IS 804 Project

Project Title: Predicting the likelihood of developing depression.

Submitted By

Hasan Mahmud Prottoy

Campus Id: QU47611

Email: qu47611@umbc.edu

Table of contents

Project Title: Predicting the likelihood of developing depression.

Introduction

In this project I intend to do statistical learning on a dataset related to depression. The dataset contains various attributes related to life conditions of the individuals who live in rural zones and also indicates whether they developed depression or not. According to the data source, this data originated from a study undertaken by the Busara Center in rural Siaya County, Kenya, near Lake Victoria in 2015.

Depression is a troubling illness that affects the mental health of millions of individuals throughout the world. Approximately 280 millions of people suffer from depression which constitutes 3.8% of the world population, 5.0% of adults and 5.7% of adults older than 60 years [1]. According to a 2021 WHO article [2], depression can result from a complex interaction of social, psychological and biological factors.

As mentioned above, the project is about doing statistical analysis on depression data. In doing so, I would apply various statistical learning models to make sense of the data. Applying different statistical learning techniques, I would like to know which approach works best and whether it is possible to build a predictive model to detect the likelihood of developing depression from some personal, social and economical parameters included in the dataset. This analysis may help us to get some sense on what contributing factors can contribute to development of depression. The platform used for this analysis is R.

Dataset Description

Name of the dataset: b_depressed

The dataset I am using for this project is available at Kaggle here:
https://www.kaggle.com/diegobabativa/depression

It can also be seen at google sheet (with UMBC email) here:
https://docs.google.com/spreadsheets/d/1Apfaow3l4Z7Qo2amHsOrpimq7FQ23A5eB6LK_jjNkx Y/edit?usp=sharing

These data originated from a study undertaken by the Busara Center in rural Siaya County, Kenya, near Lake Victoria in 2015. Original source:
https://zindi.africa/competitions/busara-mental-health-prediction-challenge/data

The dataset contains 23 columns and 1432 rows in total.

Some characteristics of the data:
• Task: Classification.
• Number of observations (n): 1432.

The columns (variables) and their short explanations:

Survey_id : Individual Identifier
Village_id : Village Identifier
sex : Male or female, expressed as 0 or 1, 0= female, 1= male
Age : Age
Married : Marital status, expressed as 0 or 1, 0= not married, 1= married
Number_children : Number of children <=18 or younger in Household
education_level : Years of education completed (respondent)
total_members : Household size
gained_asset: Gained asset
durable_asset : Durable asset
save_asset : Saved asset
living_expenses : Living expenses
other_expenses : Other expenses
incoming_salary : Incoming salary, 0 or 1
incoming_own_farm : Incoming own farm, 0 or 1
incoming_business : Incoming business, 0 or 1
incoming_no_business : Incoming no business, 0 or 1
incoming_agricultural : Incoming agricultural
farm_expenses : Farm expenses
labor_primary : labor primary, 0 or 1
lasting_investment : lasting investment
no_lasting_investmen : no lasting investment

Target variable:

depressed : binary; expressed as 0 or 1, 0 means the individual doesn't have depression and 1 means the individual has depression.

Snapshot of the dataset:

1	Survey_id	Village_id	sex	Age	Married	Number_childre	education_level	total_members	gained_asset	durable_asset	save_asset	living_expenses	other_expenses	incoming_salary	incoming_own_f	incoming_busine	incoming_no_bu	incoming_agricu	farm_expenses	labor_primary	lasting_investme	no_lasting_inves	depressed
2	926	91	1	28	1	4	10	5	28912201	22861940	23399979	26692283	28203066	0	0	0	0	30028818	31363432	0	28411718	28292707	0
3	747	57	1	23	1	3	8	5	28912201	22861940	23399979	26692283	28203066	0	0	0	0	30028818	31363432	0	28411718	28292707	1
4	1190	115	1	22	1	3	9	5	28912201	22861940	23399979	26692283	28203066	0	0	0	0	30028818	31363432	0	28411718	28292707	0
5	1065	97	1	27	1	2	10	4	52667108	19698904	49647648	397715	44042267	0	1	0	1	22288055	18751329	0	7781123	69219765	0
6	806	42	0	59	0	4	10	6	82606287	17352654	23399979	80877619	74503502	1	0	0	0	53384566	20731006	1	20100562	43419447	0
7	483	25	1	35	1	6	10	8	35937466	736707	23399979	30696127	11531066	0	1	0	1	22688441	18907036	0	4442561	76629095	0
8	849	130	0	34	0	1	9	3	41303144	21925041	23399979	66730708	10890451	0	0	0	0	26692283	22243569	0	22562288	55608922	1
9	1386	72	1	21	1	2	10	4	12013633	20323505	48046108	80076849	58456101	0	0	1	0	9275569	36979933	0	33922659	54600174	0
10	930	195	1	32	1	7	9	9	11087568	25224208	80076851	30162281	67184479	1	0	0	0	32564587	28738691	1	14018381	15117619	0
11	390	33	1	29	1	4	10	5	28912201	22861940	23399979	26692283	28203066	0	0	0	0	30028818	31363432	0	28411718	28292707	0
12	540	52	1	84	0	0	1	5	28912201	22861940	23399979	26692283	28203066	0	0	0	0	30028818	31363432	0	28411718	28292707	1
13	557	93	1	59	0	2	9	3	1018915	47245342	23399979	262919	30108896	0	0	0	0	66730709	13968961	1	15714453	20214956	0
14	1280	232	1	38	1	4	10	6	12390944	19186414	23399979	10810375	498078	0	0	0	0	72069163	56721101	0	20745816	15708408	0
15	1195	92	1	27	1	4	10	6	16521259	37155658	23399979	21220366	10506083	0	1	0	0	3109651	22688441	0	62405292	12144989	0
16	603	100	1	56	1	0	12	2	93596368	21140288	5925687	34566505	72469551	0	1	0	0	43775349	77808008	0	12402556	71201668	1
17	729	54	1	24	1	2	10	5	1108353	12219727	1601537	38169963	37860336	0	1	0	0	21353827	37814063	0	23991919	48624439	0
18	770	102	1	25	1	3	10	5	37172832	75432396	80076847	40705733	40278656	0	1	0	0	6406148	44843035	0	11596846	12491988	0
19	76	15	1	44	1	5	12	5	28912201	22861940	23399979	26692283	28203066	0	0	0	0	30028818	31363432	0	28411718	28292707	0
20	1374	267	1	32	1	4	9	5	28912201	22861940	23399979	26692283	28203066	0	0	0	0	30028818	31363432	0	28411718	28292707	0
21	379	22	1	26	1	2	7	4	82606287	20419597	23399979	25357668	99295292	0	1	0	1	42707653	26247411	0	26450653	36790862	0
22	1001	207	1	40	0	0	7	5	28912201	22861940	23399979	26692283	28203066	0	0	0	0	30028818	31363432	0	28411718	28292707	1
23	1356	198	1	55	0	0	6	1	17142671	83440079	24023056	16495831	42985252	0	0	0	0	48046109	58456101	0	25742926	12091604	1
24	137	9	1	34	1	3	10	5	28912201	1905829	23399979	49380727	30221003	0	1	0	0	30028818	31363432	0	249039	26558821	0
25	840	102	1	43	1	4	4	9	12390944	18978214	80076847	18684598	23542593	1	0	0	1	58322639	24212127	1	2117823	29246958	1
26	309	25	1	51	1	2	12	5	28912201	22861940	23399979	26692283	28203066	0	0	0	0	30028818	31363432	0	28411718	28292707	1
27	1078	164	1	28	1	3	10	5	75696259	46604727	23399979	53384566	67264552	0	1	0	0	92088375	67842889	0	13351174	20130429	0
28	519	50	1	53	1	4	9	5	28912201	22861940	23399979	26692283	28203066	0	0	0	0	30028818	31363432	0	28411718	28292707	0
29	1156	69	1	26	1	3	11	5	37172832	27146051	23399979	15481524	41639961	1	0	0	0	44042268	330317	1	32064487	81077814	0
30	1324	281	1	23	1	2	8	5	28912201	22861940	23399979	26692283	28203066	0	0	0	0	30028818	31363432	0	28411718	28292707	0
31	404	86	1	36	1	5	12	5	28912201	22861940	23399979	26692283	28203066	0	0	0	0	30028818	31363432	0	28411718	28292707	0
32	1385	108	1	27	1	3	8	5	28912201	22861940	23399979	26692283	28203066	0	0	0	0	30028818	31363432	0	28411718	28292707	0
33	150	31	1	44	0	3	8	4	53694088	44843037	24023056	30696127	39557964	0	0	0	0	80076847	26024976	1	10334174	28627474	0
34	1237	20	0	40	1	5	11	9	24781887	51393323	23399979	60057635	28909344	0	1	0	0	25144131	4902483	0	63993225	35331688	0
35	993	92	1	19	1	2	10	5	28912201	22861940	23399979	26692283	28203066	0	0	0	0	30028818	31363432	0	28411718	28292707	0
36	810	135	1	27	1	3	8	5	82477118	21941057	80076847	77407622	15086478	0	1	0	0	93422985	40705733	0	11130479	18484406	0
37	823	27	1	31	1	2	10	4	20651573	3119794	32030739	11611143	28026898	0	0	1	1	1134422	10104141	0	36786478	7572879	0
38	574	23	1	27	1	3	10	5	28912201	68065323	23399979	28694205	52049952	1	0	0	0	28827665	10409991	1	68065323	30429201	0
39	403	5	1	27	1	4	8	6	11151849	17360661	11210759	48046112	28187052	0	0	1	1	3203074	36034581	0	298578	46671452	0
40	632	57	0	35	0	0	13	5	28912201	22861940	23399979	26692283	28203066	0	0	0	0	30028818	31363432	0	28411718	28292707	0
41	1401	23	1	43	0	2	1	4	41303144	14093526	23399979	37369196	88084536	1	0	0	0	18417675	28694203	1	15787787	22888632	1
42	569	50	1	22	1	2	10	5	28912201	22861940	23399979	26692283	28203066	0	0	0	0	30028818	31363432	0	28411718	28292707	0
43	387	49	1	41	1	6	10	8	86736603	13677126	23399979	41373043	16623955	1	0	0	0	14914313	20603106	1	31463531	1770199	0
44	533	101	1	20	1	2	9	5	28912201	22861940	23399979	26692283	28203066	0	0	0	0	30028818	31363432	0	28411718	28292707	1
45	122	22	1	18	1	2	9	3	22375139	16816139	40038424	26158438	3867712	0	0	0	1	64061481	18684599	0	19053653	41906881	0
46	554	30	1	38	1	5	9	8	10429825	12892372	23399979	25357668	87283764	0	1	0	0	10276529	8563774	0	24843657	22430082	0
47	222	21	0	24	0	2	7	5	28912201	22861940	23399979	26692283	28203066	0	0	0	0	30028818	31363432	0	28411718	28292707	0
48	399	21	1	26	1	1	8	3	22375139	17616907	23399979	13346142	86483002	1	0	0	1	28026898	42040348	1	21616112	15250636	0
49	1107	76	1	26	1	3	6	5	28912201	22861940	23399979	26692283	28203066	0	0	0	0	30028818	31363432	0	28411718	28292707	1
50	1133	176	1	37	1	5	10	8	47847064	59609204	1601537	15881909	81838539	0	0	0	0	41639962	64862247	1	11525576	18927498	1
51	1152	18	1	23	1	4	14	6	82606287	59625221	23399979	66730708	17616907	0	1	0	0	21353827	17794855	0	74292	84525561	1
52	156	15	1	28	0	4	10	5	57142133	25064053	14413834	36034582	15695063	0	0	1	1	10009606	19018252	0	89573254	14324413	0
53	467	30	1	19	1	1	7	3	22375139	20819981	23399979	66730708	23382441	1	0	0	0	30028818	31363432	1	24659033		0
54	1010	11	1	35	1	3	9	5	41303144	37556042	23399979	20019212	32030739	0	0	0	0	26692283	2224357	0	42533453	22243569	0
55	588	52	1	53	1	3	8	5	28912201	22861940	23399979	26692283	28203066	0	0	0	0	30028818	31363432	0	28411718	28292707	0
56	296	55	1	33	1	3	9	5	28912201	22861940	23399979	26692283	28203066	0	0	0	0	30028818	31363432	0	28411718	28292707	0
57	2	1	1	19	1	2	6	4	41303144	13372833	23399979	26692283	42280579	1	0	0	1	24023056	21353827	1	18750629	53384566	0
58	124	8	1	26	1	3	10	5	28912201	18417676	23399979	53384566	38757195	0	1	0	0	33365355	27804461	0	23702748	94535172	0

