Machine Learning A-Z

Contents

Course Outline

Interested in the field of Machine Learning? Then this course is for you!

This course has been designed by two professional Data Scientists so that we can share our knowledge and help you learn complex theory, algorithms, and coding libraries in a simple way.

We will walk you step-by-step into the World of Machine Learning. With every tutorial, you will develop new skills and improve your understanding of this challenging yet lucrative sub-field of Data Science.

This course is fun and exciting, but at the same time, we dive deep into Machine Learning. It is structured the following way:

- Part 1 - Data Preprocessing
- Part 2 - Regression: Simple Linear Regression, Multiple Linear Regression, Polynomial Regression, SVR, Decision Tree Regression, Random Forest Regression
- Part 3 - Classification: Logistic Regression, K-NN, SVM, Kernel SVM, Naive Bayes, Decision Tree Classification, Random Forest Classification
- Part 4 - Clustering: K-Means, Hierarchical Clustering
- Part 5 - Association Rule Learning: Apriori, Eclat
- Part 6 - Reinforcement Learning: Upper Confidence Bound, Thompson Sampling
- Part 7 - Natural Language Processing: Bag-of-words model and algorithms for NLP
- Part 8 - Deep Learning: Artificial Neural Networks, Convolutional Neural Networks
- Part 9 - Dimensionality Reduction: PCA, LDA, Kernel PCA
- Part 10 - Model Selection & Boosting: k-fold Cross Validation, Parameter Tuning, Grid Search, XGBoost

Moreover, the course is packed with practical exercises that are based on real-life examples. So not only will you learn the theory, but you will also get some hands-on practice building your own models.

And as a bonus, this course includes both Python and R code templates which you can download and use on your own projects.

Important updates (June 2020):

- CODES ALL UP TO DATE
- DEEP LEARNING CODED IN TENSORFLOW 2.0
- TOP GRADIENT BOOSTING MODELS INCLUDING XGBOOST AND EVEN CATBOOST!

What you'll learn:

- Master Machine Learning on Python & R
- Have a great intuition of many Machine Learning models
- Make accurate predictions
- Make powerful analysis
- Make robust Machine Learning models
- Create strong added value to your business
- Use Machine Learning for personal purpose
- Handle specific topics like Reinforcement Learning, NLP and Deep Learning
- Handle advanced techniques like Dimensionality Reduction
- Know which Machine Learning model to choose for each type of problem
- Build an army of powerful Machine Learning models and know how to combine them to solve any problem

Are there any course requirements or prerequisites?

- Just some high school mathematics level.

Who this course is for:

- Anyone interested in Machine Learning.

- Students who have at least high school knowledge in math and who want to start learning Machine Learning.
- Any intermediate level people who know the basics of machine learning, including the classical algorithms like linear regression or logistic regression, but who want to learn more about it and explore all the different fields of Machine Learning.
- Any people who are not that comfortable with coding but who are interested in Machine Learning and want to apply it easily on datasets.
- Any students in college who want to start a career in Data Science.
- Any data analysts who want to level up in Machine Learning.
- Any people who are not satisfied with their job and who want to become a Data Scientist.
- Any people who want to create added value to their business by using powerful Machine Learning tools.

Chapter 1

Data Preprocessing

1.1 Importing the libraries

Python

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

1.2 Importing the dataset

Python

```
dataset = pd.read_csv('Data.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
print(X)
## [['France' 44.0 72000.0]
##  ['Spain' 27.0 48000.0]
##  ['Germany' 30.0 54000.0]
##  ['Spain' 38.0 61000.0]
##  ['Germany' 40.0 nan]
##  ['France' 35.0 58000.0]
##  ['Spain' nan 52000.0]
##  ['France' 48.0 79000.0]
##  ['Germany' 50.0 83000.0]
##  ['France' 37.0 67000.0]]
print(y)
## ['No' 'Yes' 'No' 'No' 'Yes' 'Yes' 'No' 'Yes' 'No' 'Yes']
```

R

```
dataset = read.csv('Data.csv')
```

1.3 Taking care of missing data

Python

```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
imputer.fit(X[:, 1:3])
## SimpleImputer()
X[:, 1:3] = imputer.transform(X[:, 1:3])
print(X)
## [['France' 44.0 72000.0]
##  ['Spain' 27.0 48000.0]
##  ['Germany' 30.0 54000.0]
##  ['Spain' 38.0 61000.0]
##  ['Germany' 40.0 63777.77777777778]
##  ['France' 35.0 58000.0]
##  ['Spain' 38.77777777777778 52000.0]
##  ['France' 48.0 79000.0]
##  ['Germany' 50.0 83000.0]
##  ['France' 37.0 67000.0]]
```

R

```
dataset$Age = ifelse(is.na(dataset$Age),
                     ave(dataset$Age, FUN = function(x) mean(x, na.rm = TRUE)),
                     dataset$Age)
dataset$Salary = ifelse(is.na(dataset$Salary),
                        ave(dataset$Salary, FUN = function(x) mean(x, na.rm = TRUE)),
                        dataset$Salary)
```

1.4 Encoding categorical data

Python

```
# Independent Variable
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [0])], remainder='pas
X = np.array(ct.fit_transform(X))
print(X)

# Dependent Variable
## [[1.0 0.0 0.0 44.0 72000.0]
```

```
##  [0.0 0.0 1.0 27.0 48000.0]
##  [0.0 1.0 0.0 30.0 54000.0]
##  [0.0 0.0 1.0 38.0 61000.0]
##  [0.0 1.0 0.0 40.0 63777.77777777778]
##  [1.0 0.0 0.0 35.0 58000.0]
##  [0.0 0.0 1.0 38.777777777777778 52000.0]
##  [1.0 0.0 0.0 48.0 79000.0]
##  [0.0 1.0 0.0 50.0 83000.0]
##  [1.0 0.0 0.0 37.0 67000.0]]
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)
print(y)
## [0 1 0 0 1 1 0 1 0 1]
```

R

```
# Independent Variable
dataset$Country = factor(dataset$Country,
                         levels = c('France', 'Spain', 'Germany'),
                         labels = c(1, 2, 3))

# Dependent Variable
dataset$Purchased = factor(dataset$Purchased,
                           levels = c('No', 'Yes'),
                           labels = c(0, 1))
```

1.5 Splitting the dataset into the Training set and Test set

Python

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 1)
print(X_train)
## [[0.0 0.0 1.0 38.777777777777778 52000.0]
##  [0.0 1.0 0.0 40.0 63777.77777777778]
##  [1.0 0.0 0.0 44.0 72000.0]
##  [0.0 0.0 1.0 38.0 61000.0]
##  [0.0 0.0 1.0 27.0 48000.0]
##  [1.0 0.0 0.0 48.0 79000.0]
##  [0.0 1.0 0.0 50.0 83000.0]
##  [1.0 0.0 0.0 35.0 58000.0]]
print(X_test)
## [[0.0 1.0 0.0 30.0 54000.0]
```

```
##  [1.0 0.0 0.0 37.0 67000.0]]
print(y_train)
## [0 1 0 0 1 1 0 1]
print(y_test)
## [0 1]
```

R

```
# install.packages('caTools')
library(caTools)
set.seed(123)
split = sample.split(dataset$Purchased, SplitRatio = 0.8)
training_set = subset(dataset, split == TRUE)
test_set = subset(dataset, split == FALSE)
```

1.6 Feature Scaling

Python

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train[:, 3:] = sc.fit_transform(X_train[:, 3:])
X_test[:, 3:] = sc.transform(X_test[:, 3:])
print(X_train)
## [[0.0 0.0 1.0 -0.19159184384578545 -1.0781259408412425]
##  [0.0 1.0 0.0 -0.014117293757057777 -0.07013167641635372]
##  [1.0 0.0 0.0 0.566708506533324 0.633562432710455]
##  [0.0 0.0 1.0 -0.30453019390224867 -0.30786617274297867]
##  [0.0 0.0 1.0 -1.9018011447007988 -1.420463615551582]
##  [1.0 0.0 0.0 1.1475343068237058 1.232653363453549]
##  [0.0 1.0 0.0 1.4379472069688968 1.5749910381638885]
##  [1.0 0.0 0.0 -0.7401495441200351 -0.56461942877573327]]
print(X_test)
## [[0.0 1.0 0.0 -1.4661817944830124 -0.9069571034860727]
##  [1.0 0.0 0.0 -0.44973664397484414 0.2056403393225306]]
```

R

```
# training_set = scale(training_set)
# test_set = scale(test_set)
```

Chapter 2

Regression

2.1 Simple Linear Regression

2.1.1 Importing the libraries

Python

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

2.1.2 Importing the dataset

Python

```
dataset = pd.read_csv('Salary_Data.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
```

R

```
dataset = read.csv('Salary_Data.csv')
```

2.1.3 Splitting the dataset into the Training set and Test set

Python

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 1/3, random_state = 0)
```

R

```
# install.packages('caTools')
library(caTools)
set.seed(123)
split = sample.split(dataset$Salary, SplitRatio = 2/3)
training_set = subset(dataset, split == TRUE)
test_set = subset(dataset, split == FALSE)
```

2.1.4 Training the Simple Linear Regression model on the Training set

Python

```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
## LinearRegression()
```

R

```
regressor = lm(formula = Salary ~ YearsExperience,
               data = training_set)
```

2.1.5 Predicting the Test set results

Python

```
y_pred = regressor.predict(X_test)
```

R

```
y_pred = predict(regressor, newdata = test_set)
```

2.1.6 Visualising the Training set results

Python

```
plt.scatter(X_train, y_train, color = 'red')
plt.plot(X_train, regressor.predict(X_train), color = 'blue')
plt.title('Salary vs Experience (Training set)')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.show()
```

R

```r
# install.packages('ggplot2')
library(ggplot2)
ggplot() +
  geom_point(aes(x = training_set$YearsExperience, y = training_set$Salary),
             colour = 'red') +
  geom_line(aes(x = training_set$YearsExperience, y = predict(regressor, newdata = training_set))
            colour = 'blue') +
  ggtitle('Salary vs Experience (Training set)') +
  xlab('Years of experience') +
  ylab('Salary')
```

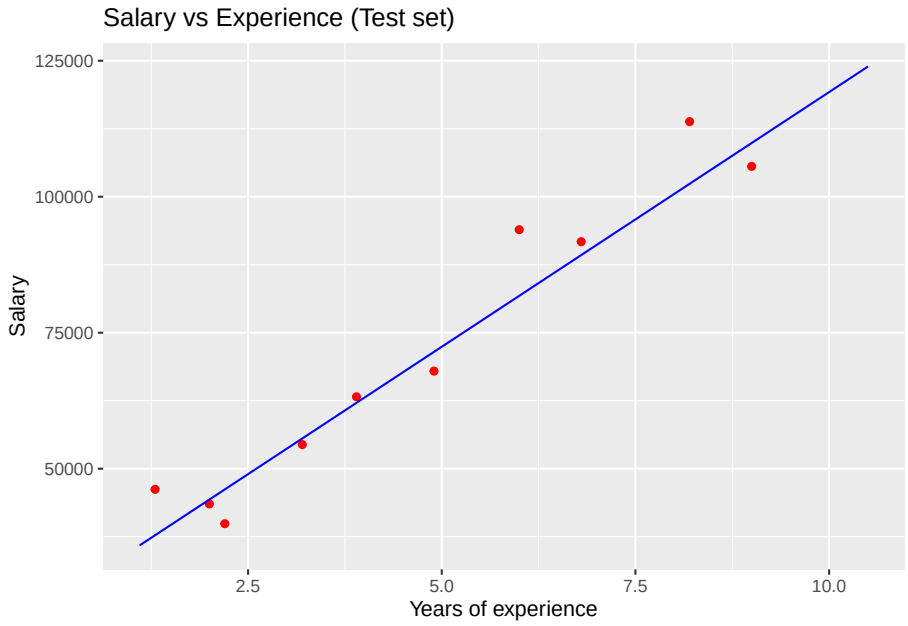
2.1.7 Visualising the Test set results

Python

```
plt.scatter(X_test, y_test, color = 'red')
plt.plot(X_train, regressor.predict(X_train), color = 'blue')
plt.title('Salary vs Experience (Test set)')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.show()
```

R

```
# install.packages('ggplot2')
library(ggplot2)
ggplot() +
  geom_point(aes(x = test_set$YearsExperience, y = test_set$Salary),
             colour = 'red') +
  geom_line(aes(x = training_set$YearsExperience, y = predict(regressor, newdata = training_set))
            colour = 'blue') +
  ggtitle('Salary vs Experience (Test set)') +
  xlab('Years of experience') +
  ylab('Salary')
```



2.2 Multiple Linear Regression

2.2.1 Importing the libraries

Python

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

2.2.2 Importing the dataset

Python

```
dataset = pd.read_csv('50_Startups.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
print(X)
## [[165349.2 136897.8 471784.1 'New York']
##  [162597.7 151377.59 443898.53 'California']
##  [153441.51 101145.55 407934.54 'Florida']
##  [144372.41 118671.85 383199.62 'New York']
##  [142107.34 91391.77 366168.42 'Florida']
##  [131876.9 99814.71 362861.36 'New York']
##  [134615.46 147198.87 127716.82 'California']
```

```
##  [130298.13 145530.06 323876.68 'Florida']
##  [120542.52 148718.95 311613.29 'New York']
##  [123334.88 108679.17 304981.62 'California']
##  [101913.08 110594.11 229160.95 'Florida']
##  [100671.96 91790.61 249744.55 'California']
##  [93863.75 127320.38 249839.44 'Florida']
##  [91992.39 135495.07 252664.93 'California']
##  [119943.24 156547.42 256512.92 'Florida']
##  [114523.61 122616.84 261776.23 'New York']
##  [78013.11 121597.55 264346.06 'California']
##  [94657.16 145077.58 282574.31 'New York']
##  [91749.16 114175.79 294919.57 'Florida']
##  [86419.7 153514.11 0.0 'New York']
##  [76253.86 113867.3 298664.47 'California']
##  [78389.47 153773.43 299737.29 'New York']
##  [73994.56 122782.75 303319.26 'Florida']
##  [67532.53 105751.03 304768.73 'Florida']
##  [77044.01 99281.34 140574.81 'New York']
##  [64664.71 139553.16 137962.62 'California']
##  [75328.87 144135.98 134050.07 'Florida']
##  [72107.6 127864.55 353183.81 'New York']
##  [66051.52 182645.56 118148.2 'Florida']
##  [65605.48 153032.06 107138.38 'New York']
##  [61994.48 115641.28 91131.24 'Florida']
##  [61136.38 152701.92 88218.23 'New York']
##  [63408.86 129219.61 46085.25 'California']
##  [55493.95 103057.49 214634.81 'Florida']
##  [46426.07 157693.92 210797.67 'California']
##  [46014.02 85047.44 205517.64 'New York']
##  [28663.76 127056.21 201126.82 'Florida']
##  [44069.95 51283.14 197029.42 'California']
##  [20229.59 65947.93 185265.1 'New York']
##  [38558.51 82982.09 174999.3 'California']
##  [28754.33 118546.05 172795.67 'California']
##  [27892.92 84710.77 164470.71 'Florida']
##  [23640.93 96189.63 148001.11 'California']
##  [15505.73 127382.3 35534.17 'New York']
##  [22177.74 154806.14 28334.72 'California']
##  [1000.23 124153.04 1903.93 'New York']
##  [1315.46 115816.21 297114.46 'Florida']
##  [0.0 135426.92 0.0 'California']
##  [542.05 51743.15 0.0 'New York']
##  [0.0 116983.8 45173.06 'California']]
```

R

```
dataset = read.csv('50_Startups.csv')
```

2.2.3 Encoding categorical data

Python

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [3])], remainder='pas
X = np.array(ct.fit_transform(X))
print(X)
## [[0.0 0.0 1.0 165349.2 136897.8 471784.1]
##  [1.0 0.0 0.0 162597.7 151377.59 443898.53]
##  [0.0 1.0 0.0 153441.51 101145.55 407934.54]
##  [0.0 0.0 1.0 144372.41 118671.85 383199.62]
##  [0.0 1.0 0.0 142107.34 91391.77 366168.42]
##  [0.0 0.0 1.0 131876.9 99814.71 362861.36]
##  [1.0 0.0 0.0 134615.46 147198.87 127716.82]
##  [0.0 1.0 0.0 130298.13 145530.06 323876.68]
##  [0.0 0.0 1.0 120542.52 148718.95 311613.29]
##  [1.0 0.0 0.0 123334.88 108679.17 304981.62]
##  [0.0 1.0 0.0 101913.08 110594.11 229160.95]
##  [1.0 0.0 0.0 100671.96 91790.61 249744.55]
##  [0.0 1.0 0.0 93863.75 127320.38 249839.44]
##  [1.0 0.0 0.0 91992.39 135495.07 252664.93]
##  [0.0 1.0 0.0 119943.24 156547.42 256512.92]
##  [0.0 0.0 1.0 114523.61 122616.84 261776.23]
##  [1.0 0.0 0.0 78013.11 121597.55 264346.06]
##  [0.0 0.0 1.0 94657.16 145077.58 282574.31]
##  [0.0 1.0 0.0 91749.16 114175.79 294919.57]
##  [0.0 0.0 1.0 86419.7 153514.11 0.0]
##  [1.0 0.0 0.0 76253.86 113867.3 298664.47]
##  [0.0 0.0 1.0 78389.47 153773.43 299737.29]
##  [0.0 1.0 0.0 73994.56 122782.75 303319.26]
##  [0.0 1.0 0.0 67532.53 105751.03 304768.73]
##  [0.0 0.0 1.0 77044.01 99281.34 140574.81]
##  [1.0 0.0 0.0 64664.71 139553.16 137962.62]
##  [0.0 1.0 0.0 75328.87 144135.98 134050.07]
##  [0.0 0.0 1.0 72107.6 127864.55 353183.81]
##  [0.0 1.0 0.0 66051.52 182645.56 118148.2]
##  [0.0 0.0 1.0 65605.48 153032.06 107138.38]
##  [0.0 1.0 0.0 61994.48 115641.28 91131.24]
##  [0.0 0.0 1.0 61136.38 152701.92 88218.23]
##  [1.0 0.0 0.0 63408.86 129219.61 46085.25]
##  [0.0 1.0 0.0 55493.95 103057.49 214634.81]
```

```
##  [1.0 0.0 0.0 46426.07 157693.92 210797.67]
##  [0.0 0.0 1.0 46014.02 85047.44 205517.64]
##  [0.0 1.0 0.0 28663.76 127056.21 201126.82]
##  [1.0 0.0 0.0 44069.95 51283.14 197029.42]
##  [0.0 0.0 1.0 20229.59 65947.93 185265.1]
##  [1.0 0.0 0.0 38558.51 82982.09 174999.3]
##  [1.0 0.0 0.0 28754.33 118546.05 172795.67]
##  [0.0 1.0 0.0 27892.92 84710.77 164470.71]
##  [1.0 0.0 0.0 23640.93 96189.63 148001.11]
##  [0.0 0.0 1.0 15505.73 127382.3 35534.17]
##  [1.0 0.0 0.0 22177.74 154806.14 28334.72]
##  [0.0 0.0 1.0 1000.23 124153.04 1903.93]
##  [0.0 1.0 0.0 1315.46 115816.21 297114.46]
##  [1.0 0.0 0.0 0.0 135426.92 0.0]
##  [0.0 0.0 1.0 542.05 51743.15 0.0]
##  [1.0 0.0 0.0 0.0 116983.8 45173.06]]
```

R

```
dataset$State = factor(dataset$State,
                       levels = c('New York', 'California', 'Florida'),
                       labels = c(1, 2, 3))
```

2.2.4 Splitting the dataset into the Training set and Test set

Python

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
```

R

```
# install.packages('caTools')
library(caTools)
set.seed(123)
split = sample.split(dataset$Profit, SplitRatio = 0.8)
training_set = subset(dataset, split == TRUE)
test_set = subset(dataset, split == FALSE)
```

2.2.5 Training the Multiple Linear Regression model on the Training set

Python

```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
```

```
regressor.fit(X_train, y_train)
## LinearRegression()
```

R

```
regressor = lm(formula = Profit ~ .,
               data = training_set)
```

2.2.6 Predicting the Test set results

Python

```
y_pred = regressor.predict(X_test)
np.set_printoptions(precision=2)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1)
## [[103015.2  103282.38]
##  [132582.28 144259.4 ]
##  [132447.74 146121.95]
##  [ 71976.1   77798.83]
##  [178537.48 191050.39]
##  [116161.24 105008.31]
##  [ 67851.69  81229.06]
##  [ 98791.73  97483.56]
##  [113969.44 110352.25]
##  [167921.07 166187.94]]
```

R

```
y_pred = predict(regressor, newdata = test_set)
```

2.3 Polynomial Regression

2.3.1 Importing the libraries

Python

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

2.3.2 Importing the dataset

Python

```
dataset = pd.read_csv('Position_Salaries.csv')
X = dataset.iloc[:, 1:-1].values
y = dataset.iloc[:, -1].values
```

R

```
dataset = read.csv('Position_Salaries.csv')
dataset = dataset[2:3]
```

2.3.3 Training the Linear Regression model on the whole dataset

Python

```
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(X, y)
## LinearRegression()
```

R

```
lin_reg = lm(formula = Salary ~ .,
             data = dataset)
```

2.3.4 Training the Polynomial Regression model on the whole dataset

Python

```
from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree = 4)
X_poly = poly_reg.fit_transform(X)
lin_reg_2 = LinearRegression()
lin_reg_2.fit(X_poly, y)
## LinearRegression()
```

R

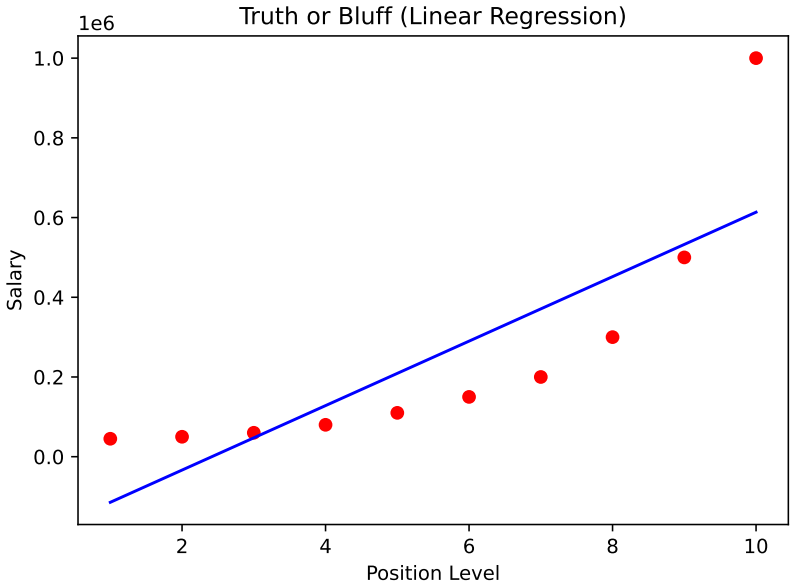
```
dataset$Level2 = dataset$Level^2
dataset$Level3 = dataset$Level^3
dataset$Level4 = dataset$Level^4
poly_reg = lm(formula = Salary ~ .,
              data = dataset)
```

2.3.5 Visualising the Linear Regression results

Python

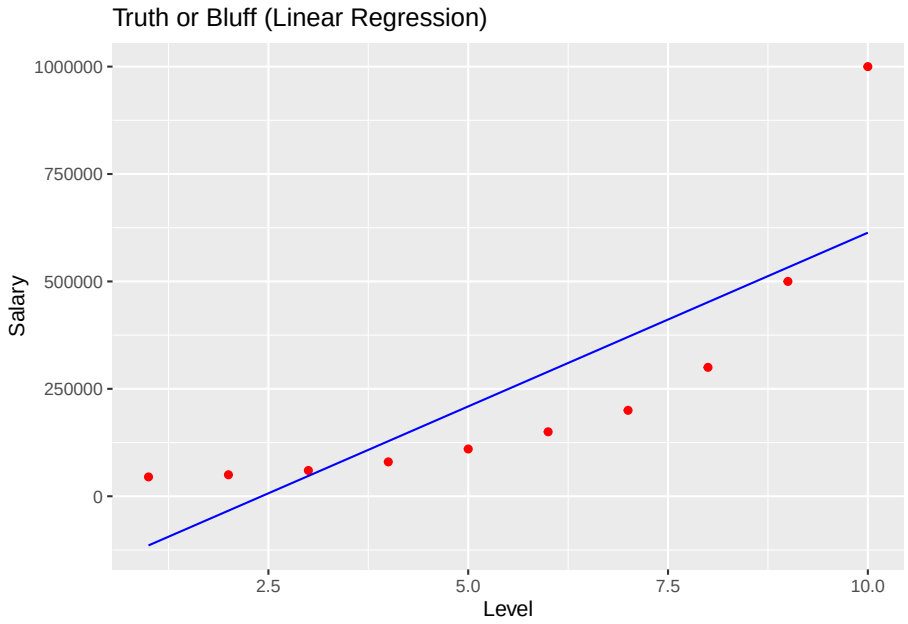
```
plt.scatter(X, y, color = 'red')
plt.plot(X, lin_reg.predict(X), color = 'blue')
plt.title('Truth or Bluff (Linear Regression)')
plt.xlabel('Position Level')
```

```
plt.ylabel('Salary')
plt.show()
```



R

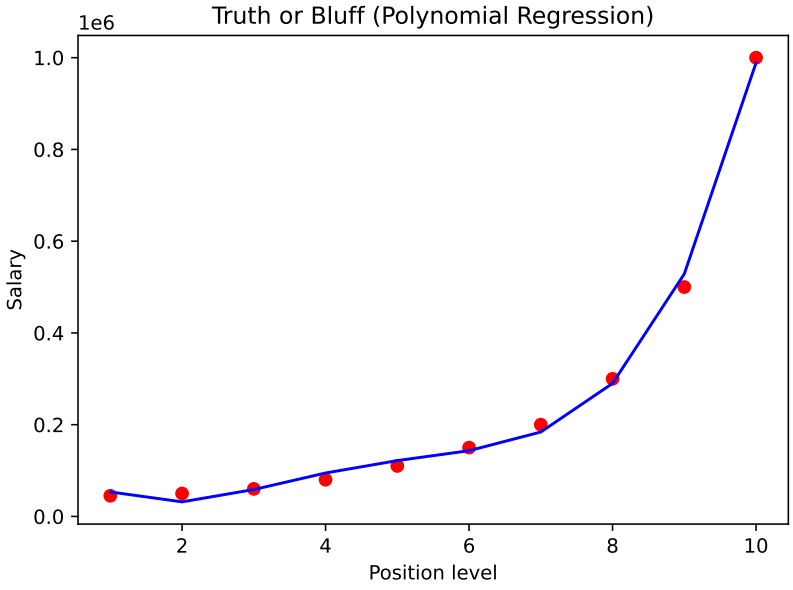
```
# install.packages('ggplot2')
library(ggplot2)
ggplot() +
  geom_point(aes(x = dataset$Level, y = dataset$Salary),
             colour = 'red') +
  geom_line(aes(x = dataset$Level, y = predict(lin_reg, newdata = dataset)),
            colour = 'blue') +
  ggtitle('Truth or Bluff (Linear Regression)') +
  xlab('Level') +
  ylab('Salary')
```



2.3.6 Visualising the Polynomial Regression results

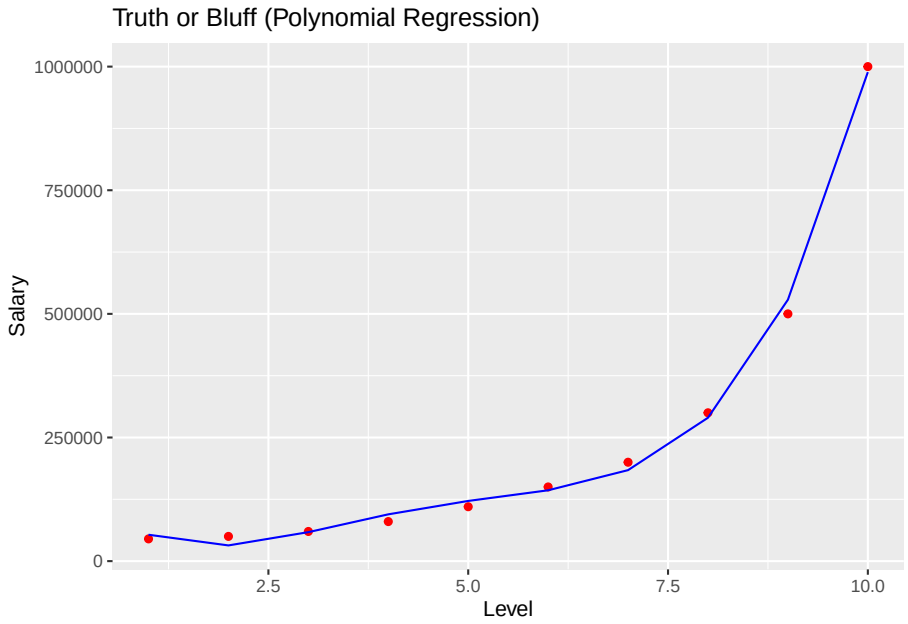
Python

```
plt.scatter(X, y, color = 'red')
plt.plot(X, lin_reg_2.predict(poly_reg.fit_transform(X)), color = 'blue')
plt.title('Truth or Bluff (Polynomial Regression)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()
```



R

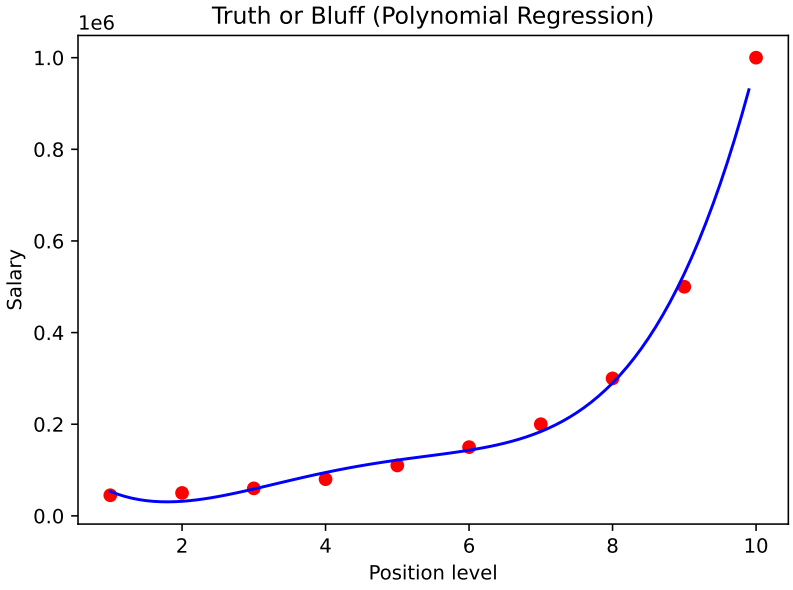
```
# install.packages('ggplot2')
library(ggplot2)
ggplot() +
  geom_point(aes(x = dataset$Level, y = dataset$Salary),
             colour = 'red') +
  geom_line(aes(x = dataset$Level, y = predict(poly_reg, newdata = dataset)),
            colour = 'blue') +
  ggtitle('Truth or Bluff (Polynomial Regression)') +
  xlab('Level') +
  ylab('Salary')
```



2.3.7 Visualising the Polynomial Regression results (for higher resolution and smoother curve)

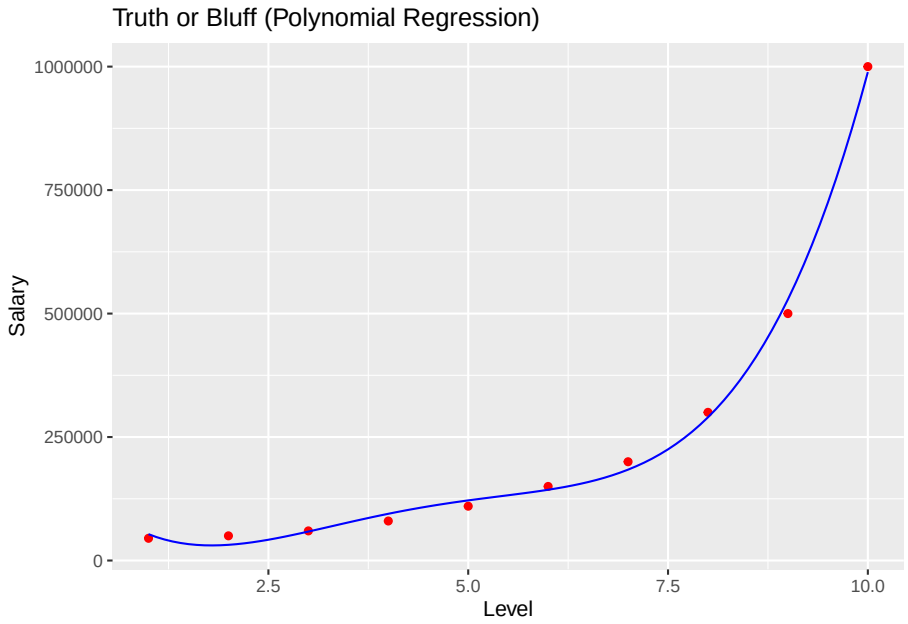
Python

```
X_grid = np.arange(min(X), max(X), 0.1)
X_grid = X_grid.reshape((len(X_grid), 1))
plt.scatter(X, y, color = 'red')
plt.plot(X_grid, lin_reg_2.predict(poly_reg.fit_transform(X_grid)), color = 'blue')
plt.title('Truth or Bluff (Polynomial Regression)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()
```



R

```
# install.packages('ggplot2')
library(ggplot2)
x_grid = seq(min(dataset$Level), max(dataset$Level), 0.1)
ggplot() +
  geom_point(aes(x = dataset$Level, y = dataset$Salary),
             colour = 'red') +
  geom_line(aes(x = x_grid, y = predict(poly_reg,
                                        newdata = data.frame(Level = x_grid,
                                                             Level2 = x_grid^2,
                                                             Level3 = x_grid^3,
                                                             Level4 = x_grid^4))),
            colour = 'blue') +
  ggtitle('Truth or Bluff (Polynomial Regression)') +
  xlab('Level') +
  ylab('Salary')
```



2.3.8 Predicting a new result with Linear Regression

Python

```
lin_reg.predict([[6.5]])
## array([330378.79])
```

R

```
predict(lin_reg, data.frame(Level = 6.5))
##        1
## 330378.8
```

2.3.9 Predicting a new result with Polynomial Regression

Python

```
lin_reg_2.predict(poly_reg.fit_transform([[6.5]]))
## array([158862.45])
```

R

```
predict(poly_reg, data.frame(Level = 6.5,
                             Level2 = 6.5^2,
                             Level3 = 6.5^3,
                             Level4 = 6.5^4))
##        1
```

```
## 158862.5
```

2.4 Support Vector Regression

2.4.1 Importing the libraries

Python

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

2.4.2 Importing the dataset

Python

