181363	0.010131668
other_expenses	0.055457861	0.026269959	0.0316764046	0.0016111496	-0.043731750
incoming_salary	-0.032949561	-0.055883704	0.0236902344	-0.0194145591	0.010842835
incoming_own_farm	0.067466927	0.125569489	0.0072122118	0.0611403291	-0.033861024
incoming_business	0.087322045	-0.028203527	-0.0321710580	0.0325144506	0.014032496
incoming_no_business	0.101542889	-0.087054481	0.0441497166	0.0659354226	0.037304559
incoming_agricultural	0.017828127	-0.009172562	-0.0430320166	0.0185123002	-0.054515481
farm_expenses	0.070757255	-0.009427753	0.0369185758	0.0522733869	0.010647703
labor_primary	-0.025091571	-0.066346850	0.0177920216	-0.0167310300	0.044067278
lasting_investment	0.037923032	0.045713674	-0.0006062229	0.0429875911	0.005361286
no_lasting_investmen	0.047778990	-0.023023662	0.0497487741	0.0144801849	0.013782302
depressed	-0.003518778	0.105721084	-0.0621551180	0.0038229010	-0.098043324

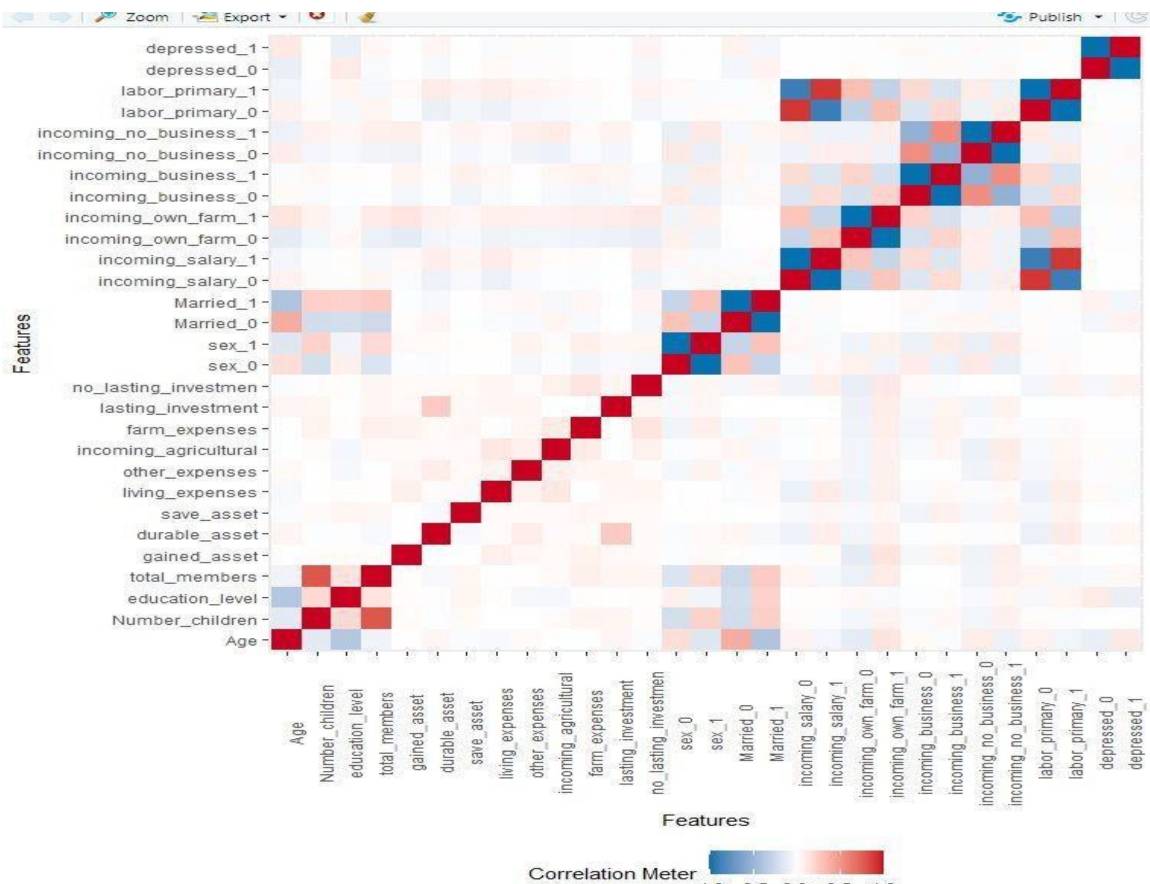
	total_members	gained_asset	durable_asset	save_asset	living_expenses	other_expenses
sex	0.180664189	0.022315573	0.028602285	0.006636670	-0.0034927299	0.055457861
Age	-0.073935572	0.008315619	0.045368312	-0.026684583	-0.0365992658	0.026269959
Married	0.246808268	-0.015316773	-0.045356200	0.009163957	0.0245176427	0.031676405
Number_children	0.781730768	0.016194501	-0.014357966	0.027849932	-0.0006181363	0.001611150
education_level	0.130235801	0.014533575	-0.011658119	0.046625981	0.0101316684	-0.043731750
total_members	1.000000000	0.015471788	-0.040372023	0.036629820	-0.0059374721	0.009988096
gained_asset	0.015471788	1.000000000	-0.005729581	-0.004477290	0.0739814188	0.039910943
durable_asset	-0.040372023	-0.005729581	1.000000000	-0.038217884	0.0209838824	0.086410117
save_asset	0.036629820	-0.004477290	-0.038217884	1.000000000	0.0237222706	0.028680476
living_expenses	-0.005937472	0.073981419	0.020983882	0.023722271	1.0000000000	0.055056622
other_expenses	0.009988096	0.039910943	0.086410117	0.028680476	0.0550566217	1.000000000
incoming_salary	-0.047026153	0.028493447	0.078161625	0.039431033	0.0897687407	0.039328844
incoming_own_farm	0.093959391	0.126403287	0.064366399	0.038604582	0.0741433200	0.063620397
incoming_business	0.008520253	0.050641461	0.015376179	0.067390126	0.0318583025	0.009415269
incoming_no_business	0.070957974	0.075120357	0.020217559	0.053794778	0.0234100109	0.072545013
incoming_agricultural	0.026157050	0.028655896	0.024394841	0.022900329	0.1155530270	0.071318802
farm_expenses	0.072907055	0.058568596	0.027891701	0.040168245	0.0035119394	0.042088483
labor_primary	-0.043622369	0.022878845	0.094493952	0.061983737	0.0836389845	0.051381200
lasting_investment	0.044340935	0.033085068	0.246893524	0.034958507	0.0402465329	0.048172855
no_lasting_investmen	0.047658461	0.030592325	0.022040307	0.028493081	0.0467023854	0.018639865

sex	-0.0329495615	0.067466927	0.087322045	0.101542889
Age	-0.0558837037	0.125569489	-0.028203527	-0.087054481
Married	0.0236902344	0.007212212	-0.032171058	0.044149717
Number_children	-0.0194145591	0.061140329	0.032514451	0.065935423
education_level	0.0108428349	-0.033861024	0.014032496	0.037304559
total_members	-0.0470261526	0.093959391	0.008520253	0.070957974
gained_asset	0.0284934472	0.126403287	0.050641461	0.075120357
durable_asset	0.0781616251	0.064366399	0.015376179	0.020217559
save_asset	0.0394310328	0.038604582	0.067390126	0.053794778
living_expenses	0.0897687407	0.074143320	0.031858303	0.023410011
other_expenses	0.0393288444	0.063620397	0.009415269	0.072545013
incoming_salary	1.0000000000	-0.271747247	-0.162745138	-0.078490557
incoming_own_farm	-0.2717472473	1.000000000	-0.201682244	0.081861283
incoming_business	-0.1627451384	-0.201682244	1.000000000	0.534400605
incoming_no_business	-0.0784905568	0.081861283	0.534400605	1.000000000
incoming_agricultural	0.0160587353	0.064287042	0.040302220	0.087570702
farm_expenses	0.0208395358	0.075559739	0.061622555	0.030318031
labor_primary	0.8989509418	-0.302293746	-0.181038954	-0.091069359
lasting_investment	0.0007152039	0.086145739	0.042626080	0.057760192
no_lasting_investmen	0.0754223247	0.106167984	-0.023941272	0.004462138
depressed	-0.0039288655	0.013161494	-0.028158171	-0.025496366

	incoming_agricultural	farm_expenses	labor_primary	lasting_investment
sex	0.017828127	0.070757255	-0.025091571	0.0379230319
Age	-0.009172562	-0.009427753	-0.066346850	0.0457136743
Married	-0.043032017	0.036918576	0.017792022	-0.0006062229
Number_children	0.018512300	0.052273387	-0.016731030	0.0429875911
education_level	-0.054515481	0.010647703	0.044067278	0.0053612863
total_members	0.026157050	0.072907055	-0.043622369	0.0443409354
gained_asset	0.028655896	0.058568596	0.022878845	0.0330850677
durable_asset	0.024394841	0.027891701	0.094493952	0.2468935241
save_asset	0.022900329	0.040168245	0.061983737	0.0349585073
living_expenses	0.115553027	0.003511939	0.083638984	0.0402465329
other_expenses	0.071318802	0.042088483	0.051381200	0.0481728546
incoming_salary	0.016058735	0.020839536	0.898950942	0.0007152039
incoming_own_farm	0.064287042	0.075559739	-0.302293746	0.0861457392
incoming_business	0.040302220	0.061622555	-0.181038954	0.0426260798
incoming_no_business	0.087570702	0.030318031	-0.091069359	0.0577601921
incoming_agricultural	1.000000000	0.093705636	0.039126869	-0.0032136145
farm_expenses	0.093705636	1.000000000	0.023941109	-0.0017718697
labor_primary	0.039126869	0.023941109	1.000000000	0.0080666929
lasting_investment	-0.003213614	-0.001771870	0.008066693	1.0000000000
no_lasting_investmen	0.066964097	0.125171679	0.056362128	0.0426971173
depressed	-0.019147452	-0.004901205	-0.012825236	0.0041360672

```
                       no_lasting_investmen    depressed
sex                             0.047778990 -0.003518778
Age                            -0.023023662  0.105721084
Married                         0.049748774 -0.062155118
Number_children                 0.014480185  0.003822901
education_level                 0.013782302 -0.098043324
total_members                   0.047658461  0.035055892
gained_asset                    0.030592325 -0.004401936
durable_asset                   0.022040307  0.040505375
save_asset                      0.028493081  0.009059126
living_expenses                 0.046702385 -0.028213284
other_expenses                  0.018639865  0.017116850
incoming_salary                 0.075422325 -0.003928865
incoming_own_farm               0.106167984  0.013161494
incoming_business              -0.023941272 -0.028158171
incoming_no_business            0.004462138 -0.025496366
incoming_agricultural           0.066964097 -0.019147452
farm_expenses                   0.125171679 -0.004901205
labor_primary                   0.056362128 -0.012825236
lasting_investment              0.042697117  0.004136067
no_lasting_investmen            1.000000000  0.051650536
depressed                       0.051650536  1.000000000
>
```

