MACHINE LEARNING WITH R

AchyuthReddy.k

MTECH DATASCIENCE

Model	Package	Class
LinearRegression	NaN	lm()
Multiple Linear Regression	NaN	lm()
Polynomial Regression	Nan	lm()
Support Vector Regression	E1071	svm()
Decision Tree Regression	rpart	rpart()
RandomForest Regression	randomForest	randomForest()
Logistic Regression	NaN	glm()
KNN	class	knn()
Support Vector Machine(SVM)	e1071,kernlab	svm()
Kernel SVM	Kernlab,e1071	Ksvm()
Decision Tree Classification	rpart	rpart()
Naïve Bayes	e1071	naiveBayes()
K-Means Cluster	Nan	kmeans()
Hierarchical Clustering	Nan	hclust()
Apriori-ARM	arules	apriori()
Eclat	arules	eclat()
Upper Confidence Bound(UCB)	NaN	NaN
Thompson Sampling	NaN	NaN
Principal Component Analysis	caret	preProcess()

LinearDiscriminant Analysis	MASS	lda()
Kernel PCA	kernlab	kpca()
K-Fold Cross Validation	caret	createFolds()
Grid Search	caret	train()
XGBoost	xgboost	xgboost()

Data Preprocessing in R

Step 1:Installing packages

```
install.packages("caTools")

library(caTools)
```

Step 2:Importing datasets

```
data=read.csv("FileName.csv")
```

Step 3:Taking Care of Missing values

```
data$Col1=ifelse(is.na(data$Col1),
        ave(data$Col1,FUN = function(x) mean(x,na.rm = TRUE)),
        data$Col1)
```

Step 4:Categorical Data

```
data$ColName=factor(data$ColName,
          levels = c('France','Spain','Germany'),
          labels = c(1,2,3))
```

Step5:Splitting data in to train and test sets

```
set.seed(123)

Split=sample.split(data$Purchased,SplitRatio = 0.8)

train_set=subset(data,Split==TRUE)

test_set=subset(data,Split==FALSE)
```

Step 6:Feature Scaling

```
train_set[2:3]=scale(train_set[2:3])

test_set[2:3]=scale(test_set[2:3])
```

Simple Linear Regression

```
#importing data
data=read.csv("Salary_Data.csv")
#splitting train and test set
library(caTools)
Split=sample.split(data$Salary,SplitRatio = 0.75)
train_set=subset(data,Split==TRUE)
test_set=subset(data,Split==FALSE)
#model building
reg=lm(formula =Salary ~ YearsExperience,data = train_set)
summary(reg)
#prediction
y_pred=predict(reg,newdata = test_set)
#visualising data
install.packages("ggplot2")
library(ggplot2)
#on traing data
ggplot()+
  geom_point(aes(train_set$YearsExperience,train_set$Salary),colour="red")+
  geom_line(aes(train_set$YearsExperience,predict(reg,newdata=train_set)),
            colour="blue")+
  ggtitle("Sal vs Exp(traingSet")+
  xlab("Years of Expr")+
  ylab("Salary")
```

```
#on test set
ggplot()+
  geom_point(aes(test_set$YearsExperience,test_set$Salary),colour="blue")+
  geom_line(aes(train_set$YearsExperience,predict(reg,newdata=train_set)),
            colour="red")+
  ggtitle("Sal vs Exp(testSet")+
  xlab("Years of Expr")+
  ylab("Salary")
```

Multiple Linear Regression

```
#importing packages
install.packages("caTools")
library(caTools)
#importing Dataset
data=read.csv("50_Startups.csv")
#categorical data
data$State=factor(data$State,levels  = c("New York","California","Florida"),
           labels = c(1,2,3))
#splitting data into train and test sets
set.seed(123)
values=sample.split(data$Profit,SplitRatio = 0.8)
train_set=subset(data,values==TRUE)
test_set=subset(data,values==FALSE)
```

```
#build the model back propagation
reg=lm(formula = Profit ~ R.D.Spend + Administration +
         Marketing.Spend + State,data = data)
summary(reg)
reg=lm(formula = Profit ~ R.D.Spend + Administration +
         Marketing.Spend,data = data)
summary(reg)
reg=lm(formula = Profit ~ R.D.Spend  +
         Marketing.Spend,data = data)
summary(reg)
reg=lm(formula = Profit ~ R.D.Spend  ,data = data)
summary(reg)
```