```
dataset = pd.read_csv('Position_Salaries.csv')
X = dataset.iloc[:, 1:-1].values
y = dataset.iloc[:, -1].values
print(X)
## [[ 1]
##  [ 2]
##  [ 3]
##  [ 4]
##  [ 5]
##  [ 6]
##  [ 7]
##  [ 8]
##  [ 9]
##  [10]]
print(y)
## [  45000   50000   60000   80000  110000  150000  200000  300000  500000
##  1000000]
y = y.reshape(len(y),1)
print(y)
## [[  45000]
##  [  50000]
##  [  60000]
##  [  80000]
##  [ 110000]
##  [ 150000]
##  [ 200000]
##  [ 300000]
##  [ 500000]
##  [1000000]]
```

R

```
dataset = read.csv('Position_Salaries.csv')
dataset = dataset[2:3]
```

2.4.3 Feature Scaling

Python

```
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
sc_y = StandardScaler()
X = sc_X.fit_transform(X)
y = sc_y.fit_transform(y)
print(X)
## [[-1.57]
##  [-1.22]
##  [-0.87]
##  [-0.52]
##  [-0.17]
##  [ 0.17]
##  [ 0.52]
##  [ 0.87]
##  [ 1.22]
##  [ 1.57]]
print(y)
## [[-0.72]
##  [-0.7 ]
##  [-0.67]
##  [-0.6 ]
##  [-0.49]
##  [-0.35]
##  [-0.17]
##  [ 0.18]
##  [ 0.88]
##  [ 2.64]]
```

R

```
# training_set = scale(training_set)
# test_set = scale(test_set)
```

2.4.4 Training the SVR model on the whole dataset

Python

```
from sklearn.svm import SVR
regressor = SVR(kernel = 'rbf')
```

```
regressor.fit(X, y)
## SVR()
##
## C:\Users\murph\mambaforge\lib\site-packages\sklearn\utils\validation.py:63: DataCon
##   return f(*args, **kwargs)
```

R

```
# install.packages('e1071')
library(e1071)
regressor = svm(formula = Salary ~ .,
                data = dataset,
                type = 'eps-regression',
                kernel = 'radial')
```

2.4.5 Predicting a new result

Python

```
sc_y.inverse_transform(regressor.predict(sc_X.transform([[6.5]])))
## array([170370.02])
```

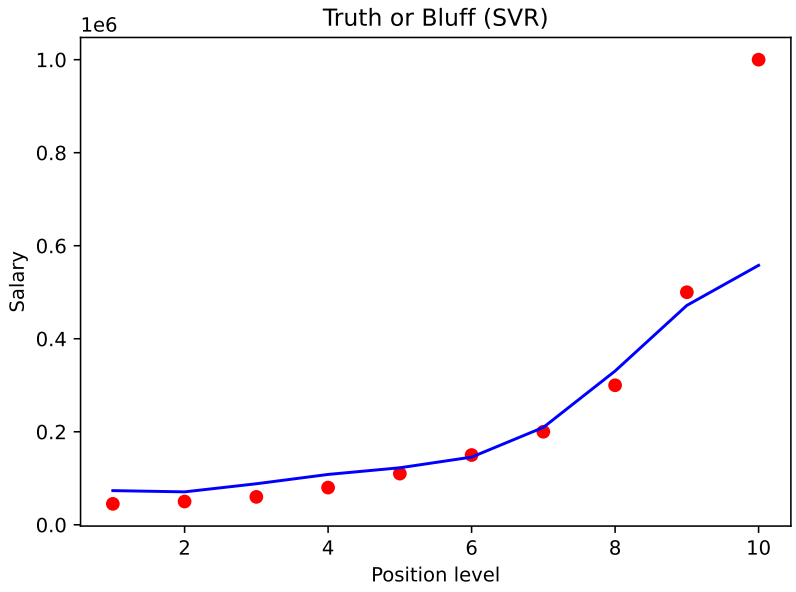
R

```
y_pred = predict(regressor, data.frame(Level = 6.5))
```

2.4.6 Visualising the SVR results

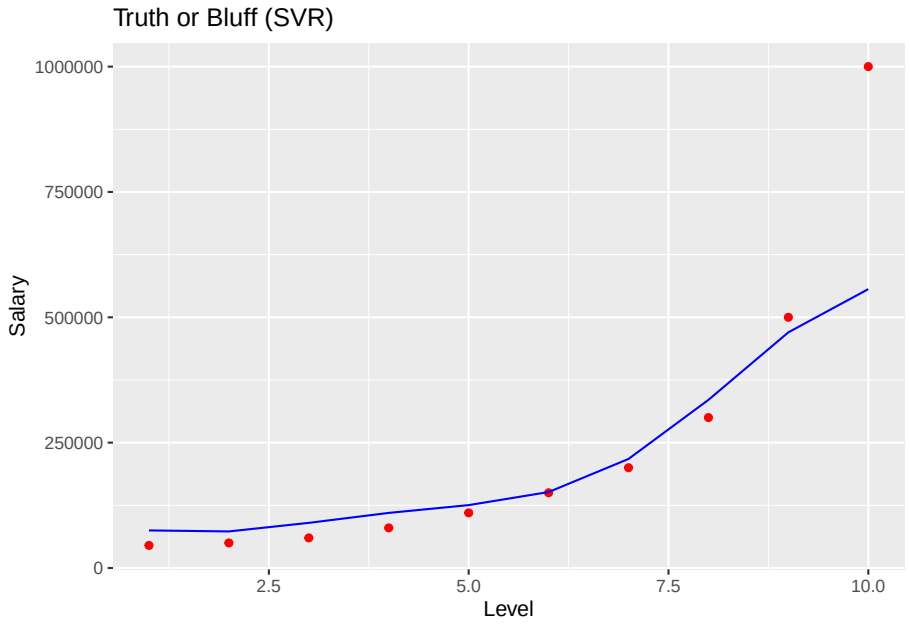
Python

```
plt.scatter(sc_X.inverse_transform(X), sc_y.inverse_transform(y), color = 'red')
plt.plot(sc_X.inverse_transform(X), sc_y.inverse_transform(regressor.predict(X)), colo
plt.title('Truth or Bluff (SVR)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()
```



R

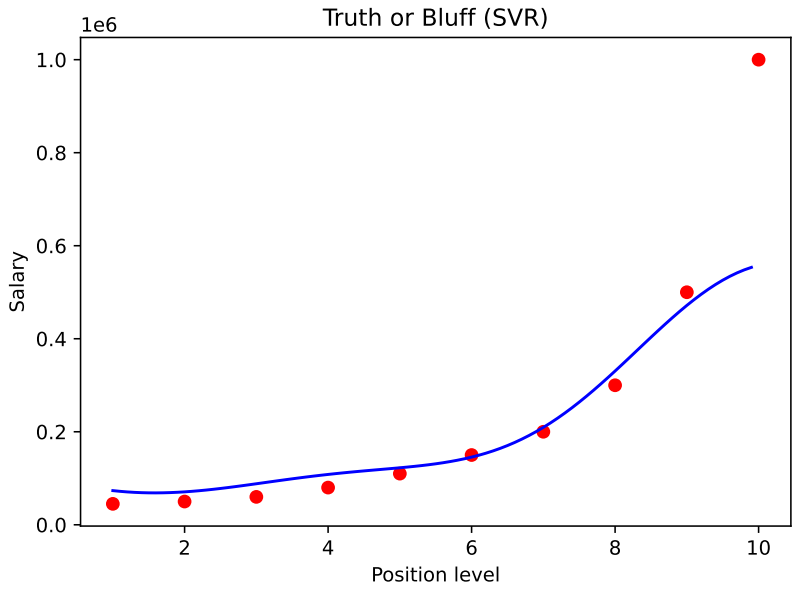
```
# install.packages('ggplot2')
library(ggplot2)
ggplot() +
  geom_point(aes(x = dataset$Level, y = dataset$Salary),
             colour = 'red') +
  geom_line(aes(x = dataset$Level, y = predict(regressor, newdata = dataset)),
            colour = 'blue') +
  ggtitle('Truth or Bluff (SVR)') +
  xlab('Level') +
  ylab('Salary')
```



2.4.7 Visualising the SVR results (for higher resolution and smoother curve)

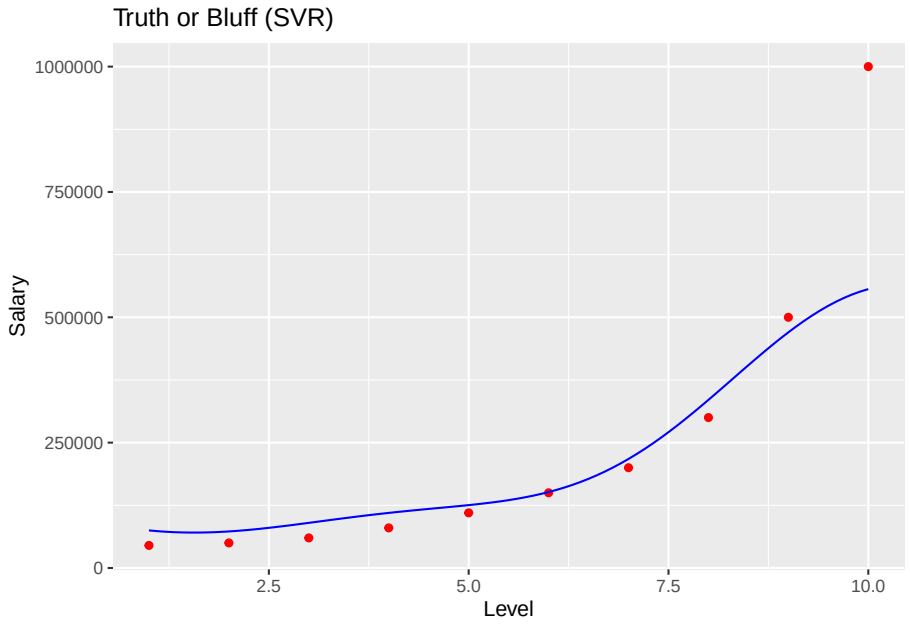
Python

```
X_grid = np.arange(min(sc_X.inverse_transform(X)), max(sc_X.inverse_transform(X)), 0.1)
X_grid = X_grid.reshape((len(X_grid), 1))
plt.scatter(sc_X.inverse_transform(X), sc_y.inverse_transform(y), color = 'red')
plt.plot(X_grid, sc_y.inverse_transform(regressor.predict(sc_X.transform(X_grid))), col
plt.title('Truth or Bluff (SVR)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()
```



R

```r
# install.packages('ggplot2')
library(ggplot2)
x_grid = seq(min(dataset$Level), max(dataset$Level), 0.1)
ggplot() +
  geom_point(aes(x = dataset$Level, y = dataset$Salary),
             colour = 'red') +
  geom_line(aes(x = x_grid, y = predict(regressor, newdata = data.frame(Level = x_grid))),
            colour = 'blue') +
  ggtitle('Truth or Bluff (SVR)') +
  xlab('Level') +
  ylab('Salary')
```



2.5 Decision Tree Regression

2.5.1 Importing the libraries

Python

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

2.5.2 Importing the dataset

Python

```
dataset = pd.read_csv('Position_Salaries.csv')
X = dataset.iloc[:, 1:-1].values
y = dataset.iloc[:, -1].values
```

R

```
dataset = read.csv('Position_Salaries.csv')
dataset = dataset[2:3]
```

2.5.3 Training the Decision Tree Regression model on the whole dataset

Python

```
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor(random_state = 0)
regressor.fit(X, y)
## DecisionTreeRegressor(random_state=0)
```

R

```
# install.packages('rpart')
library(rpart)
regressor = rpart(formula = Salary ~ .,
                  data = dataset,
                  control = rpart.control(minsplit = 1))
```

2.5.4 Predicting a new result

Python

```
regressor.predict([[6.5]])
## array([150000.])
```

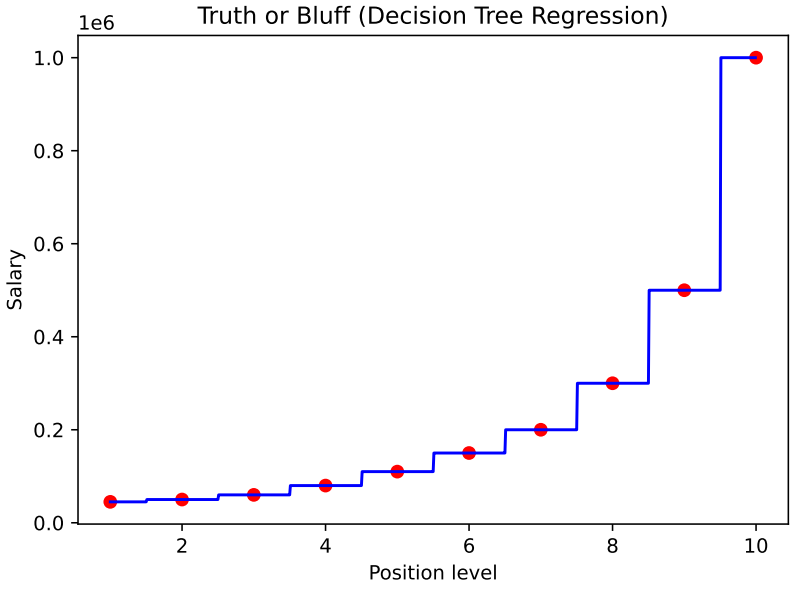
R

```
y_pred = predict(regressor, data.frame(Level = 6.5))
```

2.5.5 Visualising the Decision Tree Regression results (higher resolution)

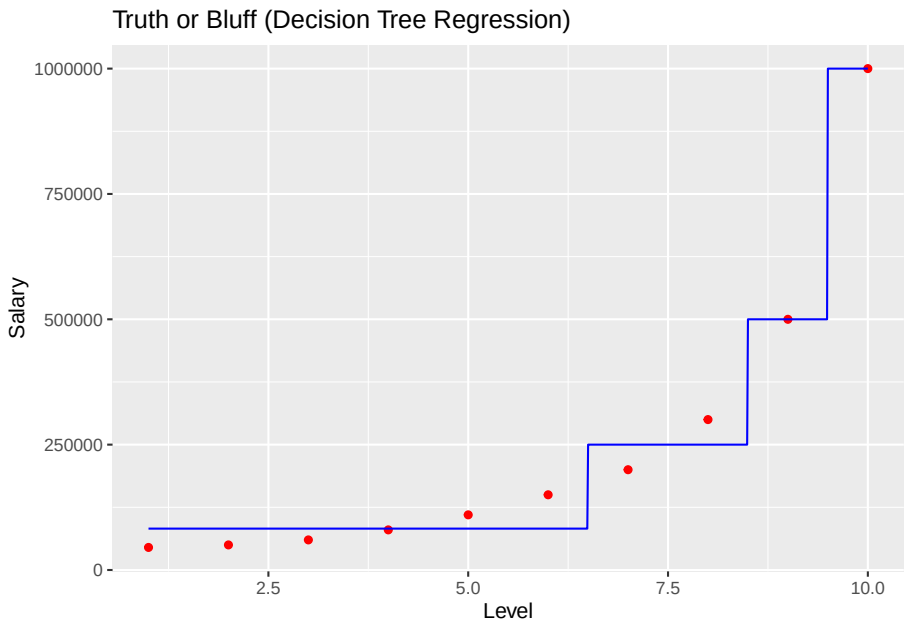
Python

```
X_grid = np.arange(min(X), max(X), 0.01)
X_grid = X_grid.reshape((len(X_grid), 1))
plt.scatter(X, y, color = 'red')
plt.plot(X_grid, regressor.predict(X_grid), color = 'blue')
plt.title('Truth or Bluff (Decision Tree Regression)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()
```



R

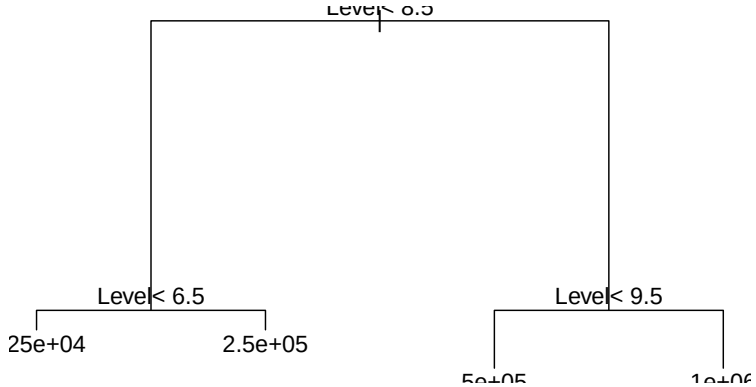
```
# install.packages('ggplot2')
library(ggplot2)
x_grid = seq(min(dataset$Level), max(dataset$Level), 0.01)
ggplot() +
  geom_point(aes(x = dataset$Level, y = dataset$Salary),
             colour = 'red') +
  geom_line(aes(x = x_grid, y = predict(regressor, newdata = data.frame(Level = x_grid
            colour = 'blue') +
  ggtitle('Truth or Bluff (Decision Tree Regression)') +
  xlab('Level') +
  ylab('Salary')
```



2.5.6 Plotting the tree

R

```
plot(regressor)
text(regressor)
```



2.6 Random Forest Regression

2.6.1 Importing the libraries

Python

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

2.6.2 Importing the dataset

Python

```
dataset = pd.read_csv('Position_Salaries.csv')
X = dataset.iloc[:, 1:-1].values
y = dataset.iloc[:, -1].values
```

R

```
dataset = read.csv('Position_Salaries.csv')
dataset = dataset[2:3]
```

2.6.3 Training the Random Forest Regression model on the whole dataset

Python

```
from sklearn.ensemble import RandomForestRegressor
regressor = RandomForestRegressor(n_estimators = 10, random_state = 0)
regressor.fit(X, y)
## RandomForestRegressor(n_estimators=10, random_state=0)
```

R

```
# install.packages('randomForest')
library(randomForest)
## randomForest 4.6-14
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
## The following object is masked from 'package:ggplot2':
##
##     margin
set.seed(1234)
regressor = randomForest(x = dataset[-2],
                         y = dataset$Salary,
                         ntree = 500)
```

2.6.4 Predicting a new result

Python

```
regressor.predict([[6.5]])
## array([167000.])
```

R

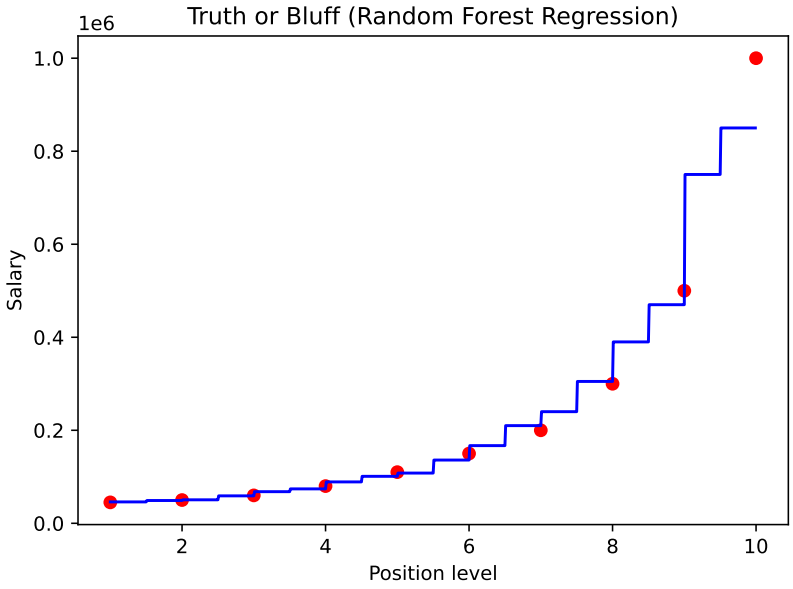
```
y_pred = predict(regressor, data.frame(Level = 6.5))
```

2.6.5 Visualising the Random Forest Regression results (higher resolution)

Python

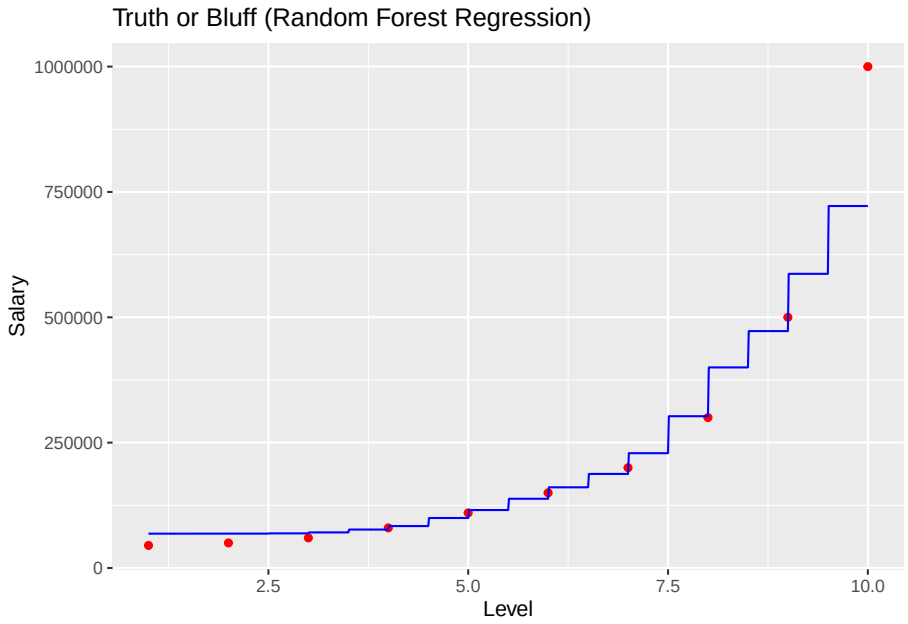
```
X_grid = np.arange(min(X), max(X), 0.01)
X_grid = X_grid.reshape((len(X_grid), 1))
plt.scatter(X, y, color = 'red')
plt.plot(X_grid, regressor.predict(X_grid), color = 'blue')
plt.title('Truth or Bluff (Random Forest Regression)')
plt.xlabel('Position level')
```

```
plt.ylabel('Salary')
plt.show()
```



R

```
# install.packages('ggplot2')
library(ggplot2)
x_grid = seq(min(dataset$Level), max(dataset$Level), 0.01)
ggplot() +
  geom_point(aes(x = dataset$Level, y = dataset$Salary),
             colour = 'red') +
  geom_line(aes(x = x_grid, y = predict(regressor, newdata = data.frame(Level = x_grid
            colour = 'blue') +
  ggtitle('Truth or Bluff (Random Forest Regression)') +
  xlab('Level') +
  ylab('Salary')
```


Truth or Bluff (Random Forest Regression)
1000000
750000
500000
250000
0
Salary
2.5
5.0
7.5
10.0
Level

Chapter 3

Classification

3.1 Logistic Regression

3.1.1 Importing the libraries

Python

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

3.1.2 Importing the dataset

Python

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

R

```
dataset = read.csv('Social_Network_Ads.csv')
# dataset = dataset[3:5]
```

3.1.3 Encoding the target feature as factor

R

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

3.1.4 Splitting the dataset into the Training set and Test set

Python

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_stat
print(X_train)
## [[    44  39000]
##  [    32 120000]
##  [    38  50000]
##  [    32 135000]
##  [    52  21000]
##  [    53 104000]
##  [    39  42000]
##  [    38  61000]
##  [    36  50000]
##  [    36  63000]
##  [    35  25000]
##  [    35  50000]
##  [    42  73000]
##  [    47  49000]
##  [    59  29000]
##  [    49  65000]
##  [    45 131000]
##  [    31  89000]
##  [    46  82000]
##  [    47  51000]
##  [    26  15000]
##  [    60 102000]
##  [    38 112000]
##  [    40 107000]
##  [    42  53000]
##  [    35  59000]
##  [    48  41000]
##  [    48 134000]
##  [    38 113000]
##  [    29 148000]
##  [    26  15000]
##  [    60  42000]
##  [    24  19000]
##  [    42 149000]
##  [    46  96000]
##  [    28  59000]
##  [    39  96000]
##  [    28  89000]
```

```
##  [    41  72000]
##  [    45  26000]
##  [    33  69000]
##  [    20  82000]
##  [    31  74000]
##  [    42  80000]
##  [    35  72000]
##  [    33 149000]
##  [    40  71000]
##  [    51 146000]
##  [    46  79000]
##  [    35  75000]
##  [    38  51000]
##  [    36  75000]
##  [    37  78000]
##  [    38  61000]
##  [    60 108000]
##  [    20  82000]
##  [    57  74000]
##  [    42  65000]
##  [    26  80000]
##  [    46 117000]
##  [    35  61000]
##  [    21  68000]
##  [    28  44000]
##  [    41  87000]
##  [    37  33000]
##  [    27  90000]
##  [    39  42000]
##  [    28 123000]
##  [    31 118000]
##  [    25  87000]
##  [    35  71000]
##  [    37  70000]
##  [    35  39000]
##  [    47  23000]
##  [    35 147000]
##  [    48 138000]
##  [    26  86000]
##  [    25  79000]
##  [    52 138000]
##  [    51  23000]
##  [    35  60000]
##  [    33 113000]
##  [    30 107000]
```

```
##  [    48  33000]
##  [    41  80000]
##  [    48  96000]
##  [    31  18000]
##  [    31  71000]
##  [    43 129000]
##  [    59  76000]
##  [    18  44000]
##  [    36 118000]
##  [    42  90000]
##  [    47  30000]
##  [    26  43000]
##  [    40  78000]
##  [    46  59000]
##  [    59  42000]
##  [    46  74000]
##  [    35  91000]
##  [    28  59000]
##  [    40  57000]
##  [    59 143000]
##  [    57  26000]
##  [    52  38000]
##  [    47 113000]
##  [    53 143000]
##  [    35  27000]
##  [    58 101000]
##  [    45  45000]
##  [    23  82000]
##  [    46  23000]
##  [    42  65000]
##  [    28  84000]
##  [    38  59000]
##  [    26  84000]
##  [    29  28000]
##  [    37  71000]
##  [    22  55000]
##  [    48  35000]
##  [    49  28000]
##  [    38  65000]
##  [    27  17000]
##  [    46  28000]
##  [    48 141000]
##  [    26  17000]
##  [    35  97000]
##  [    39  59000]
```

```
##  [    24  27000]
##  [    32  18000]
##  [    46  88000]
##  [    35  58000]
##  [    56  60000]
##  [    47  34000]
##  [    40  72000]
##  [    32 100000]
##  [    19  21000]
##  [    25  90000]
##  [    35  88000]
##  [    28  32000]
##  [    50  20000]
##  [    40  59000]
##  [    50  44000]
##  [    35  72000]
##  [    40 142000]
##  [    46  32000]
##  [    39  71000]
##  [    20  74000]
##  [    29  75000]
##  [    31  76000]
##  [    47  25000]
##  [    40  61000]
##  [    34 112000]
##  [    38  80000]
##  [    42  75000]
##  [    47  47000]
##  [    39  75000]
##  [    19  25000]
##  [    37  80000]
##  [    36  60000]
##  [    41  52000]
##  [    36 125000]
##  [    48  29000]
##  [    36 126000]
##  [    51 134000]
##  [    27  57000]
##  [    38  71000]
##  [    39  61000]
##  [    22  27000]
##  [    33  60000]
##  [    48  74000]
##  [    58  23000]
##  [    53  72000]
```

```
##  [    32 117000]
##  [    54  70000]
##  [    30  80000]
##  [    58  95000]
##  [    26  52000]
##  [    45  79000]
##  [    24  55000]
##  [    40  75000]
##  [    33  28000]
##  [    44 139000]
##  [    22  18000]
##  [    33  51000]
##  [    43 133000]
##  [    24  32000]
##  [    46  22000]
##  [    35  55000]
##  [    54 104000]
##  [    48 119000]
##  [    35  53000]
##  [    37 144000]
##  [    23  66000]
##  [    37 137000]
##  [    31  58000]
##  [    33  41000]
##  [    45  22000]
##  [    30  15000]
##  [    19  19000]
##  [    49  74000]
##  [    39 122000]
##  [    35  73000]
##  [    39  71000]
##  [    24  23000]
##  [    41  72000]
##  [    29  83000]
##  [    54  26000]
##  [    35  44000]
##  [    37  75000]
##  [    29  47000]
##  [    31  68000]
##  [    42  54000]
##  [    30 135000]
##  [    52 114000]
##  [    50  36000]
##  [    56 133000]
##  [    29  61000]
```

```
##  [    30  89000]
##  [    26  16000]
##  [    33  31000]
##  [    41  72000]
##  [    36  33000]
##  [    55 125000]
##  [    48 131000]
##  [    41  71000]
##  [    30  62000]
##  [    37  72000]
##  [    41  63000]
##  [    58  47000]
##  [    30 116000]
##  [    20  49000]
##  [    37  74000]
##  [    41  59000]
##  [    49  89000]
##  [    28  79000]
##  [    53  82000]
##  [    40  57000]
##  [    60  34000]
##  [    35 108000]
##  [    21  72000]
##  [    38  71000]
##  [    39 106000]
##  [    37  57000]
##  [    26  72000]
##  [    35  23000]
##  [    54 108000]
##  [    30  17000]
##  [    39 134000]
##  [    29  43000]
##  [    33  43000]
##  [    35  38000]
##  [    41  45000]
##  [    41  72000]
##  [    39 134000]
##  [    27 137000]
##  [    21  16000]
##  [    26  32000]
##  [    31  66000]
##  [    39  73000]
##  [    41  79000]
##  [    47  50000]
##  [    41  30000]
```

```
##  [    37  93000]
##  [    60  46000]
##  [    25  22000]
##  [    28  37000]
##  [    38  55000]
##  [    36  54000]
##  [    20  36000]
##  [    56 104000]
##  [    40  57000]
##  [    42 108000]
##  [    20  23000]
##  [    40  65000]
##  [    47  20000]
##  [    18  86000]
##  [    35  79000]
##  [    57  33000]
##  [    34  72000]
##  [    49  39000]
##  [    27  31000]
##  [    19  70000]
##  [    39  79000]
##  [    26  81000]
##  [    25  80000]
##  [    28  85000]
##  [    55  39000]
##  [    50  88000]
##  [    49  88000]
##  [    52 150000]
##  [    35  65000]
##  [    42  54000]
##  [    34  43000]
##  [    37  52000]
##  [    48  30000]
##  [    29  43000]
##  [    36  52000]
##  [    27  54000]
##  [    26 118000]]
print(y_train)
## [0 1 0 1 1 1 0 0 0 0 0 0 1 1 1 0 1 0 0 1 0 1 0 1 0 0 1 1 1 1 0 1 0 1 0 0 1
##  0 0 1 0 0 0 0 0 1 1 1 1 0 0 0 1 0 1 0 1 0 0 1 0 0 0 1 0 0 0 1 1 0 0 1 0 1
##  1 1 0 0 1 1 0 0 1 1 0 1 0 0 1 1 0 1 1 1 0 0 0 0 0 1 0 0 1 1 1 1 1 0 1 1 0
##  1 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 1 0 0 0 1 0 1 1 0 1 0 0 0 0 1 0 0 0 1 1 0 0
##  0 0 1 0 1 0 0 0 1 0 0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 1 1 0 1 0 0 0 0 0 1 0 0
##  0 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 1 1 0 0 0 0 0
##  0 1 1 0 0 0 0 1 0 0 0 0 1 0 1 0 1 0 0 0 1 0 0 0 1 0 1 0 0 0 0 0 1 1 0 0 0
```

```
##  0 0 1 0 1 1 0 0 0 0 0 1 0 1 0 0 1 0 0 1 0 1 0 0 0 0 0 0 1 1 1 1 0 0 0 0 1
##  0 0 0 0]
print(X_test)
## [[    30  87000]
##  [    38  50000]
##  [    35  75000]
##  [    30  79000]
##  [    35  50000]
##  [    27  20000]
##  [    31  15000]
##  [    36 144000]
##  [    18  68000]
##  [    47  43000]
##  [    30  49000]
##  [    28  55000]
##  [    37  55000]
##  [    39  77000]
##  [    20  86000]
##  [    32 117000]
##  [    37  77000]
##  [    19  85000]
##  [    55 130000]
##  [    35  22000]
##  [    35  47000]
##  [    47 144000]
##  [    41  51000]
##  [    47 105000]
##  [    23  28000]
##  [    49 141000]
##  [    28  87000]
##  [    29  80000]
##  [    37  62000]
##  [    32  86000]
##  [    21  88000]
##  [    37  79000]
##  [    57  60000]
##  [    37  53000]
##  [    24  58000]
##  [    18  52000]
##  [    22  81000]
##  [    34  43000]
##  [    31  34000]
##  [    49  36000]
##  [    27  88000]
##  [    41  52000]
```

```
##  [    27  84000]
##  [    35  20000]
##  [    43 112000]
##  [    27  58000]
##  [    37  80000]
##  [    52  90000]
##  [    26  30000]
##  [    49  86000]
##  [    57 122000]
##  [    34  25000]
##  [    35  57000]
##  [    34 115000]
##  [    59  88000]
##  [    45  32000]
##  [    29  83000]
##  [    26  80000]
##  [    49  28000]
##  [    23  20000]
##  [    32  18000]
##  [    60  42000]
##  [    19  76000]
##  [    36  99000]
##  [    19  26000]
##  [    60  83000]
##  [    24  89000]
##  [    27  58000]
##  [    40  47000]
##  [    42  70000]
##  [    32 150000]
##  [    35  77000]
##  [    22  63000]
##  [    45  22000]
##  [    27  89000]
##  [    18  82000]
##  [    42  79000]
##  [    40  60000]
##  [    53  34000]
##  [    47 107000]
##  [    58 144000]
##  [    59  83000]
##  [    24  55000]
##  [    26  35000]
##  [    58  38000]
##  [    42  80000]
##  [    40  75000]
```

```
##  [    59 130000]
##  [    46  41000]
##  [    41  60000]
##  [    42  64000]
##  [    37 146000]
##  [    23  48000]
##  [    25  33000]
##  [    24  84000]
##  [    27  96000]
##  [    23  63000]
##  [    48  33000]
##  [    48  90000]
##  [    42 104000]]
print(y_test)
## [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 1 0 0 0 0 0 1 1 0 0 0 0
##  0 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 0 1 1 0 0 1 0 0 1 0 1 0 1 0 0 0 0 1 0 0 1
##  0 0 0 0 1 1 1 0 0 0 1 1 0 1 1 0 0 1 0 0 0 1 0 1 1 1]
```

R

```
# 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)
```

3.1.5 Feature Scaling

Python

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
print(X_train)
## [[ 0.58 -0.89]
##  [-0.61  1.46]
##  [-0.01 -0.57]
##  [-0.61  1.9 ]
##  [ 1.37 -1.41]
##  [ 1.47  1.  ]
##  [ 0.09 -0.8 ]
##  [-0.01 -0.25]
##  [-0.21 -0.57]
##  [-0.21 -0.19]
```