Correlation Matrix



Building the Models

There are three models proposed as the solution to the given problem statement.

1. Logistic Regression
2. Naïve Bayes Classification

To build the models we need to preprocess the dataset according to each model. Each model requires an input to it; either the data should be all numeric or all categorical. The preprocessing of the dataset, splitting of it and calculating its accuracy is done for each model separately. To select which model to be used to get better results and accuracy while predicting the output different models are used and the approach to decide which all models can be used depending on the quality of data, the quantity of data, and the nature of the data.

Logistic Regression

Logistic Regression is the appropriate regression analysis that is to be conducted when the dependent variable is binary. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval, or ratio-level independent variables.

Data Preprocessing:

The dataset required to apply logistic regression should be numeric. After handling the null values to apply the logistic model the dataset should be numeric. As we had to convert our predict variable into as factor, we had to again convert that back to numeric.

Splitting of data:

To perform classification, we need to divide the data into a training set and a testing set. The following screenshot shows the code for evaluation and the number of rows after splitting the results. Until the data set is split, the sample function is performed on the data. This randomizes the rows of the dataset to provide a better collection of results. We have divided the dataset into 80% training data and 20% testing data.

Building the model:

The logistic model was build using glm() function. The input to this function is the regression model we need to build on the train dataset.

The summary of this model tells us a lot about the model and the feature variables and their contribution towards building the model. The first feature of the summary shows the intercept for each x variable which is the beta value in the regression model. The last feature shows the individual p values. From this we can conclude that the p value for Married, education_level, total_membaer, no_last_investment. is less than the alpha value. If the p value which is probability value of x the variable is less than alpha value 0.05, then it implies that the x variable does contributes much towards the y variable.

```r

## fit a logistic regression model with the training dataset
log.model <- glm(depressed ~., data = train, family = binomial(link = "logit"))
summary(log.model)

## to predict using logistic regression model, probablilities obtained
log.predictions <- predict(log.model, test, type="response")

## Look at probability output
log.predictions

#Below we are going to assign our labels with decision rule that
#if the prediction is greater than 0.5, assign it 1 else 0.
log.prediction.rd <- ifelse(log.predictions > 0.5, 1, 0)
log.prediction.rd

#confusion matrix
table(log.prediction.rd, test[,21])
```

88:1 (Top Level)

Console Terminal Jobs

C:/Users/dharm/Desktop/Fall 20 SEM3/IS 777/Project/

```
                        Estimate Std. Error z value Pr(>|z|)
(Intercept)           -1.614e+00  6.827e-01  -2.364  0.01809 *
sex                   -6.785e-02  3.117e-01  -0.218  0.82770
Age                    5.281e-03  6.801e-03   0.776  0.43749
Married               -4.255e-01  2.114e-01  -2.013  0.04416 *
Number_children       -4.717e-02  6.998e-02  -0.674  0.50030
education_level       -9.167e-02  3.127e-02  -2.932  0.00337 **
total_members          1.723e-01  7.153e-02   2.409  0.01600 *
gained_asset          -1.071e-09  4.135e-09  -0.259  0.79556
durable_asset          7.663e-09  4.414e-09   1.736  0.08256 .
save_asset             5.530e-09  5.145e-09   1.075  0.28245
living_expenses       -3.825e-09  4.138e-09  -0.924  0.35528
other_expenses         1.864e-09  3.743e-09   0.498  0.61848
incoming_salary        8.175e-02  4.912e-01   0.166  0.86782
incoming_own_farm     -2.785e-01  2.254e-01  -1.235  0.21674
incoming_business     -3.305e-01  3.523e-01  -0.938  0.34812
incoming_no_business   6.842e-02  2.257e-01   0.303  0.76177
incoming_agricultural -4.668e-09  4.087e-09  -1.142  0.25333
farm_expenses          1.616e-09  3.908e-09   0.413  0.67925
labor_primary         -1.858e-01  4.748e-01  -0.391  0.69564
lasting_investment    -2.366e-09  4.019e-09  -0.589  0.55612
no_lasting_investmen   8.233e-09  3.707e-09   2.221  0.02637 *
```

Predicted values by the model:

```
> log.predictions
      1143       1144       1145       1146       1147       1148       1149       1150       1151
0.11170913 0.14234299 0.22884784 0.12349540 0.27898435 0.20242601 0.18973222 0.20895163 0.23124948
      1152       1153       1154       1155       1156       1157       1158       1159       1160
0.15535306 0.17883808 0.11923617 0.28399373 0.11787930 0.16666033 0.11473211 0.25557253 0.26626239
      1161       1162       1163       1164       1165       1166       1167       1168       1169
0.10752004 0.16891743 0.13464453 0.19830291 0.10123875 0.11257202 0.13537418 0.13327380 0.17269159
      1170       1171       1172       1173       1174       1175       1176       1177       1178
0.13383460 0.12569867 0.15037581 0.16117802 0.09276137 0.22946454 0.17774576 0.20834375 0.10893797
      1179       1180       1181       1182       1183       1184       1185       1186       1187
0.39223344 0.26242056 0.13512796 0.36430757 0.13325237 0.12467371 0.11993250 0.18578262 0.19498315
      1188       1189       1190       1191       1192       1193       1194       1195       1196
0.18683016 0.17234594 0.13444715 0.22265872 0.09437220 0.13519439 0.13521606 0.18640569 0.14515525
```

Round the values to the nearest value as 0 or 1

```
> log.prediction.rd
1143 1144 1145 1146 1147 1148 1149 1150 1151 1152 1153 1154 1155 1156 1157 1158 1159 1160 1161 1162 1163
   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
1164 1165 1166 1167 1168 1169 1170 1171 1172 1173 1174 1175 1176 1177 1178 1179 1180 1181 1182 1183 1184
   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
1185 1186 1187 1188 1189 1190 1191 1192 1193 1194 1195 1196 1197 1198 1199 1200 1201 1202 1203 1204 1205
   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
1206 1207 1208 1209 1210 1211 1212 1213 1214 1215 1216 1217 1218 1219 1220 1221 1222 1223 1224 1225 1226
   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
1227 1228 1229 1230 1231 1232 1233 1234 1235 1236 1237 1238 1239 1240 1241 1242 1243 1244 1245 1246 1247
   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
1248 1249 1250 1251 1252 1253 1254 1255 1256 1257 1258 1259 1260 1261 1262 1263 1264 1265 1266 1267 1268
   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
1269 1270 1271 1272 1273 1274 1275 1276 1277 1278 1279 1280 1281 1282 1283 1284 1285 1286 1287 1288 1289
```

Prediction and Accuracy:

The prediction of the value is done using predict (). The values predicted are not classified into 0 or 1. The values are decimal values and hence need to be classified into labels. The values below 0.5 will be classified into 0 and the values greater than 0.5 will be classified into 1 label. The accuracy of the model will be predicted using a confusion matrix and then the accuracy is calculated.

The accuracy of the Logistic model 83.97%

```
> #Accuracy
> accuracy <- table(log.prediction.rd, test[,11])
> sum(diag(accuracy))/sum(accuracy)
[1] 0.8397213
> 
```

Naïve Bayes Classification

Naive Bayes is among the simplest and most efficient classification algorithms based on Bayes Theorem. The Bayes theorem states that "The posterior probability is equal to the probability times the probability ratio before." The Naive Bayes model is simple to create and especially helpful for very large data sets. All these characteristics in Naïve Bayes contribute independently to the probability of the outcome. Firstly, using each attribute of the dataset, we will construct a frequency table. We will generate probability tables for each frequency table.

Data Preprocessing:

```
> naive.data$incoming_no_business <- as.factor(naive.data$incoming_no_business)
> naive.data$labor_primary <- as.factor(naive.data$labor_primary)
> str(naive.data)
'data.frame':   1429 obs. of  11 variables:
 $ Age                 : num  0.1486 0.0811 0.0676 0.1351 0.5676 ...
 $ Married             : Factor w/ 2 levels "0","1": 2 2 2 2 1 2 1 2 2 2 ...
 $ Number_children     : num  0.364 0.273 0.273 0.182 0.364 ...
 $ education_level     : num  0.5 0.389 0.444 0.5 0.5 ...
 $ total_members       : num  0.364 0.364 0.364 0.273 0.455 ...
 $ incoming_salary     : Factor w/ 2 levels "0","1": 1 1 1 1 2 1 1 1 2 1 ...
 $ incoming_own_farm   : Factor w/ 2 levels "0","1": 1 1 1 2 1 2 1 1 1 1 ...
 $ incoming_business   : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 2 1 1 ...
 $ incoming_no_business: Factor w/ 2 levels "0","1": 1 1 1 2 1 2 1 1 1 1 ...
 $ labor_primary       : Factor w/ 2 levels "0","1": 1 1 1 1 2 1 1 1 2 1 ...
 $ depressed           : Factor w/ 2 levels "0","1": 1 2 1 1 1 1 2 1 1 1 ...
> 
```

Naive Bayes model works on calculating the probability of the feature variables. It works on both categorical as well as numeric datatype. Hence, we will be converting num to factor for the binary data type.