Research Questions

The research questions I want to answer from the analysis:

- Is it possible to successfully predict the likelihood of developing depression using the used parameters and to what extent?
- Which variable contributes most for developing depression?
- How is the relative importance among variables in contributing to the development of depression?
- Is there any correlation between different personal & social factors (which are included in the dataset) and developing depression?
- How much is a person's financial situation responsible or not responsible for depression?

Analysis

Data preprocessing

Name of the dataset: b_depressed

To get sense of the dataset, first I run the ‘summary’ function on the dataset, which results in:

```
> summary(b_depressed)
  Survey_id    Village_id       sex              Age            Married
 Min.   :   1  Min.   :  1.00  Min.   :0.0000  Min.   :17.00  Min.   :0.0000
 1st Qu.: 358  1st Qu.: 24.00  1st Qu.:1.0000  1st Qu.:25.00  1st Qu.:1.0000
 Median : 715  Median : 57.00  Median :1.0000  Median :30.00  Median :1.0000
 Mean   : 715  Mean   : 76.29  Mean   :0.9181  Mean   :34.78  Mean   :0.7726
 3rd Qu.:1072  3rd Qu.:105.00  3rd Qu.:1.0000  3rd Qu.:42.00  3rd Qu.:1.0000
 Max.   :1429  Max.   :292.00  Max.   :1.0000  Max.   :91.00  Max.   :1.0000

 Number_children  education_level  total_members gained_asset
 Min.   : 0.000   Min.   : 1.000   Min.   : 1.000   Min.   :  325112
 1st Qu.: 2.000   1st Qu.: 8.000   1st Qu.: 4.000   1st Qu.:23269824
 Median : 3.000   Median : 9.000   Median : 5.000   Median :28912201
 Mean   : 2.883   Mean   : 8.687   Mean   : 4.969   Mean   :33634478
 3rd Qu.: 4.000   3rd Qu.:10.000   3rd Qu.: 6.000   3rd Qu.:37172832
 Max.   :11.000   Max.   :19.000   Max.   :12.000   Max.   :99127548

 durable_asset        save_asset      living_expenses      other_expenses
 Min.   :  162556   Min.   :  172966   Min.   :  262919   Min.   :  172966
 1st Qu.:19298521   1st Qu.:23399979   1st Qu.:20886711   1st Qu.:20980135
 Median :22861940   Median :23399979   Median :26692283   Median :28203066
 Mean   :27172957   Mean   :27424708   Mean   :32482566   Mean   :33666324
 3rd Qu.:26569498   3rd Qu.:23399979   3rd Qu.:38436887   3rd Qu.:40518887
 Max.   :99615601   Max.   :99926758   Max.   :99295282   Max.   :99823799

 incoming_salary  incoming_own_farm incoming_business incoming_no_business
 Min.   :0.0000   Min.   :0.0000    Min.   :0.0000  Min.   :0.0000
 1st Qu.:0.0000   1st Qu.:0.0000    1st Qu.:0.0000    1st Qu.:0.0000
 Median :0.0000   Median :0.0000    Median :0.0000    Median :0.0000
 Mean   :0.1798   Mean   :0.2519    Mean   :0.1078    Mean   :0.2603
 3rd Qu.:0.0000   3rd Qu.:1.0000    3rd Qu.:0.0000    3rd Qu.:1.0000
 Max.   :1.0000   Max.   :1.0000    Max.   :1.0000  Max.   :1.0000

 incoming_agricultural farm_expenses      labor_primary  lasting_investment
 Min.   :  325112      Min.   :  271505   Min.   :0.0000   Min.   :   74292
```

```
 1st Qu.:23222287     1st Qu.:22799659   1st Qu.:0.0000   1st Qu.:20019113
 Median :30028818     Median :31363432   Median :0.0000   Median :28411718
 Mean   :34510389     Mean   :35491526   Mean   :0.2134   Mean   :32992215
 3rd Qu.:40038424     3rd Qu.:43485844   3rd Qu.:0.0000   3rd Qu.:39826862
 Max.   :99789095     Max.   :99651194   Max.   :1.0000   Max.   :99446667

 no_lasting_investmen   depressed
 Min.   :  126312       Min.   :0.0000
 1st Qu.:20642033       1st Qu.:0.0000
 Median :28292707       Median :0.0000
 Mean   :33603851       Mean   :0.1666
 3rd Qu.:41517625       3rd Qu.:0.0000
 Max.   :99651194       Max.   :1.0000
 NA's   :20
```

I can see there are eight variables (including the target variable ‘depressed’) which are expressed as binary (0 and 1, as they are categorical). I need to express them as factors for the next steps of the analysis. For converting these eight variables into factors, I run the following codes:

```
b_depressed$depressed<- as.factor(b_depressed$depressed)
b_depressed$sex<- as.factor(b_depressed$sex)
b_depressed$Married<- as.factor(b_depressed$Married)
b_depressed$incoming_salary<- as.factor(b_depressed$incoming_salary)
b_depressed$incoming_own_farm<- as.factor(b_depressed$incoming_own_farm)
b_depressed$incoming_business<- as.factor(b_depressed$incoming_business)
b_depressed$incoming_no_business<- as.factor(b_depressed$incoming_no_business)
b_depressed$labor_primary<- as.factor(b_depressed$labor_primary)
```

Then I run the ‘summary’ function again which results in:

```
> summary(b_depressed)
   Survey_id     Village_id       sex              Age      Married  Number_children
 Min.   :   1   Min.   :  1.00   0: 117   Min.   :17.00   0: 325   Min.   : 0.000
 1st Qu.: 358   1st Qu.: 24.00   1:1312   1st Qu.:25.00   1:1104   1st Qu.: 2.000
 Median : 715   Median : 57.00            Median :30.00            Median : 3.000
 Mean   : 715   Mean   : 76.29            Mean   :34.78            Mean   : 2.883
 3rd Qu.:1072   3rd Qu.:105.00            3rd Qu.:42.00            3rd Qu.: 4.000
 Max.   :1429   Max.   :292.00            Max.   :91.00            Max.   :11.000
```

```
education_level  total_members      gained_asset  durable_asset
Min.   : 1.000   Min.   : 1.000   Min.   :  325112   Min.   :  162556
1st Qu.: 8.000   1st Qu.: 4.000   1st Qu.:23269824   1st Qu.:19298521
Median : 9.000   Median : 5.000   Median :28912201   Median :22861940
Mean   : 8.687   Mean   : 4.969   Mean   :33634478   Mean   :27172957
3rd Qu.:10.000   3rd Qu.: 6.000   3rd Qu.:37172832   3rd Qu.:26569498
Max.   :19.000   Max.   :12.000   Max.   :99127548   Max.   :99615601

  save_asset           living_expenses        other_expenses        incoming_salary
Min.   :  172966   Min.   :  262919   Min.   :  172966   0:1172
1st Qu.:23399979   1st Qu.:20886711   1st Qu.:20980135   1: 257
Median :23399979   Median :26692283   Median :28203066
Mean   :27424708   Mean   :32482566   Mean   :33666324
3rd Qu.:23399979   3rd Qu.:38436887   3rd Qu.:40518887
Max.   :99926758   Max.   :99295282   Max.   :99823799

incoming_own_farm incoming_business incoming_no_business incoming_agricultural
0:1069            0:1275            0:1057            Min.   :  325112
1: 360            1: 154            1: 372            1st Qu.:23222287
                                                Median :30028818
                                                Mean   :34510389
                                                3rd Qu.:40038424
                                                Max.   :99789095

farm_expenses        labor_primary lasting_investment no_lasting_investmen depressed
Min.   :  271505   0:1124      Min.   :   74292   Min.   :  126312       0:1191
1st Qu.:22799659   1: 305      1st Qu.:20019113   1st Qu.:20642033       1: 238
Median :31363432               Median :28411718   Median :28292707
Mean   :35491526               Mean   :32992215   Mean   :33603851
3rd Qu.:43485844               3rd Qu.:39826862   3rd Qu.:41517625
Max.   :99651194               Max.   :99446667   Max.   :99651194
                                                NA's   :20
```

So all the variables which were expressed as binary categorical values are now converted into factors.

Omitting missing values

From the summary, I can see there are missing values in the data. For omitting missing values I apply the following codes:

```
#omitting missing values
is.na(b_depressed)
sum(is.na(b_depressed))
b_depressed<-na.omit(b_depressed)
sum(is.na(b_depressed))
dim(b_depressed)
```

Which results in:

```
> sum(is.na(b_depressed))
[1] 20
> b_depressed<-na.omit(b_depressed)
> sum(is.na(b_depressed))
[1] 0
> dim(b_depressed)
[1] 1409   23
```

Dividing the dataset into Training and Testing set

For dividing the dataset into training and testing set, I apply following codes:

```
set.seed(123)
split <- sort(sample(nrow(b_depressed), nrow(b_depressed)*0.5))

training <- b_depressed[split,]

testing <- b_depressed[-split,]

summary(training)
summary(testing)

dim(training)
dim(testing)
```

Which results in:

```
> dim(training)
[1] 704  23
> dim(testing)
[1] 705  23
```

In my analysis while applying different models, in most cases I used ‘training[,c(3:23)]’ as the training dataset which means I am taking the column 3 to column 23 for training the model. The reason I am doing that is, the first two columns are Survey_id and Village_id which are individual and village identifiers respectively. The identifiers are usually randomly assigned, so there should not be any effect on the target variable by them.

