Lab experiment - 5

Name: Prithviraj Guntha

Reg. No.: 20BRS1188

Subject: Essentials of data analytics

Subject code: CSE3506

Professor: A Sheik Abdullah

Slot: L55+L56

1. **Implement Logistic regression using different functions given below**
    a. **Logit function**
    b. **Odds function**
    c. **Sigmoid function**
    d. **Likelihood function**

a. Logit function

The logit function, also known as the log-odds function or logistic function, is a common link function used in logistic regression. It is used to model the probability of a binary response variable by taking the natural logarithm of the odds of the response variable being equal to 1.

The formula for the logit function is as follows:

$$logit(p) = log(p / (1-p))$$

Where: p = probability of the response variable being equal to 1

Input:

We will try to predict if a female has the risk of diabetes or not, the input will be a csv file containing various parameters that contribute to the final decision on whether the person will suffer from diabetes.

	A	B	C	D	E	F	G	H	I
1	Pregnancies	Glucose	BloodPressui	SkinThicknes	Insulin	BMI	DiabetesPed	Age	Outcome
2	6	148	72	35	0	33.6	0.627	50	1
3	1	85	66	29	0	26.6	0.351	31	0
4	8	183	64	0	0	23.3	0.672	32	1
5	1	89	66	23	94	28.1	0.167	21	0
6	0	137	40	35	168	43.1	2.288	33	1
7	5	116	74	0	0	25.6	0.201	30	0
8	3	78	50	32	88	31	0.248	26	1
9	10	115	0	0	0	35.3	0.134	29	0
10	2	197	70	45	543	30.5	0.158	53	1
11	8	125	96	0	0	0	0.232	54	1
12	4	110	92	0	0	37.6	0.191	30	0
13	10	168	74	0	0	38	0.537	34	1
14	10	139	80	0	0	27.1	1.441	57	0
15	1	189	60	23	846	30.1	0.398	59	1
16	5	166	72	19	175	25.8	0.587	51	1
17	7	100	0	0	0	30	0.484	32	1
18	0	118	84	47	230	45.8	0.551	31	1
19	7	107	74	0	0	29.6	0.254	31	1

Implementation steps:

1. We first import the dataset into Rstudio.
2. We then clean the dataset and take only the necessary variables.

3. Once we have the data ready, we can now split the data into train and test sets so that we can train the model using the training dataset and test the model using the testing dataset.
4. Apply the glm() function on the training dataset with respect to the logit function.
5. We will now evaluate the model by applying the predict() function on the test dataset.
6. We can now calculate the confusion matrix and also plot the necessary graphs to interpret the results.

Rcode:

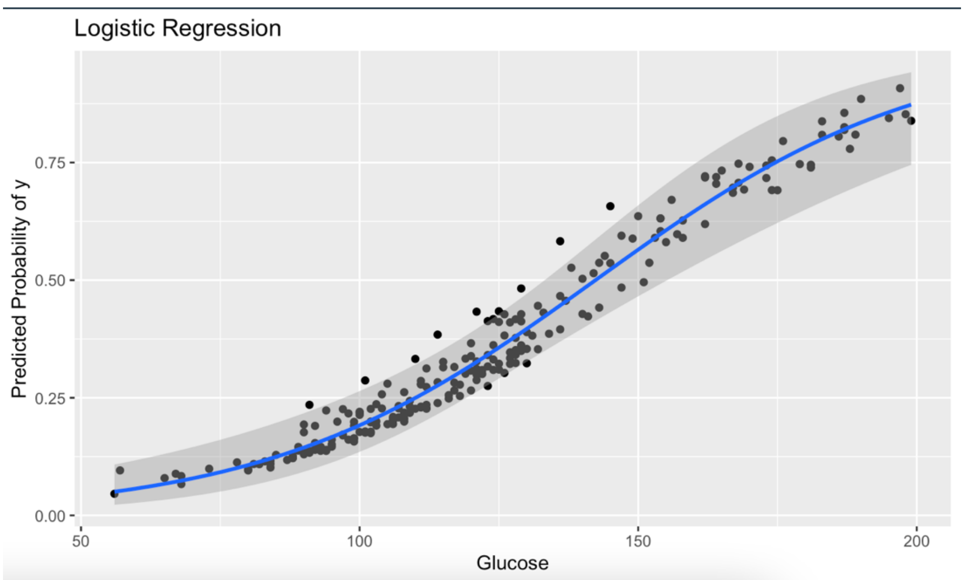
```r
library(stats)

#importing the dataset
data <- read.csv(("/Users/prithviraj/Downloads/diabetes2.csv"))
head(data)

#taking only required variables
data <- data.frame(Glucose = c(data$Glucose),
                   BP = c(data$BloodPressure),
                   Age = c(data$Age),
                   Outcome = c(data$Outcome))

#splitting the data into train and test
split_index <- floor(0.7 * nrow(data))
train_data <- data[1:split_index, ]
test_data <- data[(split_index+1):nrow(data),]

#fitting model
logitmodel <- glm(Outcome ~ Glucose + BP + Age, data = train_data,
                  family = binomial(link = "logit"))

#model summary
summary(logitmodel)

#making presictions
predictions <- predict(logitmodel,newdata = test_data,type = "response")

test_data$preds <- round(predictions)
library(caret)
print(test_data$Outcome)
print(round(predictions))
confusionMatrix(as.factor(test_data$Outcome),as.factor(test_data$preds))

#plotting
library(ggplot2)
```

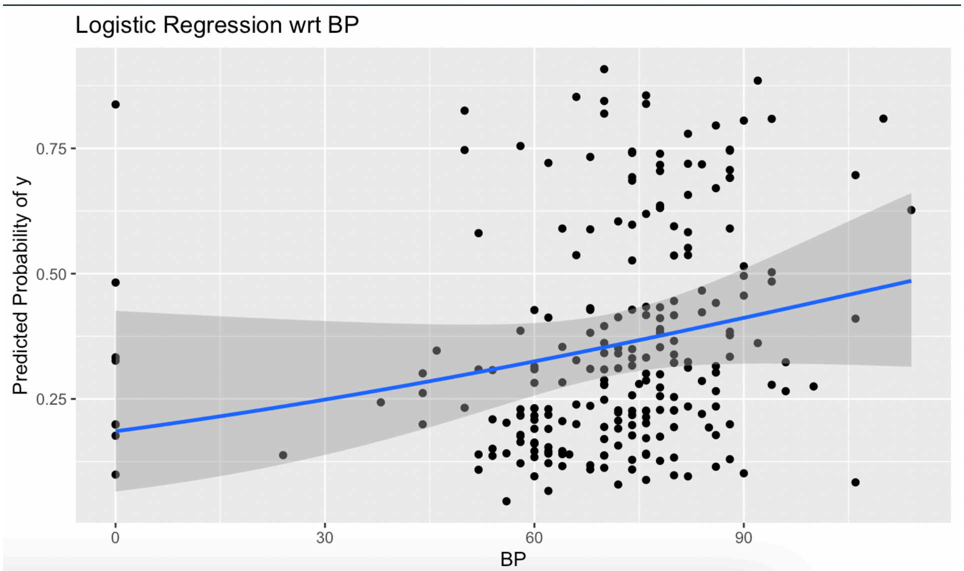
```r
ggplot(test_data, aes(x = Glucose, y = predictions)) +
  geom_point() +
  geom_smooth(method = "glm", method.args = list(family = binomial(link = "logit"))) +
  xlab("Glucose") +
  ylab("Predicted Probability of y") +
  ggtitle("Logistic Regression wrt Glucose")

ggplot(test_data, aes(x = BP, y = predictions)) +
  geom_point() +
  geom_smooth(method = "glm", method.args = list(family = binomial(link = "logit"))) +
  xlab("BP") +
  ylab("Predicted Probability of y") +
  ggtitle("Logistic Regression wrt BP")

ggplot(test_data, aes(x = Age, y = predictions)) +
  geom_point() +
  geom_smooth(method = "glm", method.args = list(family = binomial(link = "logit"))) +
  xlab("Age") +
  ylab("Predicted Probability of y") +
  ggtitle("Logistic Regression wrt Age")
```