Splitting of the data:

We need to break the information into a training set and a test set to conduct the classification. As the dataset is small, for this reason, holdout evaluation will be used. The screenshot below shows the train test Evaluation code and the number of rows after splitting the results. The sample function is carried out on the data before the data set is broken. This will shuffle the rows of the dataset to give a better data collection.

```
install.packages("e1071")
library(e1071)

naive <- naiveBayes(train.naive$depressed~. , data = train.naive)
naive

#predict
#install.packages("caret")
library(caret)
pre_naive = predict(naive, test.naive)
head(pre_naive)

confusionMatrix(table(pre_naive,test.naive$depressed))

```

```
94:1    naive Bayes:

Console   Terminal   Jobs
C:/Users/dharm/Desktop/Fall 20 SEM3/IS 777/Project/
> naive

Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
        0         1 
0.8425197 0.1574803 

Conditional probabilities:
   Age
Y       [,1]     [,2]
  0 34.04465 13.51326
  1 38.16111 16.05722
```

Building the Naïve Bayes model:

We are using the Naïve Bayes Classification to construct a naïve Bayes model. The parameter for this is a model of total regression. As we are just training the model to predict the class, the information used is the train data. In the adjacent screenshot, the model's output is given. There is a table with values for lasting investment and non-lasting investment under the Apriori Probabilities.

Prediction and Accuracy:

We use the predict () function to predict the test output. The parameters are the naïve Bayes model and the testing dataset. To evaluate the model, we are using a confusion matrix. The accuracy of this model is 83.2%. On the top left corner there is a 0,1 (non-depressed, depressed) table. It tells that there are 233 non-depressed observations which were correctly predicted by our model into the non-depressed label and there are 5 depressed observations correctly predicted. Whereas 37 depressed observations were wrongly predicted as non-depressed and vice versa for 11 observations. Accuracy is given by sum of the correctly predicted values the total observations. The CI entry says that the model is 95% confident of giving an accuracy between 78.3% and 87.3%. The accuracy of the Naïve Bayes model is 83.2%. With the positive class being non- depressed.

The accuracy of the Naïve Bayes model is 83.22%

```
> confusionMatrix(table(pre_naive,test.naive$depressed))
Confusion Matrix and Statistics


pre_naive   0   1
        0 233  37
        1  11   5

               Accuracy : 0.8322
                 95% CI : (0.7837, 0.8736)
    No Information Rate : 0.8531
    P-Value [Acc > NIR] : 0.860638

                  Kappa : 0.0994

 Mcnemar's Test P-Value : 0.000308

            Sensitivity : 0.9549
            Specificity : 0.1190
         Pos Pred Value : 0.8630
         Neg Pred Value : 0.3125
             Prevalence : 0.8531
         Detection Rate : 0.8147
   Detection Prevalence : 0.9441
      Balanced Accuracy : 0.5370

       'Positive' Class : 0

> 
```

Approach to be adopted to maximize model outcomes

Naïve Bayes: When the amount of training data is limited compared to the number of features, historical likelihood information / data tends to improve the results.

Logistic regression: Compared to the number of features, if the training data size is limited, logic function will help minimize overfitting and result in a more generalized model.

Accuracy - The number of correct predictions (true positives and true negatives) divided by the number of total predictions. The best model for accuracy achieved a score of 83.97%: Logistic regression This model made the correct prediction of having Depression or not for 83.97% of the individuals in the test set whereas the accuracy of the Naïve Bayes model is 83.22%

Resampling:

In any statistical learning process, a training error and a test error must be computed in order to determine the consistency of the fit. It is also especially important to estimate the minimum point of error curves (training and testing) for the fit function to evaluate the under-or over-fitting, as well as the accuracy of that process.

In statistics, re-sampling is any of a few methods for conducting one of the following: estimating the accuracy of sample statistics (medians, variances, percentiles) by using subsets of available data or drawing randomly by replacing a set of data points.

Resampling requires the collection of randomized cases to be replaced from the original data sample in such a way that each number of the sample taken has several cases that are identical to the original data sample. Owing to the substitution, the number of samples taken using the re-sampling process consists of repeated instances.

There are times when there is a need to recognize the feasibility of the model without resorting to the test collection. Simply rescheduling the training set is troublesome, so a process is required to get an assessment using the training set. For this reason, re-sampling methods will be used.

Polynomial Degree:

Polynomial Regression is also known as Polynomial Linear Regression since it depends on the linearly arranged coefficients rather than the variables. In R, to implement polynomial regression, following packages were installed:

- **tidy verse** package for better visualization and manipulation.
- **caret** package for a smoother and easier machine learning workflow.

After proper installation of the packages, data is set properly which was done by splitting the data into two sets (train set and test set). Then one can visualize the data into various plots. In R, in order to fit a polynomial regression, first one needs to generate pseudo random numbers using the set.seed(n) function.

The resampling methods used in the Model are:

1. *The 100% train data*
2. *Validation Set Approach*
3. *Leave One Out Cross Validation*
4. *10-Fold Cross Validation*

POLYNOMIAL DEGREE

In modern statistics, resampling methods have become an integral part. In order to get new insights into the model, resampling is based on repeatedly drawing samples from a training collection of observations and refitting a model on each sample.

In other words, to calculate approximate p probability values, the resampling approach does not require the use of generic distribution tables (for example, regular distribution tables).

- Sampling is used if data needs to be obtained.
- Periodically, sampling should be checked.

Why it is not used for Naïve Bayes?

There are no such parameters for polynomial degree for Naïve Bayes.

A. THE ENTIRE DATA SET AS A TRAINING SET

When we used our model for prediction, then it is important to keep our training and test set separate to avoid Data Leakage i.e., having overly confident estimates of prediction accuracy because the model was evaluated on the same data it was trained on. The more data our deployed model has seen, the better is should generalize. So, we trained the model on the full set of data, which is available, that should generalize better than a model which only saw train/validation sets (e.g., ~ 100%) from the full data set.

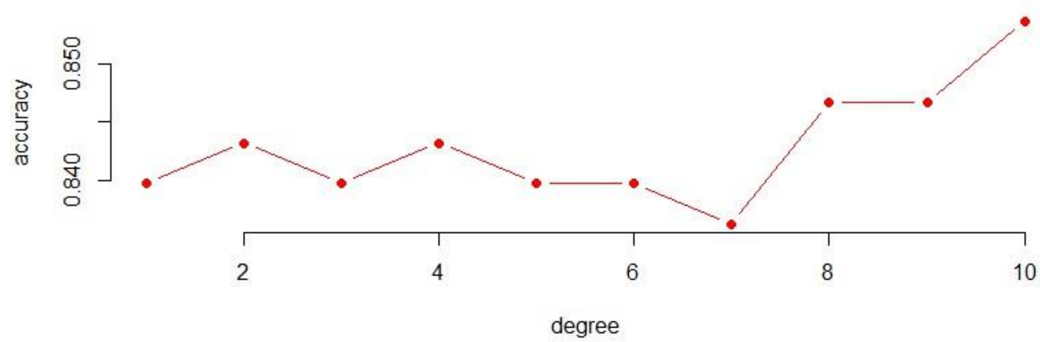
1.100% training set
Logistic Regression:

The more data our deployed model has the better it is to get the better outcome. So, we trained the model on the full set of data, which is available, that should generalize better than a model which only saw train/validation sets (e.g., ~ 100%) from the full data set. The model was trained on the 100% training dataset taking into consideration after the resamples we got the accuracy for the model as 85% as seen below.The old accuracy of the model was 82.6% which increased while applying this sampling method.

```
+   print(i)
+ }
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
[1] 6
[1] 7
[1] 8
[1] 9
[1] 10
Warning messages:
1: In predict.lm(object, newdata, se.fit, scale = 1, type = if (type ==  :
  prediction from a rank-deficient fit may be misleading
2: In predict.lm(object, newdata, se.fit, scale = 1, type = if (type ==  :
  prediction from a rank-deficient fit may be misleading
3: In predict.lm(object, newdata, se.fit, scale = 1, type = if (type ==  :
  prediction from a rank-deficient fit may be misleading
4: In predict.lm(object, newdata, se.fit, scale = 1, type = if (type ==  :
  prediction from a rank-deficient fit may be misleading
5: In predict.lm(object, newdata, se.fit, scale = 1, type = if (type ==  :
  prediction from a rank-deficient fit may be misleading
6: In predict.lm(object, newdata, se.fit, scale = 1, type = if (type ==  :
  prediction from a rank-deficient fit may be misleading
7: In predict.lm(object, newdata, se.fit, scale = 1, type = if (type ==  :
  prediction from a rank-deficient fit may be misleading
8: In predict.lm(object, newdata, se.fit, scale = 1, type = if (type ==  :
  prediction from a rank-deficient fit may be misleading
9: glm.fit: fitted probabilities numerically 0 or 1 occurred
10: In predict.lm(object, newdata, se.fit, scale = 1, type = if (type ==  :
  prediction from a rank-deficient fit may be misleading
> acc
 [1] 0.8397213 0.8432056 0.8397213 0.8432056 0.8397213 0.8397213 0.8362369 0.8466899 0.8466899 0.8536585
> 
```

Polynomial Degree graph for Logistic Regression:

As seen in the below graph the accuracy decreases for the first 5 variables and later it increases, and it is highest for the degree 10 with 85.36% accuracy.



