Introduction:

Here we first work on the data called Job Satisfaction. This data set have four columns namely- Job Satisfaction(Coded as 0 and 1: 0 for not satisfied and 1 for satisfied). The first column is dichotomous or binary in nature. The second column gives the income of the person. It is a numerical column. The third column is also a numerical column called Expenditure. The second and third column is continuous in nature. The fourth column contains Years_In_Job in terms of numbers 1,2,3,4. This column is ordinal type qualitative variable.

Our Goal:

Here we are to construct a Binary Multiple Logistic Regression Model by using the glm() function. This whole process has three steps. They are-

- First step is to understand the data very well.
- Second step is changing the data as per requirement.
- The last step is to construct the glm() function.

Starting of the project:

First, we install, and load required packages for this project.

```
install.packages("glmtoolbox")
install.packages("tidyverse")
install.packages("aod")
library(dplyr)
library(glmtoolbox)
library(aod)
```

Reading the Job_Satisfaction.csv file

```
data <- as.data.frame(read.csv("Job_Satisfaction.csv"))
```

Basic data Exploratory works

- Observing first few rows
- Observing the structure
- Getting the summary of the data frame

- Getting the dimensions
- Finding standard deviation of columns
- Getting the column names of the data frame

```
# Observing the first few rows of the data frame
head(data,10)
```

	Job_Satisfection <int>	Income <int>	Expenditure <int>
1	1	85000	48000
2	0	56000	45000
3	0	25000	5628
4	0	24015	15685
5	0	45623	39856
6	0	15000	9462
7	0	26459	58000
8	1	118796	11254
9	0	22667	105000
10	0	37379	88608

1-10 of 10 rows

```
# Observing the data structure and type of elements
str(data)
'data.frame':	100 obs. of  4 variables:
 $ Job_Satisfection: int  1 0 0 0 0 0 0 1 0 0 ...
 $ Income          : int  85000 56000 25000 24015 45623 15000 26459 118796 22667 3737
9 ...
 $ Expenditure     : int  48000 45000 5628 15685 39856 9462 58000 11254 105000 88608
...
 $ Years_In_Job    : int  3 4 4 3 4 3 2 1 4 4 ...
```

```
# Getting the summary of the data frame
summary(data)
 Job_Satisfection     Income          Expenditure       Years_In_Job
 Min.   :0.00     Min.   : 15000   Min.   :  5628   Min.   :1.00
```

```
 1st Qu.:0.00     1st Qu.: 45411   1st Qu.: 42302   1st Qu.:1.00  
 Median :0.00     Median : 73355   Median : 76323   Median :3.00  
 Mean   :0.31     Mean   : 78576   Mean   : 71949   Mean   :2.58  
 3rd Qu.:1.00     3rd Qu.:115601   3rd Qu.:102147   3rd Qu.:4.00  
 Max.   :1.00     Max.   :144858   Max.   :124659   Max.   :4.00  
```

```
# Getting the dimensions of the data frame
dim(data)
[1] 100   4
```

Hide

```
# Finding the standard deviation of the columns
round(sapply(data,sd),4)
Job_Satisfection           Income      Expenditure      Years_In_Job 
          0.4648       40179.7093       34155.3049          1.1992 
```

Hide

```
# Getting the column names of the data frame
colnames(data)
[1] "Job_Satisfection" "Income"           "Expenditure"      "Years_In_Job"
```

By mistake we write the spelling of the first column wrong. To correct this, this should be job_Satisfaction. The code for this job is given below. By using the head()function we confirm that the below code is correctly executed.

```
data <- data %>% rename(Job_Satisfaction=Job_Satisfection)
head(data,5)
```

	Job_Satisfaction <int>	Income <int>	Expenditure <int>
1	1	85000	48000
2	0	56000	45000

	Job_Satisfaction <int>	Income <int>	Expenditure <int>
3	0	25000	5628
4	0	24015	15685
5	0	45623	39856

5 rows

Now we check for any missing values in the data frame.

Hide

```
sum(is.na(data))
```

```
[1] 0
```

Next, we generate cross tabulations to understand pattern in the data set.