Logistic regression

As my data is a classification one, I apply the logistic regression first. Logistic regression basically models the probability of one event (out of two alternatives) taking place. I applied the following codes to perform logistic regression.

Code

```
attach(b_depressed)
names(b_depressed)
summary(b_depressed)
b_depressed$depressed<- as.factor(b_depressed$depressed)
b_depressed$sex<- as.factor(b_depressed$sex)
b_depressed$Married<- as.factor(b_depressed$Married)
b_depressed$incoming_salary<- as.factor(b_depressed$incoming_salary)
b_depressed$incoming_own_farm<- as.factor(b_depressed$incoming_own_farm)
b_depressed$incoming_business<- as.factor(b_depressed$incoming_business)
b_depressed$incoming_no_business<- as.factor(b_depressed$incoming_no_business)
b_depressed$labor_primary<- as.factor(b_depressed$labor_primary)

summary(b_depressed)


#omitting missing values
is.na(b_depressed)
sum(is.na(b_depressed))
b_depressed<-na.omit(b_depressed)
sum(is.na(b_depressed))
dim(b_depressed)

set.seed(123)
split <- sort(sample(nrow(b_depressed), nrow(b_depressed)*0.5))

training <- b_depressed[split,]
```

```
testing <- b_depressed[-split,]

summary(training)
summary(testing)

dim(training)
dim(testing)



#implementing logistic regression model

log_model <- glm(depressed ~.,family = binomial,data =training[,c(3:23)])

summary(log_model)


#observing the performance of the logistic regression model

probabilities <- predict(log_model,
                    newdata = testing,
                    type = "response")
depressedPred<- ifelse(probabilities > 0.5, "1", "0")

#the confusion matrix

table(depressedPred, testing$depressed)

#accuracy

mean(depressedPred==testing$depressed)
```

Output

I observed the confusion matrix and the mean from the analysis.

```
> table(depressedPred, testing$depressed)

depressedPred   0   1
            0 582 121
            1   1   1
```

```
> mean(depressedPred==testing$depressed)
[1] 0.8269504
```

As we can see, the mean (accuracy) is 0.8269504, which is quite good. The confusion matrix shows that the model correctly predicted 582 cases where the person hasn't developed depression. But there is only one case where the model correctly predicted someone has depression, and incorrectly predicted 121 individuals having depression. So, although the mean is quite good, the number of correctly predicting cases of someone having depression is low in this model.

Applying Logistic regression to a different random training and testing set (Validation set approach)

I also did the logistic regression using a different random training and testing set. The seed function is set as ‘set.seed(1)’ this time, whereas the previous one was ‘set.seed(123)’. I applied the following code.

Code:

```
#using a different seed sampling
set.seed(1)
split <- sort(sample(nrow(b_depressed), nrow(b_depressed)*0.5))

training <- b_depressed[split,]

testing <- b_depressed[-split,]

summary(training)
summary(testing)

dim(training)
dim(testing)



#implementing logistic regression model

log_model <- glm(depressed ~.,family = binomial,data =training[,c(3:23)])

summary(log_model)


#observing the performance of the logistic regression model
```

```
probabilities <- predict(log_model,
                         newdata = testing,
                         type = "response")
depressedPred<- ifelse(probabilities > 0.5, "1", "0")

#the confusion matrix

table(depressedPred, testing$depressed)

#accuracy

mean(depressedPred==testing$depressed)
```

Output

I observed the confusion matrix and the mean from the analysis.

```
depressedPred   0   1
          0 598 106
          1   1   0
> mean(depressedPred==testing$depressed)
[1] 0.848227
```

As we can see, the mean (accuracy) is 0.848227, which is quite good and better than the previous one. The confusion matrix shows that the model correctly predicted 598 cases where the person hasn't developed depression. But there is zero case where the model correctly predicted someone has depression, and incorrectly predicted 106 individuals having depression. So, although the mean is better than the previous one, the number of correctly predicting cases of someone having depression actually dropped from 1 to zero.

KNN- k-nearest neighbors method

In k-nearest neighbors (KNN) method, an object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its *k* nearest neighbors. For implementing the KNN method, I have applied the following codes and observed k for k=3,5,7, and 10.

Code:

```
library(class)
attach(b_depressed)

#missing values
is.na(b_depressed)
sum(is.na(b_depressed))
b_depressed<-na.omit(b_depressed)
sum(is.na(b_depressed))
dim(b_depressed)

names(b_depressed)
summary(b_depressed)
b_depressed$depressed<- as.factor(b_depressed$depressed)
b_depressed$sex<- as.factor(b_depressed$sex)
b_depressed$Married<- as.factor(b_depressed$Married)
b_depressed$incoming_salary<- as.factor(b_depressed$incoming_salary)
b_depressed$incoming_own_farm<- as.factor(b_depressed$incoming_own_farm)
b_depressed$incoming_business<- as.factor(b_depressed$incoming_business)
b_depressed$incoming_no_business<- as.factor(b_depressed$incoming_no_business)
b_depressed$labor_primary<- as.factor(b_depressed$labor_primary)

summary(b_depressed)

set.seed(1)
split <- sort(sample(nrow(b_depressed), nrow(b_depressed)*0.7))

training <- b_depressed[split,]
testing <- b_depressed[-split,]


train_feat <- training[,3:23]
test_feat <- testing[,3:23]

#knn model for k=3

set.seed(1)
train_pred <- knn(train_feat, train_feat, training$depressed, k=3)
train_acc <- mean(train_pred == training$depressed)


set.seed(1)
test_pred <- knn(train_feat, test_feat, training$depressed, k=3)
```

```r
test_acc <- mean(test_pred == testing$depressed)


#Confusion Matrix for k=3
table(test_pred,testing$depressed)

cat('Training Accuracy for k=3:   ', train_acc, '\n',
        'Testing Accuracy for k=3: ', test_acc, sep='')
```

Output

I observed the confusion matrix, the training and testing accuracy for each k = 3, 5, 7, and 10.

For k=3

```
test_pred   0   1
        0 336  57
        1  23   7
> cat('Training Accuracy for k=3:   ', train_acc, '\n',
+        'Testing Accuracy for k=3: ', test_acc, sep='')
Training Accuracy for k=3:   0.8539554
Testing Accuracy for k=3: 0.8108747
```

For k=5

```
test_pred   0   1
        0 349  62
        1  10   2
> cat('Training Accuracy for k=5:   ', train_acc, '\n',
+        'Testing Accuracy for k=5: ', test_acc, sep='')
Training Accuracy for k=5:   0.836714
Testing Accuracy for k=5: 0.8297872
```

For k=7

```
test_pred   0   1
        0 355  62
        1   4   2
> cat('Training Accuracy for k=7:   ', train_acc, '\n',
+        'Testing Accuracy for k=7: ', test_acc, sep='')
Training Accuracy for k=7:   0.8286004
Testing Accuracy for k=7: 0.8439716
```

For k=10

```
test_pred   0   1
        0 356  62
        1   3   2
> cat('Training Accuracy for k=10:   ', train_acc, '\n',
+         'Testing Accuracy for k=10: ', test_acc, sep='')
Training Accuracy for k=10:   0.8296146
Testing Accuracy for k=10: 0.8463357
```

As we can see, for k=3 the testing accuracy is 0.8108747, for k=5 the testing accuracy is 0.8297872, for k=7 the testing accuracy is 0.8439716, for k=10 the testing accuracy is 0.8463357. So, the testing accuracy actually improves by increasing the k value although the increase of accuracy from k=7 to k=10 is not very significant.

On the other hand, if we look at the confusion matrix, k=3 giving the highest number of cases (7) where the model correctly predicted someone having depression.

LDA- Linear Discriminant Analysis

LDA is a method to find a linear combination of features that characterizes or separates two or more classes of objects or events. The LDA model assumes that the observations are random samples, each predictor is normally distributed, and every class has the same variance/ covariance. I applied the following codes for applying the LDA model.

Code

```
library(class)
attach(b_depressed)
library(MASS)
library(ROCR)

library(tidyverse)
library(caret)

#missing values
is.na(b_depressed)
sum(is.na(b_depressed))
b_depressed<-na.omit(b_depressed)
sum(is.na(b_depressed))
```

```
dim(b_depressed)

names(b_depressed)
summary(b_depressed)
b_depressed$depressed<- as.factor(b_depressed$depressed)
b_depressed$sex<- as.factor(b_depressed$sex)
b_depressed$Married<- as.factor(b_depressed$Married)
b_depressed$incoming_salary<- as.factor(b_depressed$incoming_salary)
b_depressed$incoming_own_farm<- as.factor(b_depressed$incoming_own_farm)
b_depressed$incoming_business<- as.factor(b_depressed$incoming_business)
b_depressed$incoming_no_business<- as.factor(b_depressed$incoming_no_business)
b_depressed$labor_primary<- as.factor(b_depressed$labor_primary)

summary(b_depressed)

set.seed(1)
split <- sort(sample(nrow(b_depressed), nrow(b_depressed)*0.7))

training <- b_depressed[split,]
testing <- b_depressed[-split,]


#LDA

ldamodel <- lda(depressed~., data = training[,c(3:23)])

# Make predictions
ldapredictions <- ldamodel %>% predict(testing)

table(ldapredictions$class,testing$depressed)

# Model accuracy
mean(ldapredictions$class==testing$depressed)
```

Output

I observed the confusion matrix and the mean from the analysis.

```
> table(ldapredictions$class,testing$depressed)

      0   1
```

```
  0 358  63
  1   1   1
> # Model accuracy
> mean(ldapredictions$class==testing$depressed)
[1] 0.8486998
```

QDA- Quadratic Discriminant Analysis

QDA works identically as LDA except that it estimates separate variances/ covariance for each class. I have applied the following code to perform the QDA model.

