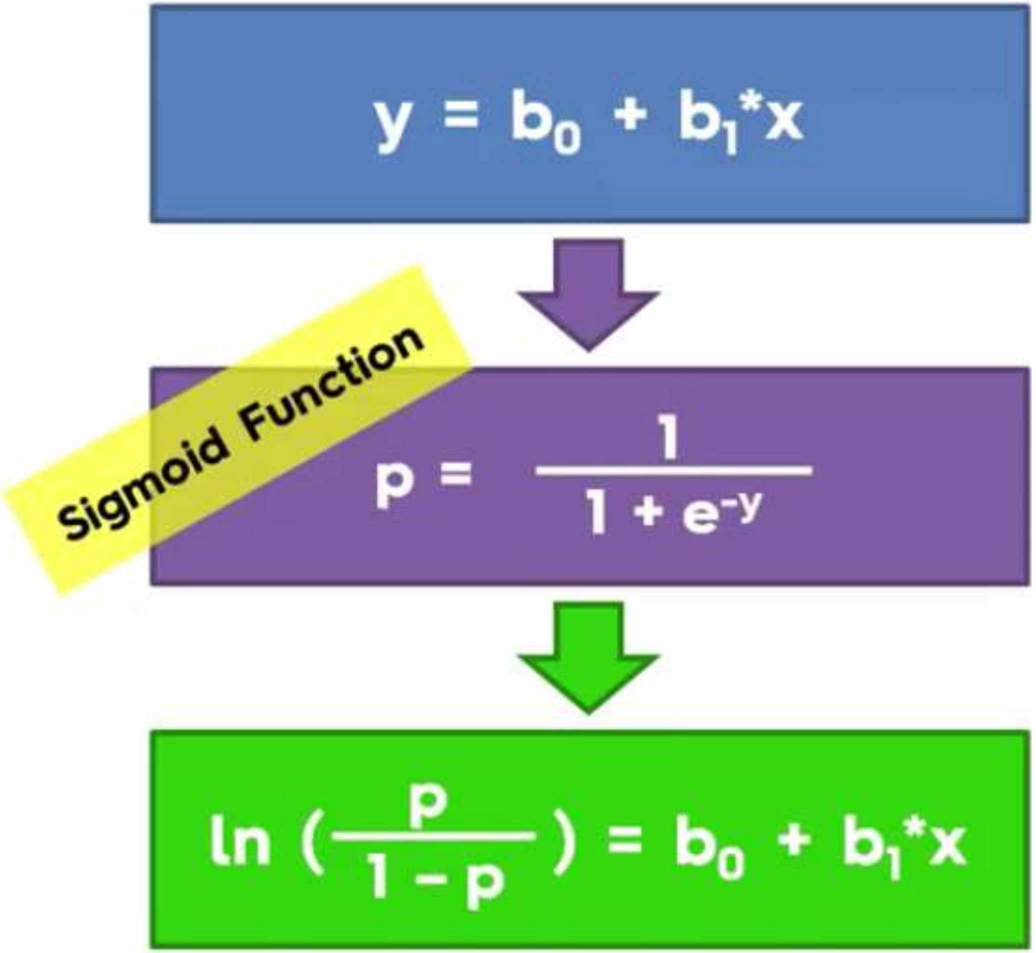
LOGISTIC REGRESSION

Logistic Regression

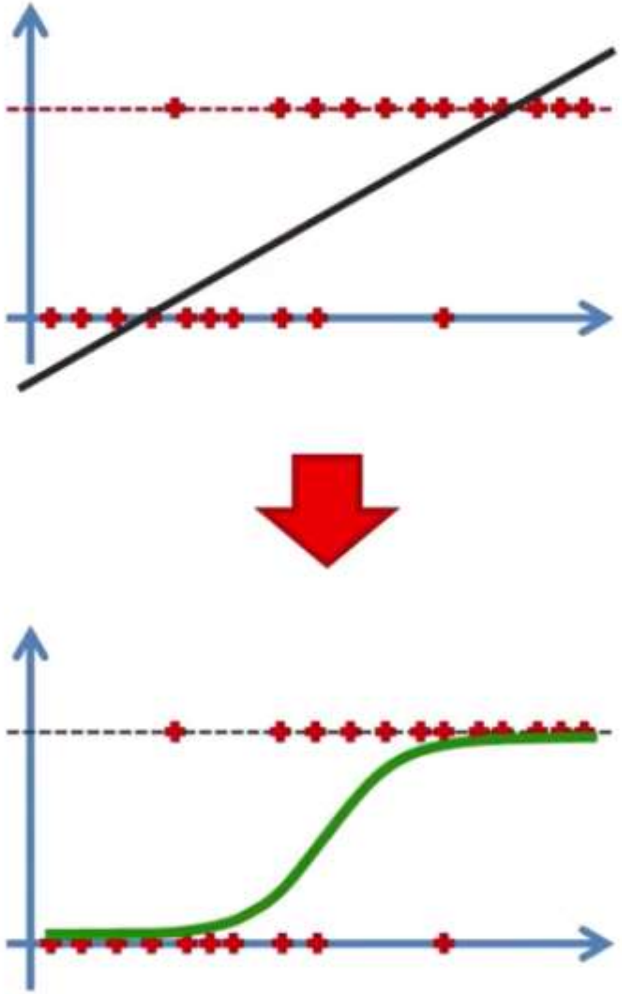