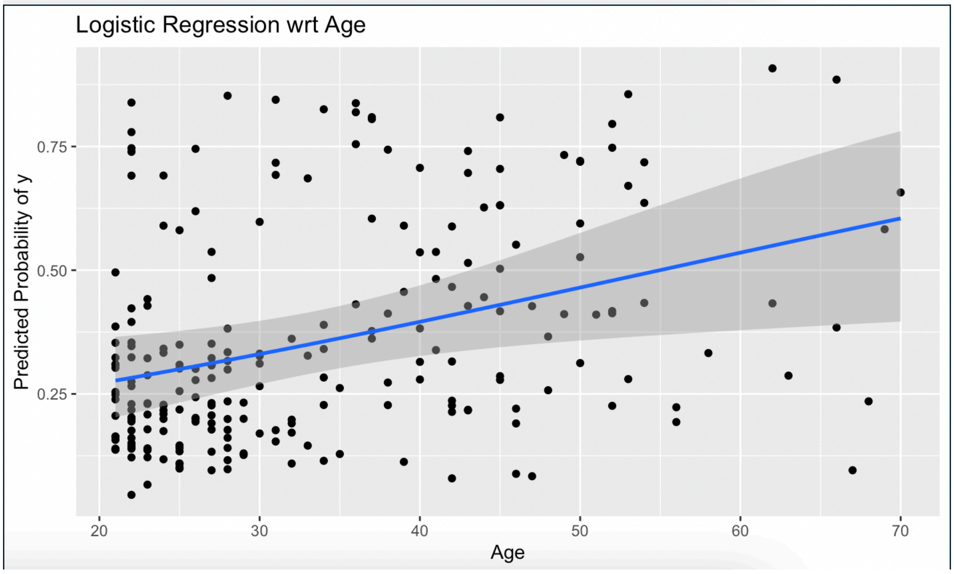
Output:

```
> summary(logitmodel)

Call:
glm(formula = Outcome ~ Glucose + BP + Age, family = binomial(link = "logit"),
    data = train_data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.1625  -0.8000  -0.5451   0.9175   3.0786

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -5.063239   0.580716  -8.719   <2e-16 ***
Glucose      0.033267   0.003839   8.665   <2e-16 ***
BP          -0.003742   0.005106  -0.733   0.4636
Age          0.017097   0.008861   1.930   0.0537 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 696.65  on 536  degrees of freedom
Residual deviance: 578.45  on 533  degrees of freedom
AIC: 586.45

Number of Fisher Scoring iterations: 4
```

```
> confusionMatrix(as.factor(test_data$Outcome),as.factor(test_data$preds))
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 139  13
         1  37  42

               Accuracy : 0.7835
                 95% CI : (0.7248, 0.8349)
    No Information Rate : 0.7619
    P-Value [Acc > NIR] : 0.245714

                  Kappa : 0.4812

 Mcnemar's Test P-Value : 0.001143

            Sensitivity : 0.7898
            Specificity : 0.7636
         Pos Pred Value : 0.9145
         Neg Pred Value : 0.5316
             Prevalence : 0.7619
         Detection Rate : 0.6017
   Detection Prevalence : 0.6580
      Balanced Accuracy : 0.7767

       'Positive' Class : 0
```







b. Odds function

The odds function is a measure of the probability of an event happening (e.g. a binary response variable equal to 1) versus the probability of it not happening (e.g. a binary response variable equal to 0). In logistic regression, it is used as a link function to model the relationship between the predictor variables and the odds of the response variable being equal to 1.
The formula for the odds function is as follows:

$$odds(p) = p / (1-p)$$

Where: p = probability of the response variable being equal to 1

Input:
The input for this model is same as the input given for the logit function,