Code:

```
#QDA

qdamodel <- qda(depressed~., data = training[,c(3:23)])

# Make predictions
qdapredictions <- qdamodel %>% predict(testing)

table(qdapredictions$class,testing$depressed)

# Model accuracy
mean(qdapredictions$class==testing$depressed)
```

Output:

I observed the confusion matrix and the mean from the analysis.

```
> table(qdapredictions$class,testing$depressed)

       0   1
  0 319  56
  1  40   8
> # Model accuracy
> mean(qdapredictions$class==testing$depressed)
[1] 0.7730496
```

As we can see, the LDA model gives a higher accuracy (0.8486998) than the QDA model (0.7730496). But from the confusion matrix we can see that the QDA model correctly predicts 8 cases of someone having depression whereas the LDA model correctly predicts just one. In terms of accuracy LDA is a better model, so it can be inferred that the variances are similar among classes or we don't have enough data to accurately estimate the variances. The QDA model is also doing well as it also has quite good accuracy.

Leave-One-Out Cross-Validation

Cross-validation (CV) is a resampling method that uses different portions of the data to test and train a model on different iterations. I applied the following codes to implement leave one out cross-validation.

Code

```
glm.fit=glm(depressed~.,family = binomial,data=training[,c(3:23)])
#leave one out cross validation, k unspecified
cv.err=cv.glm(training[,c(3:23)],glm.fit)
cv.err$delta
```

Output

Observed the cross validation error.

```
> cv.err$delta
[1] 0.1433677 0.1433643
```

As we can see, the cross validation error is 0.1433677. Which is quite good.

CV to evaluate Polynomial models with different degrees

I have implemented the following codes for CV to evaluate polynomial models with different degrees. In doing so, I applied glm function and as variables I took five columns named as the Age, no_lasting_investmen, education_level, arm_expenses, and other_expenses.

Code

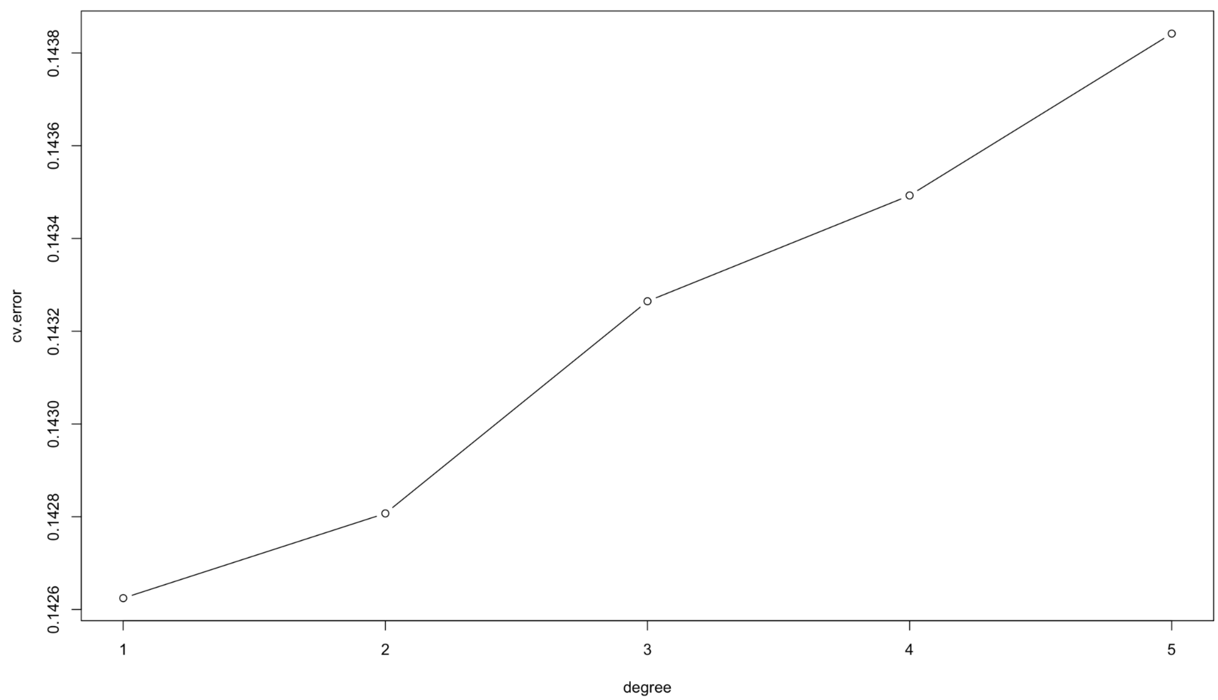
```
#cv to evaluate Polynomial models with different degrees
cv.error=rep(0,5)
for (i in 1:5){
```

```
glm.fit=glm(depressed~poly(Age+no_lasting_investmen+education_level+farm_expenses+other
_expenses,i),family = binomial,data=training[,c(3:23)])
  cv.error[i]=cv.glm(training[,c(3:23)],glm.fit)$delta[1]
}
cv.error
degree=1:5
plot(degree,cv.error, type="b")
```

Output

I observed the plot for CV error with degrees from 1 to 5.

```
> plot(degree,cv.error, type="b")
```



As we can see from the plot, actually the degree 1 gives the lowest CV error.

10-Fold Cross-Validation

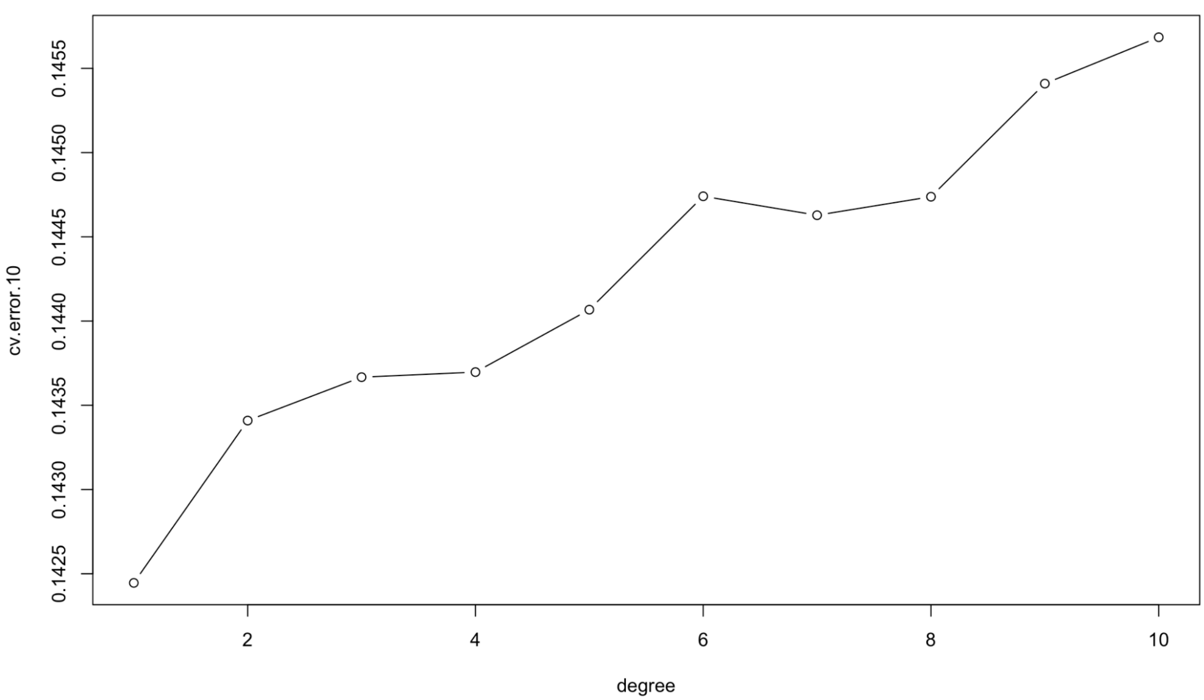
I have applied the following codes for implementing 10 (k)- fold cross validation.

Code

```
set.seed(17)
cv.error.10=rep(0,10)
for (i in 1:10){

glm.fit=glm(depressed~poly(Age+no_lasting_investmen+education_level+farm_expenses+other
_expenses,i),family = binomial,data=training[,c(3:23)])
  cv.error.10[i]=cv.glm(training[,c(3:23)],glm.fit,K=10)$delta[1]
}
cv.error.10
degree=1:10

#type="b": plot for both points and lines
plot(degree,cv.error.10, type="b")
```

Output

I observed the plot for CV error for 10-fold cross-validation for degrees 1 to 10



As we can see from the plot, the 10-fold CV plot is a bit different from the leave-one-out CV plot but still the degree 1 is giving the lowest CV error.

Best subset selection

“Best subset selection is a method that aims to find the subset of independent variables that best predict the outcome and it does so by considering all possible combinations of independent variables” [3]. I applied the following codes to implement best subset selection method

Code

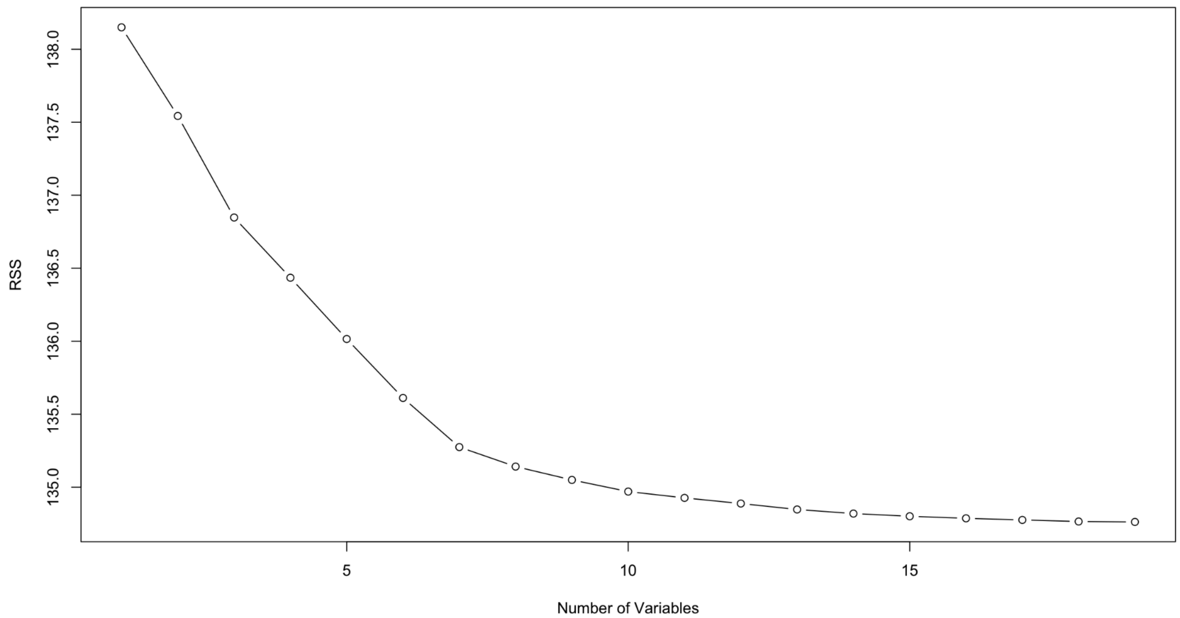
```
regfit.full=regsubsets(depressed~.,training[,c(3:23)])
?regsubsets
summary(regfit.full)
regfit.full=regsubsets(depressed~.,data=training[,c(3:23)],nvmax=19)
reg.summary=summary(regfit.full)
reg.summary
names(reg.summary)
reg.summary$rsq
plot(reg.summary$rss,xlab="Number of Variables",ylab="RSS",type="b")
plot(reg.summary$adjr2,xlab="Number of Variables",ylab="Adjusted RSq",type="b")
which.max(reg.summary$adjr2)
points(11,reg.summary$adjr2[11], col="red",pch=20)
plot(reg.summary$cp,xlab="Number of Variables",ylab="Cp",type='b')
which.min(reg.summary$cp)
points(10,reg.summary$cp[10],col="red",pch=20)
plot(reg.summary$bic,xlab="Number of Variables",ylab="BIC",type='b')
which.min(reg.summary$bic)
points(6,reg.summary$bic[6],col="red",pch=20)
#regsubsets function to plot all subsets and their corresponding statistics specified in "scale"
plot(regfit.full,scale="r2")
plot(regfit.full,scale="adjr2")
plot(regfit.full,scale="Cp")
plot(regfit.full,scale="bic")

coef(regfit.full,1)
coef(regfit.full,7)
```