$$y = b_0 + b_1{*}x$$

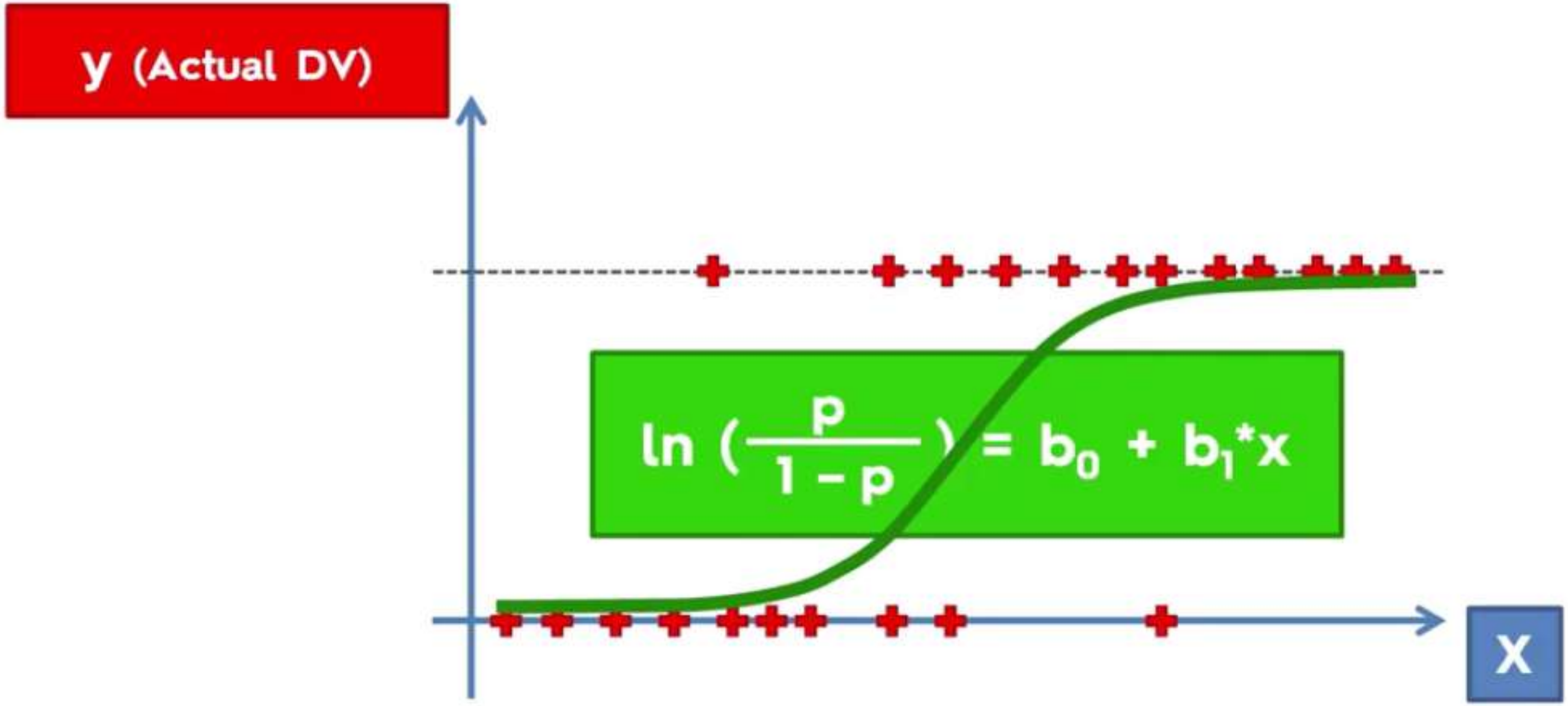
Sigmoid Function

$$p = \frac{1}{1 + e^{-y}}$$