Naïve Bayes Classification

This sampling method was applied to the Naïve Bayes Model with taking samples as 100 % data as a train data. The accuracy comes out to 83.34% much better then then the accuracy which we got before applying 100% train set for the model. The accuracy was tested on 20% of the data and considering 100% data in the train model. Applying this resampling method increases the accuracy of the model to 83.34% compared to 83.22.

```
Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
      0       1 
0.83345 0.16655 
```

THE VALIDATION SET APPROACH

The Validation Set Approach is a type of method that calculates the model error rate by keeping out a subset of data from the fitting process (creating a test dataset). The model is then constructed using the other set of observations (training dataset). The model is trained on the training dataset and its accuracy is calculated by predicting the target variable for those data points which is not present during the training that is validation set.

Steps Involved in the Validation Set Approach

1. A random splitting of the dataset into a certain ratio (generally 70-30 or 80-20 ratio is preferred)
2. Training of the model on the training data set.
3. The resultant model is applied to the validation set
4. Model’s accuracy is calculated through prediction error by using model performance metrics

Logistic Regression:

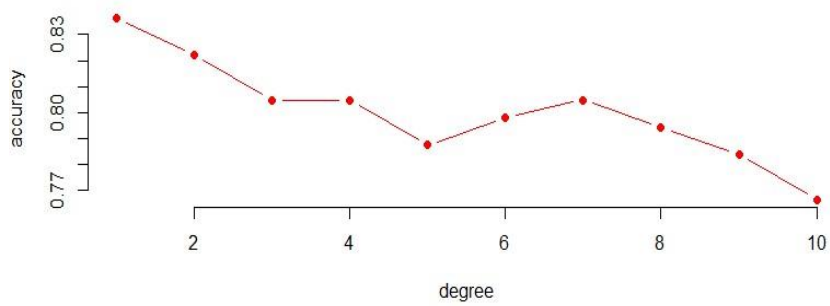
The validation set approach was done by splitting our data into 80% as the train data and the 20% as the test data. The first 80% of the data was taken to train the model and the rest 20% was used to test the results and the accuracy of the model. The model after building was applied to the predict the outcome of unseen observations. Quantify the prediction error as the mean squared difference between the observed and the predicted outcome values. The model that produces least RMSE test model was preferred. The greater accuracy was found with 83.62% accuracy.

```
  prediction from a rank-deficient fit may be misleading
9: glm.fit: fitted probabilities numerically 0 or 1 occurred
10: In predict.lm(object, newdata, se.fit, scale = 1, type = if (type ==  :
  prediction from a rank-deficient fit may be misleading
> acc
 [1] 0.8362369 0.8222997 0.8048780 0.8048780 0.7874564 0.7979094 0.8048780 0.7944251 0.7839721 0.7665505
> plot(c(1:10),acc,type = "b", frame = FALSE, pch = 19,
+      col = "red", xlab = "degree", ylab = "accuracy")
> plot(c(1:10),acc,type = "b", frame = FALSE, pch = 19,
+      col = "red", xlab = "degree", ylab = "accuracy")
> |
```

Polynomial Degree Graph for Logistic Regression

Polynomial Degree graph for Logistic Regression:

The degree graph for the model was calculated as shown below. As we can are increasing the flexibility of the model the accuracy is decreasing that shows the original model flexibility is the best.



Naïve Bayes:

The validation set approach was done by splitting our data into 80% as the train data and the 20% as the test data. The first 80% of the data was taken to train the model and the rest 20% was used to test the results and the accuracy of the model. The model after building was applied to the predict the outcome of unseen observations. Naive Bayes model works on calculating the probability of the feature variables. It works on both categorical as well as numeric datatype. The accuracy of the model was found to be 83.62% accuracy.

```
Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
        0         1
0.8337708 0.1662292

Conditional probabilities:
   Age
Y   [,1] [,2]
```

Advantages of the Validation Set approach

- One of the most basic and simple techniques for evaluating a model.
- No complex steps for implementation.

Disadvantages of the Validation Set approach

- Predictions done by the model is highly dependent upon the subset of observations used for training and validation.
- Using only one subset of the data for training purposes can make the model biased.

LEAVE-ONE-OUT CROSS VALIDATION

LOOCV (Leave One Out Cross-Validation) is a form of cross-validation method in which each observation is a validation set and the remaining (N-1) observations are a training set. In LOOCV, the fitting of the model is performed, and the prediction is made using a single observation validation package. In addition, repeat this for N times for each observation as a validation package. Model is fitted and the model is used to estimate the observation value.

Advantage:

- Leave-one-out cross-validation is approximately unbiased, since the difference in size between the training set used in each fold and the entire data set is just one pattern.
- Much less bias, since we used the entire data set for training compared to the validation set method, where we use only a subset of data for training.
- No uncertainty in training / test data when running LOOCV several times would produce the same performance.

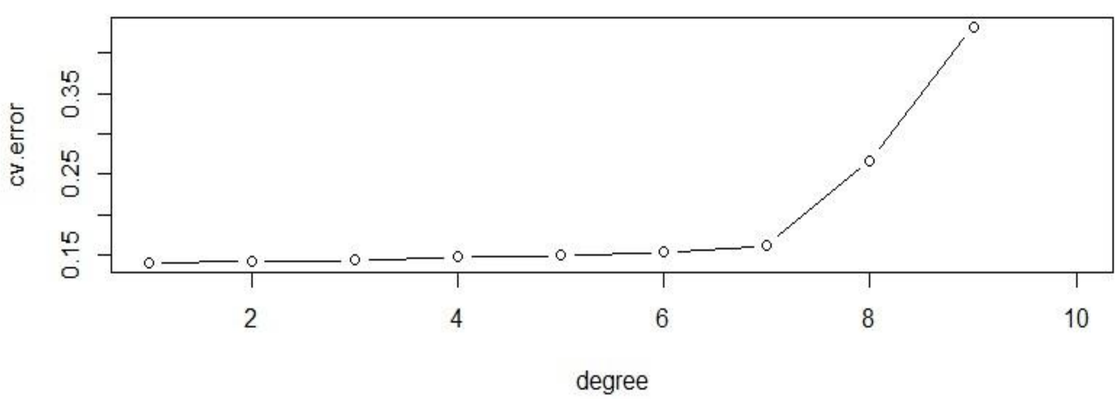
Logistic Regression:

The leave-one-out cross validation is executed by the leaving one row and executing other rows because of that the bias for the model is less which will in turn gives better accuracy. In our case the accuracy for the model after applying this resampling model is 83.97%.

```
9: In predict.lm(object, newdata, se.fit, scale = 1, type = if (type ==  :
  prediction from a rank-deficient fit may be misleading
> acc
 [1] 0.8397213 0.8362369 0.8327526 0.8327526 0.8257840 0.8153310 0.8153310 0.8118467 0.8083624 0.8048780
>
```

Polynomial Degree Graph for Logistic Regression

The graph depicts that for every degree the flexibility increases gradually and for this model the accuracy will increase gradually with the increase of the degree. The accuracy will increase at its maximum level between degree 8 and degree 10.



Naïve Bayes:

The model has the accuracy of 81.87% which is less than the other model after applying this resampling method.

```
Summary of sample sizes: 1428, 1428, 1428, 1428, 1428, 1428, ...
Resampling results across tuning parameters:

  usekernel  Accuracy   Kappa
  FALSE      0.7998600   0.06134101
   TRUE      0.8187544  -0.01699242

Tuning parameter 'fL' was held constant at a value of 0
Tuning parameter 'adjust' was held constant at
 a value of 1
```

10-FOLD CROSS VALIDATION

The alternative to LOOCV is the k-fold cross-validation method. This re-sampling method involves randomly dividing the data into k groups (aka folds) of approximately the same size. The first fold shall be treated as a validation set and the statistical method shall be adapted to the remaining data. The accuracy of the model can be evaluated by the graph plotted for different degrees for the model.

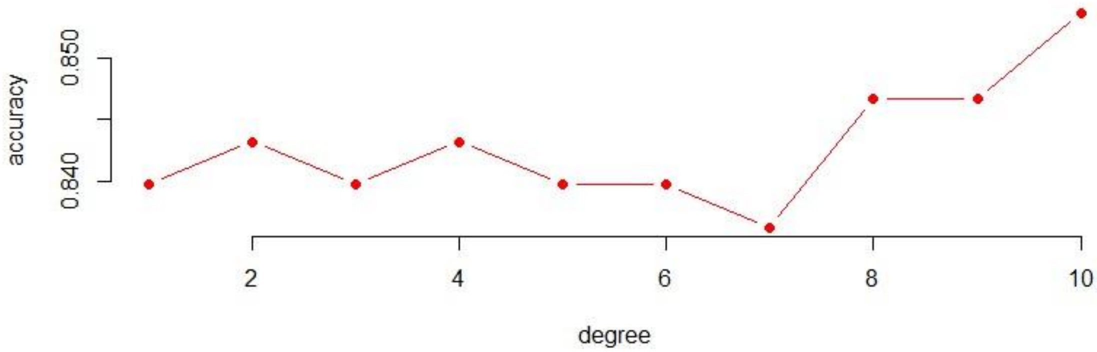
Logistic Regression:

The accuracy for the model after applying this resampling method is 83.97% which is better than other models.

```
> acc
 [1] 0.8397213 0.8432056 0.8397213 0.8432056 0.8397213 0.8397213 0.8362369 0.8466899 0.8466899 0.8536585
>
```

Polynomial Degree Graph for Logistic Regression

The graph depicts that for every degree the flexibility increases gradually, but between degree 6 and degree 8 the accuracy decrease. After degree 8 the accuracy increases gradually. This will depict that at degree 10 the model has maximum accuracy for the model.