Output

```
> names(reg.summary)
[1] "which"  "rsq"    "rss"    "adjr2"  "cp"     "bic"    "outmat" "obj"
> reg.summary$rsq
 [1] 0.01344751 0.01778394 0.02275089 0.02569736 0.02869445 0.03157832 0.03398271
 [8] 0.03493630 0.03558941 0.03615864 0.03646795 0.03674395 0.03703588 0.03723580
```

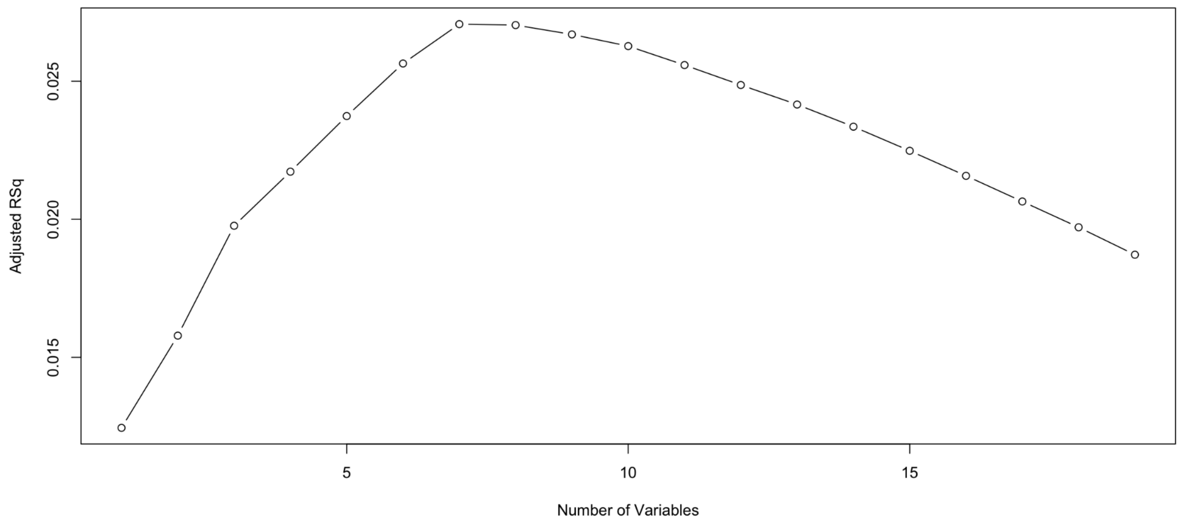
[15] 0.03736726 0.03746762 0.03754691 0.03762391 0.03764500

> plot(reg.summary$rss,xlab="Number of Variables",ylab="RSS",type="b")



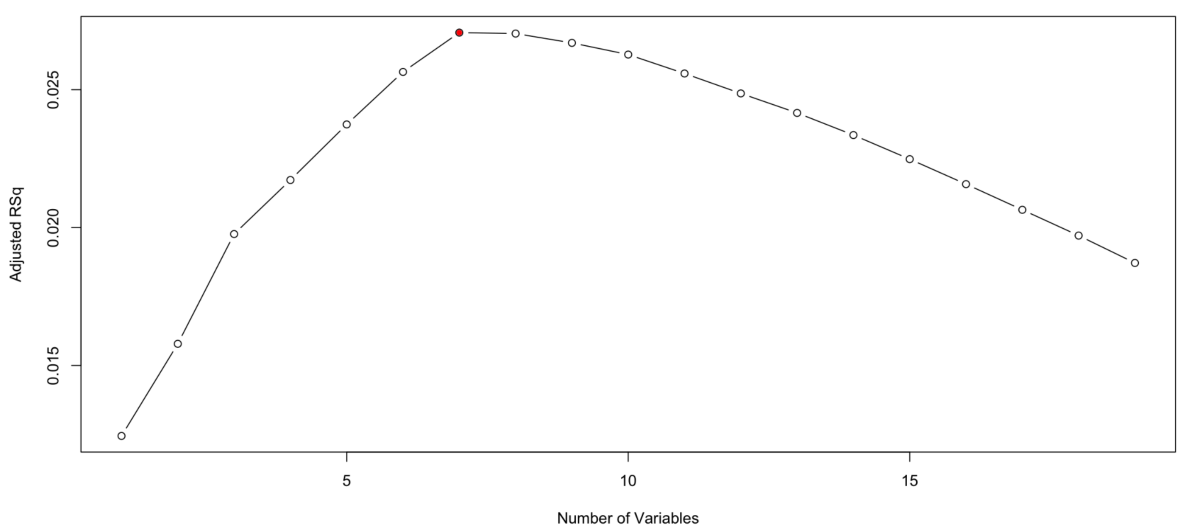
As we can see from the plot, with the increase of number of variables, the RSS goes down.

> plot(reg.summary$adjr2,xlab="Number of Variables",ylab="Adjusted RSq",type="b")



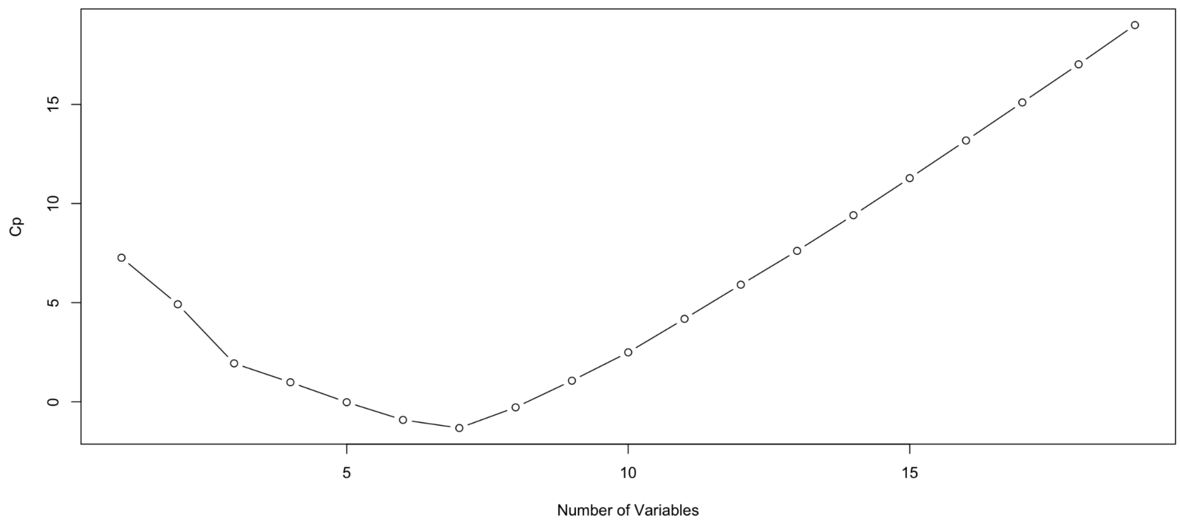
> which.max(reg.summary$adjr2)
[1] 7

> points(7,reg.summary$adjr2[7], col="red",pch=20)

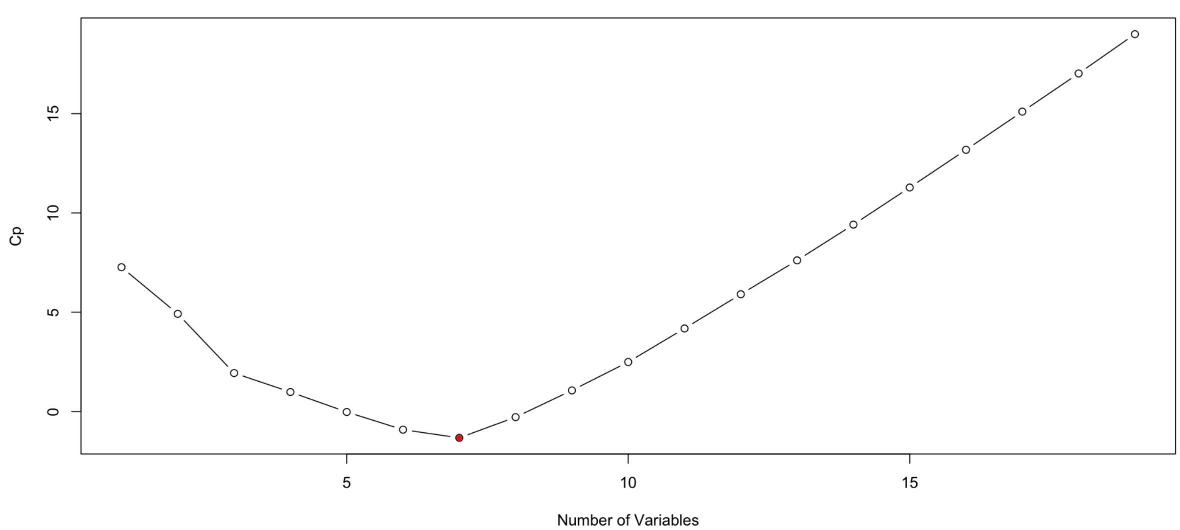


As we can see from the plot, the highest Adjusted RSq happens when the number of variables is 7.

```
> plot(reg.summary$cp,xlab="Number of Variables",ylab="Cp",type='b')
```