```
##  [-0.31 -1.29]
##  [-0.31 -0.57]
##  [ 0.38  0.1 ]
##  [ 0.88 -0.6 ]
##  [ 2.07 -1.18]
##  [ 1.08 -0.13]
##  [ 0.68  1.78]
##  [-0.71  0.56]
##  [ 0.78  0.36]
##  [ 0.88 -0.54]
##  [-1.2  -1.58]
##  [ 2.17  0.94]
##  [-0.01  1.23]
##  [ 0.19  1.08]
##  [ 0.38 -0.48]
##  [-0.31 -0.31]
##  [ 0.98 -0.83]
##  [ 0.98  1.87]
##  [-0.01  1.26]
##  [-0.9   2.27]
##  [-1.2  -1.58]
##  [ 2.17 -0.8 ]
##  [-1.4  -1.47]
##  [ 0.38  2.3 ]
##  [ 0.78  0.77]
##  [-1.   -0.31]
##  [ 0.09  0.77]
##  [-1.    0.56]
##  [ 0.28  0.07]
##  [ 0.68 -1.26]
##  [-0.51 -0.02]
##  [-1.8   0.36]
##  [-0.71  0.13]
##  [ 0.38  0.3 ]
##  [-0.31  0.07]
##  [-0.51  2.3 ]
##  [ 0.19  0.04]
##  [ 1.27  2.22]
##  [ 0.78  0.27]
##  [-0.31  0.16]
##  [-0.01 -0.54]
##  [-0.21  0.16]
##  [-0.11  0.24]
##  [-0.01 -0.25]
##  [ 2.17  1.11]
```

```
##  [-1.8   0.36]
##  [ 1.87  0.13]
##  [ 0.38 -0.13]
##  [-1.2   0.3 ]
##  [ 0.78  1.37]
##  [-0.31 -0.25]
##  [-1.7  -0.05]
##  [-1.   -0.74]
##  [ 0.28  0.5 ]
##  [-0.11 -1.06]
##  [-1.1   0.59]
##  [ 0.09 -0.8 ]
##  [-1.    1.55]
##  [-0.71  1.4 ]
##  [-1.3   0.5 ]
##  [-0.31  0.04]
##  [-0.11  0.01]
##  [-0.31 -0.89]
##  [ 0.88 -1.35]
##  [-0.31  2.24]
##  [ 0.98  1.98]
##  [-1.2   0.48]
##  [-1.3   0.27]
##  [ 1.37  1.98]
##  [ 1.27 -1.35]
##  [-0.31 -0.28]
##  [-0.51  1.26]
##  [-0.8   1.08]
##  [ 0.98 -1.06]
##  [ 0.28  0.3 ]
##  [ 0.98  0.77]
##  [-0.71 -1.5 ]
##  [-0.71  0.04]
##  [ 0.48  1.72]
##  [ 2.07  0.19]
##  [-1.99 -0.74]
##  [-0.21  1.4 ]
##  [ 0.38  0.59]
##  [ 0.88 -1.15]
##  [-1.2  -0.77]
##  [ 0.19  0.24]
##  [ 0.78 -0.31]
##  [ 2.07 -0.8 ]
##  [ 0.78  0.13]
##  [-0.31  0.62]
```

```
##  [-1.   -0.31]
##  [ 0.19 -0.36]
##  [ 2.07  2.13]
##  [ 1.87 -1.26]
##  [ 1.37 -0.92]
##  [ 0.88  1.26]
##  [ 1.47  2.13]
##  [-0.31 -1.23]
##  [ 1.97  0.91]
##  [ 0.68 -0.71]
##  [-1.5   0.36]
##  [ 0.78 -1.35]
##  [ 0.38 -0.13]
##  [-1.    0.42]
##  [-0.01 -0.31]
##  [-1.2   0.42]
##  [-0.9  -1.21]
##  [-0.11  0.04]
##  [-1.6  -0.42]
##  [ 0.98 -1.  ]
##  [ 1.08 -1.21]
##  [-0.01 -0.13]
##  [-1.1  -1.52]
##  [ 0.78 -1.21]
##  [ 0.98  2.07]
##  [-1.2  -1.52]
##  [-0.31  0.79]
##  [ 0.09 -0.31]
##  [-1.4  -1.23]
##  [-0.61 -1.5 ]
##  [ 0.78  0.53]
##  [-0.31 -0.34]
##  [ 1.77 -0.28]
##  [ 0.88 -1.03]
##  [ 0.19  0.07]
##  [-0.61  0.88]
##  [-1.89 -1.41]
##  [-1.3   0.59]
##  [-0.31  0.53]
##  [-1.   -1.09]
##  [ 1.18 -1.44]
##  [ 0.19 -0.31]
##  [ 1.18 -0.74]
##  [-0.31  0.07]
##  [ 0.19  2.1 ]
```

```
##  [ 0.78 -1.09]
##  [ 0.09  0.04]
##  [-1.8   0.13]
##  [-0.9   0.16]
##  [-0.71  0.19]
##  [ 0.88 -1.29]
##  [ 0.19 -0.25]
##  [-0.41  1.23]
##  [-0.01  0.3 ]
##  [ 0.38  0.16]
##  [ 0.88 -0.65]
##  [ 0.09  0.16]
##  [-1.89 -1.29]
##  [-0.11  0.3 ]
##  [-0.21 -0.28]
##  [ 0.28 -0.51]
##  [-0.21  1.61]
##  [ 0.98 -1.18]
##  [-0.21  1.64]
##  [ 1.27  1.87]
##  [-1.1  -0.36]
##  [-0.01  0.04]
##  [ 0.09 -0.25]
##  [-1.6  -1.23]
##  [-0.51 -0.28]
##  [ 0.98  0.13]
##  [ 1.97 -1.35]
##  [ 1.47  0.07]
##  [-0.61  1.37]
##  [ 1.57  0.01]
##  [-0.8   0.3 ]
##  [ 1.97  0.74]
##  [-1.2  -0.51]
##  [ 0.68  0.27]
##  [-1.4  -0.42]
##  [ 0.19  0.16]
##  [-0.51 -1.21]
##  [ 0.58  2.01]
##  [-1.6  -1.5 ]
##  [-0.51 -0.54]
##  [ 0.48  1.84]
##  [-1.4  -1.09]
##  [ 0.78 -1.38]
##  [-0.31 -0.42]
##  [ 1.57  1.  ]
```

```
##  [ 0.98  1.43]
##  [-0.31 -0.48]
##  [-0.11  2.16]
##  [-1.5  -0.1 ]
##  [-0.11  1.95]
##  [-0.71 -0.34]
##  [-0.51 -0.83]
##  [ 0.68 -1.38]
##  [-0.8  -1.58]
##  [-1.89 -1.47]
##  [ 1.08  0.13]
##  [ 0.09  1.52]
##  [-0.31  0.1 ]
##  [ 0.09  0.04]
##  [-1.4  -1.35]
##  [ 0.28  0.07]
##  [-0.9   0.39]
##  [ 1.57 -1.26]
##  [-0.31 -0.74]
##  [-0.11  0.16]
##  [-0.9  -0.65]
##  [-0.71 -0.05]
##  [ 0.38 -0.45]
##  [-0.8   1.9 ]
##  [ 1.37  1.29]
##  [ 1.18 -0.97]
##  [ 1.77  1.84]
##  [-0.9  -0.25]
##  [-0.8   0.56]
##  [-1.2  -1.55]
##  [-0.51 -1.12]
##  [ 0.28  0.07]
##  [-0.21 -1.06]
##  [ 1.67  1.61]
##  [ 0.98  1.78]
##  [ 0.28  0.04]
##  [-0.8  -0.22]
##  [-0.11  0.07]
##  [ 0.28 -0.19]
##  [ 1.97 -0.65]
##  [-0.8   1.35]
##  [-1.8  -0.6 ]
##  [-0.11  0.13]
##  [ 0.28 -0.31]
##  [ 1.08  0.56]
```

```
##  [-1.    0.27]
##  [ 1.47  0.36]
##  [ 0.19 -0.36]
##  [ 2.17 -1.03]
##  [-0.31  1.11]
##  [-1.7   0.07]
##  [-0.01  0.04]
##  [ 0.09  1.06]
##  [-0.11 -0.36]
##  [-1.2   0.07]
##  [-0.31 -1.35]
##  [ 1.57  1.11]
##  [-0.8  -1.52]
##  [ 0.09  1.87]
##  [-0.9  -0.77]
##  [-0.51 -0.77]
##  [-0.31 -0.92]
##  [ 0.28 -0.71]
##  [ 0.28  0.07]
##  [ 0.09  1.87]
##  [-1.1   1.95]
##  [-1.7  -1.55]
##  [-1.2  -1.09]
##  [-0.71 -0.1 ]
##  [ 0.09  0.1 ]
##  [ 0.28  0.27]
##  [ 0.88 -0.57]
##  [ 0.28 -1.15]
##  [-0.11  0.68]
##  [ 2.17 -0.68]
##  [-1.3  -1.38]
##  [-1.   -0.94]
##  [-0.01 -0.42]
##  [-0.21 -0.45]
##  [-1.8  -0.97]
##  [ 1.77  1.  ]
##  [ 0.19 -0.36]
##  [ 0.38  1.11]
##  [-1.8  -1.35]
##  [ 0.19 -0.13]
##  [ 0.88 -1.44]
##  [-1.99  0.48]
##  [-0.31  0.27]
##  [ 1.87 -1.06]
##  [-0.41  0.07]
```

```
##  [ 1.08 -0.89]
##  [-1.1  -1.12]
##  [-1.89  0.01]
##  [ 0.09  0.27]
##  [-1.2   0.33]
##  [-1.3   0.3 ]
##  [-1.    0.45]
##  [ 1.67 -0.89]
##  [ 1.18  0.53]
##  [ 1.08  0.53]
##  [ 1.37  2.33]
##  [-0.31 -0.13]
##  [ 0.38 -0.45]
##  [-0.41 -0.77]
##  [-0.11 -0.51]
##  [ 0.98 -1.15]
##  [-0.9  -0.77]
##  [-0.21 -0.51]
##  [-1.1  -0.45]
##  [-1.2   1.4 ]]
print(X_test)
## [[-0.8   0.5 ]
##  [-0.01 -0.57]
##  [-0.31  0.16]
##  [-0.8   0.27]
##  [-0.31 -0.57]
##  [-1.1  -1.44]
##  [-0.71 -1.58]
##  [-0.21  2.16]
##  [-1.99 -0.05]
##  [ 0.88 -0.77]
##  [-0.8  -0.6 ]
##  [-1.   -0.42]
##  [-0.11 -0.42]
##  [ 0.09  0.22]
##  [-1.8   0.48]
##  [-0.61  1.37]
##  [-0.11  0.22]
##  [-1.89  0.45]
##  [ 1.67  1.75]
##  [-0.31 -1.38]
##  [-0.31 -0.65]
##  [ 0.88  2.16]
##  [ 0.28 -0.54]
##  [ 0.88  1.03]
```

```
##  [-1.5  -1.21]
##  [ 1.08  2.07]
##  [-1.    0.5 ]
##  [-0.9   0.3 ]
##  [-0.11 -0.22]
##  [-0.61  0.48]
##  [-1.7   0.53]
##  [-0.11  0.27]
##  [ 1.87 -0.28]
##  [-0.11 -0.48]
##  [-1.4  -0.34]
##  [-1.99 -0.51]
##  [-1.6   0.33]
##  [-0.41 -0.77]
##  [-0.71 -1.03]
##  [ 1.08 -0.97]
##  [-1.1   0.53]
##  [ 0.28 -0.51]
##  [-1.1   0.42]
##  [-0.31 -1.44]
##  [ 0.48  1.23]
##  [-1.1  -0.34]
##  [-0.11  0.3 ]
##  [ 1.37  0.59]
##  [-1.2  -1.15]
##  [ 1.08  0.48]
##  [ 1.87  1.52]
##  [-0.41 -1.29]
##  [-0.31 -0.36]
##  [-0.41  1.32]
##  [ 2.07  0.53]
##  [ 0.68 -1.09]
##  [-0.9   0.39]
##  [-1.2   0.3 ]
##  [ 1.08 -1.21]
##  [-1.5  -1.44]
##  [-0.61 -1.5 ]
##  [ 2.17 -0.8 ]
##  [-1.89  0.19]
##  [-0.21  0.85]
##  [-1.89 -1.26]
##  [ 2.17  0.39]
##  [-1.4   0.56]
##  [-1.1  -0.34]
##  [ 0.19 -0.65]
```

```
##  [ 0.38  0.01]
##  [-0.61  2.33]
##  [-0.31  0.22]
##  [-1.6  -0.19]
##  [ 0.68 -1.38]
##  [-1.1   0.56]
##  [-1.99  0.36]
##  [ 0.38  0.27]
##  [ 0.19 -0.28]
##  [ 1.47 -1.03]
##  [ 0.88  1.08]
##  [ 1.97  2.16]
##  [ 2.07  0.39]
##  [-1.4  -0.42]
##  [-1.2  -1.  ]
##  [ 1.97 -0.92]
##  [ 0.38  0.3 ]
##  [ 0.19  0.16]
##  [ 2.07  1.75]
##  [ 0.78 -0.83]
##  [ 0.28 -0.28]
##  [ 0.38 -0.16]
##  [-0.11  2.22]
##  [-1.5  -0.63]
##  [-1.3  -1.06]
##  [-1.4   0.42]
##  [-1.1   0.77]
##  [-1.5  -0.19]
##  [ 0.98 -1.06]
##  [ 0.98  0.59]
##  [ 0.38  1.  ]]
```

R

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

3.1.6 Training the Logistic Regression model on the Training set

Python

```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train, y_train)
## LogisticRegression(random_state=0)
```

R

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

3.1.7 Predicting a new result

Python

```
print(classifier.predict(sc.transform([[30,87000]])))
## [0]
```

3.1.8 Predicting the Test set results

Python

```
y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))
## [[0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [1 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [1 1]
##  [0 0]
##  [1 1]
##  [0 0]
##  [0 0]
```

```
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 1]
##  [1 1]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [1 1]
##  [1 1]
##  [0 0]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 1]
##  [0 0]
##  [0 0]
##  [0 1]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [0 1]
##  [0 0]
##  [1 1]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [0 0]
```

```
##  [0 1]
##  [0 0]
##  [0 0]
##  [1 0]
##  [0 0]
##  [1 1]
##  [1 1]
##  [1 1]
##  [1 0]
##  [0 0]
##  [0 0]
##  [1 1]
##  [1 1]
##  [0 0]
##  [1 1]
##  [0 1]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 1]
##  [0 0]
##  [0 1]
##  [1 1]
##  [1 1]]
```

R

```
prob_pred = predict(classifier, type = 'response', newdata = test_set[-3])
y_pred = ifelse(prob_pred > 0.5, 1, 0)
```

3.1.9 Making the Confusion Matrix

Python

```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
## [[65  3]
##  [ 8 24]]
accuracy_score(y_test, y_pred)
## 0.89
```

R

```
cm = table(test_set[, 3], y_pred > 0.5)
```

3.1.10 Visualising the Training set results

Python

```
from matplotlib.colors import ListedColormap
X_set, y_set = sc.inverse_transform(X_train), y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 10, stop = X_set[:, 0].max()
                     np.arange(start = X_set[:, 1].min() - 1000, stop = X_set[:, 1].ma
plt.contourf(X1, X2, classifier.predict(sc.transform(np.array([X1.ravel(), X2.ravel()]
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], c = ListedColormap(('red',
plt.title('Logistic Regression (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

R

```
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'))
```

3.1.11 Visualising the Test set results

Python

```
from matplotlib.colors import ListedColormap
X_set, y_set = sc.inverse_transform(X_test), y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 10, stop = X_set[:, 0].max() + 10, ste
                     np.arange(start = X_set[:, 1].min() - 1000, stop = X_set[:, 1].max() + 1000,
plt.contourf(X1, X2, classifier.predict(sc.transform(np.array([X1.ravel(), X2.ravel()]).T)).resha
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], c = ListedColormap(('red', 'green'))(
plt.title('Logistic Regression (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

R

```
library(ElemStatLearn)
set = test_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 (Test 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'))
```

3.2 K-Nearest Neighbours (K-NN)

3.2.1 Importing the libraries

Python

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

3.2.2 Importing the dataset

Python

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

R

```
dataset = read.csv('Social_Network_Ads.csv')
# dataset = dataset[3:5]
```

3.2.3 Encoding the target feature as factor

R

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

3.2.4 Splitting the dataset into the Training set and Test set

Python

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
print(X_train)
## [[    44  39000]
##  [    32 120000]
##  [    38  50000]
##  [    32 135000]
##  [    52  21000]
##  [    53 104000]
##  [    39  42000]
##  [    38  61000]
##  [    36  50000]
##  [    36  63000]
##  [    35  25000]
##  [    35  50000]
##  [    42  73000]
##  [    47  49000]
##  [    59  29000]
##  [    49  65000]
##  [    45 131000]
##  [    31  89000]
##  [    46  82000]
##  [    47  51000]
##  [    26  15000]
##  [    60 102000]
##  [    38 112000]
##  [    40 107000]
```

```
##  [    42  53000]
##  [    35  59000]
##  [    48  41000]
##  [    48 134000]
##  [    38 113000]
##  [    29 148000]
##  [    26  15000]
##  [    60  42000]
##  [    24  19000]
##  [    42 149000]
##  [    46  96000]
##  [    28  59000]
##  [    39  96000]
##  [    28  89000]
##  [    41  72000]
##  [    45  26000]
##  [    33  69000]
##  [    20  82000]
##  [    31  74000]
##  [    42  80000]
##  [    35  72000]
##  [    33 149000]
##  [    40  71000]
##  [    51 146000]
##  [    46  79000]
##  [    35  75000]
##  [    38  51000]
##  [    36  75000]
##  [    37  78000]
##  [    38  61000]
##  [    60 108000]
##  [    20  82000]
##  [    57  74000]
##  [    42  65000]
##  [    26  80000]
##  [    46 117000]
##  [    35  61000]
##  [    21  68000]
##  [    28  44000]
##  [    41  87000]
##  [    37  33000]
##  [    27  90000]
##  [    39  42000]
##  [    28 123000]
##  [    31 118000]
```

```
##  [    25  87000]
##  [    35  71000]
##  [    37  70000]
##  [    35  39000]
##  [    47  23000]
##  [    35 147000]
##  [    48 138000]
##  [    26  86000]
##  [    25  79000]
##  [    52 138000]
##  [    51  23000]
##  [    35  60000]
##  [    33 113000]
##  [    30 107000]
##  [    48  33000]
##  [    41  80000]
##  [    48  96000]
##  [    31  18000]
##  [    31  71000]
##  [    43 129000]
##  [    59  76000]
##  [    18  44000]
##  [    36 118000]
##  [    42  90000]
##  [    47  30000]
##  [    26  43000]
##  [    40  78000]
##  [    46  59000]
##  [    59  42000]
##  [    46  74000]
##  [    35  91000]
##  [    28  59000]
##  [    40  57000]
##  [    59 143000]
##  [    57  26000]
##  [    52  38000]
##  [    47 113000]
##  [    53 143000]
##  [    35  27000]
##  [    58 101000]
##  [    45  45000]
##  [    23  82000]
##  [    46  23000]
##  [    42  65000]
##  [    28  84000]
```

```
##  [    38  59000]
##  [    26  84000]
##  [    29  28000]
##  [    37  71000]
##  [    22  55000]
##  [    48  35000]
##  [    49  28000]
##  [    38  65000]
##  [    27  17000]
##  [    46  28000]
##  [    48 141000]
##  [    26  17000]
##  [    35  97000]
##  [    39  59000]
##  [    24  27000]
##  [    32  18000]
##  [    46  88000]
##  [    35  58000]
##  [    56  60000]
##  [    47  34000]
##  [    40  72000]
##  [    32 100000]
##  [    19  21000]
##  [    25  90000]
##  [    35  88000]
##  [    28  32000]
##  [    50  20000]
##  [    40  59000]
##  [    50  44000]
##  [    35  72000]
##  [    40 142000]
##  [    46  32000]
##  [    39  71000]
##  [    20  74000]
##  [    29  75000]
##  [    31  76000]
##  [    47  25000]
##  [    40  61000]
##  [    34 112000]
##  [    38  80000]
##  [    42  75000]
##  [    47  47000]
##  [    39  75000]
##  [    19  25000]
##  [    37  80000]
```

```
##  [    36  60000]
##  [    41  52000]
##  [    36 125000]
##  [    48  29000]
##  [    36 126000]
##  [    51 134000]
##  [    27  57000]
##  [    38  71000]
##  [    39  61000]
##  [    22  27000]
##  [    33  60000]
##  [    48  74000]
##  [    58  23000]
##  [    53  72000]
##  [    32 117000]
##  [    54  70000]
##  [    30  80000]
##  [    58  95000]
##  [    26  52000]
##  [    45  79000]
##  [    24  55000]
##  [    40  75000]
##  [    33  28000]
##  [    44 139000]
##  [    22  18000]
##  [    33  51000]
##  [    43 133000]
##  [    24  32000]
##  [    46  22000]
##  [    35  55000]
##  [    54 104000]
##  [    48 119000]
##  [    35  53000]
##  [    37 144000]
##  [    23  66000]
##  [    37 137000]
##  [    31  58000]
##  [    33  41000]
##  [    45  22000]
##  [    30  15000]
##  [    19  19000]
##  [    49  74000]
##  [    39 122000]
##  [    35  73000]
##  [    39  71000]
```

```
##  [    24  23000]
##  [    41  72000]
##  [    29  83000]
##  [    54  26000]
##  [    35  44000]
##  [    37  75000]
##  [    29  47000]
##  [    31  68000]
##  [    42  54000]
##  [    30 135000]
##  [    52 114000]
##  [    50  36000]
##  [    56 133000]
##  [    29  61000]
##  [    30  89000]
##  [    26  16000]
##  [    33  31000]
##  [    41  72000]
##  [    36  33000]
##  [    55 125000]
##  [    48 131000]
##  [    41  71000]
##  [    30  62000]
##  [    37  72000]
##  [    41  63000]
##  [    58  47000]
##  [    30 116000]
##  [    20  49000]
##  [    37  74000]
##  [    41  59000]
##  [    49  89000]
##  [    28  79000]
##  [    53  82000]
##  [    40  57000]
##  [    60  34000]
##  [    35 108000]
##  [    21  72000]
##  [    38  71000]
##  [    39 106000]
##  [    37  57000]
##  [    26  72000]
##  [    35  23000]
##  [    54 108000]
##  [    30  17000]
##  [    39 134000]
```

```
##  [    29  43000]
##  [    33  43000]
##  [    35  38000]
##  [    41  45000]
##  [    41  72000]
##  [    39 134000]
##  [    27 137000]
##  [    21  16000]
##  [    26  32000]
##  [    31  66000]
##  [    39  73000]
##  [    41  79000]
##  [    47  50000]
##  [    41  30000]
##  [    37  93000]
##  [    60  46000]
##  [    25  22000]
##  [    28  37000]
##  [    38  55000]
##  [    36  54000]
##  [    20  36000]
##  [    56 104000]
##  [    40  57000]
##  [    42 108000]
##  [    20  23000]
##  [    40  65000]
##  [    47  20000]
##  [    18  86000]
##  [    35  79000]
##  [    57  33000]
##  [    34  72000]
##  [    49  39000]
##  [    27  31000]
##  [    19  70000]
##  [    39  79000]
##  [    26  81000]
##  [    25  80000]
##  [    28  85000]
##  [    55  39000]
##  [    50  88000]
##  [    49  88000]
##  [    52 150000]
##  [    35  65000]
##  [    42  54000]
##  [    34  43000]
```

```
##  [    37  52000]
##  [    48  30000]
##  [    29  43000]
##  [    36  52000]
##  [    27  54000]
##  [    26 118000]]
print(y_train)
## [0 1 0 1 1 1 0 0 0 0 0 0 1 1 1 0 1 0 0 1 0 1 0 1 0 0 1 1 1 1 0 1 0 1 0 0 1
##  0 0 1 0 0 0 0 0 1 1 1 1 0 0 0 1 0 1 0 1 0 0 1 0 0 0 1 0 0 0 1 1 0 0 1 0 1
##  1 1 0 0 1 1 0 0 1 1 0 1 0 0 1 1 0 1 1 1 0 0 0 0 0 1 0 0 1 1 1 1 1 0 1 1 0
##  1 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 1 0 0 0 1 0 1 1 0 1 0 0 0 0 1 0 0 0 1 1 0 0
##  0 0 1 0 1 0 0 0 1 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 1 1 1 1 1 0 1 0 0 0 0 0 1 0 0
##  0 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0
##  0 1 1 0 0 0 0 1 0 0 0 0 0 1 0 1 0 1 0 0 0 1 0 0 0 1 0 1 0 0 0 0 0 0 1 1 0 0 0
##  0 0 1 0 1 1 0 0 0 0 0 0 1 0 1 0 0 1 0 0 1 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 1
##  0 0 0 0]
print(X_test)
## [[    30  87000]
##  [    38  50000]
##  [    35  75000]
##  [    30  79000]
##  [    35  50000]
##  [    27  20000]
##  [    31  15000]
##  [    36 144000]
##  [    18  68000]
##  [    47  43000]
##  [    30  49000]
##  [    28  55000]
##  [    37  55000]
##  [    39  77000]
##  [    20  86000]
##  [    32 117000]
##  [    37  77000]
##  [    19  85000]
##  [    55 130000]
##  [    35  22000]
##  [    35  47000]
##  [    47 144000]
##  [    41  51000]
##  [    47 105000]
##  [    23  28000]
##  [    49 141000]
##  [    28  87000]
##  [    29  80000]
```

```
##  [    37  62000]
##  [    32  86000]
##  [    21  88000]
##  [    37  79000]
##  [    57  60000]
##  [    37  53000]
##  [    24  58000]
##  [    18  52000]
##  [    22  81000]
##  [    34  43000]
##  [    31  34000]
##  [    49  36000]
##  [    27  88000]
##  [    41  52000]
##  [    27  84000]
##  [    35  20000]
##  [    43 112000]
##  [    27  58000]
##  [    37  80000]
##  [    52  90000]
##  [    26  30000]
##  [    49  86000]
##  [    57 122000]
##  [    34  25000]
##  [    35  57000]
##  [    34 115000]
##  [    59  88000]
##  [    45  32000]
##  [    29  83000]
##  [    26  80000]
##  [    49  28000]
##  [    23  20000]
##  [    32  18000]
##  [    60  42000]
##  [    19  76000]
##  [    36  99000]
##  [    19  26000]
##  [    60  83000]
##  [    24  89000]
##  [    27  58000]
##  [    40  47000]
##  [    42  70000]
##  [    32 150000]
##  [    35  77000]
##  [    22  63000]
```

```
##  [    45  22000]
##  [    27  89000]
##  [    18  82000]
##  [    42  79000]
##  [    40  60000]
##  [    53  34000]
##  [    47 107000]
##  [    58 144000]
##  [    59  83000]
##  [    24  55000]
##  [    26  35000]
##  [    58  38000]
##  [    42  80000]
##  [    40  75000]
##  [    59 130000]
##  [    46  41000]
##  [    41  60000]
##  [    42  64000]
##  [    37 146000]
##  [    23  48000]
##  [    25  33000]
##  [    24  84000]
##  [    27  96000]
##  [    23  63000]
##  [    48  33000]
##  [    48  90000]
##  [    42 104000]]
print(y_test)
## [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 1 0 0 0 0 0 1 1 0 0 0 0
##  0 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 0 1 1 0 0 1 0 0 1 0 1 0 1 0 0 0 0 1 0 0 1
##  0 0 0 0 1 1 1 0 0 0 1 1 0 1 1 0 0 1 0 0 0 1 0 1 1 1]
```

R

```
# 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)
```

3.2.5 Feature Scaling

Python

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
print(X_train)
## [[ 0.58 -0.89]
##  [-0.61  1.46]
##  [-0.01 -0.57]
##  [-0.61  1.9 ]
##  [ 1.37 -1.41]
##  [ 1.47  1.  ]
##  [ 0.09 -0.8 ]
##  [-0.01 -0.25]
##  [-0.21 -0.57]
##  [-0.21 -0.19]
##  [-0.31 -1.29]
##  [-0.31 -0.57]
##  [ 0.38  0.1 ]
##  [ 0.88 -0.6 ]
##  [ 2.07 -1.18]
##  [ 1.08 -0.13]
##  [ 0.68  1.78]
##  [-0.71  0.56]
##  [ 0.78  0.36]
##  [ 0.88 -0.54]
##  [-1.2  -1.58]
##  [ 2.17  0.94]
##  [-0.01  1.23]
##  [ 0.19  1.08]
##  [ 0.38 -0.48]
##  [-0.31 -0.31]
##  [ 0.98 -0.83]
##  [ 0.98  1.87]
##  [-0.01  1.26]
##  [-0.9   2.27]
##  [-1.2  -1.58]
##  [ 2.17 -0.8 ]
##  [-1.4  -1.47]
##  [ 0.38  2.3 ]
##  [ 0.78  0.77]
##  [-1.   -0.31]
##  [ 0.09  0.77]
##  [-1.    0.56]
##  [ 0.28  0.07]
##  [ 0.68 -1.26]
```

```
##  [-0.51 -0.02]
##  [-1.8   0.36]
##  [-0.71  0.13]
##  [ 0.38  0.3 ]
##  [-0.31  0.07]
##  [-0.51  2.3 ]
##  [ 0.19  0.04]
##  [ 1.27  2.22]
##  [ 0.78  0.27]
##  [-0.31  0.16]
##  [-0.01 -0.54]
##  [-0.21  0.16]
##  [-0.11  0.24]
##  [-0.01 -0.25]
##  [ 2.17  1.11]
##  [-1.8   0.36]
##  [ 1.87  0.13]
##  [ 0.38 -0.13]
##  [-1.2   0.3 ]
##  [ 0.78  1.37]
##  [-0.31 -0.25]
##  [-1.7  -0.05]
##  [-1.   -0.74]
##  [ 0.28  0.5 ]
##  [-0.11 -1.06]
##  [-1.1   0.59]
##  [ 0.09 -0.8 ]
##  [-1.    1.55]
##  [-0.71  1.4 ]
##  [-1.3   0.5 ]
##  [-0.31  0.04]
##  [-0.11  0.01]
##  [-0.31 -0.89]
##  [ 0.88 -1.35]
##  [-0.31  2.24]
##  [ 0.98  1.98]
##  [-1.2   0.48]
##  [-1.3   0.27]
##  [ 1.37  1.98]
##  [ 1.27 -1.35]
##  [-0.31 -0.28]
##  [-0.51  1.26]
##  [-0.8   1.08]
##  [ 0.98 -1.06]
##  [ 0.28  0.3 ]
```

```
##  [ 0.98  0.77]
##  [-0.71 -1.5 ]
##  [-0.71  0.04]
##  [ 0.48  1.72]
##  [ 2.07  0.19]
##  [-1.99 -0.74]
##  [-0.21  1.4 ]
##  [ 0.38  0.59]
##  [ 0.88 -1.15]
##  [-1.2  -0.77]
##  [ 0.19  0.24]
##  [ 0.78 -0.31]
##  [ 2.07 -0.8 ]
##  [ 0.78  0.13]
##  [-0.31  0.62]
##  [-1.   -0.31]
##  [ 0.19 -0.36]
##  [ 2.07  2.13]
##  [ 1.87 -1.26]
##  [ 1.37 -0.92]
##  [ 0.88  1.26]
##  [ 1.47  2.13]
##  [-0.31 -1.23]
##  [ 1.97  0.91]
##  [ 0.68 -0.71]
##  [-1.5   0.36]
##  [ 0.78 -1.35]
##  [ 0.38 -0.13]
##  [-1.    0.42]
##  [-0.01 -0.31]
##  [-1.2   0.42]
##  [-0.9  -1.21]
##  [-0.11  0.04]
##  [-1.6  -0.42]
##  [ 0.98 -1.  ]
##  [ 1.08 -1.21]
##  [-0.01 -0.13]
##  [-1.1  -1.52]
##  [ 0.78 -1.21]
##  [ 0.98  2.07]
##  [-1.2  -1.52]
##  [-0.31  0.79]
##  [ 0.09 -0.31]
##  [-1.4  -1.23]
##  [-0.61 -1.5 ]
```

```
##  [ 0.78  0.53]
##  [-0.31 -0.34]
##  [ 1.77 -0.28]
##  [ 0.88 -1.03]
##  [ 0.19  0.07]
##  [-0.61  0.88]
##  [-1.89 -1.41]
##  [-1.3   0.59]
##  [-0.31  0.53]
##  [-1.   -1.09]
##  [ 1.18 -1.44]
##  [ 0.19 -0.31]
##  [ 1.18 -0.74]
##  [-0.31  0.07]
##  [ 0.19  2.1 ]
##  [ 0.78 -1.09]
##  [ 0.09  0.04]
##  [-1.8   0.13]
##  [-0.9   0.16]
##  [-0.71  0.19]
##  [ 0.88 -1.29]
##  [ 0.19 -0.25]
##  [-0.41  1.23]
##  [-0.01  0.3 ]
##  [ 0.38  0.16]
##  [ 0.88 -0.65]
##  [ 0.09  0.16]
##  [-1.89 -1.29]
##  [-0.11  0.3 ]
##  [-0.21 -0.28]
##  [ 0.28 -0.51]
##  [-0.21  1.61]
##  [ 0.98 -1.18]
##  [-0.21  1.64]
##  [ 1.27  1.87]
##  [-1.1  -0.36]
##  [-0.01  0.04]
##  [ 0.09 -0.25]
##  [-1.6  -1.23]
##  [-0.51 -0.28]
##  [ 0.98  0.13]
##  [ 1.97 -1.35]
##  [ 1.47  0.07]
##  [-0.61  1.37]
##  [ 1.57  0.01]
```

```
##  [-0.8   0.3 ]
##  [ 1.97  0.74]
##  [-1.2  -0.51]
##  [ 0.68  0.27]
##  [-1.4  -0.42]
##  [ 0.19  0.16]
##  [-0.51 -1.21]
##  [ 0.58  2.01]
##  [-1.6  -1.5 ]
##  [-0.51 -0.54]
##  [ 0.48  1.84]
##  [-1.4  -1.09]
##  [ 0.78 -1.38]
##  [-0.31 -0.42]
##  [ 1.57  1.  ]
##  [ 0.98  1.43]
##  [-0.31 -0.48]
##  [-0.11  2.16]
##  [-1.5  -0.1 ]
##  [-0.11  1.95]
##  [-0.71 -0.34]
##  [-0.51 -0.83]
##  [ 0.68 -1.38]
##  [-0.8  -1.58]
##  [-1.89 -1.47]
##  [ 1.08  0.13]
##  [ 0.09  1.52]
##  [-0.31  0.1 ]
##  [ 0.09  0.04]
##  [-1.4  -1.35]
##  [ 0.28  0.07]
##  [-0.9   0.39]
##  [ 1.57 -1.26]
##  [-0.31 -0.74]
##  [-0.11  0.16]
##  [-0.9  -0.65]
##  [-0.71 -0.05]
##  [ 0.38 -0.45]
##  [-0.8   1.9 ]
##  [ 1.37  1.29]
##  [ 1.18 -0.97]
##  [ 1.77  1.84]
##  [-0.9  -0.25]
##  [-0.8   0.56]
##  [-1.2  -1.55]
```