Hide

```
table(data$Job_Satisfaction , data$Years_In_Job)
```

```
     1  2  3  4
  0 20 15 10 24
  1  6  8  8  9
```

Note:

We observe that R calculates the standard deviation of each column. This is something we don’t need. We need the response variable Job_Satisfaction as categorical variable having two categories “0” and “1”. On the other hand, we want the column named Yeas_In_Job as an ordinal variable having 4 levels “1”,“2”,“3” and “4” respectively. To convert them in categorical variables we use the following lines of codes.

Hide

```
data$Job_Satisfaction <- factor(data$Job_Satisfaction)
data$Years_In_Job <- factor(data$Years_In_Job , levels = c(1,2,3,4) , ordered=TRUE)
attach(data)
```

Now after converting them into categorical column, we check the structure of the data

frame again to confirm that above code is executed properly.

```
str(data)
```

```
'data.frame':   100 obs. of  4 variables:
 $ Job_Satisfaction: Factor w/ 2 levels "0","1": 2 1 1 1 1 1 1 2 1 1 ...
 $ Income          : int  85000 56000 25000 24015 45623 15000 26459 118796 22667 3737
9 ...
 $ Expenditure     : int  48000 45000 5628 15685 39856 9462 58000 11254 105000 88608
...
 $ Years_In_Job    : Ord.factor w/ 4 levels "1"<"2"<"3"<"4": 3 4 4 3 4 3 2 1 4 4 ...
```

Using the summary function again

To check the difference in results before and after converting the column into categorical variable.

```
summary(data)
```

```
 Job_Satisfaction     Income        Expenditure     Years_In_Job
 0:69             Min.   : 15000   Min.   :  5628   1:26
 1:31             1st Qu.: 45411   1st Qu.: 42302   2:23
                  Median : 73355   Median : 76323   3:18
                  Mean   : 78576   Mean   : 71949   4:33
                  3rd Qu.:115601   3rd Qu.:102147
                  Max.   :144858   Max.   :124659
```

Note

Now observe that after converting the first and fourth column into categorical variables the summary function does not give all six characteristics of these columns as previous. Now the summary function only returns the frequencies. It says there are 69 persons that are not job satisfied, 31 persons are job satisfied and so on.

Constructing Binary Multiple Logistic Regression Model

In this case we consider the Job_Satisfaction as response or dependent variable and the other three columns namely the Income, the Expenditure and Years_In_Job are considered as predictor, exploratory or independent variable. The code given below constructs the model. To build this model we use the glm()function.

Hide

```
model <- glm(data=data , Job_Satisfaction ~ Income + Expenditure + Years_In_Job, fami
ly="binomial")
```

```
Warning: glm.fit: algorithm did not converge Warning: glm. Fit: fitted probabilities
numerically 0 or 1 occurred
```

Hide

```
summary(model)
```

```

Call:
glm(formula = Job_Satisfaction ~ Income + Expenditure + Years_In_Job,
    family = "binomial", data = data)

Coefficients:
                 Estimate Std. Error z value Pr(>|z|)
(Intercept)    -7.741e+01  3.824e+05   0.000    1.000
Income          3.056e-03  1.033e+00   0.003    0.998
Expenditure    -3.309e-03  1.134e+00  -0.003    0.998
Years_In_Job.L  1.548e+01  3.432e+05   0.000    1.000
Years_In_Job.Q -2.079e+01  7.620e+05   0.000    1.000
Years_In_Job.C -1.877e+01  1.022e+06   0.000    1.000

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1.2382e+02  on 99  degrees of freedom
Residual deviance: 1.6386e-08  on 94  degrees of freedom
AIC: 12

Number of Fisher Scoring iterations: 25
```

```
round(coef(model),4)
```

```
   (Intercept)         Income    Expenditure Years_In_Job.L Years_In_Job.Q Years_In_J
ob.C
      -77.4146         0.0031        -0.0033        15.4800       -20.7910       -18.
7701
```

Interpretation of the model:

From the model we can say that * For 1 unit increase in Income, the log odds of being Job satisfied increases by 0.0031 unit with respect to be not job satisfied.

- For 1 unit increase in Expenditure, the log odds of being Job satisfied decreases by 0.0033 unit with respect to be not job satisfied.
- For 1 unit increase in Years_In_Job(2 years , denoted by L),the log odds of being Job satisfied increases by 15.4800 unit with respect to be not job satisfied.
- For 1 unit increase in Years_In_Job(3 years , denoted by Q),the log odds of being Job satisfied decreases by 20.7910 unit with respect to be not job satisfied.
- For 1 unit increase in Years_In_Job(4 years , denoted by C),the log odds of being Job satisfied decreases by 18.7701 unit with respect to be not job satisfied.