$$\ln\left(\frac{p}{1 - p}\right) = b_0 + b_1{*}x$$



Logistic Regression

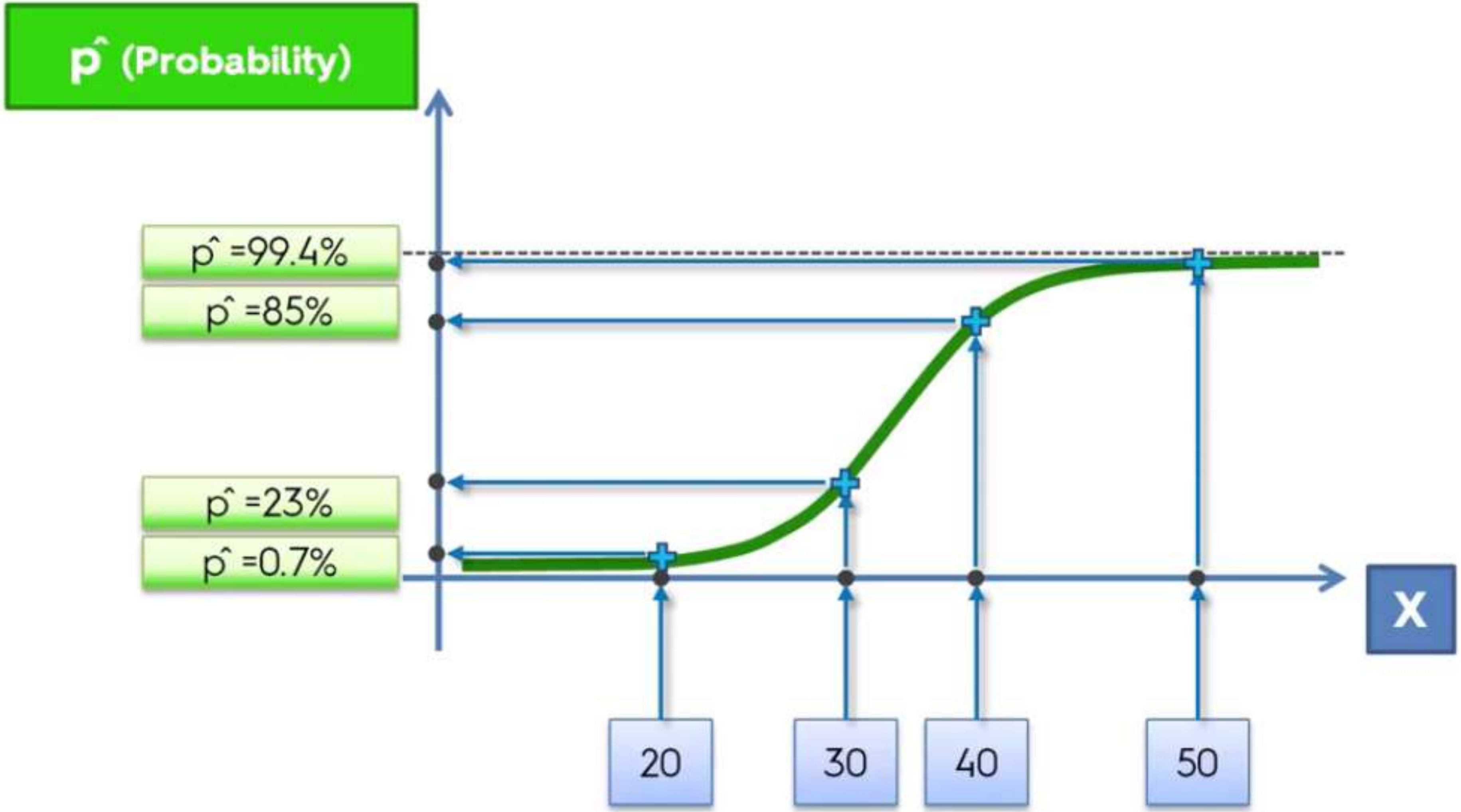