Polynomial Regression

```
#importing dataset
data=read.csv("Position_Salaries.csv")
data=data[2:3]
#building linear model
lin_reg=lm(formula=Salary ~ .,data = data)
#building a polynomial reg
data$Level2=data$Level^2
data$Level3=data$Level^3
data$Level4=data$Level^4

ploy_reg=lm(formula = Salary ~ .,data=data)
```

```
#visualising the data
library(ggplot2)
ggplot()+
  geom_point(aes(x=data$Level,y=data$Salary),color="blue")+
  geom_line(aes(x=data$Level,y=predict(lin_reg,newdata = data)),color="red")+
  geom_line(aes(x=data$Level,y=predict(ploy_reg,newdata =
data)),color="green")+
  ggtitle("Salary Detect")+
  xlab("Expericence")+
  ylab("Salary")
```

Support Vector Regression

```
#importing dataset
data=read.csv("Position_Salaries.csv")
data=data[2:3]
#installing lib
install.packages("e1071")
library("e1071")

#building the model
reg=svm(formula=Salary ~ .,data = data,
        type="eps-regression")

#predicting
y_pred=predict(reg,newdata = data.frame(Level=6.5))
```

```
#visualising the data
library(ggplot2)
ggplot()+
  geom_point(aes(x=data$Level,y=data$Salary),color="blue")+
  geom_line(aes(x=data$Level,y=predict(reg,newdata = data)),color="black")+
  ggtitle("Sal vs Exp using SVR")+
  xlab("Exp")+
  ylab("Salary")
```

Decision Tree Regression

```
#importing dataset
data=read.csv("Position_Salaries.csv")
data=data[2:3]
#importing req pack
install.packages("rpart")
library("rpart")
#dev model
req=rpart(formula = Salary ~ .,
          data=data,control = rpart.control(minsplit = 1))
#prediction
y_pred=predict(req,data.frame(Level=6.5))
```

```
#visualising model
library("ggplot2")
X_grid=seq(min(data$Level),max(data$Level),0.01)
ggplot()+
  geom_point(aes(x=data$Level,y=data$Salary),color="blue")+
  geom_line(aes(x=X_grid,y=predict(req,newdata =
data.frame(Level=X_grid))),color="black")+
  ggtitle("Sal vs Exp using DecsionTreeRegression")+
  xlab("Exp")+
  ylab("Salary")
```

Random Forest Regression

```
#importing data to r inv
data=read.csv("Position_Salaries.csv")
dataset=data[2:3]
#importing lib
install.packages("randomForest")
library("randomForest")
req=randomForest(x=dataset[1],
                 y=dataset$Salary,
                 ntree = 500)
#prediction
y_pred=predict(req,data.frame(Level=6.5))
```

```
#visualising
library("ggplot2")
X_grid=seq(min(dataset$Level),max(dataset$Level),0.1)
ggplot()+
  geom_point(aes(x=dataset$Level,y=dataset$Salary),
            color='red')+
  geom_line(aes(x=X_grid,y=predict(req,newdata = data.frame(Level=X_grid))),
           color="blue")+
  ggtitle("RandomForestReg")+
  xlab("Position")+
  ylab("Salary")
```

Classification Algorithms

Logistic Regression

```
#importing data
data=read.csv("Social_Network_Ads.csv")
dataset=data[,3:5]
#splitting data
library(caTools)
set.seed(123)
split=sample.split(dataset$Purchased,SplitRatio = 0.75)
train_set=subset(dataset,split ==TRUE )
test_set=subset(dataset,split == FALSE)
#feature scaling
train_set[,1:2]=scale(train_set[,1:2])
test_set[,1:2]=scale(test_set[,1:2])
#Build model
classifier=glm(formula = Purchased ~ .,
            data = train_set,
            family = binomial)
# Predicting the Test set results
#new_test_set=data.frame(test_set[,1:2])
prob_pred = predict(classifier, type = 'response', newdata = test_set[-3])
y_pred = ifelse(prob_pred > 0.5, 1, 0)
```

```
# Making the Confusion Matrix
cm = table(test_set[, 3], y_pred > 0.5)
```

```
# Visualising the Training set results
install.packages('ElemStatLearn')
library(ElemStatLearn)
set = train_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'))
```

```
# Visualising the Test set results
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'))
```

K-Nearest Neighbor(KNN)

#importingdata

```
data=read.csv("Social_Network_Ads.csv")
dataset=data[,3:5]
```

#splitting data

```
set.seed(123)
library(caTools)
split=sample.split(dataset$Purchased,SplitRatio = 0.25)
train_set=subset(dataset,split==TRUE)
test_set=subset(dataset,split==FALSE)
```

#feature scale

```
train_set[,1:2]=scale(train_set[,1:2])
test_set[,1:2]=scale(test_set[,1:2])
```

#build the model and test the model

```
install.packages('class')
library(class)

y_pred=knn(train=train_set[,-3],
        test=test_set[,-3],
        cl=train_set[,3],
        k=5)
```