	A	B	C	D	E	F	G	H	I
1	Pregnancies	Glucose	BloodPressu	SkinThicknes	Insulin	BMI	DiabetesPed	Age	Outcome
2	6	148	72	35	0	33.6	0.627	50	1
3	1	85	66	29	0	26.6	0.351	31	0
4	8	183	64	0	0	23.3	0.672	32	1
5	1	89	66	23	94	28.1	0.167	21	0
6	0	137	40	35	168	43.1	2.288	33	1
7	5	116	74	0	0	25.6	0.201	30	0
8	3	78	50	32	88	31	0.248	26	1

Implementation steps:

1. We first import the dataset into Rstudio.
2. We then clean the dataset and take only the necessary variables.
3. Once we have the data ready, we can now split the data into train and test sets so that we can train the model using the training dataset and test the model using the testing dataset.
4. Apply the glm() function on the training dataset with respect to the odds function.
5. We will now evaluate the model by applying the predict() function on the test dataset.
6. We can now calculate the confusion matrix and also plot the necessary graphs to interpret the results.

Rcode:

```r
#fitting model
install.packages("brglm")
library(brglm)
oddsmodel <- brglm(Outcome ~ Glucose + BP + Age, data = train_data,
                   family = binomial(),link = "power", power = 1)

#model summary
summary(oddsmodel)

#making presictions
predictions <- predict(oddsmodel,newdata = test_data,type = "response")

test_data$preds <- round(predictions)
library(caret)
print(test_data$Outcome)
print(round(predictions))
confusionMatrix(as.factor(test_data$Outcome),as.factor(test_data$preds))
```

```r
ggplot(test_data, aes(x = Glucose, y = predictions)) +
  geom_point() +
  geom_smooth(method = "brglm", method.args = list(family = binomial(),link="power",power=1)) +
  xlab("Glucose") +
  ylab("Predicted Probability of y") +
  ggtitle("Logistic Regression wrt Glucose")

ggplot(test_data, aes(x = BP, y = predictions)) +
  geom_point() +
  geom_smooth(method = "brglm", method.args = list(family = binomial(),link="power",power=1)) +
  xlab("BP") +
  ylab("Predicted Probability of y") +
  ggtitle("Logistic Regression wrt BP")

ggplot(test_data, aes(x = Age, y = predictions)) +
  geom_point() +
  geom_smooth(method = "brglm", method.args = list(family = binomial(),link="power",power=1)) +
  xlab("Age") +
  ylab("Predicted Probability of y") +
  ggtitle("Logistic Regression wrt Age")
```

Output:

```
> summary(oddsmodel)

Call:
brglm(formula = Outcome ~ Glucose + BP + Age, family = binomial(),
    data = train_data, link = "power", power = 1)


Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -5.008981   0.578128  -8.664   <2e-16 ***
Glucose      0.032909   0.003822   8.611   <2e-16 ***
BP          -0.003774   0.005096  -0.741   0.4590
Age          0.016979   0.008845   1.920   0.0549 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 677.26  on 536  degrees of freedom
Residual deviance: 578.46  on 533  degrees of freedom
Penalized deviance: 542.6461
AIC:  586.46
```

```
> confusionMatrix(as.factor(test_data$Outcome),as.factor(test_data$preds))
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 139  13
         1  37  42

               Accuracy : 0.7835
                 95% CI : (0.7248, 0.8349)
    No Information Rate : 0.7619
    P-Value [Acc > NIR] : 0.245714

                  Kappa : 0.4812

 Mcnemar's Test P-Value : 0.001143

            Sensitivity : 0.7898
            Specificity : 0.7636
         Pos Pred Value : 0.9145
         Neg Pred Value : 0.5316
             Prevalence : 0.7619
         Detection Rate : 0.6017
   Detection Prevalence : 0.6580
      Balanced Accuracy : 0.7767

       'Positive' Class : 0
```

c. Sigmoid function

The sigmoid function, also known as the logistic function, is a common function used in logistic regression. It is used to model the probability of a binary response variable by taking the sigmoid or logistic function of the linear predictor.