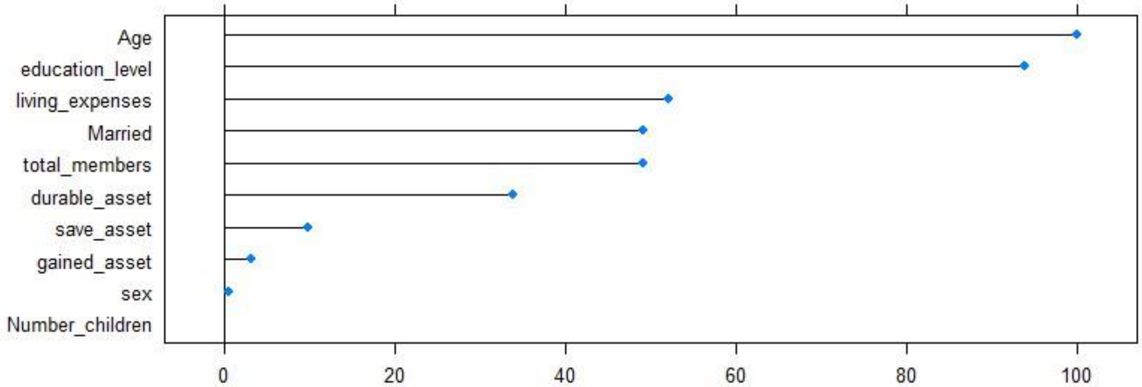
Naïve Bayes:

The accuracy for the model is 82.09% which is less than the other models.

```
(entries are percentual average cell counts across resamples)

          Reference
Prediction    0    1
         0 81.9 16.5
         1  1.4  0.1

 Accuracy (average) : 0.8209
```

```
Summary of sample sizes: 1286, 1286, 1286, 1286, 1286, 1286, ...
Resampling results across tuning parameters:

  usekernel  Accuracy   Kappa
  FALSE      0.7977642   0.05196046
   TRUE      0.8208510  -0.01289302

Tuning parameter 'fL' was held constant at a value of 0
Tuning parameter 'adjust' was held constant at
 a value of 1
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were fL = 0, usekernel = TRUE and adjust = 1.
> print("naive built")
```

This graph shows that the variables with the higher values are very important for the dataset.



Advantages

- The computation time is reduced as we repeat the procedure only 10 times when the value of k is 10.
- Reduced bias.
- Each data point will be checked exactly once and will be used in training k-1 times.
- The variance of the resultant estimate is decreased as k increases.

In my opinion, leave one out cross validation is better when you have a small set of training data. In this case, you cannot really make 10 folds to make predictions on using the rest of your data to train the model.

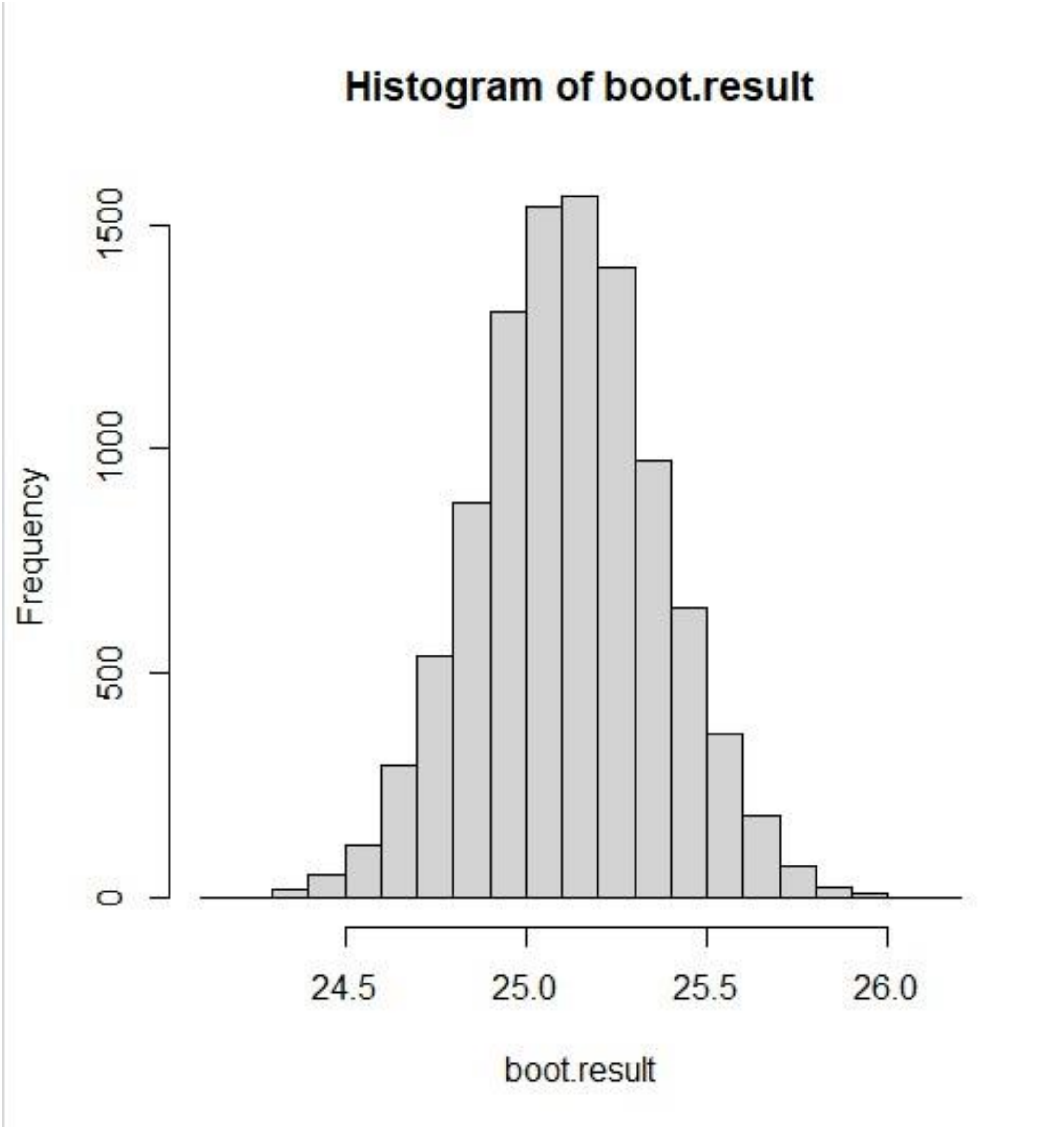
If you have a large amount of training data on the other hand, 10-fold cross validation would be a better bet, because there will be too many iterations for leave one out cross-validation and considering these many results to tune your hyperparameters might not be such a good idea.

Bootstrapping

Bootstrapping is any test or metric that uses random sampling with substitution and falls within the wider class of re-sampling methods. Bootstrapping assigns metrics of precision (bias, variance, confidence intervals, error of estimation, etc. to the estimates of the sample. This technique allows the estimation of the distribution of sampling of almost any statistic using random sampling methods.

The feature boot in R sets the bias, which implies that there is a discrepancy between the two. The regression coefficients for the single model and the mean of the bootstrap samples. The output of the bootstrap displays the original regression coefficients and their bias, which is the distinction between the original and the bootstrap coefficients. Normal errors are also included over. Also, the bootstrapped standard error output is greater than the original standard problems.

```
Bootstrap Statistics :
          original        bias     std. error
t1*  -2.001481e+00 -1.148876e-02 6.053495e-01
t2*   1.647772e-02  1.223031e-02 3.133832e-01
t3*   1.139698e-02  1.128150e-04 6.053263e-03
t4*  -2.358344e-01  2.800434e-03 1.994230e-01
t5*  -2.527187e-02 -5.294699e-03 6.976483e-02
t6*  -6.088174e-02 -1.448732e-03 2.756440e-02
t7*   1.116935e-01  6.442835e-03 6.585459e-02
t8*   2.274907e-10 -2.321057e-10 3.660718e-09
t9*   6.556526e-09 -1.456453e-10 4.152715e-09
t10*  2.836271e-09 -1.861225e-10 4.552155e-09
t11* -3.168369e-09  3.482225e-10 4.035691e-09
t12*  1.672889e-09 -2.256983e-10 3.574669e-09
t13*  2.230749e-01  7.830503e-02 5.425197e-01
t14* -1.448039e-01 -2.255988e-02 1.968375e-01
t15* -2.969247e-01 -3.360397e-02 3.321625e-01
t16* -1.846031e-02 -5.781905e-03 2.071319e-01
t17* -3.136369e-09 -2.451707e-10 3.882083e-09
t18* -1.461593e-09 -2.507725e-10 3.682964e-09
t19* -3.433474e-01 -9.295223e-02 5.200330e-01
t20* -1.471507e-09 -7.298424e-11 3.740067e-09
t21*  7.401471e-09  5.350290e-11 3.522095e-09
> boot::boot.ci(b,index=1, type = "perc")
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 1000 bootstrap replicates

CALL :
boot::boot.ci(boot.out = b, type = "perc", index = 1)

Intervals :
Level     Percentile
95%   (-3.245, -0.787 )
Calculations and Intervals on Original Scale
```



In the output above, original column corresponds to the regression coefficients. The associated standard errors are given in the column St. Error.

t1 corresponds to the sex, t2 corresponds to age and so on…

By using the k-fold cross validation to assess predictive performance, choose the model with the size leading to the best prediction performance by using the following techniques:

We implemented forward selection, backward selection, Ridge regression and Lasso regression on logistic regression.