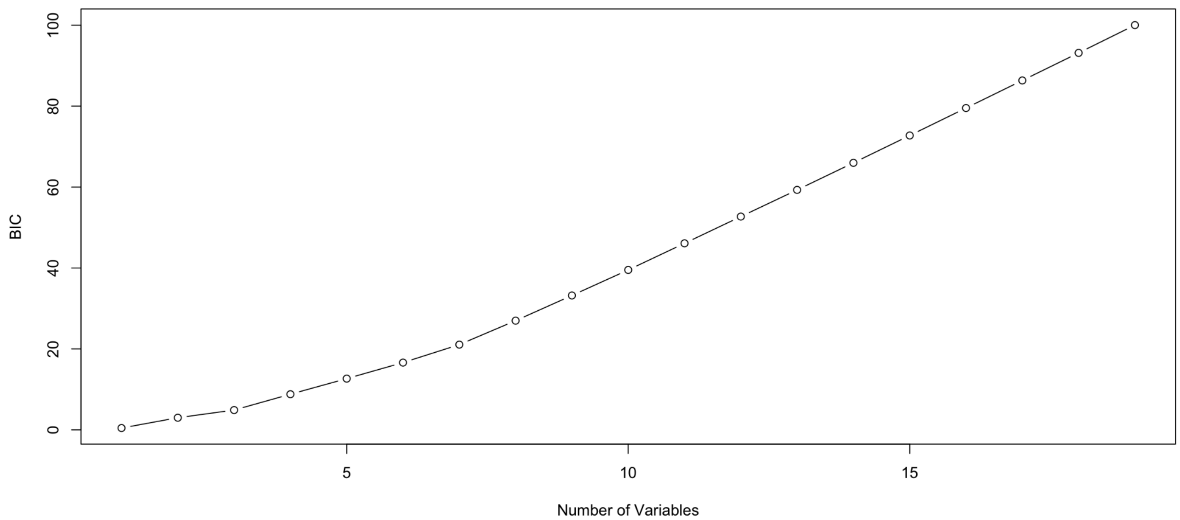



```
> which.min(reg.summary$cp)
[1] 7

> points(7,reg.summary$cp[7],col="red",pch=20)
```

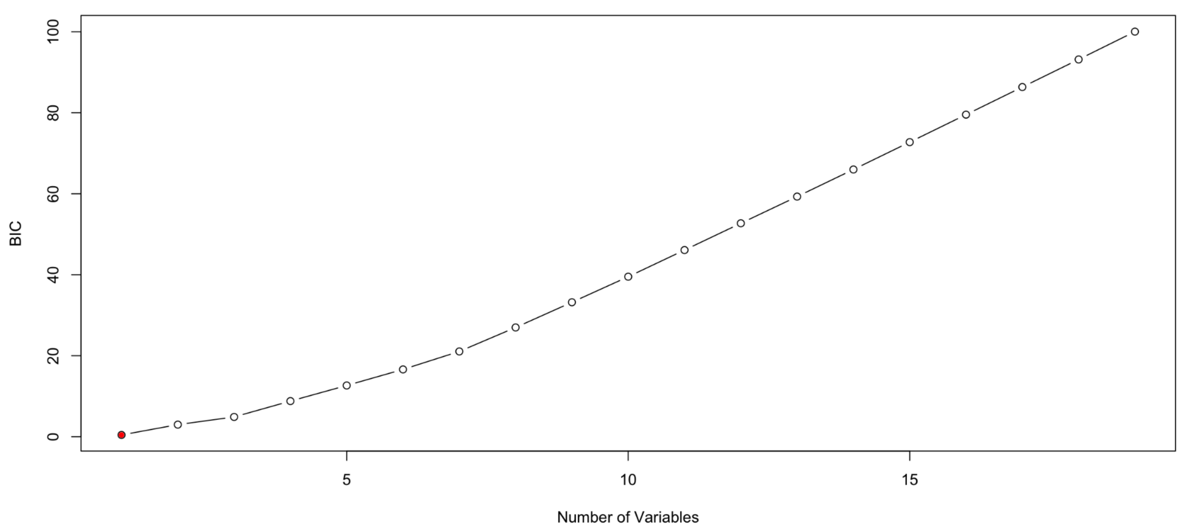


As we can see from the plot, the lowest Cp happens when the number of variables is 7.

```
> plot(reg.summary$bic,xlab="Number of Variables",ylab="BIC",type='b')
```

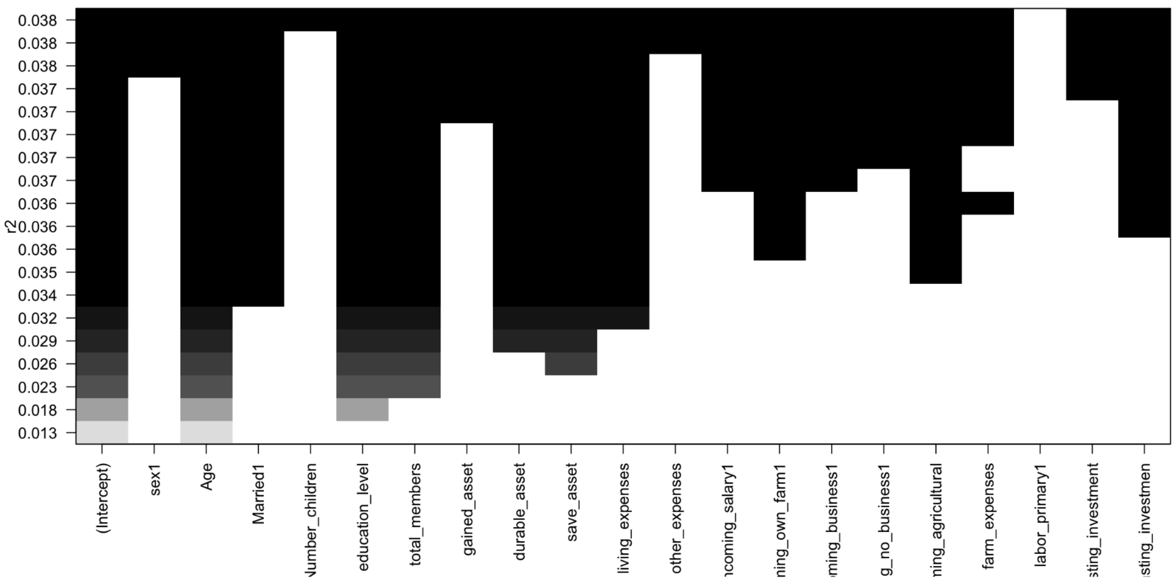



```
> which.min(reg.summary$bic)
[1] 1
```

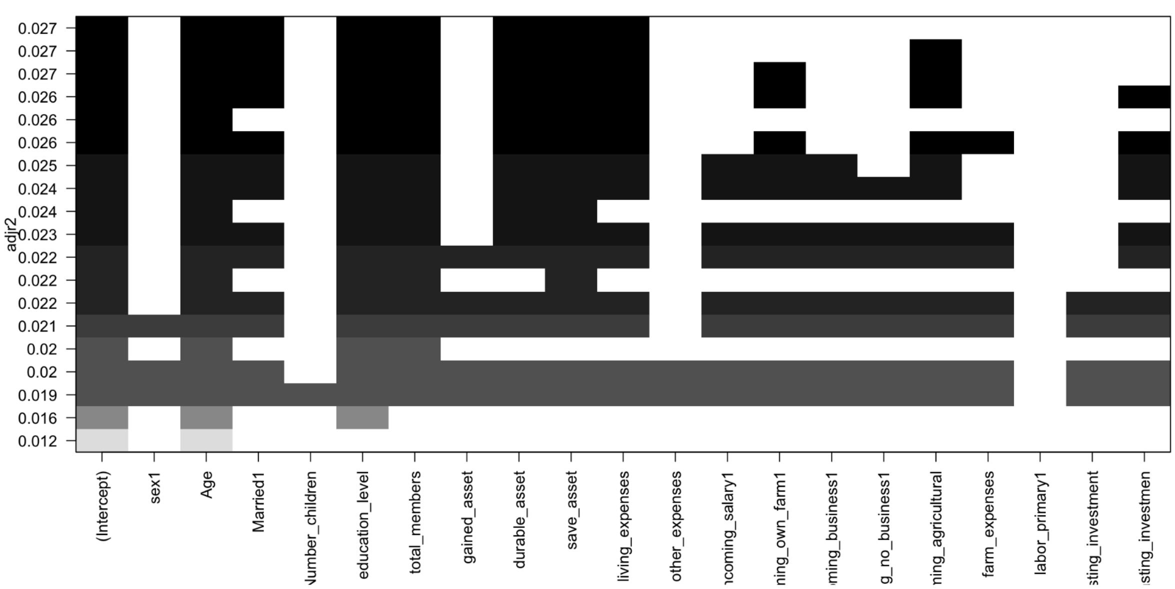


As we can see from the plot, the lowest BIC happens when the number of variables is 1.

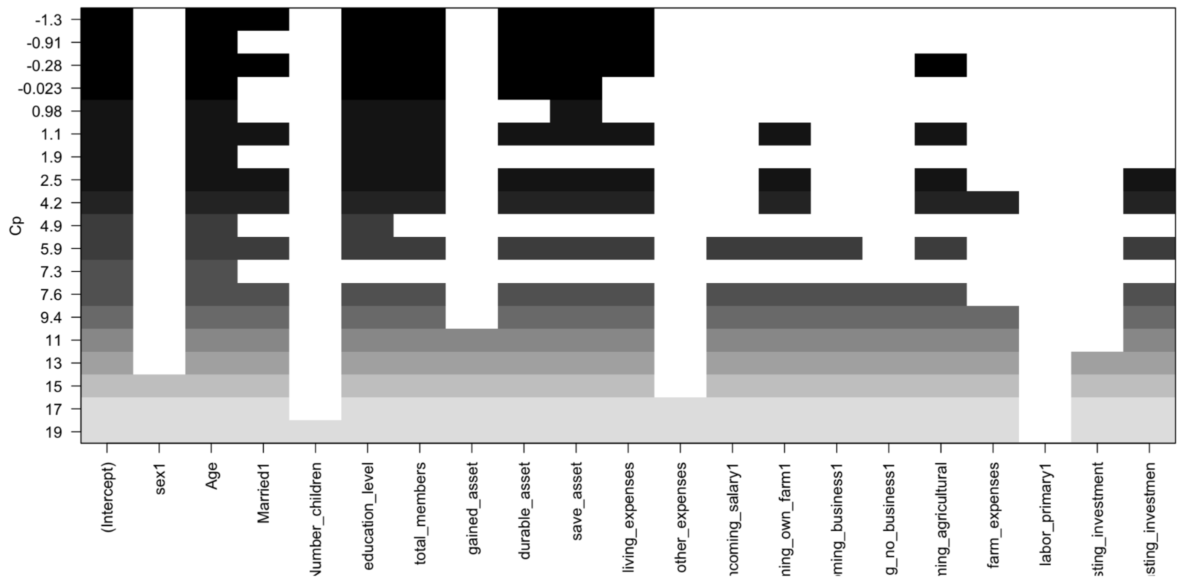
> plot(regfit.full,scale="r2")