y (Actual DV)

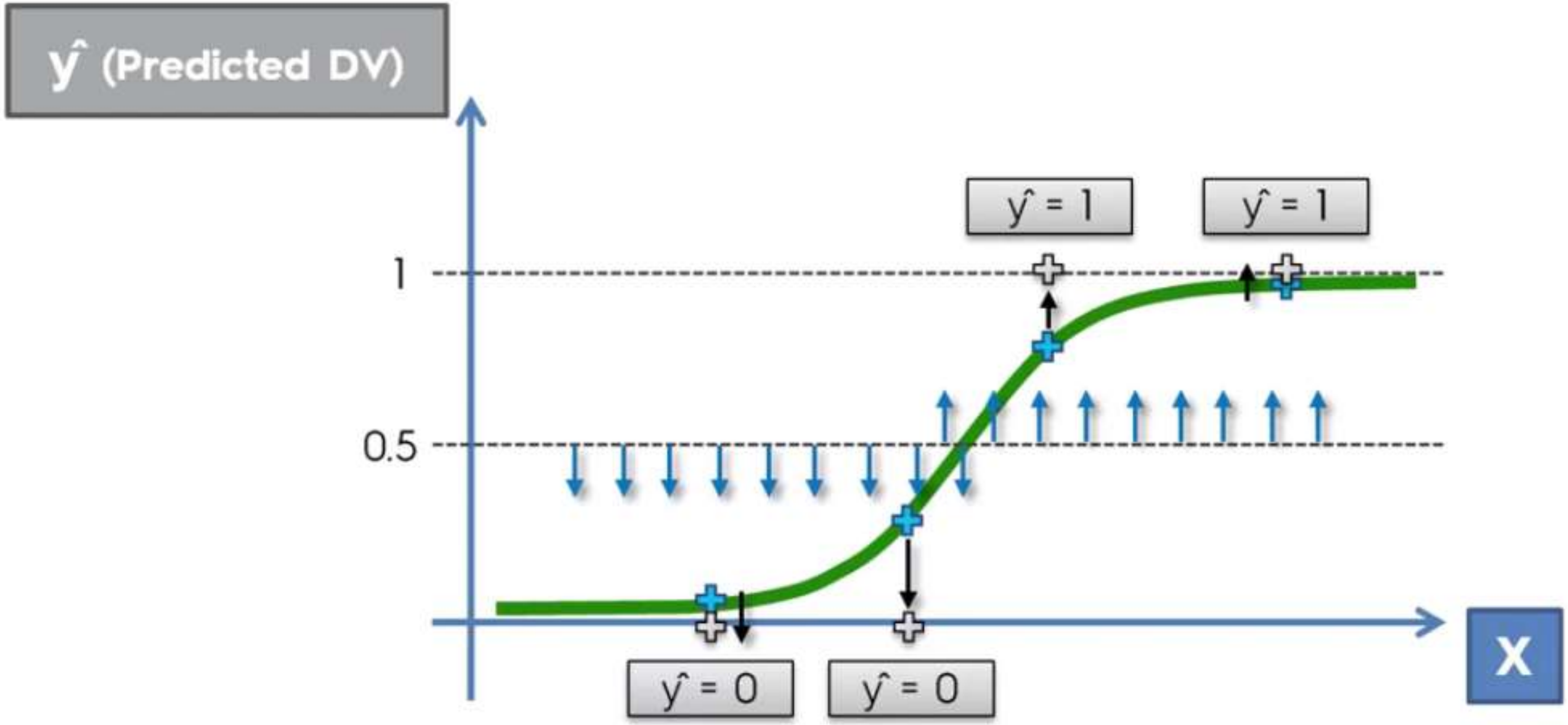
$$\ln\left(\frac{p}{1-p}\right) = b_0 + b_1{}^{*}x$$