```
##  [-0.51 -1.12]
##  [ 0.28  0.07]
##  [-0.21 -1.06]
##  [ 1.67  1.61]
##  [ 0.98  1.78]
##  [ 0.28  0.04]
##  [-0.8  -0.22]
##  [-0.11  0.07]
##  [ 0.28 -0.19]
##  [ 1.97 -0.65]
##  [-0.8   1.35]
##  [-1.8  -0.6 ]
##  [-0.11  0.13]
##  [ 0.28 -0.31]
##  [ 1.08  0.56]
##  [-1.    0.27]
##  [ 1.47  0.36]
##  [ 0.19 -0.36]
##  [ 2.17 -1.03]
##  [-0.31  1.11]
##  [-1.7   0.07]
##  [-0.01  0.04]
##  [ 0.09  1.06]
##  [-0.11 -0.36]
##  [-1.2   0.07]
##  [-0.31 -1.35]
##  [ 1.57  1.11]
##  [-0.8  -1.52]
##  [ 0.09  1.87]
##  [-0.9  -0.77]
##  [-0.51 -0.77]
##  [-0.31 -0.92]
##  [ 0.28 -0.71]
##  [ 0.28  0.07]
##  [ 0.09  1.87]
##  [-1.1   1.95]
##  [-1.7  -1.55]
##  [-1.2  -1.09]
##  [-0.71 -0.1 ]
##  [ 0.09  0.1 ]
##  [ 0.28  0.27]
##  [ 0.88 -0.57]
##  [ 0.28 -1.15]
##  [-0.11  0.68]
##  [ 2.17 -0.68]
```

```
##  [-1.3  -1.38]
##  [-1.   -0.94]
##  [-0.01 -0.42]
##  [-0.21 -0.45]
##  [-1.8  -0.97]
##  [ 1.77  1.  ]
##  [ 0.19 -0.36]
##  [ 0.38  1.11]
##  [-1.8  -1.35]
##  [ 0.19 -0.13]
##  [ 0.88 -1.44]
##  [-1.99  0.48]
##  [-0.31  0.27]
##  [ 1.87 -1.06]
##  [-0.41  0.07]
##  [ 1.08 -0.89]
##  [-1.1  -1.12]
##  [-1.89  0.01]
##  [ 0.09  0.27]
##  [-1.2   0.33]
##  [-1.3   0.3 ]
##  [-1.    0.45]
##  [ 1.67 -0.89]
##  [ 1.18  0.53]
##  [ 1.08  0.53]
##  [ 1.37  2.33]
##  [-0.31 -0.13]
##  [ 0.38 -0.45]
##  [-0.41 -0.77]
##  [-0.11 -0.51]
##  [ 0.98 -1.15]
##  [-0.9  -0.77]
##  [-0.21 -0.51]
##  [-1.1  -0.45]
##  [-1.2   1.4 ]]
print(X_test)
## [[-0.8   0.5 ]
##  [-0.01 -0.57]
##  [-0.31  0.16]
##  [-0.8   0.27]
##  [-0.31 -0.57]
##  [-1.1  -1.44]
##  [-0.71 -1.58]
##  [-0.21  2.16]
##  [-1.99 -0.05]
```

```
##  [ 0.88 -0.77]
##  [-0.8  -0.6 ]
##  [-1.   -0.42]
##  [-0.11 -0.42]
##  [ 0.09  0.22]
##  [-1.8   0.48]
##  [-0.61  1.37]
##  [-0.11  0.22]
##  [-1.89  0.45]
##  [ 1.67  1.75]
##  [-0.31 -1.38]
##  [-0.31 -0.65]
##  [ 0.88  2.16]
##  [ 0.28 -0.54]
##  [ 0.88  1.03]
##  [-1.5  -1.21]
##  [ 1.08  2.07]
##  [-1.    0.5 ]
##  [-0.9   0.3 ]
##  [-0.11 -0.22]
##  [-0.61  0.48]
##  [-1.7   0.53]
##  [-0.11  0.27]
##  [ 1.87 -0.28]
##  [-0.11 -0.48]
##  [-1.4  -0.34]
##  [-1.99 -0.51]
##  [-1.6   0.33]
##  [-0.41 -0.77]
##  [-0.71 -1.03]
##  [ 1.08 -0.97]
##  [-1.1   0.53]
##  [ 0.28 -0.51]
##  [-1.1   0.42]
##  [-0.31 -1.44]
##  [ 0.48  1.23]
##  [-1.1  -0.34]
##  [-0.11  0.3 ]
##  [ 1.37  0.59]
##  [-1.2  -1.15]
##  [ 1.08  0.48]
##  [ 1.87  1.52]
##  [-0.41 -1.29]
##  [-0.31 -0.36]
##  [-0.41  1.32]
```

```
##  [ 2.07  0.53]
##  [ 0.68 -1.09]
##  [-0.9   0.39]
##  [-1.2   0.3 ]
##  [ 1.08 -1.21]
##  [-1.5  -1.44]
##  [-0.61 -1.5 ]
##  [ 2.17 -0.8 ]
##  [-1.89  0.19]
##  [-0.21  0.85]
##  [-1.89 -1.26]
##  [ 2.17  0.39]
##  [-1.4   0.56]
##  [-1.1  -0.34]
##  [ 0.19 -0.65]
##  [ 0.38  0.01]
##  [-0.61  2.33]
##  [-0.31  0.22]
##  [-1.6  -0.19]
##  [ 0.68 -1.38]
##  [-1.1   0.56]
##  [-1.99  0.36]
##  [ 0.38  0.27]
##  [ 0.19 -0.28]
##  [ 1.47 -1.03]
##  [ 0.88  1.08]
##  [ 1.97  2.16]
##  [ 2.07  0.39]
##  [-1.4  -0.42]
##  [-1.2  -1.  ]
##  [ 1.97 -0.92]
##  [ 0.38  0.3 ]
##  [ 0.19  0.16]
##  [ 2.07  1.75]
##  [ 0.78 -0.83]
##  [ 0.28 -0.28]
##  [ 0.38 -0.16]
##  [-0.11  2.22]
##  [-1.5  -0.63]
##  [-1.3  -1.06]
##  [-1.4   0.42]
##  [-1.1   0.77]
##  [-1.5  -0.19]
##  [ 0.98 -1.06]
##  [ 0.98  0.59]
```

```
##  [ 0.38  1.  ]]
```

R

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

3.2.6 Training the K-NN model on the Training set

Python

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
classifier.fit(X_train, y_train)
## KNeighborsClassifier()
```

R (Fitting K-NN to the Training set and Predicting the Test set results)

```
library(class)
y_pred = knn(train = training_set[, -3],
             test = test_set[, -3],
             cl = training_set[, 3],
             k = 5,
             prob = TRUE)
```

3.2.7 Predicting a new result

Python

```
print(classifier.predict(sc.transform([[30,87000]])))
## [0]
```

3.2.8 Predicting the Test set results

Python

```
y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))
## [[0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [1 0]
```

```
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [1 0]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [1 1]
##  [0 0]
##  [1 1]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 1]
##  [1 1]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [1 1]
##  [1 1]
##  [0 0]
##  [0 0]
##  [1 0]
##  [1 1]
```

```
##  [1 1]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [1 1]
##  [0 0]
##  [1 1]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [1 1]
##  [1 1]
##  [1 1]
##  [1 0]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 1]
##  [0 0]
##  [1 1]
##  [1 1]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 1]
##  [0 0]
##  [1 1]
##  [1 1]
##  [1 1]]
```

3.2.9 Making the Confusion Matrix

Python

```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
## [[64  4]
##  [ 3 29]]
accuracy_score(y_test, y_pred)
## 0.93
```

R

```
cm = table(test_set[, 3], y_pred)
```

3.2.10 Visualising the Training set results

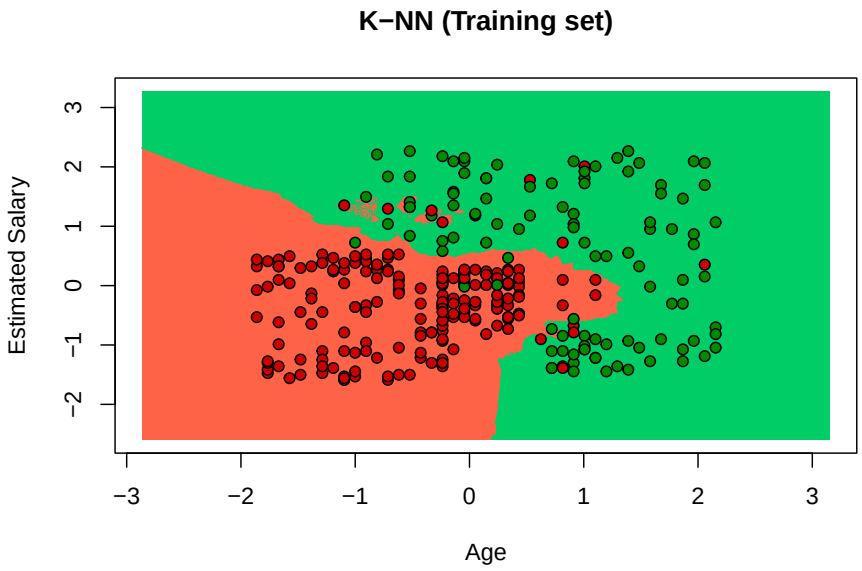
Python

```
from matplotlib.colors import ListedColormap
X_set, y_set = sc.inverse_transform(X_train), y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 10, stop = X_set[:, 0].max() + 10, ste
                     np.arange(start = X_set[:, 1].min() - 1000, stop = X_set[:, 1].max() + 1000,
plt.contourf(X1, X2, classifier.predict(sc.transform(np.array([X1.ravel(), X2.ravel()]).T)).resha
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], c = ListedColormap(('red', 'green'))(
plt.title('K-NN (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

R

```
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')
y_grid = knn(train = training_set[, -3], test = grid_set, cl = training_set[, 3], k = 5)
plot(set[, -3],
     main = 'K-NN (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'))
```



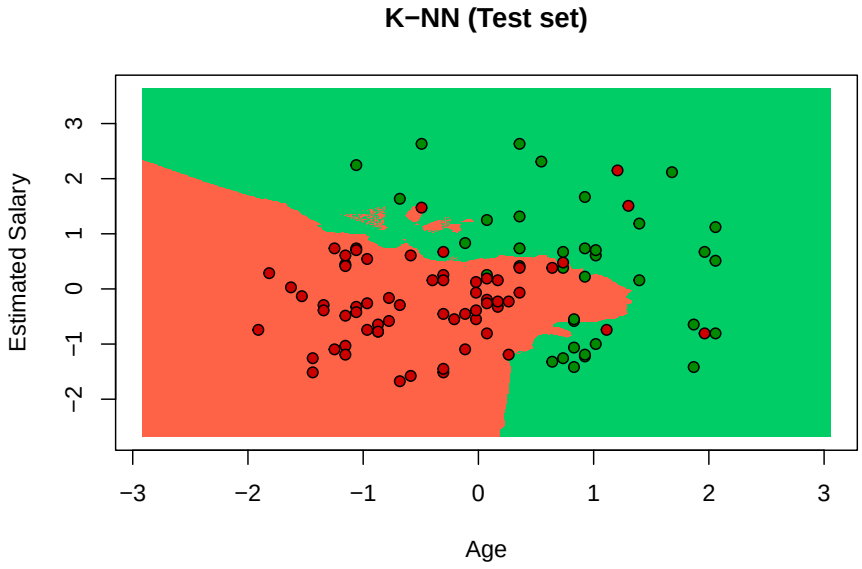
3.2.11 Visualising the Test set results

Python

```
from matplotlib.colors import ListedColormap
X_set, y_set = sc.inverse_transform(X_test), y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 10, stop = X_set[:, 0].max()
                     np.arange(start = X_set[:, 1].min() - 1000, stop = X_set[:, 1].ma
plt.contourf(X1, X2, classifier.predict(sc.transform(np.array([X1.ravel(), X2.ravel()]
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], c = ListedColormap(('red',
plt.title('K-NN (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

R

```
library(ElemStatLearn)
set = test_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')
y_grid = knn(train = training_set[, -3], test = grid_set, cl = training_set[, 3], k = 5)
plot(set[, -3],
     main = 'K-NN (Test 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'))
```



3.3 Support Vector Machine (SVM)

3.3.1 Importing the libraries

Python

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

3.3.2 Importing the dataset

Python

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

R

```
dataset = read.csv('Social_Network_Ads.csv')
# dataset = dataset[3:5]
```

3.3.3 Encoding the target feature as factor

R

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

3.3.4 Splitting the dataset into the Training set and Test set

Python

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_sta
print(X_train)
## [[    44  39000]
##  [    32 120000]
##  [    38  50000]
##  [    32 135000]
##  [    52  21000]
##  [    53 104000]
##  [    39  42000]
##  [    38  61000]
##  [    36  50000]
##  [    36  63000]
##  [    35  25000]
##  [    35  50000]
##  [    42  73000]
##  [    47  49000]
##  [    59  29000]
##  [    49  65000]
##  [    45 131000]
##  [    31  89000]
##  [    46  82000]
##  [    47  51000]
##  [    26  15000]
```

```
##  [    60 102000]
##  [    38 112000]
##  [    40 107000]
##  [    42  53000]
##  [    35  59000]
##  [    48  41000]
##  [    48 134000]
##  [    38 113000]
##  [    29 148000]
##  [    26  15000]
##  [    60  42000]
##  [    24  19000]
##  [    42 149000]
##  [    46  96000]
##  [    28  59000]
##  [    39  96000]
##  [    28  89000]
##  [    41  72000]
##  [    45  26000]
##  [    33  69000]
##  [    20  82000]
##  [    31  74000]
##  [    42  80000]
##  [    35  72000]
##  [    33 149000]
##  [    40  71000]
##  [    51 146000]
##  [    46  79000]
##  [    35  75000]
##  [    38  51000]
##  [    36  75000]
##  [    37  78000]
##  [    38  61000]
##  [    60 108000]
##  [    20  82000]
##  [    57  74000]
##  [    42  65000]
##  [    26  80000]
##  [    46 117000]
##  [    35  61000]
##  [    21  68000]
##  [    28  44000]
##  [    41  87000]
##  [    37  33000]
##  [    27  90000]
```

```
##  [    39  42000]
##  [    28 123000]
##  [    31 118000]
##  [    25  87000]
##  [    35  71000]
##  [    37  70000]
##  [    35  39000]
##  [    47  23000]
##  [    35 147000]
##  [    48 138000]
##  [    26  86000]
##  [    25  79000]
##  [    52 138000]
##  [    51  23000]
##  [    35  60000]
##  [    33 113000]
##  [    30 107000]
##  [    48  33000]
##  [    41  80000]
##  [    48  96000]
##  [    31  18000]
##  [    31  71000]
##  [    43 129000]
##  [    59  76000]
##  [    18  44000]
##  [    36 118000]
##  [    42  90000]
##  [    47  30000]
##  [    26  43000]
##  [    40  78000]
##  [    46  59000]
##  [    59  42000]
##  [    46  74000]
##  [    35  91000]
##  [    28  59000]
##  [    40  57000]
##  [    59 143000]
##  [    57  26000]
##  [    52  38000]
##  [    47 113000]
##  [    53 143000]
##  [    35  27000]
##  [    58 101000]
##  [    45  45000]
##  [    23  82000]
```

```
##  [    46  23000]
##  [    42  65000]
##  [    28  84000]
##  [    38  59000]
##  [    26  84000]
##  [    29  28000]
##  [    37  71000]
##  [    22  55000]
##  [    48  35000]
##  [    49  28000]
##  [    38  65000]
##  [    27  17000]
##  [    46  28000]
##  [    48 141000]
##  [    26  17000]
##  [    35  97000]
##  [    39  59000]
##  [    24  27000]
##  [    32  18000]
##  [    46  88000]
##  [    35  58000]
##  [    56  60000]
##  [    47  34000]
##  [    40  72000]
##  [    32 100000]
##  [    19  21000]
##  [    25  90000]
##  [    35  88000]
##  [    28  32000]
##  [    50  20000]
##  [    40  59000]
##  [    50  44000]
##  [    35  72000]
##  [    40 142000]
##  [    46  32000]
##  [    39  71000]
##  [    20  74000]
##  [    29  75000]
##  [    31  76000]
##  [    47  25000]
##  [    40  61000]
##  [    34 112000]
##  [    38  80000]
##  [    42  75000]
##  [    47  47000]
```

```
##  [    39  75000]
##  [    19  25000]
##  [    37  80000]
##  [    36  60000]
##  [    41  52000]
##  [    36 125000]
##  [    48  29000]
##  [    36 126000]
##  [    51 134000]
##  [    27  57000]
##  [    38  71000]
##  [    39  61000]
##  [    22  27000]
##  [    33  60000]
##  [    48  74000]
##  [    58  23000]
##  [    53  72000]
##  [    32 117000]
##  [    54  70000]
##  [    30  80000]
##  [    58  95000]
##  [    26  52000]
##  [    45  79000]
##  [    24  55000]
##  [    40  75000]
##  [    33  28000]
##  [    44 139000]
##  [    22  18000]
##  [    33  51000]
##  [    43 133000]
##  [    24  32000]
##  [    46  22000]
##  [    35  55000]
##  [    54 104000]
##  [    48 119000]
##  [    35  53000]
##  [    37 144000]
##  [    23  66000]
##  [    37 137000]
##  [    31  58000]
##  [    33  41000]
##  [    45  22000]
##  [    30  15000]
##  [    19  19000]
##  [    49  74000]
```

```
##  [    39 122000]
##  [    35  73000]
##  [    39  71000]
##  [    24  23000]
##  [    41  72000]
##  [    29  83000]
##  [    54  26000]
##  [    35  44000]
##  [    37  75000]
##  [    29  47000]
##  [    31  68000]
##  [    42  54000]
##  [    30 135000]
##  [    52 114000]
##  [    50  36000]
##  [    56 133000]
##  [    29  61000]
##  [    30  89000]
##  [    26  16000]
##  [    33  31000]
##  [    41  72000]
##  [    36  33000]
##  [    55 125000]
##  [    48 131000]
##  [    41  71000]
##  [    30  62000]
##  [    37  72000]
##  [    41  63000]
##  [    58  47000]
##  [    30 116000]
##  [    20  49000]
##  [    37  74000]
##  [    41  59000]
##  [    49  89000]
##  [    28  79000]
##  [    53  82000]
##  [    40  57000]
##  [    60  34000]
##  [    35 108000]
##  [    21  72000]
##  [    38  71000]
##  [    39 106000]
##  [    37  57000]
##  [    26  72000]
##  [    35  23000]
```

```
##  [    54 108000]
##  [    30  17000]
##  [    39 134000]
##  [    29  43000]
##  [    33  43000]
##  [    35  38000]
##  [    41  45000]
##  [    41  72000]
##  [    39 134000]
##  [    27 137000]
##  [    21  16000]
##  [    26  32000]
##  [    31  66000]
##  [    39  73000]
##  [    41  79000]
##  [    47  50000]
##  [    41  30000]
##  [    37  93000]
##  [    60  46000]
##  [    25  22000]
##  [    28  37000]
##  [    38  55000]
##  [    36  54000]
##  [    20  36000]
##  [    56 104000]
##  [    40  57000]
##  [    42 108000]
##  [    20  23000]
##  [    40  65000]
##  [    47  20000]
##  [    18  86000]
##  [    35  79000]
##  [    57  33000]
##  [    34  72000]
##  [    49  39000]
##  [    27  31000]
##  [    19  70000]
##  [    39  79000]
##  [    26  81000]
##  [    25  80000]
##  [    28  85000]
##  [    55  39000]
##  [    50  88000]
##  [    49  88000]
##  [    52 150000]
```

```
##  [    35  65000]
##  [    42  54000]
##  [    34  43000]
##  [    37  52000]
##  [    48  30000]
##  [    29  43000]
##  [    36  52000]
##  [    27  54000]
##  [    26 118000]]
print(y_train)
## [0 1 0 1 1 1 0 0 0 0 0 0 1 1 1 0 1 0 0 1 0 1 0 1 0 0 1 1 1 1 0 1 0 1 0 0 1
##  0 0 1 0 0 0 0 0 1 1 1 1 0 0 0 1 0 1 0 1 0 0 1 0 0 0 1 0 0 0 1 1 0 0 1 0 1
##  1 1 0 0 1 1 0 0 1 1 0 1 0 0 1 1 0 1 1 1 0 0 0 0 0 1 0 0 1 1 1 1 1 0 1 1 0
##  1 0 0 0 0 0 0 0 1 1 0 0 1 0 0 1 0 0 0 1 0 1 1 0 1 0 0 0 0 1 0 0 0 1 1 0 0
##  0 0 1 0 1 0 0 0 1 0 0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 1 1 0 1 0 0 0 0 0 1 0 0
##  0 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 1 1 0 0 0 0 0
##  0 1 1 0 0 0 0 1 0 0 0 0 1 0 1 0 1 0 0 0 1 0 0 0 1 0 1 0 0 0 0 0 1 1 0 0 0
##  0 0 1 0 1 1 0 0 0 0 0 1 0 1 0 0 1 0 0 1 0 1 0 0 0 0 0 0 1 1 1 1 0 0 0 0 1
##  0 0 0 0]
print(X_test)
## [[    30  87000]
##  [    38  50000]
##  [    35  75000]
##  [    30  79000]
##  [    35  50000]
##  [    27  20000]
##  [    31  15000]
##  [    36 144000]
##  [    18  68000]
##  [    47  43000]
##  [    30  49000]
##  [    28  55000]
##  [    37  55000]
##  [    39  77000]
##  [    20  86000]
##  [    32 117000]
##  [    37  77000]
##  [    19  85000]
##  [    55 130000]
##  [    35  22000]
##  [    35  47000]
##  [    47 144000]
##  [    41  51000]
##  [    47 105000]
##  [    23  28000]
```

```
##  [    49 141000]
##  [    28  87000]
##  [    29  80000]
##  [    37  62000]
##  [    32  86000]
##  [    21  88000]
##  [    37  79000]
##  [    57  60000]
##  [    37  53000]
##  [    24  58000]
##  [    18  52000]
##  [    22  81000]
##  [    34  43000]
##  [    31  34000]
##  [    49  36000]
##  [    27  88000]
##  [    41  52000]
##  [    27  84000]
##  [    35  20000]
##  [    43 112000]
##  [    27  58000]
##  [    37  80000]
##  [    52  90000]
##  [    26  30000]
##  [    49  86000]
##  [    57 122000]
##  [    34  25000]
##  [    35  57000]
##  [    34 115000]
##  [    59  88000]
##  [    45  32000]
##  [    29  83000]
##  [    26  80000]
##  [    49  28000]
##  [    23  20000]
##  [    32  18000]
##  [    60  42000]
##  [    19  76000]
##  [    36  99000]
##  [    19  26000]
##  [    60  83000]
##  [    24  89000]
##  [    27  58000]
##  [    40  47000]
##  [    42  70000]
```

```
##  [    32 150000]
##  [    35  77000]
##  [    22  63000]
##  [    45  22000]
##  [    27  89000]
##  [    18  82000]
##  [    42  79000]
##  [    40  60000]
##  [    53  34000]
##  [    47 107000]
##  [    58 144000]
##  [    59  83000]
##  [    24  55000]
##  [    26  35000]
##  [    58  38000]
##  [    42  80000]
##  [    40  75000]
##  [    59 130000]
##  [    46  41000]
##  [    41  60000]
##  [    42  64000]
##  [    37 146000]
##  [    23  48000]
##  [    25  33000]
##  [    24  84000]
##  [    27  96000]
##  [    23  63000]
##  [    48  33000]
##  [    48  90000]
##  [    42 104000]]
print(y_test)
## [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 1 0 0 0 0 0 1 1 0 0 0 0
##  0 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 0 1 1 0 0 1 0 0 1 0 1 0 1 0 0 0 0 1 0 0 1
##  0 0 0 0 1 1 1 0 0 0 1 1 0 1 1 0 0 1 0 0 0 1 0 1 1 1]
```

R

```
# 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)
```

3.3.5 Feature Scaling

Python

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
print(X_train)
## [[ 0.58 -0.89]
##  [-0.61  1.46]
##  [-0.01 -0.57]
##  [-0.61  1.9 ]
##  [ 1.37 -1.41]
##  [ 1.47  1.  ]
##  [ 0.09 -0.8 ]
##  [-0.01 -0.25]
##  [-0.21 -0.57]
##  [-0.21 -0.19]
##  [-0.31 -1.29]
##  [-0.31 -0.57]
##  [ 0.38  0.1 ]
##  [ 0.88 -0.6 ]
##  [ 2.07 -1.18]
##  [ 1.08 -0.13]
##  [ 0.68  1.78]
##  [-0.71  0.56]
##  [ 0.78  0.36]
##  [ 0.88 -0.54]
##  [-1.2  -1.58]
##  [ 2.17  0.94]
##  [-0.01  1.23]
##  [ 0.19  1.08]
##  [ 0.38 -0.48]
##  [-0.31 -0.31]
##  [ 0.98 -0.83]
##  [ 0.98  1.87]
##  [-0.01  1.26]
##  [-0.9   2.27]
##  [-1.2  -1.58]
##  [ 2.17 -0.8 ]
##  [-1.4  -1.47]
##  [ 0.38  2.3 ]
##  [ 0.78  0.77]
##  [-1.   -0.31]
##  [ 0.09  0.77]
```

```
##  [-1.    0.56]
##  [ 0.28  0.07]
##  [ 0.68 -1.26]
##  [-0.51 -0.02]
##  [-1.8   0.36]
##  [-0.71  0.13]
##  [ 0.38  0.3 ]
##  [-0.31  0.07]
##  [-0.51  2.3 ]
##  [ 0.19  0.04]
##  [ 1.27  2.22]
##  [ 0.78  0.27]
##  [-0.31  0.16]
##  [-0.01 -0.54]
##  [-0.21  0.16]
##  [-0.11  0.24]
##  [-0.01 -0.25]
##  [ 2.17  1.11]
##  [-1.8   0.36]
##  [ 1.87  0.13]
##  [ 0.38 -0.13]
##  [-1.2   0.3 ]
##  [ 0.78  1.37]
##  [-0.31 -0.25]
##  [-1.7  -0.05]
##  [-1.   -0.74]
##  [ 0.28  0.5 ]
##  [-0.11 -1.06]
##  [-1.1   0.59]
##  [ 0.09 -0.8 ]
##  [-1.    1.55]
##  [-0.71  1.4 ]
##  [-1.3   0.5 ]
##  [-0.31  0.04]
##  [-0.11  0.01]
##  [-0.31 -0.89]
##  [ 0.88 -1.35]
##  [-0.31  2.24]
##  [ 0.98  1.98]
##  [-1.2   0.48]
##  [-1.3   0.27]
##  [ 1.37  1.98]
##  [ 1.27 -1.35]
##  [-0.31 -0.28]
##  [-0.51  1.26]
```

```
##  [-0.8   1.08]
##  [ 0.98 -1.06]
##  [ 0.28  0.3 ]
##  [ 0.98  0.77]
##  [-0.71 -1.5 ]
##  [-0.71  0.04]
##  [ 0.48  1.72]
##  [ 2.07  0.19]
##  [-1.99 -0.74]
##  [-0.21  1.4 ]
##  [ 0.38  0.59]
##  [ 0.88 -1.15]
##  [-1.2  -0.77]
##  [ 0.19  0.24]
##  [ 0.78 -0.31]
##  [ 2.07 -0.8 ]
##  [ 0.78  0.13]
##  [-0.31  0.62]
##  [-1.   -0.31]
##  [ 0.19 -0.36]
##  [ 2.07  2.13]
##  [ 1.87 -1.26]
##  [ 1.37 -0.92]
##  [ 0.88  1.26]
##  [ 1.47  2.13]
##  [-0.31 -1.23]
##  [ 1.97  0.91]
##  [ 0.68 -0.71]
##  [-1.5   0.36]
##  [ 0.78 -1.35]
##  [ 0.38 -0.13]
##  [-1.    0.42]
##  [-0.01 -0.31]
##  [-1.2   0.42]
##  [-0.9  -1.21]
##  [-0.11  0.04]
##  [-1.6  -0.42]
##  [ 0.98 -1.  ]
##  [ 1.08 -1.21]
##  [-0.01 -0.13]
##  [-1.1  -1.52]
##  [ 0.78 -1.21]
##  [ 0.98  2.07]
##  [-1.2  -1.52]
##  [-0.31  0.79]
```

```
##  [ 0.09 -0.31]
##  [-1.4  -1.23]
##  [-0.61 -1.5 ]
##  [ 0.78  0.53]
##  [-0.31 -0.34]
##  [ 1.77 -0.28]
##  [ 0.88 -1.03]
##  [ 0.19  0.07]
##  [-0.61  0.88]
##  [-1.89 -1.41]
##  [-1.3   0.59]
##  [-0.31  0.53]
##  [-1.   -1.09]
##  [ 1.18 -1.44]
##  [ 0.19 -0.31]
##  [ 1.18 -0.74]
##  [-0.31  0.07]
##  [ 0.19  2.1 ]
##  [ 0.78 -1.09]
##  [ 0.09  0.04]
##  [-1.8   0.13]
##  [-0.9   0.16]
##  [-0.71  0.19]
##  [ 0.88 -1.29]
##  [ 0.19 -0.25]
##  [-0.41  1.23]
##  [-0.01  0.3 ]
##  [ 0.38  0.16]
##  [ 0.88 -0.65]
##  [ 0.09  0.16]
##  [-1.89 -1.29]
##  [-0.11  0.3 ]
##  [-0.21 -0.28]
##  [ 0.28 -0.51]
##  [-0.21  1.61]
##  [ 0.98 -1.18]
##  [-0.21  1.64]
##  [ 1.27  1.87]
##  [-1.1  -0.36]
##  [-0.01  0.04]
##  [ 0.09 -0.25]
##  [-1.6  -1.23]
##  [-0.51 -0.28]
##  [ 0.98  0.13]
##  [ 1.97 -1.35]
```

```
##  [ 1.47  0.07]
##  [-0.61  1.37]
##  [ 1.57  0.01]
##  [-0.8   0.3 ]
##  [ 1.97  0.74]
##  [-1.2  -0.51]
##  [ 0.68  0.27]
##  [-1.4  -0.42]
##  [ 0.19  0.16]
##  [-0.51 -1.21]
##  [ 0.58  2.01]
##  [-1.6  -1.5 ]
##  [-0.51 -0.54]
##  [ 0.48  1.84]
##  [-1.4  -1.09]
##  [ 0.78 -1.38]
##  [-0.31 -0.42]
##  [ 1.57  1.  ]
##  [ 0.98  1.43]
##  [-0.31 -0.48]
##  [-0.11  2.16]
##  [-1.5  -0.1 ]
##  [-0.11  1.95]
##  [-0.71 -0.34]
##  [-0.51 -0.83]
##  [ 0.68 -1.38]
##  [-0.8  -1.58]
##  [-1.89 -1.47]
##  [ 1.08  0.13]
##  [ 0.09  1.52]
##  [-0.31  0.1 ]
##  [ 0.09  0.04]
##  [-1.4  -1.35]
##  [ 0.28  0.07]
##  [-0.9   0.39]
##  [ 1.57 -1.26]
##  [-0.31 -0.74]
##  [-0.11  0.16]
##  [-0.9  -0.65]
##  [-0.71 -0.05]
##  [ 0.38 -0.45]
##  [-0.8   1.9 ]
##  [ 1.37  1.29]
##  [ 1.18 -0.97]
##  [ 1.77  1.84]
```

```
##  [-0.9  -0.25]
##  [-0.8   0.56]
##  [-1.2  -1.55]
##  [-0.51 -1.12]
##  [ 0.28  0.07]
##  [-0.21 -1.06]
##  [ 1.67  1.61]
##  [ 0.98  1.78]
##  [ 0.28  0.04]
##  [-0.8  -0.22]
##  [-0.11  0.07]
##  [ 0.28 -0.19]
##  [ 1.97 -0.65]
##  [-0.8   1.35]
##  [-1.8  -0.6 ]
##  [-0.11  0.13]
##  [ 0.28 -0.31]
##  [ 1.08  0.56]
##  [-1.    0.27]
##  [ 1.47  0.36]
##  [ 0.19 -0.36]
##  [ 2.17 -1.03]
##  [-0.31  1.11]
##  [-1.7   0.07]
##  [-0.01  0.04]
##  [ 0.09  1.06]
##  [-0.11 -0.36]
##  [-1.2   0.07]
##  [-0.31 -1.35]
##  [ 1.57  1.11]
##  [-0.8  -1.52]
##  [ 0.09  1.87]
##  [-0.9  -0.77]
##  [-0.51 -0.77]
##  [-0.31 -0.92]
##  [ 0.28 -0.71]
##  [ 0.28  0.07]
##  [ 0.09  1.87]
##  [-1.1   1.95]
##  [-1.7  -1.55]
##  [-1.2  -1.09]
##  [-0.71 -0.1 ]
##  [ 0.09  0.1 ]
##  [ 0.28  0.27]
##  [ 0.88 -0.57]
```

```
##  [ 0.28 -1.15]
##  [-0.11  0.68]
##  [ 2.17 -0.68]
##  [-1.3  -1.38]
##  [-1.   -0.94]
##  [-0.01 -0.42]
##  [-0.21 -0.45]
##  [-1.8  -0.97]
##  [ 1.77  1.  ]
##  [ 0.19 -0.36]
##  [ 0.38  1.11]
##  [-1.8  -1.35]
##  [ 0.19 -0.13]
##  [ 0.88 -1.44]
##  [-1.99  0.48]
##  [-0.31  0.27]
##  [ 1.87 -1.06]
##  [-0.41  0.07]
##  [ 1.08 -0.89]
##  [-1.1  -1.12]
##  [-1.89  0.01]
##  [ 0.09  0.27]
##  [-1.2   0.33]
##  [-1.3   0.3 ]
##  [-1.    0.45]
##  [ 1.67 -0.89]
##  [ 1.18  0.53]
##  [ 1.08  0.53]
##  [ 1.37  2.33]
##  [-0.31 -0.13]
##  [ 0.38 -0.45]
##  [-0.41 -0.77]
##  [-0.11 -0.51]
##  [ 0.98 -1.15]
##  [-0.9  -0.77]
##  [-0.21 -0.51]
##  [-1.1  -0.45]
##  [-1.2   1.4 ]]
print(X_test)
## [[-0.8   0.5 ]
##  [-0.01 -0.57]
##  [-0.31  0.16]
##  [-0.8   0.27]
##  [-0.31 -0.57]
##  [-1.1  -1.44]
```

```
##  [-0.71 -1.58]
##  [-0.21  2.16]
##  [-1.99 -0.05]
##  [ 0.88 -0.77]
##  [-0.8  -0.6 ]
##  [-1.   -0.42]
##  [-0.11 -0.42]
##  [ 0.09  0.22]
##  [-1.8   0.48]
##  [-0.61  1.37]
##  [-0.11  0.22]
##  [-1.89  0.45]
##  [ 1.67  1.75]
##  [-0.31 -1.38]
##  [-0.31 -0.65]
##  [ 0.88  2.16]
##  [ 0.28 -0.54]
##  [ 0.88  1.03]
##  [-1.5  -1.21]
##  [ 1.08  2.07]
##  [-1.    0.5 ]
##  [-0.9   0.3 ]
##  [-0.11 -0.22]
##  [-0.61  0.48]
##  [-1.7   0.53]
##  [-0.11  0.27]
##  [ 1.87 -0.28]
##  [-0.11 -0.48]
##  [-1.4  -0.34]
##  [-1.99 -0.51]
##  [-1.6   0.33]
##  [-0.41 -0.77]
##  [-0.71 -1.03]
##  [ 1.08 -0.97]
##  [-1.1   0.53]
##  [ 0.28 -0.51]
##  [-1.1   0.42]
##  [-0.31 -1.44]
##  [ 0.48  1.23]
##  [-1.1  -0.34]
##  [-0.11  0.3 ]
##  [ 1.37  0.59]
##  [-1.2  -1.15]
##  [ 1.08  0.48]
##  [ 1.87  1.52]
```

```
##  [-0.41 -1.29]
##  [-0.31 -0.36]
##  [-0.41  1.32]
##  [ 2.07  0.53]
##  [ 0.68 -1.09]
##  [-0.9   0.39]
##  [-1.2   0.3 ]
##  [ 1.08 -1.21]
##  [-1.5  -1.44]
##  [-0.61 -1.5 ]
##  [ 2.17 -0.8 ]
##  [-1.89  0.19]
##  [-0.21  0.85]
##  [-1.89 -1.26]
##  [ 2.17  0.39]
##  [-1.4   0.56]
##  [-1.1  -0.34]
##  [ 0.19 -0.65]
##  [ 0.38  0.01]
##  [-0.61  2.33]
##  [-0.31  0.22]
##  [-1.6  -0.19]
##  [ 0.68 -1.38]
##  [-1.1   0.56]
##  [-1.99  0.36]
##  [ 0.38  0.27]
##  [ 0.19 -0.28]
##  [ 1.47 -1.03]
##  [ 0.88  1.08]
##  [ 1.97  2.16]
##  [ 2.07  0.39]
##  [-1.4  -0.42]
##  [-1.2  -1.  ]
##  [ 1.97 -0.92]
##  [ 0.38  0.3 ]
##  [ 0.19  0.16]
##  [ 2.07  1.75]
##  [ 0.78 -0.83]
##  [ 0.28 -0.28]
##  [ 0.38 -0.16]
##  [-0.11  2.22]
##  [-1.5  -0.63]
##  [-1.3  -1.06]
##  [-1.4   0.42]
##  [-1.1   0.77]
```

```
##  [-1.5  -0.19]
##  [ 0.98 -1.06]
##  [ 0.98  0.59]
##  [ 0.38  1.  ]]
```