For forward and backward we analyzed our model on AIC value

1. Forward selection

Forward selection is a type of Stepwise Regression which begins with an empty model and adds in feature one by one. In each forward step, we added one feature that gives the single best improvement to our model.

As the model continues to improve (per that same criteria) we continue the process, adding in one variable at a time and testing at each step. Once the model no longer improves with adding more variables, the process stops. Now more the AIC value better the feature selection for the model I.e., features with higher AIC value will be much more beneficial for our model

You can see the output for forward in image given below:

```
> fits2 <- stepAIC(Fitstart1, direction = 'forward',scope=formula(FitAll))
Start:  AIC=1289.17
depressed ~ 1

                       Df Deviance    AIC
+ Age                   1   1272.2 1276.2
+ education_level       1   1273.9 1277.9
+ Married               1   1281.9 1285.9
+ no_lasting_investmen  1   1283.5 1287.5
+ durable_asset         1   1284.9 1288.9
<none>                      1287.2 1289.2
+ total_members         1   1285.4 1289.4
+ incoming_business     1   1286.0 1290.0
+ living_expenses       1   1286.0 1290.0
+ incoming_no_business  1   1286.2 1290.2
+ incoming_agricultural 1   1286.6 1290.6
+ other_expenses        1   1286.8 1290.8
+ incoming_own_farm     1   1286.9 1290.9
+ labor_primary         1   1286.9 1290.9
+ save_asset            1   1287.0 1291.0
+ farm_expenses         1   1287.1 1291.1
+ gained_asset          1   1287.1 1291.1
+ lasting_investment    1   1287.1 1291.1
+ incoming_salary       1   1287.2 1291.2
+ Number_children       1   1287.2 1291.2
+ sex                   1   1287.2 1291.2

Step:  AIC=1276.21
depressed ~ Age

                       Df Deviance    AIC
+ education_level       1   1267.2 1273.2
+ no_lasting_investmen  1   1268.0 1274.0
+ total_members         1   1269.3 1275.3
<none>                      1272.2 1276.2
+ durable_asset         1   1270.5 1276.5
+ incoming_business     1   1271.3 1277.3
+ living_expenses       1   1271.3 1277.3
+ Number_children       1   1271.5 1277.5
+ Married               1   1271.6 1277.6
+ incoming_agricultural 1   1271.8 1277.8
+ incoming_no_business  1   1271.9 1277.9
+ other_expenses        1   1271.9 1277.9
+ sex                   1   1272.0 1278.0
```

```
+ living_expenses        1   1261.8 1271.8
+ incoming_agricultural  1   1262.0 1272.0
+ incoming_business      1   1262.1 1272.1
+ Married                1   1262.4 1272.4
+ incoming_no_business   1   1262.5 1272.5
+ save_asset             1   1262.7 1272.7
+ farm_expenses          1   1262.7 1272.7
+ other_expenses         1   1262.8 1272.8
+ labor_primary          1   1262.8 1272.8
+ gained_asset           1   1262.9 1272.9
+ incoming_own_farm      1   1262.9 1272.9
+ incoming_salary        1   1262.9 1272.9
+ lasting_investment     1   1262.9 1272.9
+ sex                    1   1262.9 1272.9

Step:  AIC=1269.43
depressed ~ Age + education_level + no_lasting_investmen + total_members

                        Df Deviance    AIC
<none>                       1259.4 1269.4
+ durable_asset          1   1257.5 1269.5
+ Married                1   1258.0 1270.0
+ living_expenses        1   1258.3 1270.3
+ incoming_agricultural  1   1258.4 1270.4
+ incoming_business      1   1258.6 1270.6
+ incoming_no_business   1   1258.9 1270.9
+ farm_expenses          1   1259.1 1271.1
+ Number_children        1   1259.2 1271.2
+ save_asset             1   1259.3 1271.3
+ incoming_own_farm      1   1259.3 1271.3
+ other_expenses         1   1259.3 1271.3
+ sex                    1   1259.3 1271.3
+ gained_asset           1   1259.4 1271.4
+ labor_primary          1   1259.4 1271.4
+ lasting_investment     1   1259.4 1271.4
+ incoming_salary        1   1259.4 1271.4
>
> coefficients(Fitstart1)
(Intercept)
  -1.610278
> coefficients(fits2)
       (Intercept)                 Age     education_level no_lasting_investmen        total_members
     -2.184013e+00        1.454600e-02       -6.322360e-02          6.447199e-09         7.192692e-02
> |
```

```
>
>
>
> log.predictions <- predict(model.fits2, logistic[,1:20], type="response")
> log.prediction.rd <- ifelse(log.predictions > 0.5, 1, 0)
> accuracy <- mean(log.prediction.rd != logistic[,21])
> print(paste('Accuracy',1 -accuracy ))
[1] "Accuracy 0.833449965010497"
> |
```

It is one of two commonly used methods of stepwise regression; In our model which includes every possible variable and using this method we try eliminating the extraneous variables one by one.

Result: According to the image AIC value 1269.43 is the best we got for column Age, Education_level, no_lasting_members and Total members. These features are very important features which can be used to predict our outcome. The accuracy for this was 83.34%.

2. Backward selection

Backward selection is a stepwise regression approach that begins with a full model and at each step gradually eliminates variables from the regression model to find a reduced model that best explains the data I.e. All the independent variables are entered into the equation first and at each step the variable that is the least significant is removed if they do not contribute to the regression equation.

This process continues until no nonsignificant variables remain. We built regression model from a set of predictor variables by removing predictors based on AIC values, in a stepwise manner until there is no variable left to remove any more. Now for our model, we implemented this model on whole model except our column depressed (which is our output column) so the way this model worked is that it kept deleting every variable which is least important in predicting the outcome variable depressed

Lower the AIC value better the model is, which mean that all the features which are below AIC values will be discarded I.e., they are not as much important and will not be considered for next iteration to determine AIC. So, in our case we kept on repeating the iteration until we reached to the final best features for our model. We have 20 columns but after using backward we are only left with 4 best columns

```
> fits <- stepAIC(FitAll, direction = 'backward')
Start:  AIC=1292.83
depressed ~ sex + Age + Married + Number_children + education_level +
    total_members + gained_asset + durable_asset + save_asset +
    living_expenses + other_expenses + incoming_salary + incoming_own_farm +
    incoming_business + incoming_no_business + incoming_agricultural +
    farm_expenses + labor_primary + lasting_investment + no_lasting_investmen

                        Df Deviance    AIC
- sex                    1   1250.8 1290.8
- gained_asset           1   1250.8 1290.8
- incoming_no_business   1   1250.8 1290.8
- farm_expenses          1   1251.0 1291.0
- Number_children        1   1251.0 1291.0
- lasting_investment     1   1251.0 1291.0
- incoming_salary        1   1251.1 1291.1
- other_expenses         1   1251.1 1291.1
- save_asset             1   1251.3 1291.3
- incoming_own_farm      1   1251.4 1291.4
- labor_primary          1   1251.4 1291.4
- incoming_agricultural  1   1251.6 1291.6
- living_expenses        1   1251.6 1291.6
- incoming_business      1   1251.7 1291.7
- Married                1   1252.3 1292.3
<none>                       1250.8 1292.8
- durable_asset          1   1253.5 1293.5
- total_members          1   1254.1 1294.1
- Age                    1   1254.6 1294.6
- no_lasting_investmen   1   1255.6 1295.6
- education_level        1   1256.0 1296.0

Step:  AIC=1290.83
depressed ~ Age + Married + Number_children + education_level +
    total_members + gained_asset + durable_asset + save_asset +
    living_expenses + other_expenses + incoming_salary + incoming_own_farm +
    incoming_business + incoming_no_business + incoming_agricultural +
```

```
                        Df Deviance    AIC
- incoming_agricultural  1   1256.2 1270.2
- Married                1   1256.4 1270.4
- durable_asset          1   1256.9 1270.9
<none>                       1254.9 1270.9
- Age                    1   1259.1 1273.1
- no_lasting_investmen   1   1259.2 1273.2
- total_members          1   1259.7 1273.7
- education_level        1   1261.0 1275.0

Step:  AIC=1270.16
depressed ~ Age + Married + education_level + total_members +
    durable_asset + no_lasting_investmen

                       Df Deviance    AIC
- Married               1   1257.5 1269.5
- durable_asset         1   1258.0 1270.0
<none>                      1256.2 1270.2
- no_lasting_investmen  1   1260.2 1272.2
- Age                   1   1260.7 1272.7
- total_members         1   1260.8 1272.8
- education_level       1   1261.9 1273.9

Step:  AIC=1269.5
depressed ~ Age + education_level + total_members + durable_asset +
    no_lasting_investmen

                       Df Deviance    AIC
- durable_asset         1   1259.4 1269.4
<none>                      1257.5 1269.5
- total_members         1   1261.2 1271.2
- no_lasting_investmen  1   1261.3 1271.3
- education_level       1   1263.6 1273.6
- Age                   1   1264.7 1274.7

Step:  AIC=1269.43
depressed ~ Age + education_level + total_members + no_lasting_investmen

                       Df Deviance    AIC
<none>                      1259.4 1269.4
- total_members         1   1262.9 1270.9
- no_lasting_investmen  1   1263.4 1271.4
- education_level       1   1265.5 1273.5
- Age                   1   1266.9 1274.9
```

```
>
> accuracy <- mean(log.prediction.rd != logistic[,21])
> print(paste('Accuracy',1 -accuracy ))
[1] "Accuracy 0.833449965010497"
>
```

Result:

According to the image AIC value 1269.43 is the best we got for column Age, Education_level, no_lasting_members and Total members. These features are more key features which can be helps in the prediction of the depression in the person. The accuracy for the same was found out to be 83.34%.