x

Logistic Regression



Logistic Regression



PYTHON

READING DATASET DYNAMICALLY

```
from tkinter import *
from tkinter.filedialog import askopenfilename

root = Tk()
root.withdraw()
root.update()
file_path = askopenfilename()
root.destroy()
```

IMPORTING LIBRARIES

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

IMPORTING DATASET

```
dataset = pd.read_csv('Social_Network_Ads.csv')
X = dataset.iloc[:,2:4]
y= dataset.iloc[:,-1]
```

SPLITTING THE DATASET INTO THE TRAINING SET AND TEST SET

```
from sklearn.cross_validation import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
```

FEATURE SCALING

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

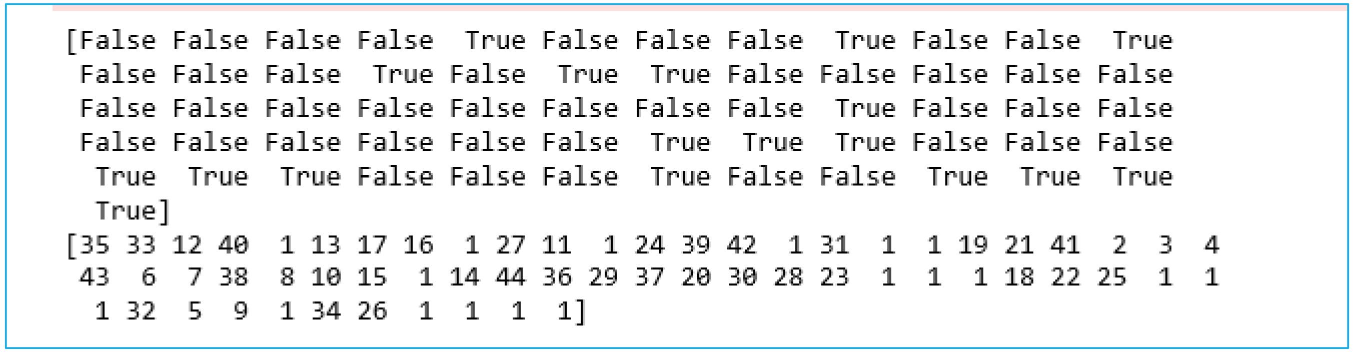
FEATURE SELECTION ...

Recursive Feature Elimination (RFE) is based on the idea to repeatedly construct a model and choose either the best or worst performing feature, setting the feature aside and then repeating the process with the rest of the features.

This process is applied until all features in the dataset are exhausted. The goal of RFE is to select features.