R

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

3.3.6 Training the SVM model on the Training set

Python

```
from sklearn.svm import SVC
classifier = SVC(kernel = 'linear', random_state = 0)
classifier.fit(X_train, y_train)
## SVC(kernel='linear', random_state=0)
```

R

```
# install.packages('e1071')
library(e1071)
classifier = svm(formula = Purchased ~ .,
                 data = training_set,
                 type = 'C-classification',
                 kernel = 'linear')
```

3.3.7 Predicting a new result

Python

```
print(classifier.predict(sc.transform([[30,87000]])))
## [0]
```

3.3.8 Predicting the Test set results

Python

```
y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))
## [[0 0]
##  [0 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]
##  [1 1]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [1 1]
##  [0 0]
##  [1 1]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 1]
##  [1 1]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [1 1]
##  [1 1]
##  [0 0]
```

```
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 1]
##  [0 0]
##  [0 0]
##  [0 1]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [0 1]
##  [0 0]
##  [1 1]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [0 0]
##  [0 1]
##  [0 0]
##  [0 0]
##  [1 0]
##  [0 0]
##  [1 1]
##  [1 1]
##  [1 1]
##  [1 0]
##  [0 0]
##  [0 0]
##  [1 1]
##  [1 1]
##  [0 0]
##  [1 1]
##  [0 1]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 1]
##  [0 0]
```

```
##  [0 1]
##  [1 1]
##  [1 1]]
```

R

```
y_pred = predict(classifier, newdata = test_set[-3])
```

3.3.9 Making the Confusion Matrix

Python

```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
## [[66  2]
##  [ 8 24]]
accuracy_score(y_test, y_pred)
## 0.9
```

R

```
cm = table(test_set[, 3], y_pred)
```

3.3.10 Visualising the Training set results

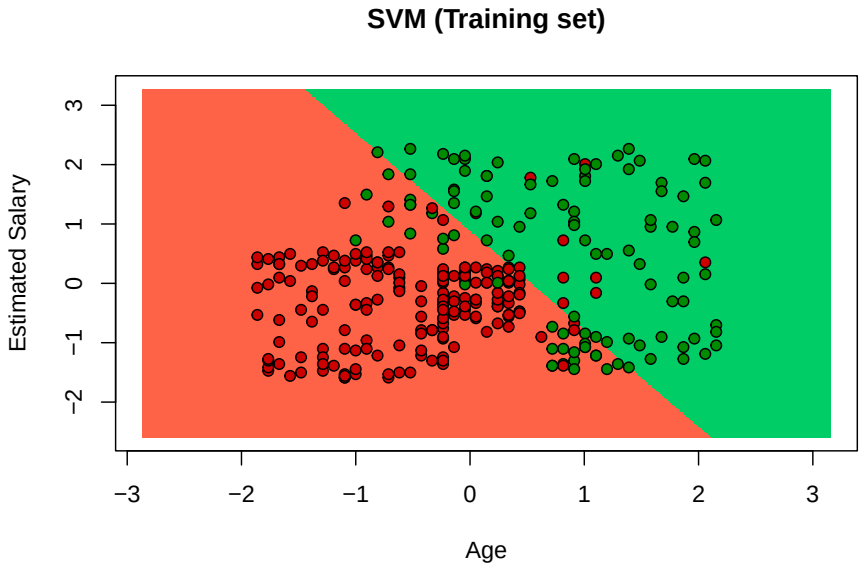
Python

```
from matplotlib.colors import ListedColormap
X_set, y_set = sc.inverse_transform(X_train), y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 10, stop = X_set[:, 0].max()
                     np.arange(start = X_set[:, 1].min() - 1000, stop = X_set[:, 1].ma
plt.contourf(X1, X2, classifier.predict(sc.transform(np.array([X1.ravel(), X2.ravel()]
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], c = ListedColormap(('red',
plt.title('SVM (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

R

```
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')
y_grid = predict(classifier, newdata = grid_set)
plot(set[, -3],
     main = 'SVM (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'))
```



3.3.11 Visualising the Test set results

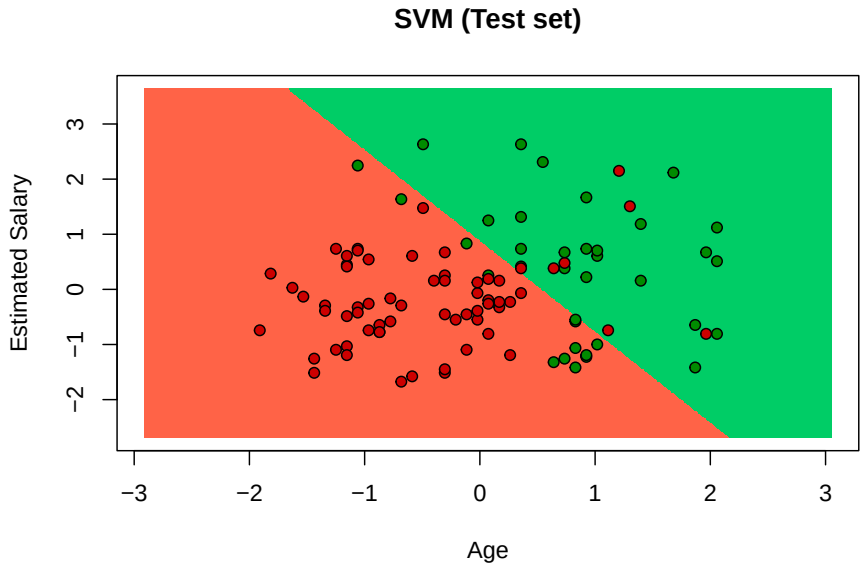
Python

```
from matplotlib.colors import ListedColormap
X_set, y_set = sc.inverse_transform(X_test), y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 10, stop = X_set[:, 0].max() + 10, ste
                     np.arange(start = X_set[:, 1].min() - 1000, stop = X_set[:, 1].max() + 1000,
plt.contourf(X1, X2, classifier.predict(sc.transform(np.array([X1.ravel(), X2.ravel()]).T)).resha
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
```

```
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], c = ListedColormap(('red',
plt.title('SVM (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

R

```
library(ElemStatLearn)
set = test_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')
y_grid = predict(classifier, newdata = grid_set)
plot(set[, -3], main = 'SVM (Test 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'))
```



3.4 Kernel SVM

3.4.1 Importing the libraries

Python

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

3.4.2 Importing the dataset

Python

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

R

```
dataset = read.csv('Social_Network_Ads.csv')
# dataset = dataset[3:5]
```

3.4.3 Encoding the target feature as factor

R

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

3.4.4 Splitting the dataset into the Training set and Test set

Python

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
print(X_train)
## [[    44  39000]
##  [    32 120000]
##  [    38  50000]
##  [    32 135000]
##  [    52  21000]
##  [    53 104000]
##  [    39  42000]
##  [    38  61000]
##  [    36  50000]
##  [    36  63000]
##  [    35  25000]
```

```
##  [    35  50000]
##  [    42  73000]
##  [    47  49000]
##  [    59  29000]
##  [    49  65000]
##  [    45 131000]
##  [    31  89000]
##  [    46  82000]
##  [    47  51000]
##  [    26  15000]
##  [    60 102000]
##  [    38 112000]
##  [    40 107000]
##  [    42  53000]
##  [    35  59000]
##  [    48  41000]
##  [    48 134000]
##  [    38 113000]
##  [    29 148000]
##  [    26  15000]
##  [    60  42000]
##  [    24  19000]
##  [    42 149000]
##  [    46  96000]
##  [    28  59000]
##  [    39  96000]
##  [    28  89000]
##  [    41  72000]
##  [    45  26000]
##  [    33  69000]
##  [    20  82000]
##  [    31  74000]
##  [    42  80000]
##  [    35  72000]
##  [    33 149000]
##  [    40  71000]
##  [    51 146000]
##  [    46  79000]
##  [    35  75000]
##  [    38  51000]
##  [    36  75000]
##  [    37  78000]
##  [    38  61000]
##  [    60 108000]
##  [    20  82000]
```

```
##  [    57  74000]
##  [    42  65000]
##  [    26  80000]
##  [    46 117000]
##  [    35  61000]
##  [    21  68000]
##  [    28  44000]
##  [    41  87000]
##  [    37  33000]
##  [    27  90000]
##  [    39  42000]
##  [    28 123000]
##  [    31 118000]
##  [    25  87000]
##  [    35  71000]
##  [    37  70000]
##  [    35  39000]
##  [    47  23000]
##  [    35 147000]
##  [    48 138000]
##  [    26  86000]
##  [    25  79000]
##  [    52 138000]
##  [    51  23000]
##  [    35  60000]
##  [    33 113000]
##  [    30 107000]
##  [    48  33000]
##  [    41  80000]
##  [    48  96000]
##  [    31  18000]
##  [    31  71000]
##  [    43 129000]
##  [    59  76000]
##  [    18  44000]
##  [    36 118000]
##  [    42  90000]
##  [    47  30000]
##  [    26  43000]
##  [    40  78000]
##  [    46  59000]
##  [    59  42000]
##  [    46  74000]
##  [    35  91000]
##  [    28  59000]
```

```
##  [    40  57000]
##  [    59 143000]
##  [    57  26000]
##  [    52  38000]
##  [    47 113000]
##  [    53 143000]
##  [    35  27000]
##  [    58 101000]
##  [    45  45000]
##  [    23  82000]
##  [    46  23000]
##  [    42  65000]
##  [    28  84000]
##  [    38  59000]
##  [    26  84000]
##  [    29  28000]
##  [    37  71000]
##  [    22  55000]
##  [    48  35000]
##  [    49  28000]
##  [    38  65000]
##  [    27  17000]
##  [    46  28000]
##  [    48 141000]
##  [    26  17000]
##  [    35  97000]
##  [    39  59000]
##  [    24  27000]
##  [    32  18000]
##  [    46  88000]
##  [    35  58000]
##  [    56  60000]
##  [    47  34000]
##  [    40  72000]
##  [    32 100000]
##  [    19  21000]
##  [    25  90000]
##  [    35  88000]
##  [    28  32000]
##  [    50  20000]
##  [    40  59000]
##  [    50  44000]
##  [    35  72000]
##  [    40 142000]
##  [    46  32000]
```

```
##  [    39  71000]
##  [    20  74000]
##  [    29  75000]
##  [    31  76000]
##  [    47  25000]
##  [    40  61000]
##  [    34 112000]
##  [    38  80000]
##  [    42  75000]
##  [    47  47000]
##  [    39  75000]
##  [    19  25000]
##  [    37  80000]
##  [    36  60000]
##  [    41  52000]
##  [    36 125000]
##  [    48  29000]
##  [    36 126000]
##  [    51 134000]
##  [    27  57000]
##  [    38  71000]
##  [    39  61000]
##  [    22  27000]
##  [    33  60000]
##  [    48  74000]
##  [    58  23000]
##  [    53  72000]
##  [    32 117000]
##  [    54  70000]
##  [    30  80000]
##  [    58  95000]
##  [    26  52000]
##  [    45  79000]
##  [    24  55000]
##  [    40  75000]
##  [    33  28000]
##  [    44 139000]
##  [    22  18000]
##  [    33  51000]
##  [    43 133000]
##  [    24  32000]
##  [    46  22000]
##  [    35  55000]
##  [    54 104000]
##  [    48 119000]
```

```
##  [    35  53000]
##  [    37 144000]
##  [    23  66000]
##  [    37 137000]
##  [    31  58000]
##  [    33  41000]
##  [    45  22000]
##  [    30  15000]
##  [    19  19000]
##  [    49  74000]
##  [    39 122000]
##  [    35  73000]
##  [    39  71000]
##  [    24  23000]
##  [    41  72000]
##  [    29  83000]
##  [    54  26000]
##  [    35  44000]
##  [    37  75000]
##  [    29  47000]
##  [    31  68000]
##  [    42  54000]
##  [    30 135000]
##  [    52 114000]
##  [    50  36000]
##  [    56 133000]
##  [    29  61000]
##  [    30  89000]
##  [    26  16000]
##  [    33  31000]
##  [    41  72000]
##  [    36  33000]
##  [    55 125000]
##  [    48 131000]
##  [    41  71000]
##  [    30  62000]
##  [    37  72000]
##  [    41  63000]
##  [    58  47000]
##  [    30 116000]
##  [    20  49000]
##  [    37  74000]
##  [    41  59000]
##  [    49  89000]
##  [    28  79000]
```

```
##  [    53  82000]
##  [    40  57000]
##  [    60  34000]
##  [    35 108000]
##  [    21  72000]
##  [    38  71000]
##  [    39 106000]
##  [    37  57000]
##  [    26  72000]
##  [    35  23000]
##  [    54 108000]
##  [    30  17000]
##  [    39 134000]
##  [    29  43000]
##  [    33  43000]
##  [    35  38000]
##  [    41  45000]
##  [    41  72000]
##  [    39 134000]
##  [    27 137000]
##  [    21  16000]
##  [    26  32000]
##  [    31  66000]
##  [    39  73000]
##  [    41  79000]
##  [    47  50000]
##  [    41  30000]
##  [    37  93000]
##  [    60  46000]
##  [    25  22000]
##  [    28  37000]
##  [    38  55000]
##  [    36  54000]
##  [    20  36000]
##  [    56 104000]
##  [    40  57000]
##  [    42 108000]
##  [    20  23000]
##  [    40  65000]
##  [    47  20000]
##  [    18  86000]
##  [    35  79000]
##  [    57  33000]
##  [    34  72000]
##  [    49  39000]
```

```
##  [    27  31000]
##  [    19  70000]
##  [    39  79000]
##  [    26  81000]
##  [    25  80000]
##  [    28  85000]
##  [    55  39000]
##  [    50  88000]
##  [    49  88000]
##  [    52 150000]
##  [    35  65000]
##  [    42  54000]
##  [    34  43000]
##  [    37  52000]
##  [    48  30000]
##  [    29  43000]
##  [    36  52000]
##  [    27  54000]
##  [    26 118000]]
print(y_train)
## [0 1 0 1 1 1 0 0 0 0 0 0 1 1 1 0 1 0 0 1 0 1 0 1 0 0 1 1 1 1 0 1 0 1 0 0 1
##  0 0 1 0 0 0 0 0 1 1 1 1 0 0 0 1 0 1 0 1 0 0 1 0 0 0 1 0 0 0 1 1 0 0 1 0 1
##  1 1 0 0 1 1 0 0 1 1 0 1 0 0 1 1 0 1 1 1 0 0 0 0 0 1 0 0 1 1 1 1 1 0 1 1 0
##  1 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 1 0 0 0 1 0 1 1 0 1 0 0 0 0 1 0 0 0 1 1 0 0
##  0 0 1 0 1 0 0 0 1 0 0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 1 1 0 1 0 0 0 0 0 1 0 0
##  0 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 1 1 0 0 0 0 0
##  0 1 1 0 0 0 0 1 0 0 0 0 1 0 1 0 1 0 0 0 1 0 0 0 1 0 1 0 0 0 0 0 1 1 0 0 0
##  0 0 1 0 1 1 0 0 0 0 0 1 0 1 0 0 1 0 0 1 0 1 0 0 0 0 0 0 1 1 1 1 0 0 0 0 1
##  0 0 0 0]
print(X_test)
## [[    30  87000]
##  [    38  50000]
##  [    35  75000]
##  [    30  79000]
##  [    35  50000]
##  [    27  20000]
##  [    31  15000]
##  [    36 144000]
##  [    18  68000]
##  [    47  43000]
##  [    30  49000]
##  [    28  55000]
##  [    37  55000]
##  [    39  77000]
##  [    20  86000]
```

```
##  [    32 117000]
##  [    37  77000]
##  [    19  85000]
##  [    55 130000]
##  [    35  22000]
##  [    35  47000]
##  [    47 144000]
##  [    41  51000]
##  [    47 105000]
##  [    23  28000]
##  [    49 141000]
##  [    28  87000]
##  [    29  80000]
##  [    37  62000]
##  [    32  86000]
##  [    21  88000]
##  [    37  79000]
##  [    57  60000]
##  [    37  53000]
##  [    24  58000]
##  [    18  52000]
##  [    22  81000]
##  [    34  43000]
##  [    31  34000]
##  [    49  36000]
##  [    27  88000]
##  [    41  52000]
##  [    27  84000]
##  [    35  20000]
##  [    43 112000]
##  [    27  58000]
##  [    37  80000]
##  [    52  90000]
##  [    26  30000]
##  [    49  86000]
##  [    57 122000]
##  [    34  25000]
##  [    35  57000]
##  [    34 115000]
##  [    59  88000]
##  [    45  32000]
##  [    29  83000]
##  [    26  80000]
##  [    49  28000]
##  [    23  20000]
```

```
##  [    32  18000]
##  [    60  42000]
##  [    19  76000]
##  [    36  99000]
##  [    19  26000]
##  [    60  83000]
##  [    24  89000]
##  [    27  58000]
##  [    40  47000]
##  [    42  70000]
##  [    32 150000]
##  [    35  77000]
##  [    22  63000]
##  [    45  22000]
##  [    27  89000]
##  [    18  82000]
##  [    42  79000]
##  [    40  60000]
##  [    53  34000]
##  [    47 107000]
##  [    58 144000]
##  [    59  83000]
##  [    24  55000]
##  [    26  35000]
##  [    58  38000]
##  [    42  80000]
##  [    40  75000]
##  [    59 130000]
##  [    46  41000]
##  [    41  60000]
##  [    42  64000]
##  [    37 146000]
##  [    23  48000]
##  [    25  33000]
##  [    24  84000]
##  [    27  96000]
##  [    23  63000]
##  [    48  33000]
##  [    48  90000]
##  [    42 104000]]
print(y_test)
## [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 1 0 0 0 0 0 1 1 0 0 0 0
##  0 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 0 1 1 0 0 1 0 0 1 0 1 0 1 0 0 0 0 1 0 0 1
##  0 0 0 0 1 1 1 0 0 0 1 1 0 1 1 0 0 1 0 0 0 1 0 1 1 1]
```

R

```
# 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)
```

3.4.5 Feature Scaling

Python

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
print(X_train)
## [[ 0.58 -0.89]
##  [-0.61  1.46]
##  [-0.01 -0.57]
##  [-0.61  1.9 ]
##  [ 1.37 -1.41]
##  [ 1.47  1.  ]
##  [ 0.09 -0.8 ]
##  [-0.01 -0.25]
##  [-0.21 -0.57]
##  [-0.21 -0.19]
##  [-0.31 -1.29]
##  [-0.31 -0.57]
##  [ 0.38  0.1 ]
##  [ 0.88 -0.6 ]
##  [ 2.07 -1.18]
##  [ 1.08 -0.13]
##  [ 0.68  1.78]
##  [-0.71  0.56]
##  [ 0.78  0.36]
##  [ 0.88 -0.54]
##  [-1.2  -1.58]
##  [ 2.17  0.94]
##  [-0.01  1.23]
##  [ 0.19  1.08]
##  [ 0.38 -0.48]
##  [-0.31 -0.31]
##  [ 0.98 -0.83]
##  [ 0.98  1.87]
```

```
##  [-0.01  1.26]
##  [-0.9   2.27]
##  [-1.2  -1.58]
##  [ 2.17 -0.8 ]
##  [-1.4  -1.47]
##  [ 0.38  2.3 ]
##  [ 0.78  0.77]
##  [-1.   -0.31]
##  [ 0.09  0.77]
##  [-1.    0.56]
##  [ 0.28  0.07]
##  [ 0.68 -1.26]
##  [-0.51 -0.02]
##  [-1.8   0.36]
##  [-0.71  0.13]
##  [ 0.38  0.3 ]
##  [-0.31  0.07]
##  [-0.51  2.3 ]
##  [ 0.19  0.04]
##  [ 1.27  2.22]
##  [ 0.78  0.27]
##  [-0.31  0.16]
##  [-0.01 -0.54]
##  [-0.21  0.16]
##  [-0.11  0.24]
##  [-0.01 -0.25]
##  [ 2.17  1.11]
##  [-1.8   0.36]
##  [ 1.87  0.13]
##  [ 0.38 -0.13]
##  [-1.2   0.3 ]
##  [ 0.78  1.37]
##  [-0.31 -0.25]
##  [-1.7  -0.05]
##  [-1.   -0.74]
##  [ 0.28  0.5 ]
##  [-0.11 -1.06]
##  [-1.1   0.59]
##  [ 0.09 -0.8 ]
##  [-1.    1.55]
##  [-0.71  1.4 ]
##  [-1.3   0.5 ]
##  [-0.31  0.04]
##  [-0.11  0.01]
##  [-0.31 -0.89]
```

```
##  [ 0.88 -1.35]
##  [-0.31  2.24]
##  [ 0.98  1.98]
##  [-1.2   0.48]
##  [-1.3   0.27]
##  [ 1.37  1.98]
##  [ 1.27 -1.35]
##  [-0.31 -0.28]
##  [-0.51  1.26]
##  [-0.8   1.08]
##  [ 0.98 -1.06]
##  [ 0.28  0.3 ]
##  [ 0.98  0.77]
##  [-0.71 -1.5 ]
##  [-0.71  0.04]
##  [ 0.48  1.72]
##  [ 2.07  0.19]
##  [-1.99 -0.74]
##  [-0.21  1.4 ]
##  [ 0.38  0.59]
##  [ 0.88 -1.15]
##  [-1.2  -0.77]
##  [ 0.19  0.24]
##  [ 0.78 -0.31]
##  [ 2.07 -0.8 ]
##  [ 0.78  0.13]
##  [-0.31  0.62]
##  [-1.   -0.31]
##  [ 0.19 -0.36]
##  [ 2.07  2.13]
##  [ 1.87 -1.26]
##  [ 1.37 -0.92]
##  [ 0.88  1.26]
##  [ 1.47  2.13]
##  [-0.31 -1.23]
##  [ 1.97  0.91]
##  [ 0.68 -0.71]
##  [-1.5   0.36]
##  [ 0.78 -1.35]
##  [ 0.38 -0.13]
##  [-1.    0.42]
##  [-0.01 -0.31]
##  [-1.2   0.42]
##  [-0.9  -1.21]
##  [-0.11  0.04]
```

```
##  [-1.6  -0.42]
##  [ 0.98 -1.  ]
##  [ 1.08 -1.21]
##  [-0.01 -0.13]
##  [-1.1  -1.52]
##  [ 0.78 -1.21]
##  [ 0.98  2.07]
##  [-1.2  -1.52]
##  [-0.31  0.79]
##  [ 0.09 -0.31]
##  [-1.4  -1.23]
##  [-0.61 -1.5 ]
##  [ 0.78  0.53]
##  [-0.31 -0.34]
##  [ 1.77 -0.28]
##  [ 0.88 -1.03]
##  [ 0.19  0.07]
##  [-0.61  0.88]
##  [-1.89 -1.41]
##  [-1.3   0.59]
##  [-0.31  0.53]
##  [-1.   -1.09]
##  [ 1.18 -1.44]
##  [ 0.19 -0.31]
##  [ 1.18 -0.74]
##  [-0.31  0.07]
##  [ 0.19  2.1 ]
##  [ 0.78 -1.09]
##  [ 0.09  0.04]
##  [-1.8   0.13]
##  [-0.9   0.16]
##  [-0.71  0.19]
##  [ 0.88 -1.29]
##  [ 0.19 -0.25]
##  [-0.41  1.23]
##  [-0.01  0.3 ]
##  [ 0.38  0.16]
##  [ 0.88 -0.65]
##  [ 0.09  0.16]
##  [-1.89 -1.29]
##  [-0.11  0.3 ]
##  [-0.21 -0.28]
##  [ 0.28 -0.51]
##  [-0.21  1.61]
##  [ 0.98 -1.18]
```

```
##  [-0.21  1.64]
##  [ 1.27  1.87]
##  [-1.1  -0.36]
##  [-0.01  0.04]
##  [ 0.09 -0.25]
##  [-1.6  -1.23]
##  [-0.51 -0.28]
##  [ 0.98  0.13]
##  [ 1.97 -1.35]
##  [ 1.47  0.07]
##  [-0.61  1.37]
##  [ 1.57  0.01]
##  [-0.8   0.3 ]
##  [ 1.97  0.74]
##  [-1.2  -0.51]
##  [ 0.68  0.27]
##  [-1.4  -0.42]
##  [ 0.19  0.16]
##  [-0.51 -1.21]
##  [ 0.58  2.01]
##  [-1.6  -1.5 ]
##  [-0.51 -0.54]
##  [ 0.48  1.84]
##  [-1.4  -1.09]
##  [ 0.78 -1.38]
##  [-0.31 -0.42]
##  [ 1.57  1.  ]
##  [ 0.98  1.43]
##  [-0.31 -0.48]
##  [-0.11  2.16]
##  [-1.5  -0.1 ]
##  [-0.11  1.95]
##  [-0.71 -0.34]
##  [-0.51 -0.83]
##  [ 0.68 -1.38]
##  [-0.8  -1.58]
##  [-1.89 -1.47]
##  [ 1.08  0.13]
##  [ 0.09  1.52]
##  [-0.31  0.1 ]
##  [ 0.09  0.04]
##  [-1.4  -1.35]
##  [ 0.28  0.07]
##  [-0.9   0.39]
##  [ 1.57 -1.26]
```

```
##  [-0.31 -0.74]
##  [-0.11  0.16]
##  [-0.9  -0.65]
##  [-0.71 -0.05]
##  [ 0.38 -0.45]
##  [-0.8   1.9 ]
##  [ 1.37  1.29]
##  [ 1.18 -0.97]
##  [ 1.77  1.84]
##  [-0.9  -0.25]
##  [-0.8   0.56]
##  [-1.2  -1.55]
##  [-0.51 -1.12]
##  [ 0.28  0.07]
##  [-0.21 -1.06]
##  [ 1.67  1.61]
##  [ 0.98  1.78]
##  [ 0.28  0.04]
##  [-0.8  -0.22]
##  [-0.11  0.07]
##  [ 0.28 -0.19]
##  [ 1.97 -0.65]
##  [-0.8   1.35]
##  [-1.8  -0.6 ]
##  [-0.11  0.13]
##  [ 0.28 -0.31]
##  [ 1.08  0.56]
##  [-1.    0.27]
##  [ 1.47  0.36]
##  [ 0.19 -0.36]
##  [ 2.17 -1.03]
##  [-0.31  1.11]
##  [-1.7   0.07]
##  [-0.01  0.04]
##  [ 0.09  1.06]
##  [-0.11 -0.36]
##  [-1.2   0.07]
##  [-0.31 -1.35]
##  [ 1.57  1.11]
##  [-0.8  -1.52]
##  [ 0.09  1.87]
##  [-0.9  -0.77]
##  [-0.51 -0.77]
##  [-0.31 -0.92]
##  [ 0.28 -0.71]
```

```
##  [ 0.28  0.07]
##  [ 0.09  1.87]
##  [-1.1   1.95]
##  [-1.7  -1.55]
##  [-1.2  -1.09]
##  [-0.71 -0.1 ]
##  [ 0.09  0.1 ]
##  [ 0.28  0.27]
##  [ 0.88 -0.57]
##  [ 0.28 -1.15]
##  [-0.11  0.68]
##  [ 2.17 -0.68]
##  [-1.3  -1.38]
##  [-1.   -0.94]
##  [-0.01 -0.42]
##  [-0.21 -0.45]
##  [-1.8  -0.97]
##  [ 1.77  1.  ]
##  [ 0.19 -0.36]
##  [ 0.38  1.11]
##  [-1.8  -1.35]
##  [ 0.19 -0.13]
##  [ 0.88 -1.44]
##  [-1.99  0.48]
##  [-0.31  0.27]
##  [ 1.87 -1.06]
##  [-0.41  0.07]
##  [ 1.08 -0.89]
##  [-1.1  -1.12]
##  [-1.89  0.01]
##  [ 0.09  0.27]
##  [-1.2   0.33]
##  [-1.3   0.3 ]
##  [-1.    0.45]
##  [ 1.67 -0.89]
##  [ 1.18  0.53]
##  [ 1.08  0.53]
##  [ 1.37  2.33]
##  [-0.31 -0.13]
##  [ 0.38 -0.45]
##  [-0.41 -0.77]
##  [-0.11 -0.51]
##  [ 0.98 -1.15]
##  [-0.9  -0.77]
##  [-0.21 -0.51]
```

```
##  [-1.1  -0.45]
##  [-1.2   1.4 ]]
print(X_test)
## [[-0.8   0.5 ]
##  [-0.01 -0.57]
##  [-0.31  0.16]
##  [-0.8   0.27]
##  [-0.31 -0.57]
##  [-1.1  -1.44]
##  [-0.71 -1.58]
##  [-0.21  2.16]
##  [-1.99 -0.05]
##  [ 0.88 -0.77]
##  [-0.8  -0.6 ]
##  [-1.   -0.42]
##  [-0.11 -0.42]
##  [ 0.09  0.22]
##  [-1.8   0.48]
##  [-0.61  1.37]
##  [-0.11  0.22]
##  [-1.89  0.45]
##  [ 1.67  1.75]
##  [-0.31 -1.38]
##  [-0.31 -0.65]
##  [ 0.88  2.16]
##  [ 0.28 -0.54]
##  [ 0.88  1.03]
##  [-1.5  -1.21]
##  [ 1.08  2.07]
##  [-1.    0.5 ]
##  [-0.9   0.3 ]
##  [-0.11 -0.22]
##  [-0.61  0.48]
##  [-1.7   0.53]
##  [-0.11  0.27]
##  [ 1.87 -0.28]
##  [-0.11 -0.48]
##  [-1.4  -0.34]
##  [-1.99 -0.51]
##  [-1.6   0.33]
##  [-0.41 -0.77]
##  [-0.71 -1.03]
##  [ 1.08 -0.97]
##  [-1.1   0.53]
##  [ 0.28 -0.51]
```

```
##  [-1.1   0.42]
##  [-0.31 -1.44]
##  [ 0.48  1.23]
##  [-1.1  -0.34]
##  [-0.11  0.3 ]
##  [ 1.37  0.59]
##  [-1.2  -1.15]
##  [ 1.08  0.48]
##  [ 1.87  1.52]
##  [-0.41 -1.29]
##  [-0.31 -0.36]
##  [-0.41  1.32]
##  [ 2.07  0.53]
##  [ 0.68 -1.09]
##  [-0.9   0.39]
##  [-1.2   0.3 ]
##  [ 1.08 -1.21]
##  [-1.5  -1.44]
##  [-0.61 -1.5 ]
##  [ 2.17 -0.8 ]
##  [-1.89  0.19]
##  [-0.21  0.85]
##  [-1.89 -1.26]
##  [ 2.17  0.39]
##  [-1.4   0.56]
##  [-1.1  -0.34]
##  [ 0.19 -0.65]
##  [ 0.38  0.01]
##  [-0.61  2.33]
##  [-0.31  0.22]
##  [-1.6  -0.19]
##  [ 0.68 -1.38]
##  [-1.1   0.56]
##  [-1.99  0.36]
##  [ 0.38  0.27]
##  [ 0.19 -0.28]
##  [ 1.47 -1.03]
##  [ 0.88  1.08]
##  [ 1.97  2.16]
##  [ 2.07  0.39]
##  [-1.4  -0.42]
##  [-1.2  -1.  ]
##  [ 1.97 -0.92]
##  [ 0.38  0.3 ]
##  [ 0.19  0.16]
```

```
##  [ 2.07  1.75]
##  [ 0.78 -0.83]
##  [ 0.28 -0.28]
##  [ 0.38 -0.16]
##  [-0.11  2.22]
##  [-1.5  -0.63]
##  [-1.3  -1.06]
##  [-1.4   0.42]
##  [-1.1   0.77]
##  [-1.5  -0.19]
##  [ 0.98 -1.06]
##  [ 0.98  0.59]
##  [ 0.38  1.  ]]
```

R

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

3.4.6 Training the Kernel SVM model on the Training set

Python

```
from sklearn.svm import SVC
classifier = SVC(kernel = 'rbf', random_state = 0)
classifier.fit(X_train, y_train)
## SVC(random_state=0)
```

R

```
# install.packages('e1071')
library(e1071)
classifier = svm(formula = Purchased ~ .,
                 data = training_set,
                 type = 'C-classification',
                 kernel = 'radial')
```

3.4.7 Predicting a new result

Python

```
print(classifier.predict(sc.transform([[30,87000]])))
## [0]
```

3.4.8 Predicting the Test set results

Python

```
y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))
## [[0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [1 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [1 0]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [1 1]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 1]
##  [1 1]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [0 0]
##  [0 0]
```

```
##  [0 0]
##  [1 1]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [1 1]
##  [1 1]
##  [0 0]
##  [0 0]
##  [1 0]
##  [1 1]
##  [1 1]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [1 1]
##  [0 0]
##  [1 1]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [1 1]
##  [1 1]
##  [1 1]
##  [1 0]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 1]
##  [0 0]
##  [1 1]
```

```
##  [1 1]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 1]
##  [0 0]
##  [1 1]
##  [1 1]
##  [1 1]]
```

R

```
y_pred = predict(classifier, newdata = test_set[-3])
```

3.4.9 Making the Confusion Matrix

Python

```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
## [[64  4]
##  [ 3 29]]
accuracy_score(y_test, y_pred)
## 0.93
```