3. Ridge regression

Ridge Regression helps to identify the more accuracy in predicting the final outcomes and reducing the overfitting. Ridge Regression estimates the best fit lines and reduces overfitting to get the more precision in predicting the outcome by the variables.

Ridge regression doesn't require unbiased estimators; it adds just enough bias to make the estimates reasonably reliable approximations to true population values. In our model by decreasing the bias Ridge regression helps us to understand how much better our model is to predict the values.

A tuning parameter (λ) controls the strength of the penalty term. When λ = 0, ridge regression equals least squares regression. If λ = ∞, all coefficients are shrunk to zero. The ideal penalty is therefore somewhere in between 0 and ∞.

Lambda regularization coefficient is 0.2361302 which is the best lambda for this model.

```
> plot(ridge.mod)
> lambda
[1] 0.2361302
> |
```

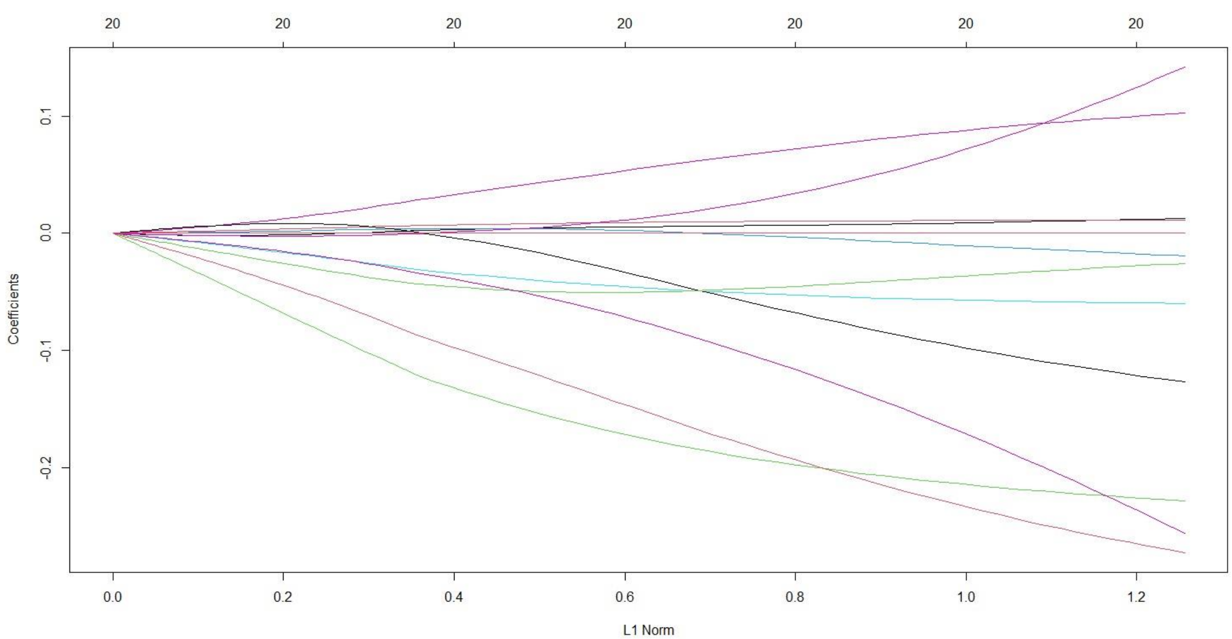
Accuracy for our ridge regression is 83.344

```
> accuracy <- mean(log.prediction.rd != logistic[,21])
> print(paste('Accuracy',1 -accuracy ))
[1] "Accuracy 0.833449965010497"
> "Accuracy 0.833449965010497"
```

In the image given below we can see Values of the coefficients is changed after introducing lambda regularization.

```
                           Estimate
(Intercept)              -1.614e+00
sex                      -6.785e-02
Age                       5.281e-03
Married                  -4.255e-01
Number_children          -4.717e-02
education_level          -9.167e-02
total_members             1.723e-01
gained_asset             -1.071e-09
durable_asset             7.663e-09
save_asset                5.530e-09
living_expenses          -3.825e-09
other_expenses            1.864e-09
incoming_salary           8.175e-02
incoming_own_farm        -2.785e-01
incoming_business        -3.305e-01
incoming_no_business      6.842e-02
incoming_agricultural    -4.668e-09
farm_expenses             1.616e-09
labor_primary            -1.858e-01
lasting_investment       -2.366e-09
no_lasting_investmen      8.233e-09
```

```
21 x 1 sparse Matrix of class "dgCMatrix"
                                  s0
(Intercept)            -1.700298e+00
sex                     1.367643e-03
Age                     5.988207e-03
Married                -1.109703e-01
Number_children         3.563125e-03
education_level        -2.798879e-02
total_members           2.482322e-02
gained_asset           -1.662499e-10
durable_asset           2.128552e-09
save_asset              6.964630e-10
living_expenses        -1.246274e-09
other_expenses          6.996021e-10
incoming_salary        -1.073955e-03
incoming_own_farm       3.617745e-03
incoming_business      -7.784402e-02
incoming_no_business   -4.017620e-02
incoming_agricultural  -1.024370e-09
farm_expenses          -3.181268e-10
labor_primary          -2.934776e-02
lasting_investment     -8.457459e-11
no_lasting_investmen    2.502656e-09
```



4.Lasso regression

In Lasso regression we are also trying to reduce the overfitting and but is also helping us to select the best feature in our model. Lasso regression is a type of linear regression that uses shrinkage. Shrinkage is where data values are shrunk towards a central point, like the mean. The lasso procedure encourages simple, sparse models (i.e., models with fewer parameters). This regression is well-suited for models showing high levels of multicollinearity or when you want to automate certain parts of model selection, like variable selection/parameter elimination.

Variables with a regression coefficient equal to zero after the shrinkage process are excluded from the model. Variables with non-zero regression coefficients variables are most strongly associated with the response variable. Explanatory variables can be either quantitative, categorical or both. Using k-fold cross validation to select the best fitting model and obtain a more accurate estimate of your model's test error rate.

- When $\lambda = 0$, no parameters are eliminated. The estimate is equal to the one found with linear regression.
- As λ increases, more and more coefficients are set to zero and eliminated (theoretically, when $\lambda = \infty$, all coefficients are eliminated).
- As λ increases, bias increases.
- As λ decreases, variance increases.

Lambda regularization coefficient is .008100374 which is the best lambda for this model.

```
> plot(lasso.mod)
> lambda
[1] 0.008100374
>
```

Accuracy for our Lasso regression is 83.344

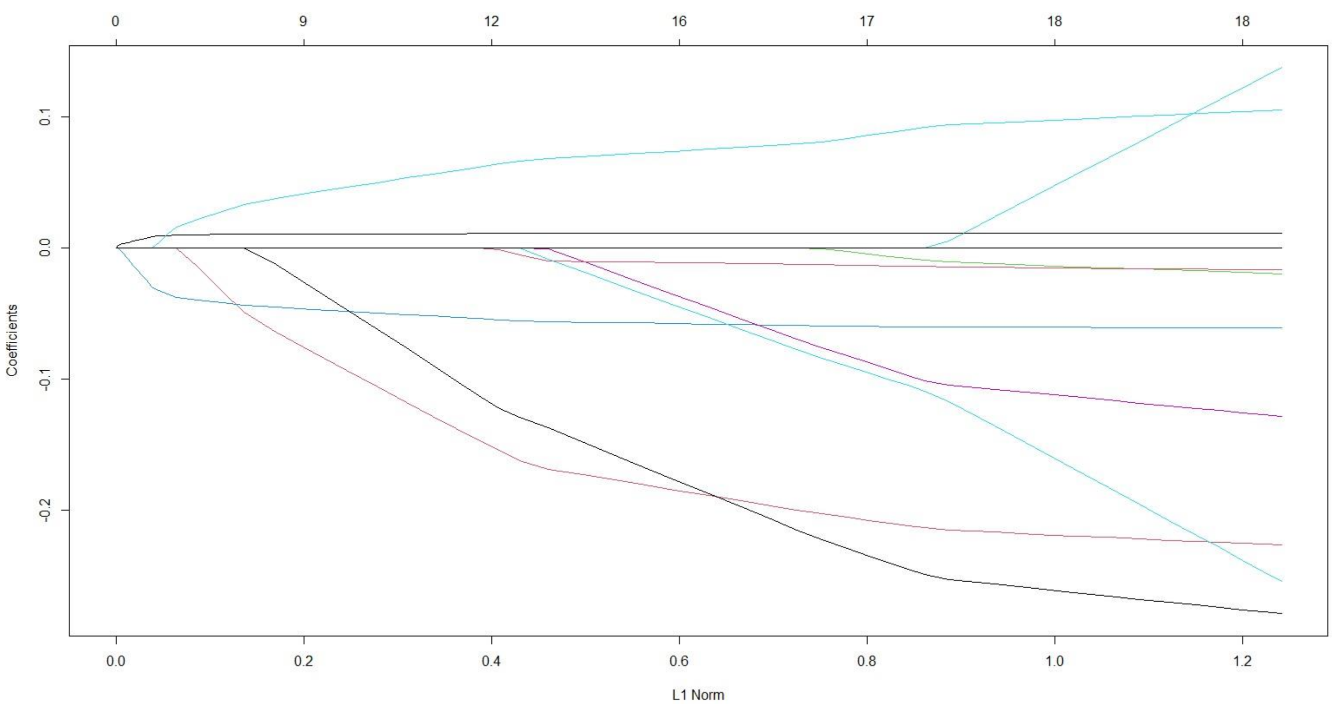
```
> print(paste('Accuracy',1 -accuracy ))
[1] "Accuracy 0.833449965010497"
> "Accuracy 0.833449965010497"
```