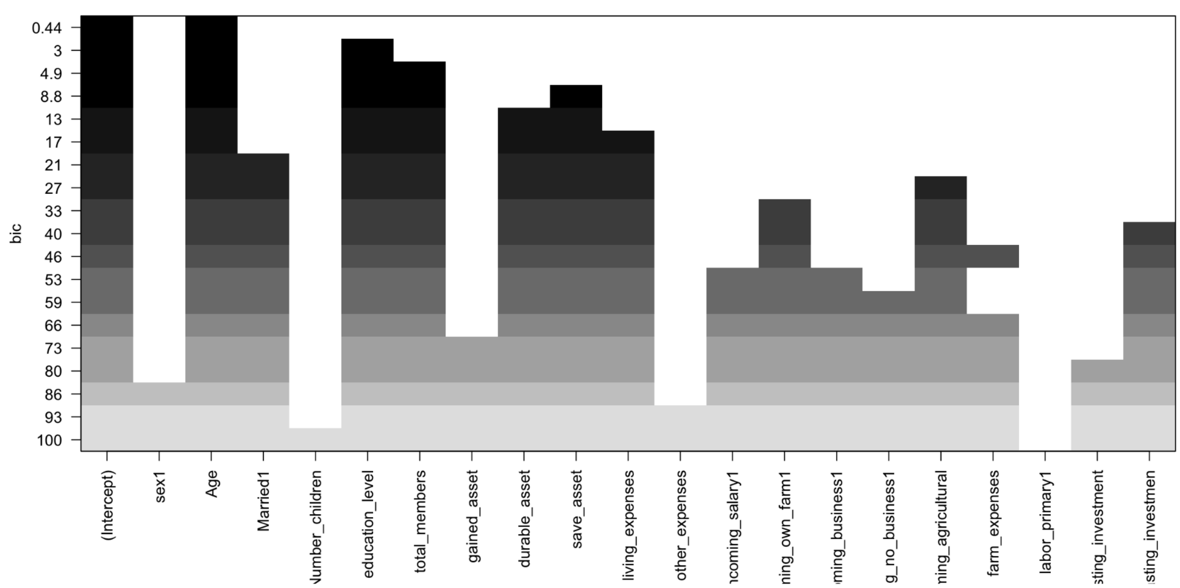
> plot(regfit.full,scale="adjr2")



> plot(regfit.full,scale="Cp")



> plot(regfit.full,scale="bic")



Now I observe the one most influential variable according to this method.

> coef(regfit.full,1)
(Intercept) Age

1.063039756 0.003128724

As we can see the one most influential variable according to this method is the ‘Age’.

Now I observe the top seven influential variable according to this method.

```
> coef(regfit.full,7)
      (Intercept)             Age     Married1 education_level   total_members
     1.118609e+00    1.835969e-03  -4.841129e-02  -1.034727e-02    1.684815e-02
    durable_asset      save_asset living_expenses
     1.133159e-09    1.198798e-09  -9.563247e-10
```

As we can see, the top seven influential variables according to this method are, Age, Married, education level, total members, durable asset, save_asset, and living_expenses.

Forward and Backward Stepwise Selection

Forward stepwise selection begins with a model containing no predictors, and then adds predictors to the model, one-at-a-time, until all of the predictors are in the model. On the other hand, backward stepwise selection begins with the full least squares model containing all p predictors, and then iteratively removes the least useful predictor, one-at-a-time. I have implemented the following codes to implement forward and backward stepwise selection.

Code:

```
regfit.fwd=regsubsets(depressed~.,data=training[,c(3:23)],nvmax=19,method="forward")
summary(regfit.fwd)
regfit.bwd=regsubsets(depressed~.,data=training[,c(3:23)],nvmax=19,method="backward")
summary(regfit.bwd)
coef(regfit.full,7)
reg.summary=summary(regfit.full)
reg.summary$adjr2[7]
coef(regfit.fwd,7)
reg.summary=summary(regfit.fwd)
reg.summary$adjr2[7]
coef(regfit.bwd,7)
reg.summary=summary(regfit.bwd)
reg.summary$adjr2[7]
```

Output

Result for best subset selection:

```
> coef(regfit.full,7)
     (Intercept)             Age      Married1 education_level   total_members
    1.118609e+00    1.835969e-03  -4.841129e-02  -1.034727e-02    1.684815e-02
   durable_asset      save_asset living_expenses
    1.133159e-09    1.198798e-09  -9.563247e-10

> reg.summary$adjr2[7]
[1] 0.02706847
```

Result for forward stepwise selection selection:

```
> coef(regfit.fwd,7)
     (Intercept)             Age      Married1 education_level   total_members
    1.118609e+00    1.835969e-03  -4.841129e-02  -1.034727e-02    1.684815e-02
   durable_asset      save_asset living_expenses
    1.133159e-09    1.198798e-09  -9.563247e-10
> reg.summary$adjr2[7]
[1] 0.02706847
```

Result for backward stepwise selection selection:

```
> coef(regfit.bwd,7)
     (Intercept)             Age      Married1 education_level   total_members
    1.118609e+00    1.835969e-03  -4.841129e-02  -1.034727e-02    1.684815e-02
   durable_asset      save_asset living_expenses
    1.133159e-09    1.198798e-09  -9.563247e-10
> reg.summary$adjr2[7]
[1] 0.02706847
```

As we can see above, the forward and backward stepwise selection gives the identical results as the best subset selection for my dataset.

Polynomial Logistic Regression

Code

```
#predicting depression using a fourth-degree polynomial in `Age`:


log_model <- glm(depressed ~ poly(Age, 4),family = binomial,data =training[,c(3:23)])
```

```
summary(log_model)

probabilities <- predict(log_model,
                         newdata = testing,
                         type = "response")
depressedPred<- ifelse(probabilities > 0.5, "1", "0")

#the confusion matrix

table(depressedPred, testing$depressed)

#accuracy

mean(depressedPred==testing$depressed)
```

Output

I observed the confusion matrix and the mean from the analysis.

```
> table(depressedPred, testing$depressed)

depressedPred   0   1
            0 356  63
            1   3   1
> mean(depressedPred==testing$depressed)
[1] 0.8439716
```

As we can see, the accuracy after applying polynomial logistic regression is very good, which is 0.8439716. The confusion matrix shows that the model is good at correctly predicting people who don't have depression (356 correct predictions). But the model could predict only one case correctly who has depression.

Classification tree

I applied the following codes to implement a simple classification tree.

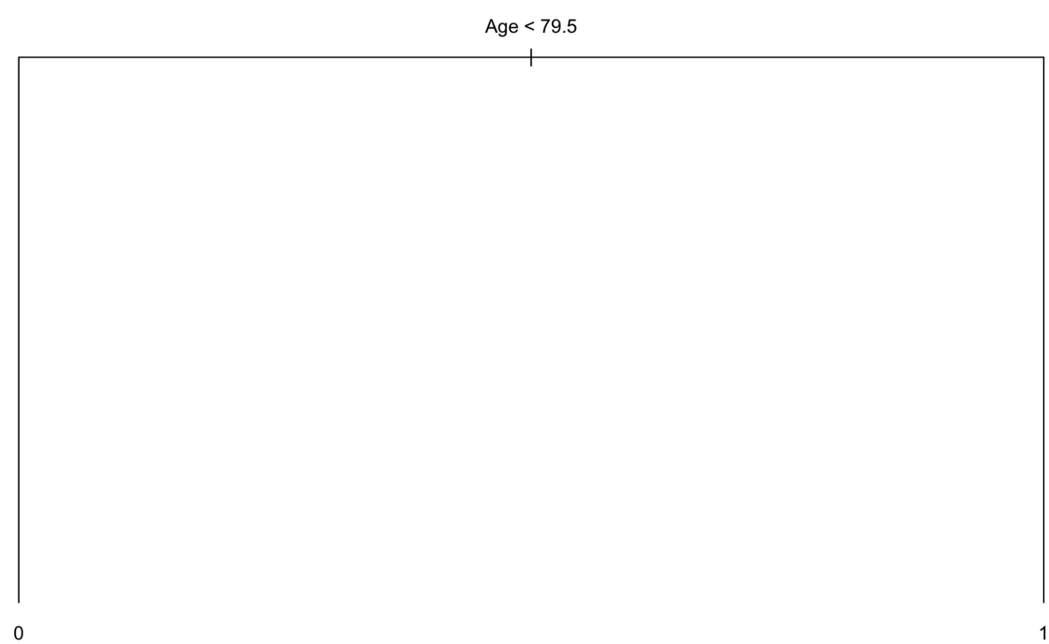
Code

```
library(tree)
```

```
tree.depressed <- tree(depressed ~ ., training[,c(3:22,23)])
###
summary(tree.depressed)
###
plot(tree.depressed)
text(tree.depressed, pretty = 0)
```

Output

```
> summary(tree.depressed)

Classification tree:
tree(formula = depressed ~ ., data = training[, c(3:22, 23)])
Variables actually used in tree construction:
[1] "Age"
Number of terminal nodes:  2
Residual mean deviance:  0.9113 = 896.7 / 984
Misclassification error rate: 0.1694 = 167 / 986

> plot(tree.depressed)
> text(tree.depressed, pretty = 0)
```



At output, I have plotted the classification tree. The variable that was used to plot the classification tree is ‘Age’, which is the most influential variable.

Another way to implement classification tree

I have implemented another code to implement the classification tree, that is the following.