FEATURE SELECTION ...

```
from sklearn import datasets
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
 rfe = RFE(logreg, 2)
rfe = rfe.fit(X, y )
print(rfe.support_)
```



```
[False False False False  True False False False  True False False  True
 False False False  True False  True  True False False False False False
 False False False False False False False False  True False False False
 False False False False False False  True  True  True False False False
  True  True  True False False False  True False False  True  True  True
  True]
[35 33 12 40  1 13 17 16  1 27 11  1 24 39 42  1 31  1  1 19 21 41  2  3  4
 43  6  7 38  8 10 15  1 14 44 36 29 37 20 30 28 23  1  1  1 18 22 25  1  1
  1 32  5  9  1 34 26  1  1  1  1]
```

IMPLEMENTING THE MODEL

```
X = sc.fit_transform(X)
import statsmodels.api as sm
logit_model=sm.Logit(y,X)
result=logit_model.fit()
print(result.summary())
```

NOTE : The p-values for most of the variables are smaller than 0.05, therefore, most of them are significant to the model.

LOGISTIC REGRESSION MODEL

```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression()
model = classifier.fit(X_train,y_train)


y_pred = classifier.predict(X_test)
model.score(X_test,y_test)
```

CONFUSION MATRIX

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

K FOLD

```
from sklearn import model_selection
from sklearn.model_selection import cross_val_score
kfold = model_selection.KFold(n_splits=10, random_state=7)
modelCV = LogisticRegression()
scoring = 'accuracy'
results = model_selection.cross_val_score(modelCV, X_train, y_train, cv=kfold,
scoring=scoring)


print("10-fold cross validation average accuracy: %.3f" % (results.mean()))
```

EVALUATING CLASSIFICATION REPORT

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

R

READ DATASET

```
library(readr)

dataset <- read_csv("D:/machine learning AZ/Machine Learning A-Z Template Folder/Part 3 - Classification/Section 14 - Logistic Regression/Logistic_Regression/Social_Network_Ads.csv")


dataset = dataset[3:5]
```

ENCODING THE TARGET FEATURE AS FACTOR

```
dataset$Purchased = factor(dataset$Purchased, levels = c(0, 1))
```

SPLITTING THE DATASET INTO THE TRAINING SET AND TEST SET

```
# install.packages('caTools')
library(caTools)
set.seed(123)
split = sample.split(dataset$Purchased, SplitRatio = 0.75)
training_set = subset(dataset, split == TRUE)
test_set = subset(dataset, split == FALSE)
```

FEATURE SCALING

```
training_set[−3] = scale(training_set[−3])
test_set[−3] = scale(test_set[−3])
```

FITTING LOGISTIC REGRESSION TO THE TRAINING SET

```
classifier = glm(formula = Purchased ~ .,
                 family = binomial,
                 data = training_set)
```

PREDICTION

```
prob_pred = predict(classifier, type = 'response', newdata = test_set[-3])

y_pred = ifelse(prob_pred > 0.5, 1, 0)
```

CONFUSION MATRIX

```
# Making the Confusion Matrix
cm = table(unlist(test_set[, 3]), y_pred )
```

PLOT

```
library(ElemStatLearn)
set = training_set
X1 = seq(min(set[, 1]) – 1, max(set[, 1]) + 1, by = 0.01)
X2 = seq(min(set[, 2]) – 1, max(set[, 2]) + 1, by = 0.01)
grid_set = expand.grid(X1, X2)
colnames(grid_set) = c('Age', 'EstimatedSalary')
prob_set = predict(classifier, type = 'response', newdata = grid_set)
y_grid = ifelse(prob_set > 0.5, 1, 0)
plot(set[, −3],
     main = 'Logistic Regression (Training set)',
     xlab = 'Age', ylab = 'Estimated Salary',
     xlim = range(X1), ylim = range(X2))
contour(X1, X2, matrix(as.numeric(y_grid), length(X1), length(X2)), add = TRUE)
points(grid_set, pch = '.', col = ifelse(y_grid == 1, 'springgreen3', 'tomato'))
points(set, pch = 21, bg = ifelse(set[, 3] == 1, 'green4', 'red3'))
```