R

```
cm = table(test_set[, 3], y_pred)
```

3.4.10 Visualising the Training set results

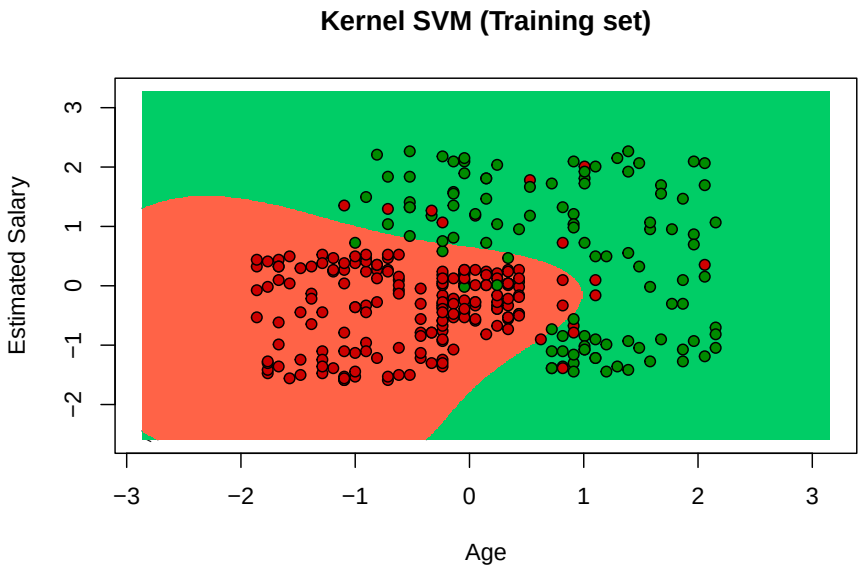
Python

```
from matplotlib.colors import ListedColormap
X_set, y_set = sc.inverse_transform(X_train), y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 10, stop = X_set[:, 0].max() + 10, ste
                     np.arange(start = X_set[:, 1].min() - 1000, stop = X_set[:, 1].max() + 1000,
plt.contourf(X1, X2, classifier.predict(sc.transform(np.array([X1.ravel(), X2.ravel()]).T)).resha
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], c = ListedColormap(('red', 'green'))(
plt.title('Kernel SVM (Training set)')
```

```
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

R

```
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')
y_grid = predict(classifier, newdata = grid_set)
plot(set[, -3],
     main = 'Kernel SVM (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'))
```



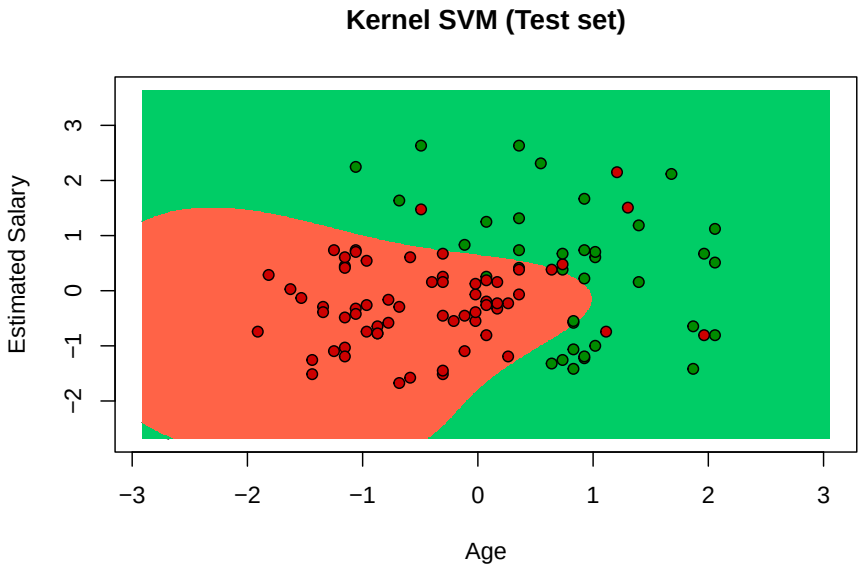
3.4.11 Visualising the Test set results

Python

```
from matplotlib.colors import ListedColormap
X_set, y_set = sc.inverse_transform(X_test), y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 10, stop = X_set[:, 0].max() + 10, ste
                     np.arange(start = X_set[:, 1].min() - 1000, stop = X_set[:, 1].max() + 1000,
plt.contourf(X1, X2, classifier.predict(sc.transform(np.array([X1.ravel(), X2.ravel()]).T)).resha
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], c = ListedColormap(('red', 'green'))(
plt.title('Kernel SVM (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

R

```
library(ElemStatLearn)
set = test_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')
y_grid = predict(classifier, newdata = grid_set)
plot(set[, -3], main = 'Kernel SVM (Test 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'))
```



3.5 Naive Bayes

3.5.1 Importing the libraries

Python

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

3.5.2 Importing the dataset

Python

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

R

```
dataset = read.csv('Social_Network_Ads.csv')
# dataset = dataset[3:5]
```

3.5.3 Encoding the target feature as factor

R

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

3.5.4 Splitting the dataset into the Training set and Test set

Python

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
print(X_train)
## [[    44  39000]
##  [    32 120000]
##  [    38  50000]
##  [    32 135000]
##  [    52  21000]
##  [    53 104000]
##  [    39  42000]
##  [    38  61000]
##  [    36  50000]
##  [    36  63000]
##  [    35  25000]
##  [    35  50000]
##  [    42  73000]
##  [    47  49000]
##  [    59  29000]
##  [    49  65000]
##  [    45 131000]
##  [    31  89000]
##  [    46  82000]
##  [    47  51000]
##  [    26  15000]
##  [    60 102000]
##  [    38 112000]
##  [    40 107000]
##  [    42  53000]
##  [    35  59000]
##  [    48  41000]
##  [    48 134000]
##  [    38 113000]
##  [    29 148000]
##  [    26  15000]
##  [    60  42000]
##  [    24  19000]
##  [    42 149000]
##  [    46  96000]
```

```
##  [    28  59000]
##  [    39  96000]
##  [    28  89000]
##  [    41  72000]
##  [    45  26000]
##  [    33  69000]
##  [    20  82000]
##  [    31  74000]
##  [    42  80000]
##  [    35  72000]
##  [    33 149000]
##  [    40  71000]
##  [    51 146000]
##  [    46  79000]
##  [    35  75000]
##  [    38  51000]
##  [    36  75000]
##  [    37  78000]
##  [    38  61000]
##  [    60 108000]
##  [    20  82000]
##  [    57  74000]
##  [    42  65000]
##  [    26  80000]
##  [    46 117000]
##  [    35  61000]
##  [    21  68000]
##  [    28  44000]
##  [    41  87000]
##  [    37  33000]
##  [    27  90000]
##  [    39  42000]
##  [    28 123000]
##  [    31 118000]
##  [    25  87000]
##  [    35  71000]
##  [    37  70000]
##  [    35  39000]
##  [    47  23000]
##  [    35 147000]
##  [    48 138000]
##  [    26  86000]
##  [    25  79000]
##  [    52 138000]
##  [    51  23000]
```

```
##  [    35  60000]
##  [    33 113000]
##  [    30 107000]
##  [    48  33000]
##  [    41  80000]
##  [    48  96000]
##  [    31  18000]
##  [    31  71000]
##  [    43 129000]
##  [    59  76000]
##  [    18  44000]
##  [    36 118000]
##  [    42  90000]
##  [    47  30000]
##  [    26  43000]
##  [    40  78000]
##  [    46  59000]
##  [    59  42000]
##  [    46  74000]
##  [    35  91000]
##  [    28  59000]
##  [    40  57000]
##  [    59 143000]
##  [    57  26000]
##  [    52  38000]
##  [    47 113000]
##  [    53 143000]
##  [    35  27000]
##  [    58 101000]
##  [    45  45000]
##  [    23  82000]
##  [    46  23000]
##  [    42  65000]
##  [    28  84000]
##  [    38  59000]
##  [    26  84000]
##  [    29  28000]
##  [    37  71000]
##  [    22  55000]
##  [    48  35000]
##  [    49  28000]
##  [    38  65000]
##  [    27  17000]
##  [    46  28000]
##  [    48 141000]
```

```
##  [    26  17000]
##  [    35  97000]
##  [    39  59000]
##  [    24  27000]
##  [    32  18000]
##  [    46  88000]
##  [    35  58000]
##  [    56  60000]
##  [    47  34000]
##  [    40  72000]
##  [    32 100000]
##  [    19  21000]
##  [    25  90000]
##  [    35  88000]
##  [    28  32000]
##  [    50  20000]
##  [    40  59000]
##  [    50  44000]
##  [    35  72000]
##  [    40 142000]
##  [    46  32000]
##  [    39  71000]
##  [    20  74000]
##  [    29  75000]
##  [    31  76000]
##  [    47  25000]
##  [    40  61000]
##  [    34 112000]
##  [    38  80000]
##  [    42  75000]
##  [    47  47000]
##  [    39  75000]
##  [    19  25000]
##  [    37  80000]
##  [    36  60000]
##  [    41  52000]
##  [    36 125000]
##  [    48  29000]
##  [    36 126000]
##  [    51 134000]
##  [    27  57000]
##  [    38  71000]
##  [    39  61000]
##  [    22  27000]
##  [    33  60000]
```

```
##  [    48  74000]
##  [    58  23000]
##  [    53  72000]
##  [    32 117000]
##  [    54  70000]
##  [    30  80000]
##  [    58  95000]
##  [    26  52000]
##  [    45  79000]
##  [    24  55000]
##  [    40  75000]
##  [    33  28000]
##  [    44 139000]
##  [    22  18000]
##  [    33  51000]
##  [    43 133000]
##  [    24  32000]
##  [    46  22000]
##  [    35  55000]
##  [    54 104000]
##  [    48 119000]
##  [    35  53000]
##  [    37 144000]
##  [    23  66000]
##  [    37 137000]
##  [    31  58000]
##  [    33  41000]
##  [    45  22000]
##  [    30  15000]
##  [    19  19000]
##  [    49  74000]
##  [    39 122000]
##  [    35  73000]
##  [    39  71000]
##  [    24  23000]
##  [    41  72000]
##  [    29  83000]
##  [    54  26000]
##  [    35  44000]
##  [    37  75000]
##  [    29  47000]
##  [    31  68000]
##  [    42  54000]
##  [    30 135000]
##  [    52 114000]
```

```
##  [    50  36000]
##  [    56 133000]
##  [    29  61000]
##  [    30  89000]
##  [    26  16000]
##  [    33  31000]
##  [    41  72000]
##  [    36  33000]
##  [    55 125000]
##  [    48 131000]
##  [    41  71000]
##  [    30  62000]
##  [    37  72000]
##  [    41  63000]
##  [    58  47000]
##  [    30 116000]
##  [    20  49000]
##  [    37  74000]
##  [    41  59000]
##  [    49  89000]
##  [    28  79000]
##  [    53  82000]
##  [    40  57000]
##  [    60  34000]
##  [    35 108000]
##  [    21  72000]
##  [    38  71000]
##  [    39 106000]
##  [    37  57000]
##  [    26  72000]
##  [    35  23000]
##  [    54 108000]
##  [    30  17000]
##  [    39 134000]
##  [    29  43000]
##  [    33  43000]
##  [    35  38000]
##  [    41  45000]
##  [    41  72000]
##  [    39 134000]
##  [    27 137000]
##  [    21  16000]
##  [    26  32000]
##  [    31  66000]
##  [    39  73000]
```

```
##  [    41  79000]
##  [    47  50000]
##  [    41  30000]
##  [    37  93000]
##  [    60  46000]
##  [    25  22000]
##  [    28  37000]
##  [    38  55000]
##  [    36  54000]
##  [    20  36000]
##  [    56 104000]
##  [    40  57000]
##  [    42 108000]
##  [    20  23000]
##  [    40  65000]
##  [    47  20000]
##  [    18  86000]
##  [    35  79000]
##  [    57  33000]
##  [    34  72000]
##  [    49  39000]
##  [    27  31000]
##  [    19  70000]
##  [    39  79000]
##  [    26  81000]
##  [    25  80000]
##  [    28  85000]
##  [    55  39000]
##  [    50  88000]
##  [    49  88000]
##  [    52 150000]
##  [    35  65000]
##  [    42  54000]
##  [    34  43000]
##  [    37  52000]
##  [    48  30000]
##  [    29  43000]
##  [    36  52000]
##  [    27  54000]
##  [    26 118000]]
print(y_train)
## [0 1 0 1 1 1 0 0 0 0 0 0 1 1 1 0 1 0 0 1 0 1 0 1 0 0 1 1 1 1 0 1 0 1 0 0 1
##  0 0 1 0 0 0 0 0 1 1 1 1 0 0 0 1 0 1 0 1 0 0 1 0 0 0 1 0 0 0 1 1 0 0 1 0 1
##  1 1 0 0 1 1 0 0 1 1 0 1 0 0 1 1 0 1 1 1 0 0 0 0 0 1 0 0 1 1 1 1 1 0 1 1 0
##  1 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 1 0 0 0 1 0 1 1 0 1 0 0 0 0 1 0 0 0 1 1 0 0
```

```
##  0 0 1 0 1 0 0 0 1 0 0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 1 1 0 1 0 0 0 0 0 1 0 0
##  0 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 1 1 0 0 0 0 0
##  0 1 1 0 0 0 0 1 0 0 0 0 1 0 1 0 1 0 0 0 1 0 0 0 1 0 1 0 0 0 0 0 1 1 0 0 0
##  0 0 1 0 1 1 0 0 0 0 0 0 1 0 1 0 0 1 0 0 1 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 1
##  0 0 0 0]
print(X_test)
## [[    30  87000]
##  [    38  50000]
##  [    35  75000]
##  [    30  79000]
##  [    35  50000]
##  [    27  20000]
##  [    31  15000]
##  [    36 144000]
##  [    18  68000]
##  [    47  43000]
##  [    30  49000]
##  [    28  55000]
##  [    37  55000]
##  [    39  77000]
##  [    20  86000]
##  [    32 117000]
##  [    37  77000]
##  [    19  85000]
##  [    55 130000]
##  [    35  22000]
##  [    35  47000]
##  [    47 144000]
##  [    41  51000]
##  [    47 105000]
##  [    23  28000]
##  [    49 141000]
##  [    28  87000]
##  [    29  80000]
##  [    37  62000]
##  [    32  86000]
##  [    21  88000]
##  [    37  79000]
##  [    57  60000]
##  [    37  53000]
##  [    24  58000]
##  [    18  52000]
##  [    22  81000]
##  [    34  43000]
##  [    31  34000]
```

```
##  [    49  36000]
##  [    27  88000]
##  [    41  52000]
##  [    27  84000]
##  [    35  20000]
##  [    43 112000]
##  [    27  58000]
##  [    37  80000]
##  [    52  90000]
##  [    26  30000]
##  [    49  86000]
##  [    57 122000]
##  [    34  25000]
##  [    35  57000]
##  [    34 115000]
##  [    59  88000]
##  [    45  32000]
##  [    29  83000]
##  [    26  80000]
##  [    49  28000]
##  [    23  20000]
##  [    32  18000]
##  [    60  42000]
##  [    19  76000]
##  [    36  99000]
##  [    19  26000]
##  [    60  83000]
##  [    24  89000]
##  [    27  58000]
##  [    40  47000]
##  [    42  70000]
##  [    32 150000]
##  [    35  77000]
##  [    22  63000]
##  [    45  22000]
##  [    27  89000]
##  [    18  82000]
##  [    42  79000]
##  [    40  60000]
##  [    53  34000]
##  [    47 107000]
##  [    58 144000]
##  [    59  83000]
##  [    24  55000]
##  [    26  35000]
```

```
##  [    58  38000]
##  [    42  80000]
##  [    40  75000]
##  [    59 130000]
##  [    46  41000]
##  [    41  60000]
##  [    42  64000]
##  [    37 146000]
##  [    23  48000]
##  [    25  33000]
##  [    24  84000]
##  [    27  96000]
##  [    23  63000]
##  [    48  33000]
##  [    48  90000]
##  [    42 104000]]
print(y_test)
## [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 1 0 0 0 0 0 1 1 0 0 0 0
##  0 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 0 1 1 0 0 1 0 0 1 0 1 0 1 0 0 0 0 1 0 0 1
##  0 0 0 0 1 1 1 0 0 0 1 1 0 1 1 0 0 1 0 0 0 1 0 1 1 1]
```

R

```
# 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)
```

3.5.5 Feature Scaling

Python

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
print(X_train)
## [[ 0.58 -0.89]
##  [-0.61  1.46]
##  [-0.01 -0.57]
##  [-0.61  1.9 ]
##  [ 1.37 -1.41]
##  [ 1.47  1.  ]
##  [ 0.09 -0.8 ]
```

```
##  [-0.01 -0.25]
##  [-0.21 -0.57]
##  [-0.21 -0.19]
##  [-0.31 -1.29]
##  [-0.31 -0.57]
##  [ 0.38  0.1 ]
##  [ 0.88 -0.6 ]
##  [ 2.07 -1.18]
##  [ 1.08 -0.13]
##  [ 0.68  1.78]
##  [-0.71  0.56]
##  [ 0.78  0.36]
##  [ 0.88 -0.54]
##  [-1.2  -1.58]
##  [ 2.17  0.94]
##  [-0.01  1.23]
##  [ 0.19  1.08]
##  [ 0.38 -0.48]
##  [-0.31 -0.31]
##  [ 0.98 -0.83]
##  [ 0.98  1.87]
##  [-0.01  1.26]
##  [-0.9   2.27]
##  [-1.2  -1.58]
##  [ 2.17 -0.8 ]
##  [-1.4  -1.47]
##  [ 0.38  2.3 ]
##  [ 0.78  0.77]
##  [-1.   -0.31]
##  [ 0.09  0.77]
##  [-1.    0.56]
##  [ 0.28  0.07]
##  [ 0.68 -1.26]
##  [-0.51 -0.02]
##  [-1.8   0.36]
##  [-0.71  0.13]
##  [ 0.38  0.3 ]
##  [-0.31  0.07]
##  [-0.51  2.3 ]
##  [ 0.19  0.04]
##  [ 1.27  2.22]
##  [ 0.78  0.27]
##  [-0.31  0.16]
##  [-0.01 -0.54]
##  [-0.21  0.16]
```

```
##  [-0.11  0.24]
##  [-0.01 -0.25]
##  [ 2.17  1.11]
##  [-1.8   0.36]
##  [ 1.87  0.13]
##  [ 0.38 -0.13]
##  [-1.2   0.3 ]
##  [ 0.78  1.37]
##  [-0.31 -0.25]
##  [-1.7  -0.05]
##  [-1.   -0.74]
##  [ 0.28  0.5 ]
##  [-0.11 -1.06]
##  [-1.1   0.59]
##  [ 0.09 -0.8 ]
##  [-1.    1.55]
##  [-0.71  1.4 ]
##  [-1.3   0.5 ]
##  [-0.31  0.04]
##  [-0.11  0.01]
##  [-0.31 -0.89]
##  [ 0.88 -1.35]
##  [-0.31  2.24]
##  [ 0.98  1.98]
##  [-1.2   0.48]
##  [-1.3   0.27]
##  [ 1.37  1.98]
##  [ 1.27 -1.35]
##  [-0.31 -0.28]
##  [-0.51  1.26]
##  [-0.8   1.08]
##  [ 0.98 -1.06]
##  [ 0.28  0.3 ]
##  [ 0.98  0.77]
##  [-0.71 -1.5 ]
##  [-0.71  0.04]
##  [ 0.48  1.72]
##  [ 2.07  0.19]
##  [-1.99 -0.74]
##  [-0.21  1.4 ]
##  [ 0.38  0.59]
##  [ 0.88 -1.15]
##  [-1.2  -0.77]
##  [ 0.19  0.24]
##  [ 0.78 -0.31]
```

```
##  [ 2.07 -0.8 ]
##  [ 0.78  0.13]
##  [-0.31  0.62]
##  [-1.   -0.31]
##  [ 0.19 -0.36]
##  [ 2.07  2.13]
##  [ 1.87 -1.26]
##  [ 1.37 -0.92]
##  [ 0.88  1.26]
##  [ 1.47  2.13]
##  [-0.31 -1.23]
##  [ 1.97  0.91]
##  [ 0.68 -0.71]
##  [-1.5   0.36]
##  [ 0.78 -1.35]
##  [ 0.38 -0.13]
##  [-1.    0.42]
##  [-0.01 -0.31]
##  [-1.2   0.42]
##  [-0.9  -1.21]
##  [-0.11  0.04]
##  [-1.6  -0.42]
##  [ 0.98 -1.  ]
##  [ 1.08 -1.21]
##  [-0.01 -0.13]
##  [-1.1  -1.52]
##  [ 0.78 -1.21]
##  [ 0.98  2.07]
##  [-1.2  -1.52]
##  [-0.31  0.79]
##  [ 0.09 -0.31]
##  [-1.4  -1.23]
##  [-0.61 -1.5 ]
##  [ 0.78  0.53]
##  [-0.31 -0.34]
##  [ 1.77 -0.28]
##  [ 0.88 -1.03]
##  [ 0.19  0.07]
##  [-0.61  0.88]
##  [-1.89 -1.41]
##  [-1.3   0.59]
##  [-0.31  0.53]
##  [-1.   -1.09]
##  [ 1.18 -1.44]
##  [ 0.19 -0.31]
```

```
##  [ 1.18 -0.74]
##  [-0.31  0.07]
##  [ 0.19  2.1 ]
##  [ 0.78 -1.09]
##  [ 0.09  0.04]
##  [-1.8   0.13]
##  [-0.9   0.16]
##  [-0.71  0.19]
##  [ 0.88 -1.29]
##  [ 0.19 -0.25]
##  [-0.41  1.23]
##  [-0.01  0.3 ]
##  [ 0.38  0.16]
##  [ 0.88 -0.65]
##  [ 0.09  0.16]
##  [-1.89 -1.29]
##  [-0.11  0.3 ]
##  [-0.21 -0.28]
##  [ 0.28 -0.51]
##  [-0.21  1.61]
##  [ 0.98 -1.18]
##  [-0.21  1.64]
##  [ 1.27  1.87]
##  [-1.1  -0.36]
##  [-0.01  0.04]
##  [ 0.09 -0.25]
##  [-1.6  -1.23]
##  [-0.51 -0.28]
##  [ 0.98  0.13]
##  [ 1.97 -1.35]
##  [ 1.47  0.07]
##  [-0.61  1.37]
##  [ 1.57  0.01]
##  [-0.8   0.3 ]
##  [ 1.97  0.74]
##  [-1.2  -0.51]
##  [ 0.68  0.27]
##  [-1.4  -0.42]
##  [ 0.19  0.16]
##  [-0.51 -1.21]
##  [ 0.58  2.01]
##  [-1.6  -1.5 ]
##  [-0.51 -0.54]
##  [ 0.48  1.84]
##  [-1.4  -1.09]
```

```
##  [ 0.78 -1.38]
##  [-0.31 -0.42]
##  [ 1.57  1.  ]
##  [ 0.98  1.43]
##  [-0.31 -0.48]
##  [-0.11  2.16]
##  [-1.5  -0.1 ]
##  [-0.11  1.95]
##  [-0.71 -0.34]
##  [-0.51 -0.83]
##  [ 0.68 -1.38]
##  [-0.8  -1.58]
##  [-1.89 -1.47]
##  [ 1.08  0.13]
##  [ 0.09  1.52]
##  [-0.31  0.1 ]
##  [ 0.09  0.04]
##  [-1.4  -1.35]
##  [ 0.28  0.07]
##  [-0.9   0.39]
##  [ 1.57 -1.26]
##  [-0.31 -0.74]
##  [-0.11  0.16]
##  [-0.9  -0.65]
##  [-0.71 -0.05]
##  [ 0.38 -0.45]
##  [-0.8   1.9 ]
##  [ 1.37  1.29]
##  [ 1.18 -0.97]
##  [ 1.77  1.84]
##  [-0.9  -0.25]
##  [-0.8   0.56]
##  [-1.2  -1.55]
##  [-0.51 -1.12]
##  [ 0.28  0.07]
##  [-0.21 -1.06]
##  [ 1.67  1.61]
##  [ 0.98  1.78]
##  [ 0.28  0.04]
##  [-0.8  -0.22]
##  [-0.11  0.07]
##  [ 0.28 -0.19]
##  [ 1.97 -0.65]
##  [-0.8   1.35]
##  [-1.8  -0.6 ]
```

```
##  [-0.11  0.13]
##  [ 0.28 -0.31]
##  [ 1.08  0.56]
##  [-1.    0.27]
##  [ 1.47  0.36]
##  [ 0.19 -0.36]
##  [ 2.17 -1.03]
##  [-0.31  1.11]
##  [-1.7   0.07]
##  [-0.01  0.04]
##  [ 0.09  1.06]
##  [-0.11 -0.36]
##  [-1.2   0.07]
##  [-0.31 -1.35]
##  [ 1.57  1.11]
##  [-0.8  -1.52]
##  [ 0.09  1.87]
##  [-0.9  -0.77]
##  [-0.51 -0.77]
##  [-0.31 -0.92]
##  [ 0.28 -0.71]
##  [ 0.28  0.07]
##  [ 0.09  1.87]
##  [-1.1   1.95]
##  [-1.7  -1.55]
##  [-1.2  -1.09]
##  [-0.71 -0.1 ]
##  [ 0.09  0.1 ]
##  [ 0.28  0.27]
##  [ 0.88 -0.57]
##  [ 0.28 -1.15]
##  [-0.11  0.68]
##  [ 2.17 -0.68]
##  [-1.3  -1.38]
##  [-1.   -0.94]
##  [-0.01 -0.42]
##  [-0.21 -0.45]
##  [-1.8  -0.97]
##  [ 1.77  1.  ]
##  [ 0.19 -0.36]
##  [ 0.38  1.11]
##  [-1.8  -1.35]
##  [ 0.19 -0.13]
##  [ 0.88 -1.44]
##  [-1.99  0.48]
```

```
##  [-0.31  0.27]
##  [ 1.87 -1.06]
##  [-0.41  0.07]
##  [ 1.08 -0.89]
##  [-1.1  -1.12]
##  [-1.89  0.01]
##  [ 0.09  0.27]
##  [-1.2   0.33]
##  [-1.3   0.3 ]
##  [-1.    0.45]
##  [ 1.67 -0.89]
##  [ 1.18  0.53]
##  [ 1.08  0.53]
##  [ 1.37  2.33]
##  [-0.31 -0.13]
##  [ 0.38 -0.45]
##  [-0.41 -0.77]
##  [-0.11 -0.51]
##  [ 0.98 -1.15]
##  [-0.9  -0.77]
##  [-0.21 -0.51]
##  [-1.1  -0.45]
##  [-1.2   1.4 ]]
print(X_test)
## [[-0.8   0.5 ]
##  [-0.01 -0.57]
##  [-0.31  0.16]
##  [-0.8   0.27]
##  [-0.31 -0.57]
##  [-1.1  -1.44]
##  [-0.71 -1.58]
##  [-0.21  2.16]
##  [-1.99 -0.05]
##  [ 0.88 -0.77]
##  [-0.8  -0.6 ]
##  [-1.   -0.42]
##  [-0.11 -0.42]
##  [ 0.09  0.22]
##  [-1.8   0.48]
##  [-0.61  1.37]
##  [-0.11  0.22]
##  [-1.89  0.45]
##  [ 1.67  1.75]
##  [-0.31 -1.38]
##  [-0.31 -0.65]
```

```
##  [ 0.88  2.16]
##  [ 0.28 -0.54]
##  [ 0.88  1.03]
##  [-1.5  -1.21]
##  [ 1.08  2.07]
##  [-1.    0.5 ]
##  [-0.9   0.3 ]
##  [-0.11 -0.22]
##  [-0.61  0.48]
##  [-1.7   0.53]
##  [-0.11  0.27]
##  [ 1.87 -0.28]
##  [-0.11 -0.48]
##  [-1.4  -0.34]
##  [-1.99 -0.51]
##  [-1.6   0.33]
##  [-0.41 -0.77]
##  [-0.71 -1.03]
##  [ 1.08 -0.97]
##  [-1.1   0.53]
##  [ 0.28 -0.51]
##  [-1.1   0.42]
##  [-0.31 -1.44]
##  [ 0.48  1.23]
##  [-1.1  -0.34]
##  [-0.11  0.3 ]
##  [ 1.37  0.59]
##  [-1.2  -1.15]
##  [ 1.08  0.48]
##  [ 1.87  1.52]
##  [-0.41 -1.29]
##  [-0.31 -0.36]
##  [-0.41  1.32]
##  [ 2.07  0.53]
##  [ 0.68 -1.09]
##  [-0.9   0.39]
##  [-1.2   0.3 ]
##  [ 1.08 -1.21]
##  [-1.5  -1.44]
##  [-0.61 -1.5 ]
##  [ 2.17 -0.8 ]
##  [-1.89  0.19]
##  [-0.21  0.85]
##  [-1.89 -1.26]
##  [ 2.17  0.39]
```

```
##  [-1.4   0.56]
##  [-1.1  -0.34]
##  [ 0.19 -0.65]
##  [ 0.38  0.01]
##  [-0.61  2.33]
##  [-0.31  0.22]
##  [-1.6  -0.19]
##  [ 0.68 -1.38]
##  [-1.1   0.56]
##  [-1.99  0.36]
##  [ 0.38  0.27]
##  [ 0.19 -0.28]
##  [ 1.47 -1.03]
##  [ 0.88  1.08]
##  [ 1.97  2.16]
##  [ 2.07  0.39]
##  [-1.4  -0.42]
##  [-1.2  -1.  ]
##  [ 1.97 -0.92]
##  [ 0.38  0.3 ]
##  [ 0.19  0.16]
##  [ 2.07  1.75]
##  [ 0.78 -0.83]
##  [ 0.28 -0.28]
##  [ 0.38 -0.16]
##  [-0.11  2.22]
##  [-1.5  -0.63]
##  [-1.3  -1.06]
##  [-1.4   0.42]
##  [-1.1   0.77]
##  [-1.5  -0.19]
##  [ 0.98 -1.06]
##  [ 0.98  0.59]
##  [ 0.38  1.  ]]
```

R

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

3.5.6 Training the Naive Bayes model on the Training set

Python

```
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
```

```
classifier.fit(X_train, y_train)
## GaussianNB()
```

R

```
# install.packages('e1071')
library(e1071)
classifier = naiveBayes(x = training_set[-3],
                        y = training_set$Purchased)
```

3.5.7 Predicting a new result

Python

```
print(classifier.predict(sc.transform([[30,87000]])))
## [0]
```

3.5.8 Predicting the Test set results

Python

```
y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))
## [[0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [1 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [1 1]
```

```
##  [0 0]
##  [1 1]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 1]
##  [1 1]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [1 1]
##  [1 1]
##  [0 0]
##  [0 0]
##  [1 0]
##  [1 1]
##  [0 1]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [0 1]
##  [0 0]
##  [1 1]
##  [0 0]
##  [0 0]
##  [0 0]
```

```
##  [0 0]
##  [1 1]
##  [0 0]
##  [0 0]
##  [0 1]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [1 1]
##  [1 1]
##  [1 1]
##  [1 0]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 1]
##  [0 0]
##  [1 1]
##  [0 1]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 1]
##  [0 0]
##  [1 1]
##  [1 1]
##  [1 1]]
```

R

```
y_pred = predict(classifier, newdata = test_set[-3])
```

3.5.9 Making the Confusion Matrix

Python

```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
## [[65  3]
##  [ 7 25]]
accuracy_score(y_test, y_pred)
```

```
## 0.9
```

R

```
cm = table(test_set[, 3], y_pred)
```

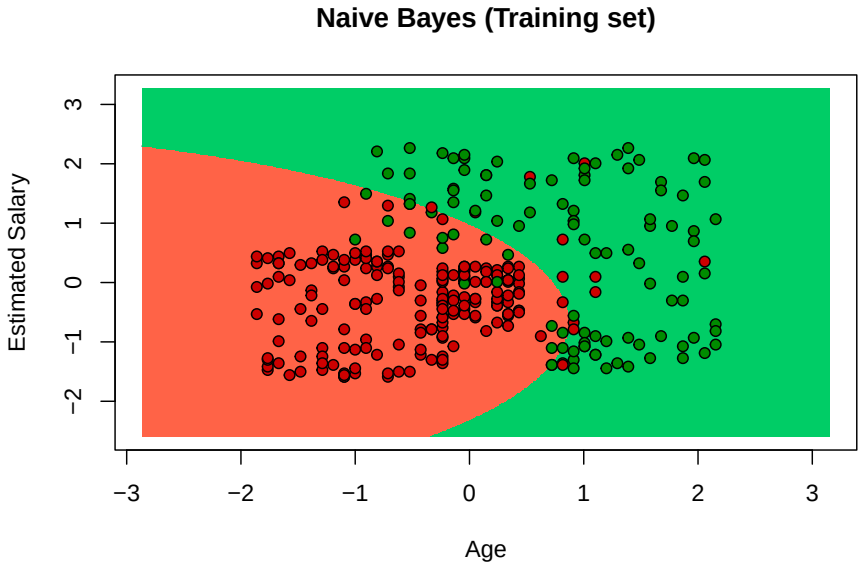
3.5.10 Visualising the Training set results

Python

```
from matplotlib.colors import ListedColormap
X_set, y_set = sc.inverse_transform(X_train), y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 10, stop = X_set[:, 0].max() + 10, ste
                     np.arange(start = X_set[:, 1].min() - 1000, stop = X_set[:, 1].max() + 1000,
plt.contourf(X1, X2, classifier.predict(sc.transform(np.array([X1.ravel(), X2.ravel()]).T)).resha
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], c = ListedColormap(('red', 'green'))(
plt.title('Naive Bayes (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

R

```
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')
y_grid = predict(classifier, newdata = grid_set)
plot(set[, -3],
     main = 'Naive Bayes (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'))
```



3.5.11 Visualising the Test set results

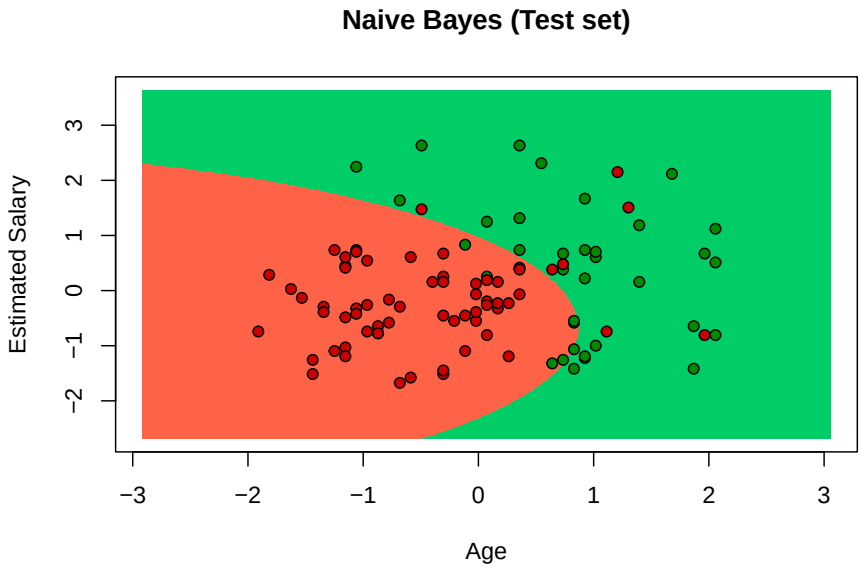
Python