The formula for the sigmoid function is as follows:

$$sigmoid(x) = 1 / (1 + exp(-x))$$

Where: x = linear predictor ($x = b0 + b1x1 + b2x2 + ... + bn*xn$)

<u>Input</u>:

The input for this model is same as the input given for the logit function,

	A	B	C	D	E	F	G	H	I
1	Pregnancies	Glucose	BloodPressu	SkinThicknes	Insulin	BMI	DiabetesPed	Age	Outcome
2	6	148	72	35	0	33.6	0.627	50	1
3	1	85	66	29	0	26.6	0.351	31	0
4	8	183	64	0	0	23.3	0.672	32	1
5	1	89	66	23	94	28.1	0.167	21	0
6	0	137	40	35	168	43.1	2.288	33	1
7	5	116	74	0	0	25.6	0.201	30	0
8	3	78	50	32	88	31	0.248	26	1

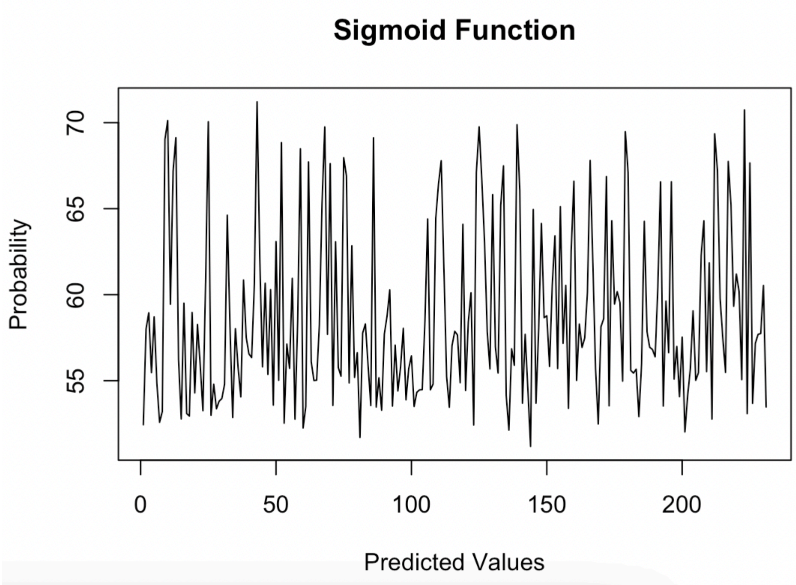
Implementation:

1. We first import the dataset into Rstudio.
2. We then clean the dataset and take only the necessary variables.
3. Once we have the data ready, we can now split the data into train and test sets so that we can train the model using the training dataset and test the model using the testing dataset.
4. Apply the glm() function on the training dataset with respect to the sigmoid function.
5. We will now evaluate the model by applying the predict() function on the test dataset.
6. We can now calculate the confusion matrix and also plot the necessary graphs to interpret the results.

Rcode:

```r

#fitting model
lr <- glm(Outcome ~ Glucose + BP + Age, data = train_data,
                  family = binomial())

#model summary
summary(lr)

#making presictions
predictions <- predict(oddsmodel,newdata = test_data,type = "response")

# Define the sigmoid function
sigmoid <- function(x) {
  1 / (1 + exp(-x))*100
}

# Apply the sigmoid function to the linear predictor values
pred_probs <- sigmoid(predictions)
plot(pred_probs, type = "l", xlab = "Predicted Values", ylab = "Probability", main = "Sigmoid Function")
```

Output:



d. Likelihood function

In logistic regression, the likelihood function is used to estimate the model parameters. The likelihood function is a probability function that describes the probability of the observed data given the model parameters. The goal of logistic regression is to find the model parameters that maximize the likelihood function.
Here is the likelihood function for logistic regression:

*Likelihood = Π P(y = 1|x)^y * (1- P(y = 1|x))^(1-y)*

Where:
Π is the product over all observations
y is the binary response variable
P(y = 1|x) is the probability of y being equal to 1 given the predictor variable x

Input:
The input for this model is same as the input given for the logit function,

	A	B	C	D	E	F	G	H	I
1	Pregnancies	Glucose	BloodPressu	SkinThicknes	Insulin	BMI	DiabetesPed	Age	Outcome
2	6	148	72	35	0	33.6	0.627	50	1
3	1	85	66	29	0	26.6	0.351	31	0
4	8	183	64	0	0	23.3	0.672	32	1
5	1	89	66	23	94	28.1	0.167	21	0
6	0	137	40	35	168	43.1	2.288	33	1
7	5	116	74	0	0	25.6	0.201	30	0
8	3	78	50	32	88	31	0.248	26	1

Implementation:

1. We first import the dataset into Rstudio.
2. We then clean the dataset and take only the necessary variables.
3. Once we have the data ready, we can now split the data into train and test sets so that we can train the model using the training dataset and test the model using the testing dataset.
4. Apply the glm() function on the training dataset with respect to the likelihood function.
5. We will now evaluate the model by applying the predict() function on the test dataset.

We can now calculate the confusion matrix and also plot the necessary graphs to interpret the results.

Rcode:

```r

#fitting model
lr <- glm(Outcome ~ Glucose + BP + Age, data = train_data,
                  family = binomial())

#model summary
summary(lr)

#making presictions
predictions <- predict(lr,newdata = test_data,type = "response")

# Define the log-likelihood function
loglik <- function(y, p) {
  sum(y * log(p) + (1 - y) * log(1 - p))
}
new_vec <- c()

for (i in 1:length(predictions)) {
  num <- as.numeric(loglik(test_data$Outcome[i],predictions[i]))
  new_vec <- c(new_vec,num)
}
```

Output:

```
> new_vec
  [1] -0.10058590 -0.38865391 -1.01755182 -1.52397006 -1.04580833 -1.64442527 -0.10728206 -0.13530528
  [9] -0.21644361 -0.15591555 -0.48129700 -1.27017074 -1.65684165 -0.28555113 -0.11597413 -0.48486306
 [17] -0.13000057 -0.12344768 -0.44913682 -0.18637420 -0.40430892 -0.26957179 -0.13756929 -0.83496610
 [25] -0.15942648 -0.12537115 -0.21125334 -0.14285012 -0.16339129 -0.17087155 -0.21106437 -0.92688713
 [33] -1.11705799 -0.11955946 -0.39050232 -0.26063372 -0.17544060 -0.58269705 -0.35836657 -0.30353985
 [41] -1.37091458 -0.56391650 -0.09660136 -0.70201527 -0.26327642 -0.56732925 -0.24039889 -0.87371419
 [49] -0.15257823 -0.62189748 -0.22282479 -0.22866040 -0.10435777 -1.25223895 -0.25894393 -0.80867082
 [57] -0.11524743 -0.41680292 -0.24965199 -0.09264690 -0.14833077 -0.29625477 -0.27875941 -0.22252871
 [65] -0.22223224 -0.40248339 -0.45250189 -0.17726493 -0.36998544 -0.30204334 -0.15195260 -0.76995365
 [73] -0.26112231 -0.23493808 -0.28146776 -0.34691194 -0.21551355 -0.64175524 -0.23159443 -0.30863240
 [81] -0.06893960 -1.16374282 -1.09924912 -0.26784679 -0.15179739 -1.65445790 -0.14753130 -0.23033427
 [89] -0.13903642 -0.37101586 -0.43578024 -0.53959162 -0.15011874 -1.26184183 -0.19220494 -0.26284738
 [97] -0.39145902 -0.16706695 -1.48116014 -0.29755000 -0.14914984 -1.76200472 -0.19516740 -0.19597180
[105] -0.41777443 -0.52006714 -0.19438103 -0.21176301 -0.91044279 -0.37744976 -0.29221128 -0.76301339
```