Code

```
modFit <- caret::train(depressed ~ ., method = "rpart", data = training[,c(3:22,23)])

predictions <- predict(modFit, newdata = testing[,c(3:22,23)])
confusionMatrix(predictions, testing$depressed)
```

Output

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 359  64
         1   0   0

               Accuracy : 0.8487
                 95% CI : (0.8109, 0.8815)
    No Information Rate : 0.8487
    P-Value [Acc > NIR] : 0.5333

                  Kappa : 0

 Mcnemar's Test P-Value : 3.407e-15

            Sensitivity : 1.0000
            Specificity : 0.0000
         Pos Pred Value : 0.8487
         Neg Pred Value :    NaN
             Prevalence : 0.8487
         Detection Rate : 0.8487
   Detection Prevalence : 1.0000
      Balanced Accuracy : 0.5000

       'Positive' Class : 0
```

The output shows that the accuracy is good, which is 0.8487. The confusion matrix showed that the model could predict zero cases correctly who has developed depression.

Random forests

“Random forests or random decision forests is an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees.” [4] I have applied the following codes to implement the random forest method on my working dataset.

Code

```
# random forest using all predictors

modFit.rf <- randomForest::randomForest(depressed ~ ., data = training[,c(3:23)])

modFit.rf

predictions.rf <- predict(modFit.rf, newdata = testing[,c(3:23)])
confusionMatrix(predictions.rf, testing$depressed)

plot(modFit.rf, main = "Error rate of random forest")

varImpPlot(modFit.rf, pch = 20, main = "Importance of Variables")
```

Output

```
> confusionMatrix(predictions.rf, testing$depressed)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
          0 357  62
          1   2   2

          Accuracy : 0.8487
                    95% CI : (0.8109, 0.8815)
          No Information Rate : 0.8487
          P-Value [Acc > NIR] : 0.5333

                    Kappa : 0.0418
```

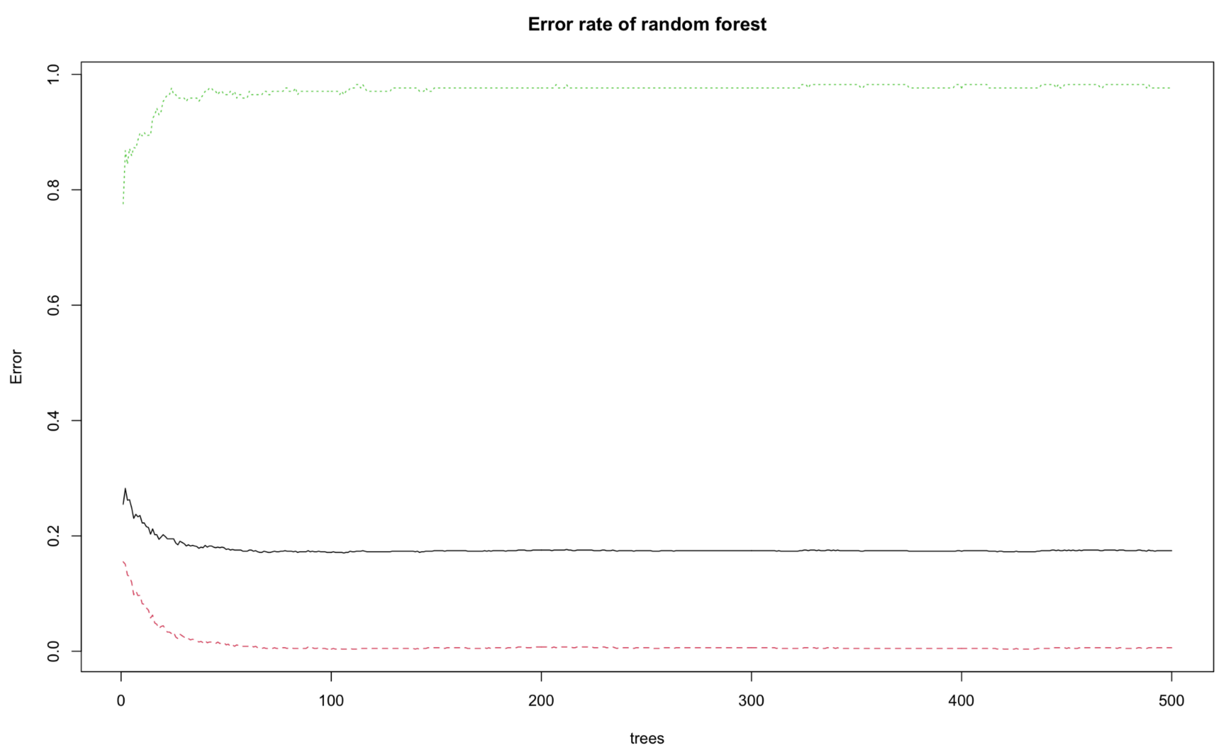
Mcnemar's Test P-Value : 1.643e-13

Sensitivity : 0.99443
Specificity : 0.03125
Pos Pred Value : 0.85203
Neg Pred Value : 0.50000
Prevalence : 0.84870
Detection Rate : 0.84397
Detection Prevalence : 0.99054
Balanced Accuracy : 0.51284

'Positive' Class : 0

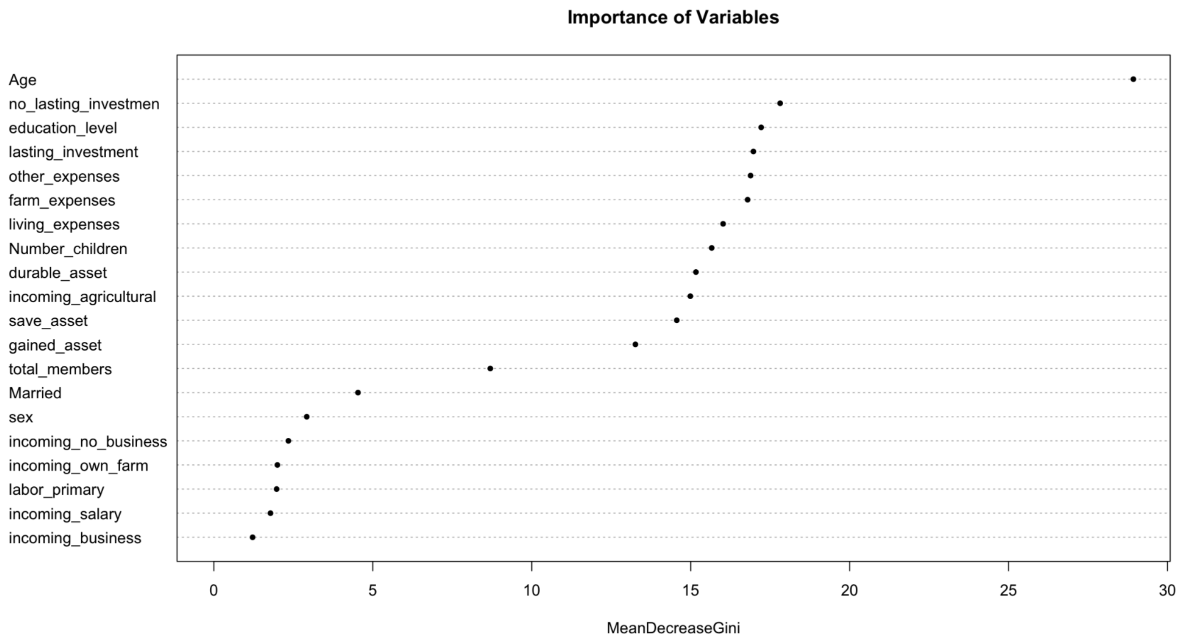
As we can see, the accuracy comes out very good using the random forest method which is 0.8487. The confusion matrix shows that the model is good at correctly predicting people who don't have depression (357 correct predictions). But the model could predict only two cases correctly who has depression.

> plot(modFit.rf, main = "Error rate of random forest")



As we can see from the Error rate of random forest plot the error rate kind of stabilizes after around 70.

> varImpPlot(modFit.rf, pch = 20, main = "Importance of Variables")



I observed the Importance of variables plot to see which variables are most important according to this method. As we can see, the most important variable is ‘Age’. According to their importance, the variables are in the following order.

Age

no_lasting_investmen

education_level

lasting_investment

other_expenses

farm_expenses

living_expenses

Number_children

durable_asset

incoming_agricultural

save_asset

gained_asset

total_members

Married

sex

incoming_no_business

Incoming_own_farm

labor_primary

incoming salary

Incoming_business

This ‘importance of Variables’ plot gives great insights about the data. The relative importance of different variables can be inferred from this plot.

In terms of the most important variable, this method (Random forest) and the best subset selection method (also forward and backward stepwise selection) agree with each other. According to both methods the most important variable is ‘Age’. Let's compare the first 7 important variables in these two methods in the following table.

Serial	Seven important variables by Random forest model	Seven important variables by best subset selection model
1	Age	Age
2	no_lasting_investmen	Married
3	education_level	education_level
4	lasting_investment	total_members
5	other_expenses	durable_asset

6	farm_expenses	save_asset
7	living_expenses	living_expenses

It looks like the ‘Age’, ‘education_level’, and ‘living_expenses’ are common in both models as important variables.

Discussion and conclusion

I have run several statistical learning models on the b_depression dataset. The accuracy of all the models implemented came out to be close, most of them are in the range of 0.80 to 0.85. The only model that scored less accuracy than that range is by the QDA model which is 0.7730496. Despite having high accuracy in most models, the case of correctly detecting whether someone has depression is consistently low among all the models which is revealed by the confusion matrices. The biggest number in this case is 8 by the QDA model and the next one is 7 by the k=3 KNN model (accuracy 0.8108747). The reason for most of the models having a high accuracy is, the models are doing very well on correctly predicting the individuals who don’t have depression. While correctly predicting the individuals who don’t have depression is important, for my analysis it is more important to correctly predict individuals who have depression where the models don’t seem to do great. Having said that, it is quite evident from the results by different models that it is possible to build good predictive models based on the data I have worked on. The best performing model among the model I ran would be the ‘random forests’ one with the accuracy of 0.8487 and predicting two cases correctly who have depression. In regards to different variables, it is quite evident from multiple analyses (random forests, best subset selection, classification tree) that the ‘Age’ is the most influential variable among all the variables. Two other influential factors would be ‘education_level’, ‘living_expenses’. Other than these three, ‘no_lasting_investmen’, ‘lasting_investment’, ‘Married’, ‘total_members’, ‘other_expenses’, ‘durable_asset’, ‘farm_expenses’, ‘save_asset’ play important roles. So as we can see, among the variables which are important towards the prediction of depression, there are different personal & social factors. Also, among these important variables several of them are related to an individual's financial situation. So that also plays a role in the process of developing depression.

In an overall sense, the statistical learning models that have been run for detecting depression performed quite good in terms of accuracy although the models could not do great in correctly predicting the individuals who have depression in many cases. So, the accuracy shows a good predictive power in the data but it is not even for all cases. As other studies [2] show that depression can result from a complex interaction of social, psychological and biological factors, quantitative statistical analysis may not always predict everything about depression correctly.

Also, many other issues for example the context of the data collection process may also influence the prediction.

References

[1] Institute of Health Metrics and Evaluation. Global Health Data Exchange (GHDx). Retrieved February 13, 2022, from http://ghdx.healthdata.org/gbd-results-tool?params=gbd-api-2019-permalink/d780dffbe8a381b25e1416884959e88b

[2] World Health Organization. (n.d.). *Depression*. World Health Organization. Retrieved February 13, 2022, from https://www.who.int/news-room/fact-sheets/detail/depression

[3] Understand Best Subset Selection – Quantifying Health. (n.d.). https://quantifyinghealth.com/best-subset-selection/

[4] Ho, Tin Kam (1995). *Random Decision Forests* (PDF). Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC, 14–16 August 1995. pp. 278–282.