```
from matplotlib.colors import ListedColormap
X_set, y_set = sc.inverse_transform(X_test), y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 10, stop = X_set[:, 0].max()
                     np.arange(start = X_set[:, 1].min() - 1000, stop = X_set[:, 1].ma
plt.contourf(X1, X2, classifier.predict(sc.transform(np.array([X1.ravel(), X2.ravel()]
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], c = ListedColormap(('red',
plt.title('Naive Bayes (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

R

```
library(ElemStatLearn)
set = test_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')
y_grid = predict(classifier, newdata = grid_set)
plot(set[, -3], main = 'Naive Bayes (Test 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'))
```



3.6 Decision Tree Classification

3.6.1 Importing the libraries

Python

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

3.6.2 Importing the dataset

Python

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

R

```
dataset = read.csv('Social_Network_Ads.csv')
# dataset = dataset[3:5]
```

3.6.3 Encoding the target feature as factor

R

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

3.6.4 Splitting the dataset into the Training set and Test set

Python

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_sta
print(X_train)
## [[    44  39000]
##  [    32 120000]
##  [    38  50000]
##  [    32 135000]
##  [    52  21000]
##  [    53 104000]
##  [    39  42000]
##  [    38  61000]
##  [    36  50000]
##  [    36  63000]
##  [    35  25000]
##  [    35  50000]
##  [    42  73000]
##  [    47  49000]
##  [    59  29000]
##  [    49  65000]
##  [    45 131000]
##  [    31  89000]
##  [    46  82000]
##  [    47  51000]
##  [    26  15000]
##  [    60 102000]
##  [    38 112000]
##  [    40 107000]
```

```
##  [    42  53000]
##  [    35  59000]
##  [    48  41000]
##  [    48 134000]
##  [    38 113000]
##  [    29 148000]
##  [    26  15000]
##  [    60  42000]
##  [    24  19000]
##  [    42 149000]
##  [    46  96000]
##  [    28  59000]
##  [    39  96000]
##  [    28  89000]
##  [    41  72000]
##  [    45  26000]
##  [    33  69000]
##  [    20  82000]
##  [    31  74000]
##  [    42  80000]
##  [    35  72000]
##  [    33 149000]
##  [    40  71000]
##  [    51 146000]
##  [    46  79000]
##  [    35  75000]
##  [    38  51000]
##  [    36  75000]
##  [    37  78000]
##  [    38  61000]
##  [    60 108000]
##  [    20  82000]
##  [    57  74000]
##  [    42  65000]
##  [    26  80000]
##  [    46 117000]
##  [    35  61000]
##  [    21  68000]
##  [    28  44000]
##  [    41  87000]
##  [    37  33000]
##  [    27  90000]
##  [    39  42000]
##  [    28 123000]
##  [    31 118000]
```

```
##  [    25  87000]
##  [    35  71000]
##  [    37  70000]
##  [    35  39000]
##  [    47  23000]
##  [    35 147000]
##  [    48 138000]
##  [    26  86000]
##  [    25  79000]
##  [    52 138000]
##  [    51  23000]
##  [    35  60000]
##  [    33 113000]
##  [    30 107000]
##  [    48  33000]
##  [    41  80000]
##  [    48  96000]
##  [    31  18000]
##  [    31  71000]
##  [    43 129000]
##  [    59  76000]
##  [    18  44000]
##  [    36 118000]
##  [    42  90000]
##  [    47  30000]
##  [    26  43000]
##  [    40  78000]
##  [    46  59000]
##  [    59  42000]
##  [    46  74000]
##  [    35  91000]
##  [    28  59000]
##  [    40  57000]
##  [    59 143000]
##  [    57  26000]
##  [    52  38000]
##  [    47 113000]
##  [    53 143000]
##  [    35  27000]
##  [    58 101000]
##  [    45  45000]
##  [    23  82000]
##  [    46  23000]
##  [    42  65000]
##  [    28  84000]
```

```
##  [    38  59000]
##  [    26  84000]
##  [    29  28000]
##  [    37  71000]
##  [    22  55000]
##  [    48  35000]
##  [    49  28000]
##  [    38  65000]
##  [    27  17000]
##  [    46  28000]
##  [    48 141000]
##  [    26  17000]
##  [    35  97000]
##  [    39  59000]
##  [    24  27000]
##  [    32  18000]
##  [    46  88000]
##  [    35  58000]
##  [    56  60000]
##  [    47  34000]
##  [    40  72000]
##  [    32 100000]
##  [    19  21000]
##  [    25  90000]
##  [    35  88000]
##  [    28  32000]
##  [    50  20000]
##  [    40  59000]
##  [    50  44000]
##  [    35  72000]
##  [    40 142000]
##  [    46  32000]
##  [    39  71000]
##  [    20  74000]
##  [    29  75000]
##  [    31  76000]
##  [    47  25000]
##  [    40  61000]
##  [    34 112000]
##  [    38  80000]
##  [    42  75000]
##  [    47  47000]
##  [    39  75000]
##  [    19  25000]
##  [    37  80000]
```

```
##  [    36  60000]
##  [    41  52000]
##  [    36 125000]
##  [    48  29000]
##  [    36 126000]
##  [    51 134000]
##  [    27  57000]
##  [    38  71000]
##  [    39  61000]
##  [    22  27000]
##  [    33  60000]
##  [    48  74000]
##  [    58  23000]
##  [    53  72000]
##  [    32 117000]
##  [    54  70000]
##  [    30  80000]
##  [    58  95000]
##  [    26  52000]
##  [    45  79000]
##  [    24  55000]
##  [    40  75000]
##  [    33  28000]
##  [    44 139000]
##  [    22  18000]
##  [    33  51000]
##  [    43 133000]
##  [    24  32000]
##  [    46  22000]
##  [    35  55000]
##  [    54 104000]
##  [    48 119000]
##  [    35  53000]
##  [    37 144000]
##  [    23  66000]
##  [    37 137000]
##  [    31  58000]
##  [    33  41000]
##  [    45  22000]
##  [    30  15000]
##  [    19  19000]
##  [    49  74000]
##  [    39 122000]
##  [    35  73000]
##  [    39  71000]
```

```
##  [    24  23000]
##  [    41  72000]
##  [    29  83000]
##  [    54  26000]
##  [    35  44000]
##  [    37  75000]
##  [    29  47000]
##  [    31  68000]
##  [    42  54000]
##  [    30 135000]
##  [    52 114000]
##  [    50  36000]
##  [    56 133000]
##  [    29  61000]
##  [    30  89000]
##  [    26  16000]
##  [    33  31000]
##  [    41  72000]
##  [    36  33000]
##  [    55 125000]
##  [    48 131000]
##  [    41  71000]
##  [    30  62000]
##  [    37  72000]
##  [    41  63000]
##  [    58  47000]
##  [    30 116000]
##  [    20  49000]
##  [    37  74000]
##  [    41  59000]
##  [    49  89000]
##  [    28  79000]
##  [    53  82000]
##  [    40  57000]
##  [    60  34000]
##  [    35 108000]
##  [    21  72000]
##  [    38  71000]
##  [    39 106000]
##  [    37  57000]
##  [    26  72000]
##  [    35  23000]
##  [    54 108000]
##  [    30  17000]
##  [    39 134000]
```

```
##  [    29  43000]
##  [    33  43000]
##  [    35  38000]
##  [    41  45000]
##  [    41  72000]
##  [    39 134000]
##  [    27 137000]
##  [    21  16000]
##  [    26  32000]
##  [    31  66000]
##  [    39  73000]
##  [    41  79000]
##  [    47  50000]
##  [    41  30000]
##  [    37  93000]
##  [    60  46000]
##  [    25  22000]
##  [    28  37000]
##  [    38  55000]
##  [    36  54000]
##  [    20  36000]
##  [    56 104000]
##  [    40  57000]
##  [    42 108000]
##  [    20  23000]
##  [    40  65000]
##  [    47  20000]
##  [    18  86000]
##  [    35  79000]
##  [    57  33000]
##  [    34  72000]
##  [    49  39000]
##  [    27  31000]
##  [    19  70000]
##  [    39  79000]
##  [    26  81000]
##  [    25  80000]
##  [    28  85000]
##  [    55  39000]
##  [    50  88000]
##  [    49  88000]
##  [    52 150000]
##  [    35  65000]
##  [    42  54000]
##  [    34  43000]
```

```
##  [    37  52000]
##  [    48  30000]
##  [    29  43000]
##  [    36  52000]
##  [    27  54000]
##  [    26 118000]]
print(y_train)
## [0 1 0 1 1 1 0 0 0 0 0 0 1 1 1 0 1 0 0 1 0 1 0 1 0 0 1 1 1 1 0 1 0 1 0 0 1
##  0 0 1 0 0 0 0 0 1 1 1 1 0 0 0 1 0 1 0 1 0 0 1 0 0 0 1 0 0 0 1 1 0 0 1 0 1
##  1 1 0 0 1 1 0 0 1 1 0 1 0 0 1 1 0 1 1 1 0 0 0 0 0 1 0 0 1 1 1 1 1 0 1 1 0
##  1 0 0 0 0 0 0 0 1 1 0 0 1 0 0 1 0 0 0 1 0 1 1 0 1 0 0 0 0 1 0 0 0 1 1 0 0
##  0 0 1 0 1 0 0 0 1 0 0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 1 1 0 1 0 0 0 0 0 1 0 0
##  0 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 1 1 0 0 0 0 0
##  0 1 1 0 0 0 0 1 0 0 0 0 1 0 1 0 1 0 0 0 1 0 0 0 1 0 1 0 0 0 0 0 1 1 0 0 0
##  0 0 1 0 1 1 0 0 0 0 0 1 0 1 0 0 1 0 0 1 0 1 0 0 0 0 0 0 1 1 1 1 0 0 0 0 1
##  0 0 0 0]
print(X_test)
## [[    30  87000]
##  [    38  50000]
##  [    35  75000]
##  [    30  79000]
##  [    35  50000]
##  [    27  20000]
##  [    31  15000]
##  [    36 144000]
##  [    18  68000]
##  [    47  43000]
##  [    30  49000]
##  [    28  55000]
##  [    37  55000]
##  [    39  77000]
##  [    20  86000]
##  [    32 117000]
##  [    37  77000]
##  [    19  85000]
##  [    55 130000]
##  [    35  22000]
##  [    35  47000]
##  [    47 144000]
##  [    41  51000]
##  [    47 105000]
##  [    23  28000]
##  [    49 141000]
##  [    28  87000]
##  [    29  80000]
```

```
##  [    37  62000]
##  [    32  86000]
##  [    21  88000]
##  [    37  79000]
##  [    57  60000]
##  [    37  53000]
##  [    24  58000]
##  [    18  52000]
##  [    22  81000]
##  [    34  43000]
##  [    31  34000]
##  [    49  36000]
##  [    27  88000]
##  [    41  52000]
##  [    27  84000]
##  [    35  20000]
##  [    43 112000]
##  [    27  58000]
##  [    37  80000]
##  [    52  90000]
##  [    26  30000]
##  [    49  86000]
##  [    57 122000]
##  [    34  25000]
##  [    35  57000]
##  [    34 115000]
##  [    59  88000]
##  [    45  32000]
##  [    29  83000]
##  [    26  80000]
##  [    49  28000]
##  [    23  20000]
##  [    32  18000]
##  [    60  42000]
##  [    19  76000]
##  [    36  99000]
##  [    19  26000]
##  [    60  83000]
##  [    24  89000]
##  [    27  58000]
##  [    40  47000]
##  [    42  70000]
##  [    32 150000]
##  [    35  77000]
##  [    22  63000]
```

```
##  [    45  22000]
##  [    27  89000]
##  [    18  82000]
##  [    42  79000]
##  [    40  60000]
##  [    53  34000]
##  [    47 107000]
##  [    58 144000]
##  [    59  83000]
##  [    24  55000]
##  [    26  35000]
##  [    58  38000]
##  [    42  80000]
##  [    40  75000]
##  [    59 130000]
##  [    46  41000]
##  [    41  60000]
##  [    42  64000]
##  [    37 146000]
##  [    23  48000]
##  [    25  33000]
##  [    24  84000]
##  [    27  96000]
##  [    23  63000]
##  [    48  33000]
##  [    48  90000]
##  [    42 104000]]
print(y_test)
## [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 1 0 0 0 0 0 1 1 0 0 0 0
##  0 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 0 1 1 0 0 1 0 0 1 0 1 0 1 0 0 0 0 1 0 0 1
##  0 0 0 0 1 1 1 0 0 0 1 1 0 1 1 0 0 1 0 0 0 1 0 1 1 1]
```

R

```
# 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)
```

3.6.5 Feature Scaling

Python

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
print(X_train)
## [[ 0.58 -0.89]
##  [-0.61  1.46]
##  [-0.01 -0.57]
##  [-0.61  1.9 ]
##  [ 1.37 -1.41]
##  [ 1.47  1.  ]
##  [ 0.09 -0.8 ]
##  [-0.01 -0.25]
##  [-0.21 -0.57]
##  [-0.21 -0.19]
##  [-0.31 -1.29]
##  [-0.31 -0.57]
##  [ 0.38  0.1 ]
##  [ 0.88 -0.6 ]
##  [ 2.07 -1.18]
##  [ 1.08 -0.13]
##  [ 0.68  1.78]
##  [-0.71  0.56]
##  [ 0.78  0.36]
##  [ 0.88 -0.54]
##  [-1.2  -1.58]
##  [ 2.17  0.94]
##  [-0.01  1.23]
##  [ 0.19  1.08]
##  [ 0.38 -0.48]
##  [-0.31 -0.31]
##  [ 0.98 -0.83]
##  [ 0.98  1.87]
##  [-0.01  1.26]
##  [-0.9   2.27]
##  [-1.2  -1.58]
##  [ 2.17 -0.8 ]
##  [-1.4  -1.47]
##  [ 0.38  2.3 ]
##  [ 0.78  0.77]
##  [-1.   -0.31]
##  [ 0.09  0.77]
##  [-1.    0.56]
##  [ 0.28  0.07]
##  [ 0.68 -1.26]
```

```
##  [-0.51 -0.02]
##  [-1.8   0.36]
##  [-0.71  0.13]
##  [ 0.38  0.3 ]
##  [-0.31  0.07]
##  [-0.51  2.3 ]
##  [ 0.19  0.04]
##  [ 1.27  2.22]
##  [ 0.78  0.27]
##  [-0.31  0.16]
##  [-0.01 -0.54]
##  [-0.21  0.16]
##  [-0.11  0.24]
##  [-0.01 -0.25]
##  [ 2.17  1.11]
##  [-1.8   0.36]
##  [ 1.87  0.13]
##  [ 0.38 -0.13]
##  [-1.2   0.3 ]
##  [ 0.78  1.37]
##  [-0.31 -0.25]
##  [-1.7  -0.05]
##  [-1.   -0.74]
##  [ 0.28  0.5 ]
##  [-0.11 -1.06]
##  [-1.1   0.59]
##  [ 0.09 -0.8 ]
##  [-1.    1.55]
##  [-0.71  1.4 ]
##  [-1.3   0.5 ]
##  [-0.31  0.04]
##  [-0.11  0.01]
##  [-0.31 -0.89]
##  [ 0.88 -1.35]
##  [-0.31  2.24]
##  [ 0.98  1.98]
##  [-1.2   0.48]
##  [-1.3   0.27]
##  [ 1.37  1.98]
##  [ 1.27 -1.35]
##  [-0.31 -0.28]
##  [-0.51  1.26]
##  [-0.8   1.08]
##  [ 0.98 -1.06]
##  [ 0.28  0.3 ]
```

```
##  [ 0.98  0.77]
##  [-0.71 -1.5 ]
##  [-0.71  0.04]
##  [ 0.48  1.72]
##  [ 2.07  0.19]
##  [-1.99 -0.74]
##  [-0.21  1.4 ]
##  [ 0.38  0.59]
##  [ 0.88 -1.15]
##  [-1.2  -0.77]
##  [ 0.19  0.24]
##  [ 0.78 -0.31]
##  [ 2.07 -0.8 ]
##  [ 0.78  0.13]
##  [-0.31  0.62]
##  [-1.   -0.31]
##  [ 0.19 -0.36]
##  [ 2.07  2.13]
##  [ 1.87 -1.26]
##  [ 1.37 -0.92]
##  [ 0.88  1.26]
##  [ 1.47  2.13]
##  [-0.31 -1.23]
##  [ 1.97  0.91]
##  [ 0.68 -0.71]
##  [-1.5   0.36]
##  [ 0.78 -1.35]
##  [ 0.38 -0.13]
##  [-1.    0.42]
##  [-0.01 -0.31]
##  [-1.2   0.42]
##  [-0.9  -1.21]
##  [-0.11  0.04]
##  [-1.6  -0.42]
##  [ 0.98 -1.  ]
##  [ 1.08 -1.21]
##  [-0.01 -0.13]
##  [-1.1  -1.52]
##  [ 0.78 -1.21]
##  [ 0.98  2.07]
##  [-1.2  -1.52]
##  [-0.31  0.79]
##  [ 0.09 -0.31]
##  [-1.4  -1.23]
##  [-0.61 -1.5 ]
```

```
##  [ 0.78  0.53]
##  [-0.31 -0.34]
##  [ 1.77 -0.28]
##  [ 0.88 -1.03]
##  [ 0.19  0.07]
##  [-0.61  0.88]
##  [-1.89 -1.41]
##  [-1.3   0.59]
##  [-0.31  0.53]
##  [-1.   -1.09]
##  [ 1.18 -1.44]
##  [ 0.19 -0.31]
##  [ 1.18 -0.74]
##  [-0.31  0.07]
##  [ 0.19  2.1 ]
##  [ 0.78 -1.09]
##  [ 0.09  0.04]
##  [-1.8   0.13]
##  [-0.9   0.16]
##  [-0.71  0.19]
##  [ 0.88 -1.29]
##  [ 0.19 -0.25]
##  [-0.41  1.23]
##  [-0.01  0.3 ]
##  [ 0.38  0.16]
##  [ 0.88 -0.65]
##  [ 0.09  0.16]
##  [-1.89 -1.29]
##  [-0.11  0.3 ]
##  [-0.21 -0.28]
##  [ 0.28 -0.51]
##  [-0.21  1.61]
##  [ 0.98 -1.18]
##  [-0.21  1.64]
##  [ 1.27  1.87]
##  [-1.1  -0.36]
##  [-0.01  0.04]
##  [ 0.09 -0.25]
##  [-1.6  -1.23]
##  [-0.51 -0.28]
##  [ 0.98  0.13]
##  [ 1.97 -1.35]
##  [ 1.47  0.07]
##  [-0.61  1.37]
##  [ 1.57  0.01]
```

```
##  [-0.8   0.3 ]
##  [ 1.97  0.74]
##  [-1.2  -0.51]
##  [ 0.68  0.27]
##  [-1.4  -0.42]
##  [ 0.19  0.16]
##  [-0.51 -1.21]
##  [ 0.58  2.01]
##  [-1.6  -1.5 ]
##  [-0.51 -0.54]
##  [ 0.48  1.84]
##  [-1.4  -1.09]
##  [ 0.78 -1.38]
##  [-0.31 -0.42]
##  [ 1.57  1.  ]
##  [ 0.98  1.43]
##  [-0.31 -0.48]
##  [-0.11  2.16]
##  [-1.5  -0.1 ]
##  [-0.11  1.95]
##  [-0.71 -0.34]
##  [-0.51 -0.83]
##  [ 0.68 -1.38]
##  [-0.8  -1.58]
##  [-1.89 -1.47]
##  [ 1.08  0.13]
##  [ 0.09  1.52]
##  [-0.31  0.1 ]
##  [ 0.09  0.04]
##  [-1.4  -1.35]
##  [ 0.28  0.07]
##  [-0.9   0.39]
##  [ 1.57 -1.26]
##  [-0.31 -0.74]
##  [-0.11  0.16]
##  [-0.9  -0.65]
##  [-0.71 -0.05]
##  [ 0.38 -0.45]
##  [-0.8   1.9 ]
##  [ 1.37  1.29]
##  [ 1.18 -0.97]
##  [ 1.77  1.84]
##  [-0.9  -0.25]
##  [-0.8   0.56]
##  [-1.2  -1.55]
```

```
##  [-0.51 -1.12]
##  [ 0.28  0.07]
##  [-0.21 -1.06]
##  [ 1.67  1.61]
##  [ 0.98  1.78]
##  [ 0.28  0.04]
##  [-0.8  -0.22]
##  [-0.11  0.07]
##  [ 0.28 -0.19]
##  [ 1.97 -0.65]
##  [-0.8   1.35]
##  [-1.8  -0.6 ]
##  [-0.11  0.13]
##  [ 0.28 -0.31]
##  [ 1.08  0.56]
##  [-1.    0.27]
##  [ 1.47  0.36]
##  [ 0.19 -0.36]
##  [ 2.17 -1.03]
##  [-0.31  1.11]
##  [-1.7   0.07]
##  [-0.01  0.04]
##  [ 0.09  1.06]
##  [-0.11 -0.36]
##  [-1.2   0.07]
##  [-0.31 -1.35]
##  [ 1.57  1.11]
##  [-0.8  -1.52]
##  [ 0.09  1.87]
##  [-0.9  -0.77]
##  [-0.51 -0.77]
##  [-0.31 -0.92]
##  [ 0.28 -0.71]
##  [ 0.28  0.07]
##  [ 0.09  1.87]
##  [-1.1   1.95]
##  [-1.7  -1.55]
##  [-1.2  -1.09]
##  [-0.71 -0.1 ]
##  [ 0.09  0.1 ]
##  [ 0.28  0.27]
##  [ 0.88 -0.57]
##  [ 0.28 -1.15]
##  [-0.11  0.68]
##  [ 2.17 -0.68]
```

```
##  [-1.3  -1.38]
##  [-1.   -0.94]
##  [-0.01 -0.42]
##  [-0.21 -0.45]
##  [-1.8  -0.97]
##  [ 1.77  1.  ]
##  [ 0.19 -0.36]
##  [ 0.38  1.11]
##  [-1.8  -1.35]
##  [ 0.19 -0.13]
##  [ 0.88 -1.44]
##  [-1.99  0.48]
##  [-0.31  0.27]
##  [ 1.87 -1.06]
##  [-0.41  0.07]
##  [ 1.08 -0.89]
##  [-1.1  -1.12]
##  [-1.89  0.01]
##  [ 0.09  0.27]
##  [-1.2   0.33]
##  [-1.3   0.3 ]
##  [-1.    0.45]
##  [ 1.67 -0.89]
##  [ 1.18  0.53]
##  [ 1.08  0.53]
##  [ 1.37  2.33]
##  [-0.31 -0.13]
##  [ 0.38 -0.45]
##  [-0.41 -0.77]
##  [-0.11 -0.51]
##  [ 0.98 -1.15]
##  [-0.9  -0.77]
##  [-0.21 -0.51]
##  [-1.1  -0.45]
##  [-1.2   1.4 ]]
print(X_test)
## [[-0.8   0.5 ]
##  [-0.01 -0.57]
##  [-0.31  0.16]
##  [-0.8   0.27]
##  [-0.31 -0.57]
##  [-1.1  -1.44]
##  [-0.71 -1.58]
##  [-0.21  2.16]
##  [-1.99 -0.05]
```

```
##  [ 0.88 -0.77]
##  [-0.8  -0.6 ]
##  [-1.   -0.42]
##  [-0.11 -0.42]
##  [ 0.09  0.22]
##  [-1.8   0.48]
##  [-0.61  1.37]
##  [-0.11  0.22]
##  [-1.89  0.45]
##  [ 1.67  1.75]
##  [-0.31 -1.38]
##  [-0.31 -0.65]
##  [ 0.88  2.16]
##  [ 0.28 -0.54]
##  [ 0.88  1.03]
##  [-1.5  -1.21]
##  [ 1.08  2.07]
##  [-1.    0.5 ]
##  [-0.9   0.3 ]
##  [-0.11 -0.22]
##  [-0.61  0.48]
##  [-1.7   0.53]
##  [-0.11  0.27]
##  [ 1.87 -0.28]
##  [-0.11 -0.48]
##  [-1.4  -0.34]
##  [-1.99 -0.51]
##  [-1.6   0.33]
##  [-0.41 -0.77]
##  [-0.71 -1.03]
##  [ 1.08 -0.97]
##  [-1.1   0.53]
##  [ 0.28 -0.51]
##  [-1.1   0.42]
##  [-0.31 -1.44]
##  [ 0.48  1.23]
##  [-1.1  -0.34]
##  [-0.11  0.3 ]
##  [ 1.37  0.59]
##  [-1.2  -1.15]
##  [ 1.08  0.48]
##  [ 1.87  1.52]
##  [-0.41 -1.29]
##  [-0.31 -0.36]
##  [-0.41  1.32]
```

```
##  [ 2.07  0.53]
##  [ 0.68 -1.09]
##  [-0.9   0.39]
##  [-1.2   0.3 ]
##  [ 1.08 -1.21]
##  [-1.5  -1.44]
##  [-0.61 -1.5 ]
##  [ 2.17 -0.8 ]
##  [-1.89  0.19]
##  [-0.21  0.85]
##  [-1.89 -1.26]
##  [ 2.17  0.39]
##  [-1.4   0.56]
##  [-1.1  -0.34]
##  [ 0.19 -0.65]
##  [ 0.38  0.01]
##  [-0.61  2.33]
##  [-0.31  0.22]
##  [-1.6  -0.19]
##  [ 0.68 -1.38]
##  [-1.1   0.56]
##  [-1.99  0.36]
##  [ 0.38  0.27]
##  [ 0.19 -0.28]
##  [ 1.47 -1.03]
##  [ 0.88  1.08]
##  [ 1.97  2.16]
##  [ 2.07  0.39]
##  [-1.4  -0.42]
##  [-1.2  -1.  ]
##  [ 1.97 -0.92]
##  [ 0.38  0.3 ]
##  [ 0.19  0.16]
##  [ 2.07  1.75]
##  [ 0.78 -0.83]
##  [ 0.28 -0.28]
##  [ 0.38 -0.16]
##  [-0.11  2.22]
##  [-1.5  -0.63]
##  [-1.3  -1.06]
##  [-1.4   0.42]
##  [-1.1   0.77]
##  [-1.5  -0.19]
##  [ 0.98 -1.06]
##  [ 0.98  0.59]
```

```
##  [ 0.38  1.  ]]
```

R

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

3.6.6 Training the Decision Tree Classification model on the Training set

Python

```
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
classifier.fit(X_train, y_train)
## DecisionTreeClassifier(criterion='entropy', random_state=0)
```

R

```
# install.packages('rpart')
library(rpart)
classifier = rpart(formula = Purchased ~ .,
                   data = training_set)
```

3.6.7 Predicting a new result

Python

```
print(classifier.predict(sc.transform([[30,87000]])))
## [0]
```

3.6.8 Predicting the Test set results

Python

```
y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))
## [[0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [0 0]
##  [0 0]
```

```
##  [0 0]
##  [0 0]
##  [1 0]
##  [0 0]
##  [1 0]
##  [1 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [1 1]
##  [0 0]
##  [0 1]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 1]
##  [1 1]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [1 1]
##  [1 1]
##  [0 0]
##  [0 0]
##  [1 0]
##  [1 1]
##  [1 1]
```

```
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [1 1]
##  [0 0]
##  [1 1]
##  [0 0]
##  [0 0]
##  [0 0]
##  [1 0]
##  [1 1]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [1 1]
##  [1 1]
##  [1 1]
##  [1 0]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 1]
##  [0 0]
##  [1 1]
##  [1 1]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [1 1]
##  [1 1]
##  [1 1]]
```

R

```
y_pred = predict(classifier, newdata = test_set[-3], type = 'class')
```

3.6.9 Making the Confusion Matrix

Python

```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
## [[62  6]
##  [ 3 29]]
accuracy_score(y_test, y_pred)
## 0.91
```

R

```
cm = table(test_set[, 3], y_pred)
```

3.6.10 Visualising the Training set results

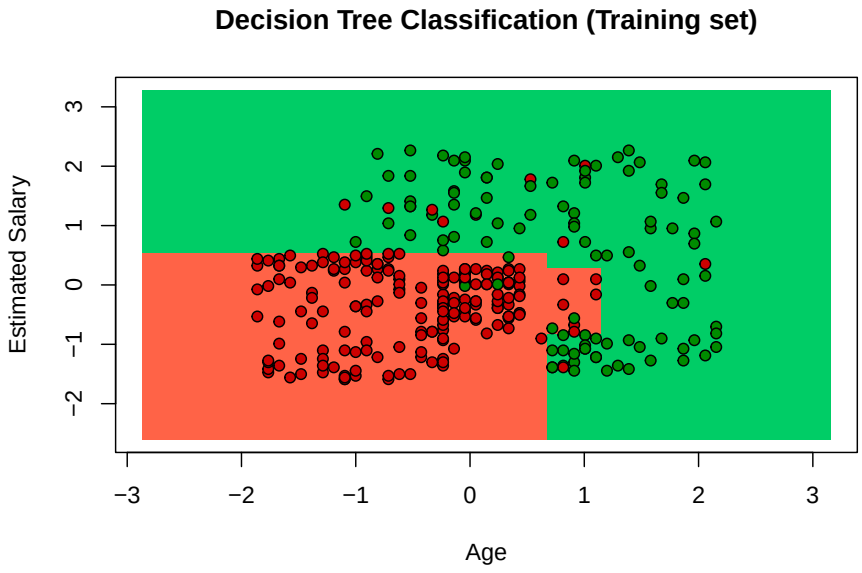
Python

```
from matplotlib.colors import ListedColormap
X_set, y_set = sc.inverse_transform(X_train), y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 10, stop = X_set[:, 0].max()
                     np.arange(start = X_set[:, 1].min() - 1000, stop = X_set[:, 1].ma
plt.contourf(X1, X2, classifier.predict(sc.transform(np.array([X1.ravel(), X2.ravel()]
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], c = ListedColormap(('red',
plt.title('Decision Tree Classification (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

R

```
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')
y_grid = predict(classifier, newdata = grid_set, type = 'class')
```

```
plot(set[, -3],
     main = 'Decision Tree Classification (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'))
```



3.6.11 Visualising the Test set results

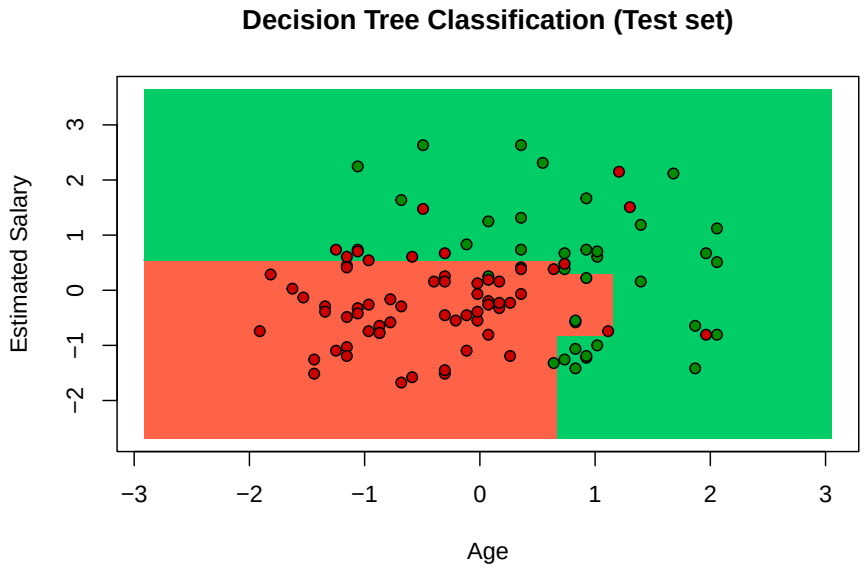
Python

```
from matplotlib.colors import ListedColormap
X_set, y_set = sc.inverse_transform(X_test), y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 10, stop = X_set[:, 0].max() + 10, ste
                     np.arange(start = X_set[:, 1].min() - 1000, stop = X_set[:, 1].max() + 1000,
plt.contourf(X1, X2, classifier.predict(sc.transform(np.array([X1.ravel(), X2.ravel()]).T)).resha
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], c = ListedColormap(('red', 'green'))(
plt.title('Decision Tree Classification (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
```

```
plt.legend()
plt.show()
```

R

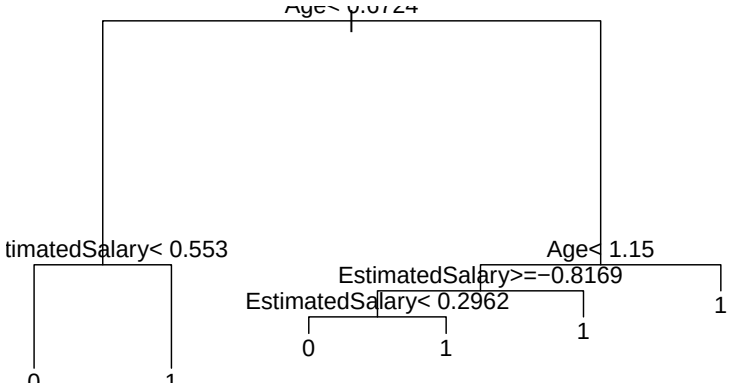
```
library(ElemStatLearn)
set = test_set
X1 = seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)
X2 = seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01)
grid_set = expand.grid(X1, X2)
colnames(grid_set) = c('Age', 'EstimatedSalary')
y_grid = predict(classifier, newdata = grid_set, type = 'class')
plot(set[, -3], main = 'Decision Tree Classification (Test 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'))
```



3.6.12 Plotting the tree

R

```
plot(classifier)
text(classifier)
```



3.7 Random Forest Classification

3.7.1 Importing the libraries

Python

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

3.7.2 Importing the dataset

Python

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

R