The Hosmer-Lemeshow test

If the P-value is calculated higher than level of significance alpha=0.05, then we conclude that the test is really a goodness of fit test. The code for this test is given as

```
hltest(model)
```

```
  The Hosmer-Lemeshow goodness-of-fit test
```

Group <dbl>	Size <dbl>	Observed <dbl>
1	63	0
2	37	31

2 rows

```
         Statistic =  0
degrees of freedom =  0
           p-value =  < 2.22e-16
```

Note

The test statistics value is 0. Also, the P-values is very less. So, the test is not best for goodness of fit test.

Using confint() function

Now we can find the confidence interval by using the confint() function. Always remember one thing that using confint() function will give the confidence intervals of the coefficients of the estimators, not the log odds

Hide

```
round(confint(model),4)
```

```
Waiting for profiling to be done...
```

```
Warning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: algorithm did not convergeWarning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: collapsing to unique 'x' values
```

	2.5 %	97.5 %
(Intercept)	-20302.0462	21959.9709
Income	-0.0960	0.4670
Expenditure	-0.0283	0.0226
Years_In_Job.L	NA	39370.0293
Years_In_Job.Q	-64596.2718	NA
Years_In_Job.C	-208290.4659	NA

Using confint.default() function

This variation of the confint() function is used to find the confidence intervals based on the standard errors of the estimates. The code gives the above understanding.

Hide

```
round(confint.default(model),4)
```

```
                        2.5 %       97.5 %
(Intercept)     -749600.4052  749445.5760
Income               -2.0208       2.0269
Expenditure          -2.2250       2.2183
Years_In_Job.L  -672579.9226  672610.8826
Years_In_Job.Q -1493493.7539 1493452.1719
Years_In_Job.C -2002874.1731 2002836.6329
```

Performing Wald test

To perform Wald test, we use the function wald.test().It is used to understand the overall effect of the Years_In_job column. The following code performs the test.

Hide

```
wald.test(b = coef(model) , vcov(model) , Terms=4:6)
```

```
Wald test:
----------


Chi-squared test:
X2 = 4.2e-08, df = 3, P(> X2) = 1.0
```

Note

Here Terms = 4:6 means from 4th variable (Years_In_Job:L) to the 6th variable (Years_In_Job:C). Here we check the effect of the three Years_In_Job (L,Q and C) with respect to YearsIn_Job:C. Here the p-value which we get is 1 and chi-squared value is equal to 4.2e-08. This means they are not statistically significant.

Generating Odds Ratio

As we know the Odds ratio is the exponential value of the coefficients of the model we created earlier. To do this we first generate the exponent values of the coefficients.

```
round(exp(coef(model)))
```

```
   (Intercept)         Income    Expenditure Years_In_Job.L Years_In_Job.Q Years_In_J
ob.C
             0              1              1        5282980              0
0
```

Now we generate the Odds ratio along with its confodence interval. To do this we use the following code.

Hide

```
exp(cbind(OR = coef(model),confint(model)))
```

```
Waiting for profiling to be done...
```

```
Warning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: algorithm did not convergeWarning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: collapsing to unique 'x' values
```

```
                          OR     2.5 %   97.5 %
(Intercept)    2.394758e-34 0.0000000      Inf
Income         1.003061e+00 0.9084440 1.595196
Expenditure    9.966969e-01 0.9720567 1.022825
Years_In_Job.L 5.282980e+06        NA      Inf
Years_In_Job.Q 9.344712e-10 0.0000000       NA
Years_In_Job.C 7.051052e-09 0.0000000       NA
```

Interpretation of the above code snippet

- For 1 unit increase in Income, the odds of being Job satisfied (versus not being job satisfied), adjusting for the effects of the other predictor variables increases by a factor of 1.0030 unit.
- For 1 unit increase in Expenditure , the odds of being Job satisfied (versus not being job satisfied), adjusting for the effects of the other predictor variables increases by a factor of 0.9966 unit.
- Thus we can also interpret the values into meaningful statements.