As we have introduced the lambda, we can see some of the coefficients have turned to 0 means these coefficients are not necessary for our final model

```
                         Estimate
(Intercept)            -1.614e+00
sex                    -6.785e-02
Age                     5.281e-03
Married                -4.255e-01
Number_children        -4.717e-02
education_level        -9.167e-02
total_members           1.723e-01
gained_asset           -1.071e-09
durable_asset           7.663e-09
save_asset              5.530e-09
living_expenses        -3.825e-09
other_expenses          1.864e-09
incoming_salary         8.175e-02
incoming_own_farm      -2.785e-01
incoming_business      -3.305e-01
incoming_no_business    6.842e-02
incoming_agricultural  -4.668e-09
farm_expenses           1.616e-09
labor_primary          -1.858e-01
lasting_investment     -2.366e-09
no_lasting_investmen    8.233e-09
```

```
> coef(lasso.mod)
21 x 1 sparse Matrix of class "dgCMatrix"
                                  s0
(Intercept)           -1.904118e+00
sex                    .
Age                    1.060072e-02
Married               -7.982675e-02
Number_children        .
education_level       -4.682890e-02
total_members          4.211777e-02
gained_asset           .
durable_asset          2.273560e-09
save_asset             .
living_expenses       -6.468867e-10
other_expenses         .
incoming_salary        .
incoming_own_farm      .
incoming_business     -3.032290e-02
incoming_no_business   .
incoming_agricultural -2.971669e-10
farm_expenses          .
labor_primary          .
lasting_investment     .
no_lasting_investmen   3.952964e-09
```

This plot indicates that about 20% of the deviance which is like R-squared is explained by 4 variables whereas the full model explains about 35% of the variance.



Part B: ACCOMODATING NON-LINEARITY

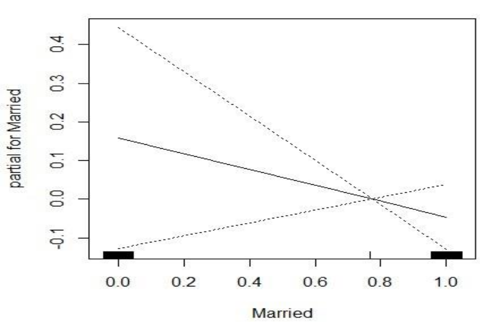
Generalized Additive Models:

GAMs are simply a class of statistical models in which many nonlinear smooth functions replace the normal linear relationship between the response and predictors to model and capture the non-linearities in the data. These are also a versatile and smooth technique that allows us to fit linear models that can rely on many Predictors Xi linearly or non-linearly to capture nonlinear relationships between predictors and response.

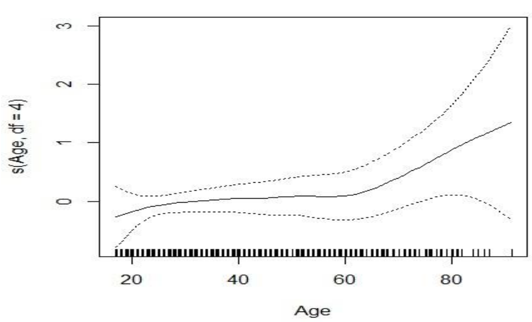
We used smoothing spline technique to relax the assumption of linearity in your generalized additive models. In Splines, the fundamental concept is that we can fit Smooth Nonlinear Functions on a bunch of Predictors Xi to capture and learn the nonlinear relationships between the variables of the model, i.e., X and Y. We used Logistic Regression Model using GAMs for predicting the Probabilities of the Binary Response values.

```
> logitgam1<-gam(I(depressed) ~ Married+s(Age,df=4)+s(Number_children,df=4)+s(education
_level,df=4)+s(total_members,df=4) ,data=mydata,family=binomial)
> summary(logitgam1)

Call: gam(formula = I(depressed) ~ Married + s(Age, df = 4) + s(Number_children,
    df = 4) + s(education_level, df = 4) + s(total_members, df = 4),
    family = binomial, data = mydata)
Deviance Residuals:
    Min      1Q  Median      3Q     Max
-1.2956 -0.6096 -0.5556 -0.4687  2.2923

(Dispersion Parameter for binomial family taken to be 1)

    Null Deviance: 1287.167 on 1428 degrees of freedom
Residual Deviance: 1236.433 on 1411 degrees of freedom
AIC: 1272.433

Number of Local Scoring Iterations: NA

Anova for Parametric Effects
                              Df  Sum Sq Mean Sq F value   Pr(>F)
Married                        1    5.40  5.3964  5.4218 0.020027 *
s(Age, df = 4)                 1    9.54  9.5421  9.5871 0.001998 **
s(Number_children, df = 4)     1    1.20  1.2022  1.2079 0.271935
s(education_level, df = 4)     1    6.46  6.4564  6.4868 0.010973 *
s(total_members, df = 4)       1    4.76  4.7632  4.7857 0.028861 *
Residuals                   1411 1404.38  0.9953
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Anova for Nonparametric Effects
                            Npar Df Npar Chisq   P(Chi)
(Intercept)
Married
s(Age, df = 4)                    3     3.3243 0.344230
s(Number_children, df = 4)        3    10.3740 0.015641 *
s(education_level, df = 4)        3     4.7991 0.187109
s(total_members, df = 4)          3    13.1958 0.004231 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
> plot(logitgam1,se=T)
```

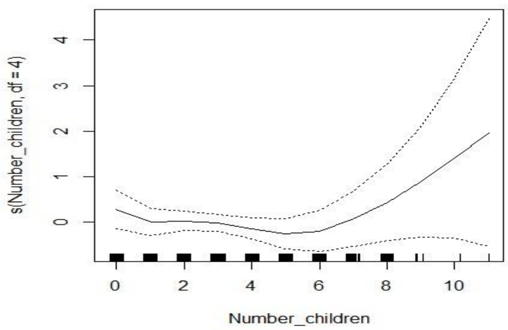
Plots:



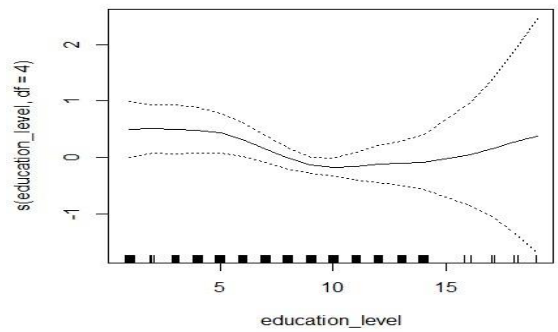
(A)



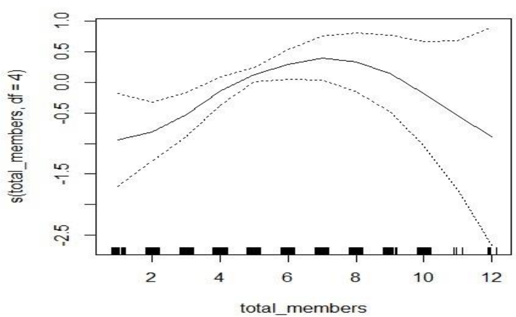
(B)



(C)



(D)



(e)

Plot(a)

For predictor variable Married, we did not use any spline technique as it is binary value. Its graph showing people tend to get less in depression when they are married.

Plot(b)

For predictor variable Age, it can be seen from the graph that depression increases with increase in age.

Plot (c)

For predictor variable Number_Children, it can be seen from the graph that depression increases with increase in number of children.

Plot (d)

For predictor variable education_level, it can be seen from the graph that people with moderate education level tends to get less depression. Although, it can also be seen from the graph that as education level goes beyond par level depression also gets increased.

`Plot (e)`

For predictor variable total_members, it can be seen from the graph that depression first increases with increase in number of members in the family then gradually decreases when total members of the family increases

Conclusion:

The main motive to build this project was to predict the values or the class for a depressed column I.e., which features, or attributes listed for it are responsible for better prediction. The two algorithms Logistic Regression and Naïve Bayes were used on our dataset to identify the depression prediction in humans where the accuracy of logistic regression is 83.97 % and for Naïve Bayes is 83.22%. The major factors contributing to the depression in the person were identified. The features which we identified and contributed most to the depression were Age, Education_level, No_lasting_ members, Total_Members. Based on the models built, the models are compared with respect to their accuracy value. This value tells us how much accurate and reliable our model is, how much perfect its predictions will be. As a real-life project, it will not be dependable if the value of accuracy of prediction is low. We used different validations techniques to get good performance and to tune the hyperparameters of the algorithms which we used on our data set. We used different resampling techniques like leave out cross validation, 10-fold validation, validation set approach and 100 percent test data where these methods increased the accuracy of the model to 83.34% compared to 83.22 and to performance of our models.

In our model we used Bootstrapping which allows the estimation of the distribution of sampling of almost any statistic using above mentioned resampling methods. The bootstrapped standard error output is greater than the original standard problems. The output of the bootstrap displays the original regression coefficients and their bias, which is the distinction between the original and the bootstrap coefficients.

We used k-fold cross validation to assess predictive performance, choose the model with the size leading to the best prediction performance by using forward selection, backward selection, ridge regression and Lasso regression for the logistic regression model. After implementing selection methods, we identified the most important features responsible for outcome depression based on AIC value. We reduced overfitting by using ridge regression and then using lasso regression we selected the best features with less overfitting in the model. Hence implementing these methods helps us to conclude the best features responsible for determining the accurate results and more accuracy in prediction. Using GAM model, we can be able to capture the non-linear relationship between predictor and response variable. Best part is that we can also mix terms on predictor variables to check the non-linear relationship. Hence from the above methods and different learnings in our model we were able to identify the major causes for the depression and were able to predict our results with more perfection and these learnings can also help society and individuals to prevent the depression.