```
dataset = read.csv('Social_Network_Ads.csv')
# dataset = dataset[3:5]
```

3.7.3 Encoding the target feature as factor

R

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

3.7.4 Splitting the dataset into the Training set and Test set

Python

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_sta
print(X_train)
## [[    44  39000]
##  [    32 120000]
##  [    38  50000]
##  [    32 135000]
##  [    52  21000]
##  [    53 104000]
##  [    39  42000]
##  [    38  61000]
##  [    36  50000]
##  [    36  63000]
##  [    35  25000]
##  [    35  50000]
##  [    42  73000]
##  [    47  49000]
##  [    59  29000]
##  [    49  65000]
##  [    45 131000]
##  [    31  89000]
##  [    46  82000]
##  [    47  51000]
##  [    26  15000]
##  [    60 102000]
##  [    38 112000]
##  [    40 107000]
##  [    42  53000]
##  [    35  59000]
##  [    48  41000]
##  [    48 134000]
##  [    38 113000]
##  [    29 148000]
##  [    26  15000]
##  [    60  42000]
##  [    24  19000]
##  [    42 149000]
##  [    46  96000]
```

```
##  [    28  59000]
##  [    39  96000]
##  [    28  89000]
##  [    41  72000]
##  [    45  26000]
##  [    33  69000]
##  [    20  82000]
##  [    31  74000]
##  [    42  80000]
##  [    35  72000]
##  [    33 149000]
##  [    40  71000]
##  [    51 146000]
##  [    46  79000]
##  [    35  75000]
##  [    38  51000]
##  [    36  75000]
##  [    37  78000]
##  [    38  61000]
##  [    60 108000]
##  [    20  82000]
##  [    57  74000]
##  [    42  65000]
##  [    26  80000]
##  [    46 117000]
##  [    35  61000]
##  [    21  68000]
##  [    28  44000]
##  [    41  87000]
##  [    37  33000]
##  [    27  90000]
##  [    39  42000]
##  [    28 123000]
##  [    31 118000]
##  [    25  87000]
##  [    35  71000]
##  [    37  70000]
##  [    35  39000]
##  [    47  23000]
##  [    35 147000]
##  [    48 138000]
##  [    26  86000]
##  [    25  79000]
##  [    52 138000]
##  [    51  23000]
```

```
##  [    35  60000]
##  [    33 113000]
##  [    30 107000]
##  [    48  33000]
##  [    41  80000]
##  [    48  96000]
##  [    31  18000]
##  [    31  71000]
##  [    43 129000]
##  [    59  76000]
##  [    18  44000]
##  [    36 118000]
##  [    42  90000]
##  [    47  30000]
##  [    26  43000]
##  [    40  78000]
##  [    46  59000]
##  [    59  42000]
##  [    46  74000]
##  [    35  91000]
##  [    28  59000]
##  [    40  57000]
##  [    59 143000]
##  [    57  26000]
##  [    52  38000]
##  [    47 113000]
##  [    53 143000]
##  [    35  27000]
##  [    58 101000]
##  [    45  45000]
##  [    23  82000]
##  [    46  23000]
##  [    42  65000]
##  [    28  84000]
##  [    38  59000]
##  [    26  84000]
##  [    29  28000]
##  [    37  71000]
##  [    22  55000]
##  [    48  35000]
##  [    49  28000]
##  [    38  65000]
##  [    27  17000]
##  [    46  28000]
##  [    48 141000]
```

```
##  [    26  17000]
##  [    35  97000]
##  [    39  59000]
##  [    24  27000]
##  [    32  18000]
##  [    46  88000]
##  [    35  58000]
##  [    56  60000]
##  [    47  34000]
##  [    40  72000]
##  [    32 100000]
##  [    19  21000]
##  [    25  90000]
##  [    35  88000]
##  [    28  32000]
##  [    50  20000]
##  [    40  59000]
##  [    50  44000]
##  [    35  72000]
##  [    40 142000]
##  [    46  32000]
##  [    39  71000]
##  [    20  74000]
##  [    29  75000]
##  [    31  76000]
##  [    47  25000]
##  [    40  61000]
##  [    34 112000]
##  [    38  80000]
##  [    42  75000]
##  [    47  47000]
##  [    39  75000]
##  [    19  25000]
##  [    37  80000]
##  [    36  60000]
##  [    41  52000]
##  [    36 125000]
##  [    48  29000]
##  [    36 126000]
##  [    51 134000]
##  [    27  57000]
##  [    38  71000]
##  [    39  61000]
##  [    22  27000]
##  [    33  60000]
```

```
##  [    48  74000]
##  [    58  23000]
##  [    53  72000]
##  [    32 117000]
##  [    54  70000]
##  [    30  80000]
##  [    58  95000]
##  [    26  52000]
##  [    45  79000]
##  [    24  55000]
##  [    40  75000]
##  [    33  28000]
##  [    44 139000]
##  [    22  18000]
##  [    33  51000]
##  [    43 133000]
##  [    24  32000]
##  [    46  22000]
##  [    35  55000]
##  [    54 104000]
##  [    48 119000]
##  [    35  53000]
##  [    37 144000]
##  [    23  66000]
##  [    37 137000]
##  [    31  58000]
##  [    33  41000]
##  [    45  22000]
##  [    30  15000]
##  [    19  19000]
##  [    49  74000]
##  [    39 122000]
##  [    35  73000]
##  [    39  71000]
##  [    24  23000]
##  [    41  72000]
##  [    29  83000]
##  [    54  26000]
##  [    35  44000]
##  [    37  75000]
##  [    29  47000]
##  [    31  68000]
##  [    42  54000]
##  [    30 135000]
##  [    52 114000]
```

```
##  [    50  36000]
##  [    56 133000]
##  [    29  61000]
##  [    30  89000]
##  [    26  16000]
##  [    33  31000]
##  [    41  72000]
##  [    36  33000]
##  [    55 125000]
##  [    48 131000]
##  [    41  71000]
##  [    30  62000]
##  [    37  72000]
##  [    41  63000]
##  [    58  47000]
##  [    30 116000]
##  [    20  49000]
##  [    37  74000]
##  [    41  59000]
##  [    49  89000]
##  [    28  79000]
##  [    53  82000]
##  [    40  57000]
##  [    60  34000]
##  [    35 108000]
##  [    21  72000]
##  [    38  71000]
##  [    39 106000]
##  [    37  57000]
##  [    26  72000]
##  [    35  23000]
##  [    54 108000]
##  [    30  17000]
##  [    39 134000]
##  [    29  43000]
##  [    33  43000]
##  [    35  38000]
##  [    41  45000]
##  [    41  72000]
##  [    39 134000]
##  [    27 137000]
##  [    21  16000]
##  [    26  32000]
##  [    31  66000]
##  [    39  73000]
```

```
##  [    41  79000]
##  [    47  50000]
##  [    41  30000]
##  [    37  93000]
##  [    60  46000]
##  [    25  22000]
##  [    28  37000]
##  [    38  55000]
##  [    36  54000]
##  [    20  36000]
##  [    56 104000]
##  [    40  57000]
##  [    42 108000]
##  [    20  23000]
##  [    40  65000]
##  [    47  20000]
##  [    18  86000]
##  [    35  79000]
##  [    57  33000]
##  [    34  72000]
##  [    49  39000]
##  [    27  31000]
##  [    19  70000]
##  [    39  79000]
##  [    26  81000]
##  [    25  80000]
##  [    28  85000]
##  [    55  39000]
##  [    50  88000]
##  [    49  88000]
##  [    52 150000]
##  [    35  65000]
##  [    42  54000]
##  [    34  43000]
##  [    37  52000]
##  [    48  30000]
##  [    29  43000]
##  [    36  52000]
##  [    27  54000]
##  [    26 118000]]
print(y_train)
## [0 1 0 1 1 1 0 0 0 0 0 0 1 1 1 0 1 0 0 1 0 1 0 1 0 0 1 1 1 1 0 1 0 1 0 0 1
##  0 0 1 0 0 0 0 0 1 1 1 1 0 0 0 1 0 1 0 1 0 0 1 0 0 0 1 0 0 0 1 1 0 0 1 0 1
##  1 1 0 0 1 1 0 0 1 1 0 1 0 0 1 1 0 1 1 1 0 0 0 0 0 1 0 0 1 1 1 1 1 0 1 1 0
##  1 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 1 0 0 0 1 0 1 1 0 1 0 0 0 0 1 0 0 0 1 1 0 0
```

```
##  0 0 1 0 1 0 0 0 1 0 0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 1 1 0 1 0 0 0 0 0 1 0 0
##  0 0 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 1 0 1 1 0 0 0 0 0
##  0 1 1 0 0 0 0 0 1 0 0 0 0 0 1 0 1 0 1 0 0 0 1 0 0 0 1 0 1 0 0 0 0 0 1 1 0 0 0
##  0 0 1 0 1 1 0 0 0 0 0 0 1 0 1 0 0 1 0 0 1 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 1
##  0 0 0 0]
print(X_test)
## [[    30  87000]
##  [    38  50000]
##  [    35  75000]
##  [    30  79000]
##  [    35  50000]
##  [    27  20000]
##  [    31  15000]
##  [    36 144000]
##  [    18  68000]
##  [    47  43000]
##  [    30  49000]
##  [    28  55000]
##  [    37  55000]
##  [    39  77000]
##  [    20  86000]
##  [    32 117000]
##  [    37  77000]
##  [    19  85000]
##  [    55 130000]
##  [    35  22000]
##  [    35  47000]
##  [    47 144000]
##  [    41  51000]
##  [    47 105000]
##  [    23  28000]
##  [    49 141000]
##  [    28  87000]
##  [    29  80000]
##  [    37  62000]
##  [    32  86000]
##  [    21  88000]
##  [    37  79000]
##  [    57  60000]
##  [    37  53000]
##  [    24  58000]
##  [    18  52000]
##  [    22  81000]
##  [    34  43000]
##  [    31  34000]
```

```
##  [    49  36000]
##  [    27  88000]
##  [    41  52000]
##  [    27  84000]
##  [    35  20000]
##  [    43 112000]
##  [    27  58000]
##  [    37  80000]
##  [    52  90000]
##  [    26  30000]
##  [    49  86000]
##  [    57 122000]
##  [    34  25000]
##  [    35  57000]
##  [    34 115000]
##  [    59  88000]
##  [    45  32000]
##  [    29  83000]
##  [    26  80000]
##  [    49  28000]
##  [    23  20000]
##  [    32  18000]
##  [    60  42000]
##  [    19  76000]
##  [    36  99000]
##  [    19  26000]
##  [    60  83000]
##  [    24  89000]
##  [    27  58000]
##  [    40  47000]
##  [    42  70000]
##  [    32 150000]
##  [    35  77000]
##  [    22  63000]
##  [    45  22000]
##  [    27  89000]
##  [    18  82000]
##  [    42  79000]
##  [    40  60000]
##  [    53  34000]
##  [    47 107000]
##  [    58 144000]
##  [    59  83000]
##  [    24  55000]
##  [    26  35000]
```

```
##  [    58  38000]
##  [    42  80000]
##  [    40  75000]
##  [    59 130000]
##  [    46  41000]
##  [    41  60000]
##  [    42  64000]
##  [    37 146000]
##  [    23  48000]
##  [    25  33000]
##  [    24  84000]
##  [    27  96000]
##  [    23  63000]
##  [    48  33000]
##  [    48  90000]
##  [    42 104000]]
print(y_test)
## [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 1 0 0 0 0 0 1 1 0 0 0 0
##  0 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 0 1 1 0 0 1 0 0 1 0 1 0 1 0 0 0 0 1 0 0 1
##  0 0 0 0 1 1 1 0 0 0 1 1 0 1 1 0 0 1 0 0 0 1 0 1 1 1]
```

R

```
# 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)
```

3.7.5 Feature Scaling

Python

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
print(X_train)
## [[ 0.58 -0.89]
##  [-0.61  1.46]
##  [-0.01 -0.57]
##  [-0.61  1.9 ]
##  [ 1.37 -1.41]
##  [ 1.47  1.  ]
##  [ 0.09 -0.8 ]
```

```
##  [-0.01 -0.25]
##  [-0.21 -0.57]
##  [-0.21 -0.19]
##  [-0.31 -1.29]
##  [-0.31 -0.57]
##  [ 0.38  0.1 ]
##  [ 0.88 -0.6 ]
##  [ 2.07 -1.18]
##  [ 1.08 -0.13]
##  [ 0.68  1.78]
##  [-0.71  0.56]
##  [ 0.78  0.36]
##  [ 0.88 -0.54]
##  [-1.2  -1.58]
##  [ 2.17  0.94]
##  [-0.01  1.23]
##  [ 0.19  1.08]
##  [ 0.38 -0.48]
##  [-0.31 -0.31]
##  [ 0.98 -0.83]
##  [ 0.98  1.87]
##  [-0.01  1.26]
##  [-0.9   2.27]
##  [-1.2  -1.58]
##  [ 2.17 -0.8 ]
##  [-1.4  -1.47]
##  [ 0.38  2.3 ]
##  [ 0.78  0.77]
##  [-1.   -0.31]
##  [ 0.09  0.77]
##  [-1.    0.56]
##  [ 0.28  0.07]
##  [ 0.68 -1.26]
##  [-0.51 -0.02]
##  [-1.8   0.36]
##  [-0.71  0.13]
##  [ 0.38  0.3 ]
##  [-0.31  0.07]
##  [-0.51  2.3 ]
##  [ 0.19  0.04]
##  [ 1.27  2.22]
##  [ 0.78  0.27]
##  [-0.31  0.16]
##  [-0.01 -0.54]
##  [-0.21  0.16]
```

```
##  [-0.11  0.24]
##  [-0.01 -0.25]
##  [ 2.17  1.11]
##  [-1.8   0.36]
##  [ 1.87  0.13]
##  [ 0.38 -0.13]
##  [-1.2   0.3 ]
##  [ 0.78  1.37]
##  [-0.31 -0.25]
##  [-1.7  -0.05]
##  [-1.   -0.74]
##  [ 0.28  0.5 ]
##  [-0.11 -1.06]
##  [-1.1   0.59]
##  [ 0.09 -0.8 ]
##  [-1.    1.55]
##  [-0.71  1.4 ]
##  [-1.3   0.5 ]
##  [-0.31  0.04]
##  [-0.11  0.01]
##  [-0.31 -0.89]
##  [ 0.88 -1.35]
##  [-0.31  2.24]
##  [ 0.98  1.98]
##  [-1.2   0.48]
##  [-1.3   0.27]
##  [ 1.37  1.98]
##  [ 1.27 -1.35]
##  [-0.31 -0.28]
##  [-0.51  1.26]
##  [-0.8   1.08]
##  [ 0.98 -1.06]
##  [ 0.28  0.3 ]
##  [ 0.98  0.77]
##  [-0.71 -1.5 ]
##  [-0.71  0.04]
##  [ 0.48  1.72]
##  [ 2.07  0.19]
##  [-1.99 -0.74]
##  [-0.21  1.4 ]
##  [ 0.38  0.59]
##  [ 0.88 -1.15]
##  [-1.2  -0.77]
##  [ 0.19  0.24]
##  [ 0.78 -0.31]
```

```
##  [ 2.07 -0.8 ]
##  [ 0.78  0.13]
##  [-0.31  0.62]
##  [-1.   -0.31]
##  [ 0.19 -0.36]
##  [ 2.07  2.13]
##  [ 1.87 -1.26]
##  [ 1.37 -0.92]
##  [ 0.88  1.26]
##  [ 1.47  2.13]
##  [-0.31 -1.23]
##  [ 1.97  0.91]
##  [ 0.68 -0.71]
##  [-1.5   0.36]
##  [ 0.78 -1.35]
##  [ 0.38 -0.13]
##  [-1.    0.42]
##  [-0.01 -0.31]
##  [-1.2   0.42]
##  [-0.9  -1.21]
##  [-0.11  0.04]
##  [-1.6  -0.42]
##  [ 0.98 -1.  ]
##  [ 1.08 -1.21]
##  [-0.01 -0.13]
##  [-1.1  -1.52]
##  [ 0.78 -1.21]
##  [ 0.98  2.07]
##  [-1.2  -1.52]
##  [-0.31  0.79]
##  [ 0.09 -0.31]
##  [-1.4  -1.23]
##  [-0.61 -1.5 ]
##  [ 0.78  0.53]
##  [-0.31 -0.34]
##  [ 1.77 -0.28]
##  [ 0.88 -1.03]
##  [ 0.19  0.07]
##  [-0.61  0.88]
##  [-1.89 -1.41]
##  [-1.3   0.59]
##  [-0.31  0.53]
##  [-1.   -1.09]
##  [ 1.18 -1.44]
##  [ 0.19 -0.31]
```

```
##  [ 1.18 -0.74]
##  [-0.31  0.07]
##  [ 0.19  2.1 ]
##  [ 0.78 -1.09]
##  [ 0.09  0.04]
##  [-1.8   0.13]
##  [-0.9   0.16]
##  [-0.71  0.19]
##  [ 0.88 -1.29]
##  [ 0.19 -0.25]
##  [-0.41  1.23]
##  [-0.01  0.3 ]
##  [ 0.38  0.16]
##  [ 0.88 -0.65]
##  [ 0.09  0.16]
##  [-1.89 -1.29]
##  [-0.11  0.3 ]
##  [-0.21 -0.28]
##  [ 0.28 -0.51]
##  [-0.21  1.61]
##  [ 0.98 -1.18]
##  [-0.21  1.64]
##  [ 1.27  1.87]
##  [-1.1  -0.36]
##  [-0.01  0.04]
##  [ 0.09 -0.25]
##  [-1.6  -1.23]
##  [-0.51 -0.28]
##  [ 0.98  0.13]
##  [ 1.97 -1.35]
##  [ 1.47  0.07]
##  [-0.61  1.37]
##  [ 1.57  0.01]
##  [-0.8   0.3 ]
##  [ 1.97  0.74]
##  [-1.2  -0.51]
##  [ 0.68  0.27]
##  [-1.4  -0.42]
##  [ 0.19  0.16]
##  [-0.51 -1.21]
##  [ 0.58  2.01]
##  [-1.6  -1.5 ]
##  [-0.51 -0.54]
##  [ 0.48  1.84]
##  [-1.4  -1.09]
```

```
##  [ 0.78 -1.38]
##  [-0.31 -0.42]
##  [ 1.57  1.  ]
##  [ 0.98  1.43]
##  [-0.31 -0.48]
##  [-0.11  2.16]
##  [-1.5  -0.1 ]
##  [-0.11  1.95]
##  [-0.71 -0.34]
##  [-0.51 -0.83]
##  [ 0.68 -1.38]
##  [-0.8  -1.58]
##  [-1.89 -1.47]
##  [ 1.08  0.13]
##  [ 0.09  1.52]
##  [-0.31  0.1 ]
##  [ 0.09  0.04]
##  [-1.4  -1.35]
##  [ 0.28  0.07]
##  [-0.9   0.39]
##  [ 1.57 -1.26]
##  [-0.31 -0.74]
##  [-0.11  0.16]
##  [-0.9  -0.65]
##  [-0.71 -0.05]
##  [ 0.38 -0.45]
##  [-0.8   1.9 ]
##  [ 1.37  1.29]
##  [ 1.18 -0.97]
##  [ 1.77  1.84]
##  [-0.9  -0.25]
##  [-0.8   0.56]
##  [-1.2  -1.55]
##  [-0.51 -1.12]
##  [ 0.28  0.07]
##  [-0.21 -1.06]
##  [ 1.67  1.61]
##  [ 0.98  1.78]
##  [ 0.28  0.04]
##  [-0.8  -0.22]
##  [-0.11  0.07]
##  [ 0.28 -0.19]
##  [ 1.97 -0.65]
##  [-0.8   1.35]
##  [-1.8  -0.6 ]
```

```
##  [-0.11  0.13]
##  [ 0.28 -0.31]
##  [ 1.08  0.56]
##  [-1.    0.27]
##  [ 1.47  0.36]
##  [ 0.19 -0.36]
##  [ 2.17 -1.03]
##  [-0.31  1.11]
##  [-1.7   0.07]
##  [-0.01  0.04]
##  [ 0.09  1.06]
##  [-0.11 -0.36]
##  [-1.2   0.07]
##  [-0.31 -1.35]
##  [ 1.57  1.11]
##  [-0.8  -1.52]
##  [ 0.09  1.87]
##  [-0.9  -0.77]
##  [-0.51 -0.77]
##  [-0.31 -0.92]
##  [ 0.28 -0.71]
##  [ 0.28  0.07]
##  [ 0.09  1.87]
##  [-1.1   1.95]
##  [-1.7  -1.55]
##  [-1.2  -1.09]
##  [-0.71 -0.1 ]
##  [ 0.09  0.1 ]
##  [ 0.28  0.27]
##  [ 0.88 -0.57]
##  [ 0.28 -1.15]
##  [-0.11  0.68]
##  [ 2.17 -0.68]
##  [-1.3  -1.38]
##  [-1.   -0.94]
##  [-0.01 -0.42]
##  [-0.21 -0.45]
##  [-1.8  -0.97]
##  [ 1.77  1.  ]
##  [ 0.19 -0.36]
##  [ 0.38  1.11]
##  [-1.8  -1.35]
##  [ 0.19 -0.13]
##  [ 0.88 -1.44]
##  [-1.99  0.48]
```

```
##  [-0.31  0.27]
##  [ 1.87 -1.06]
##  [-0.41  0.07]
##  [ 1.08 -0.89]
##  [-1.1  -1.12]
##  [-1.89  0.01]
##  [ 0.09  0.27]
##  [-1.2   0.33]
##  [-1.3   0.3 ]
##  [-1.    0.45]
##  [ 1.67 -0.89]
##  [ 1.18  0.53]
##  [ 1.08  0.53]
##  [ 1.37  2.33]
##  [-0.31 -0.13]
##  [ 0.38 -0.45]
##  [-0.41 -0.77]
##  [-0.11 -0.51]
##  [ 0.98 -1.15]
##  [-0.9  -0.77]
##  [-0.21 -0.51]
##  [-1.1  -0.45]
##  [-1.2   1.4 ]]
print(X_test)
## [[-0.8   0.5 ]
##  [-0.01 -0.57]
##  [-0.31  0.16]
##  [-0.8   0.27]
##  [-0.31 -0.57]
##  [-1.1  -1.44]
##  [-0.71 -1.58]
##  [-0.21  2.16]
##  [-1.99 -0.05]
##  [ 0.88 -0.77]
##  [-0.8  -0.6 ]
##  [-1.   -0.42]
##  [-0.11 -0.42]
##  [ 0.09  0.22]
##  [-1.8   0.48]
##  [-0.61  1.37]
##  [-0.11  0.22]
##  [-1.89  0.45]
##  [ 1.67  1.75]
##  [-0.31 -1.38]
##  [-0.31 -0.65]
```

```
##  [ 0.88  2.16]
##  [ 0.28 -0.54]
##  [ 0.88  1.03]
##  [-1.5  -1.21]
##  [ 1.08  2.07]
##  [-1.    0.5 ]
##  [-0.9   0.3 ]
##  [-0.11 -0.22]
##  [-0.61  0.48]
##  [-1.7   0.53]
##  [-0.11  0.27]
##  [ 1.87 -0.28]
##  [-0.11 -0.48]
##  [-1.4  -0.34]
##  [-1.99 -0.51]
##  [-1.6   0.33]
##  [-0.41 -0.77]
##  [-0.71 -1.03]
##  [ 1.08 -0.97]
##  [-1.1   0.53]
##  [ 0.28 -0.51]
##  [-1.1   0.42]
##  [-0.31 -1.44]
##  [ 0.48  1.23]
##  [-1.1  -0.34]
##  [-0.11  0.3 ]
##  [ 1.37  0.59]
##  [-1.2  -1.15]
##  [ 1.08  0.48]
##  [ 1.87  1.52]
##  [-0.41 -1.29]
##  [-0.31 -0.36]
##  [-0.41  1.32]
##  [ 2.07  0.53]
##  [ 0.68 -1.09]
##  [-0.9   0.39]
##  [-1.2   0.3 ]
##  [ 1.08 -1.21]
##  [-1.5  -1.44]
##  [-0.61 -1.5 ]
##  [ 2.17 -0.8 ]
##  [-1.89  0.19]
##  [-0.21  0.85]
##  [-1.89 -1.26]
##  [ 2.17  0.39]
```

```
##  [-1.4   0.56]
##  [-1.1  -0.34]
##  [ 0.19 -0.65]
##  [ 0.38  0.01]
##  [-0.61  2.33]
##  [-0.31  0.22]
##  [-1.6  -0.19]
##  [ 0.68 -1.38]
##  [-1.1   0.56]
##  [-1.99  0.36]
##  [ 0.38  0.27]
##  [ 0.19 -0.28]
##  [ 1.47 -1.03]
##  [ 0.88  1.08]
##  [ 1.97  2.16]
##  [ 2.07  0.39]
##  [-1.4  -0.42]
##  [-1.2  -1.  ]
##  [ 1.97 -0.92]
##  [ 0.38  0.3 ]
##  [ 0.19  0.16]
##  [ 2.07  1.75]
##  [ 0.78 -0.83]
##  [ 0.28 -0.28]
##  [ 0.38 -0.16]
##  [-0.11  2.22]
##  [-1.5  -0.63]
##  [-1.3  -1.06]
##  [-1.4   0.42]
##  [-1.1   0.77]
##  [-1.5  -0.19]
##  [ 0.98 -1.06]
##  [ 0.98  0.59]
##  [ 0.38  1.  ]]
```

R

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

3.7.6 Training the Random Forest Classification model on the Training set

Python

```
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators = 10, criterion = 'entropy', random_state = 0)
classifier.fit(X_train, y_train)
## RandomForestClassifier(criterion='entropy', n_estimators=10, random_state=0)
```

R

```
# install.packages('randomForest')
library(randomForest)
set.seed(123)
classifier = randomForest(x = training_set[-3],
                          y = training_set$Purchased,
                          ntree = 500)
```

3.7.7 Predicting a new result

Python

```
print(classifier.predict(sc.transform([[30,87000]])))
## [0]
```

3.7.8 Predicting the Test set results

Python

```
y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))
## [[0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [1 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [1 0]
##  [1 0]
##  [0 0]
##  [1 1]
##  [0 0]
```

```
##  [0 0]
##  [1 1]
##  [0 0]
##  [1 1]
##  [0 0]
##  [0 1]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 1]
##  [1 1]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [1 1]
##  [1 1]
##  [0 0]
##  [0 0]
##  [1 0]
##  [1 1]
##  [1 1]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [1 1]
##  [0 0]
```

```
##  [1 1]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 0]
##  [1 1]
##  [1 1]
##  [1 1]
##  [1 0]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 1]
##  [0 0]
##  [1 1]
##  [1 1]
##  [0 0]
##  [0 0]
##  [1 1]
##  [0 0]
##  [0 0]
##  [0 0]
##  [0 1]
##  [0 0]
##  [1 1]
##  [1 1]
##  [1 1]]
```

R

```
y_pred = predict(classifier, newdata = test_set[-3])
```

3.7.9 Making the Confusion Matrix

Python

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

```
print(cm)
## [[63  5]
##  [ 4 28]]
accuracy_score(y_test, y_pred)
## 0.91
```

R

```
cm = table(test_set[, 3], y_pred)
```

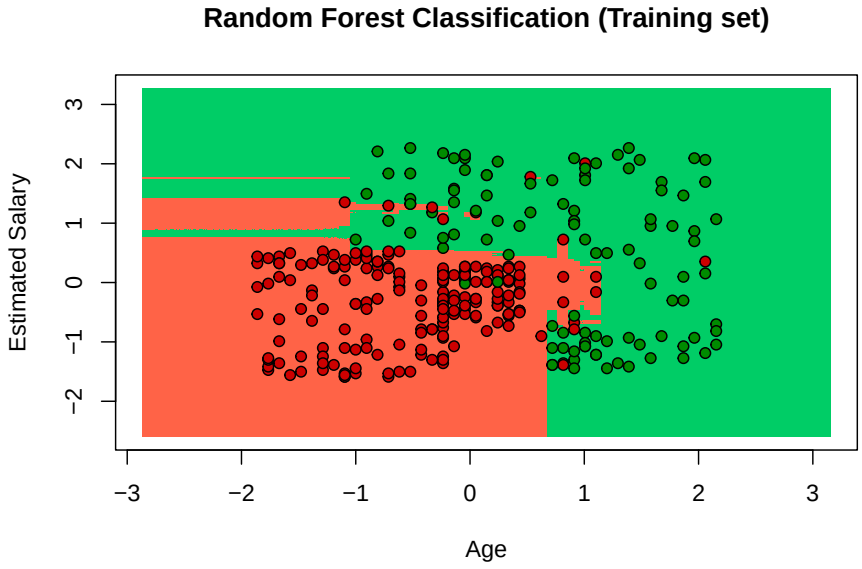
3.7.10 Visualising the Training set results

Python

```
from matplotlib.colors import ListedColormap
X_set, y_set = sc.inverse_transform(X_train), y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 10, stop = X_set[:, 0].max()
                     np.arange(start = X_set[:, 1].min() - 1000, stop = X_set[:, 1].ma
plt.contourf(X1, X2, classifier.predict(sc.transform(np.array([X1.ravel(), X2.ravel()]
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], c = ListedColormap(('red',
plt.title('Random Forest Classification (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

R

```
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')
y_grid = predict(classifier, grid_set)
plot(set[, -3],
     main = 'Random Forest Classification (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'))
```



3.7.11 Visualising the Test set results

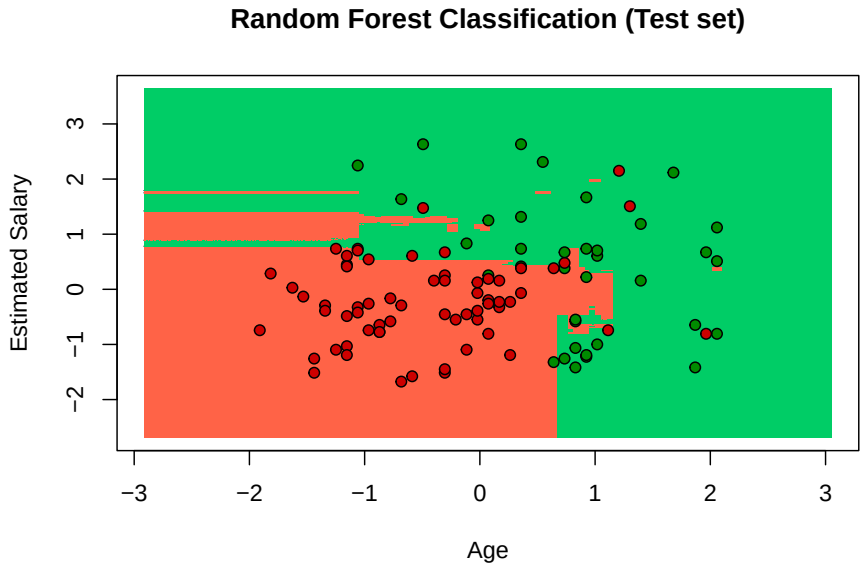
Python

```
from matplotlib.colors import ListedColormap
X_set, y_set = sc.inverse_transform(X_test), y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 10, stop = X_set[:, 0].max() + 10, ste
                     np.arange(start = X_set[:, 1].min() - 1000, stop = X_set[:, 1].max() + 1000,
plt.contourf(X1, X2, classifier.predict(sc.transform(np.array([X1.ravel(), X2.ravel()]).T)).resha
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], c = ListedColormap(('red', 'green'))(
plt.title('Random Forest Classification (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

R

```
library(ElemStatLearn)
set = test_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)
```

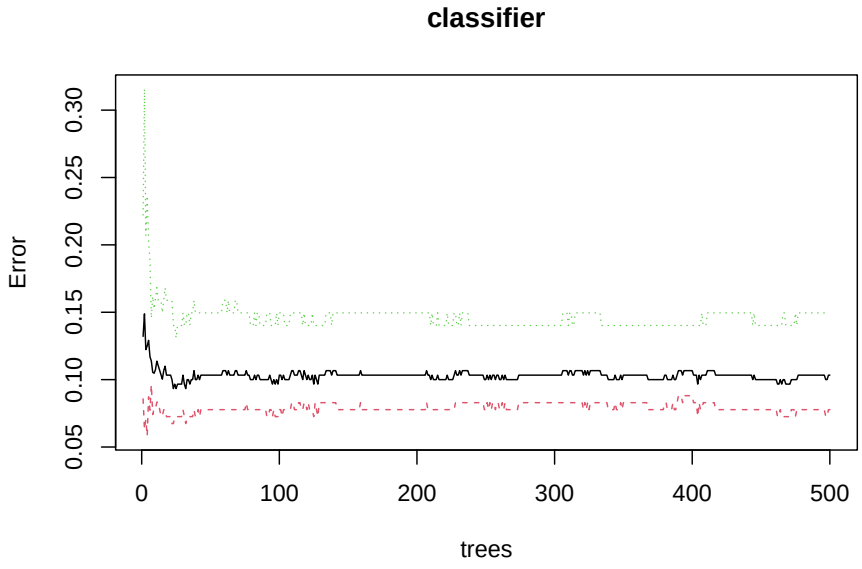
```
grid_set = expand.grid(X1, X2)
colnames(grid_set) = c('Age', 'EstimatedSalary')
y_grid = predict(classifier, grid_set)
plot(set[, -3], main = 'Random Forest Classification (Test set)',
     xlab = 'Age', ylab = 'Estimated Salary',
     xlim = range(X1), ylim = range(X2))
contour(X1, X2, matrix(as.numeric(y_grid), 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'))
```



3.7.12 Choosing the number of trees

R

```
plot(classifier)
```



3.8 Classification Model Selection in Python

3.8.1 Heading 3

Python

R

Chapter 4

Footnotes and citations

4.1 Footnotes

Footnotes are put inside the square brackets after a caret `^[]`. Like this one [1].

4.2 Citations

Reference items in your bibliography file(s) using `@key`.

For example, we are using the **bookdown** package [**?**] (check out the last code chunk in index.Rmd to see how this citation key was added) in this sample book, which was built on top of R Markdown and **knitr** [**?**] (this citation was added manually in an external file book.bib). Note that the `.bib` files need to be listed in the index.Rmd with the YAML `bibliography` key.

The RStudio Visual Markdown Editor can also make it easier to insert citations: https://rstudio.github.io/visual-markdown-editing/#/citations

[1] This is a footnote.

Chapter 5

Blocks

5.1 Equations

Here is an equation.

$$f(k) = \binom{n}{k} p^k (1-p)^{n-k} \tag{5.1}$$

You may refer to using `\@ref(eq:binom)`, like see Equation (5.1).

5.2 Theorems and proofs

Labeled theorems can be referenced in text using `\@ref(thm:tri)`, for example, check out this smart theorem 5.1.

Theorem 5.1. *For a right triangle, if c denotes the* length *of the hypotenuse and a and b denote the lengths of the* ***other*** *two sides, we have*

$$a^2 + b^2 = c^2$$

Read more here https://bookdown.org/yihui/bookdown/markdown-extensions-by-bookdown.html.

5.3 Callout blocks

The R Markdown Cookbook provides more help on how to use custom blocks to design your own callouts: https://bookdown.org/yihui/rmarkdown-cookbook/custom-blocks.html

Chapter 6

Sharing your book

6.1 Publishing

HTML books can be published online, see: https://bookdown.org/yihui/bookdown/publishing.html

6.2 404 pages

By default, users will be directed to a 404 page if they try to access a webpage that cannot be found. If you'd like to customize your 404 page instead of using the default, you may add either a `_404.Rmd` or `_404.md` file to your project root and use code and/or Markdown syntax.

6.3 Metadata for sharing

Bookdown HTML books will provide HTML metadata for social sharing on platforms like Twitter, Facebook, and LinkedIn, using information you provide in the `index.Rmd` YAML. To setup, set the `url` for your book and the path to your `cover-image` file. Your book's `title` and `description` are also used.

This `gitbook` uses the same social sharing data across all chapters in your book- all links shared will look the same.

Specify your book's source repository on GitHub using the `edit` key under the configuration options in the `_output.yml` file, which allows users to suggest an edit by linking to a chapter's source file.

Read more about the features of this output format here:

https://pkgs.rstudio.com/bookdown/reference/gitbook.html

Or use:

```
?bookdown::gitbook
```