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

```
# Visualising the Training set results
#install.packages('ElemStatLearn')
library(ElemStatLearn)
set = train_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=train_set[,-3],
          test=grid_set,
          cl=train_set[,3],
          k=5)
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'))
```

```
# Visualising the Test set results
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=train_set[,-3],
            test=grid_set,
            cl=train_set[,3],
            k=5)
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'))
```

Support Vector Machine (SVM)

```
#importingdata
data=read.csv("Social_Network_Ads.csv")
dataset=data[,3:5]

#splitting data
set.seed(123)
library(caTools)
split=sample.split(dataset$Purchased,SplitRatio = 0.75)
train_set=subset(dataset,split==TRUE)
test_set=subset(dataset,split==FALSE)

#feature scale
train_set[,1:2]=scale(train_set[,1:2])
test_set[,1:2]=scale(test_set[,1:2])

#build the model and test the model
install.packages('e1071')
library(e1071)
classifier=svm(formula=Purchased ~ .,
            data=train_set,
            type='C-classification',
            kernel = 'linear')

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

#confusion matrix

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

# Visualising the Training set results

```
#install.packages('ElemStatLearn')
library(ElemStatLearn)
set = train_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'))
```

# Visualising the Test set results

```
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)
```

```r
grid_set = expand.grid(X1, X2)
colnames(grid_set) = c('Age', 'EstimatedSalary')

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

Kernel SVM

```r
#importingdata
data=read.csv("Social_Network_Ads.csv")
dataset=data[,3:5]

#splitting data
set.seed(123)
library(caTools)
split=sample.split(dataset$Purchased,SplitRatio = 0.75)
train_set=subset(dataset,split==TRUE)
test_set=subset(dataset,split==FALSE)
```

```
#feature scale
train_set[,1:2]=scale(train_set[,1:2])
test_set[,1:2]=scale(test_set[,1:2])

#build the model and test the model
install.packages('kernlab')
library(kernlab)
classifierksvm=ksvm(Purchased ~ .,
            data=train_set,
            type='C-svc',
            kernel ='rbfdot')
#prediction
y_pred=predict(classifierksvm,newdata =test_set[-3])

#confusion matrix
cm=table(test_set[,3],y_pred)


# Visualising the Training set results
#install.packages('ElemStatLearn')
library(ElemStatLearn)
set = train_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(classifierksvm,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'))

# Visualising the Test set results
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(classifierksvm,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'))
```

Decision Tree Classification

#importingdata

```
data=read.csv("Social_Network_Ads.csv")

dataset=data[,3:5]
```

#splitting data

```
set.seed(123)

#library(caTools)

split=sample.split(dataset$Purchased,SplitRatio = 0.75)

train_set=subset(dataset,split==TRUE)

test_set=subset(dataset,split==FALSE)
```

#feature scale

```
train_set[,1:2]=scale(train_set[,1:2])

test_set[,1:2]=scale(test_set[,1:2])
```

#build the model and test the model

```
#install.packages('rpart')

library(rpart)

classifier=rpart(formula = Purchased ~.,

            data=train_set)
```

prediction

```
prob_pred=predict(classifier,newdata =test_set[-3])
y_pred = ifelse(prob_pred > 0.5, 1, 0)
#confusion matrix
cm=table(test_set[,3],y_pred)
```

# Visualising the Training set results

```
#install.packages('ElemStatLearn')
library(ElemStatLearn)
set = train_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 = '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'))
```

```
# Visualising the Test set results
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 = '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'))

#ploting DTC
plot(classifier)
text(classifier)
```

Naive Bayes

```
#importingdata
data=read.csv("Social_Network_Ads.csv")
dataset=data[,3:5]

#encoding target var
dataset$Purchased=factor(dataset$Purchased,levels = c(0,1))

#splitting data
set.seed(123)
library(caTools)
split=sample.split(dataset$Purchased,SplitRatio = 0.75)
train_set=subset(dataset,split==TRUE)
test_set=subset(dataset,split==FALSE)

#feature scale
train_set[,1:2]=scale(train_set[,1:2])
test_set[,1:2]=scale(test_set[,1:2])

#build the model and test the model
library(e1071)
classifier=naiveBayes(x=train_set[-3],
                      y=train_set$Purchased)
#prediction
y_pred=predict(classifier,newdata =test_set[-3])
#y_pred = ifelse(prob_pred > 0.5, 1, 0)
```

#confusion matrix

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

# Visualising the Training set results

```
#install.packages('ElemStatLearn')
library(ElemStatLearn)
set = train_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'))
```

# Visualising the Test set results

```
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'))
```

CLUSTERING

K-Means Clustering

```
#import data
data=read.csv("Mall_Customers.csv")
X=data[,4:5]

#Using ElBow method
set.seed(6)
wcss=vector()
for (i in 1:10) wcss[i]=sum(kmeans(X,i)$withinss)
plot(1:10,wcss,type='b',main=paste("Elbow Method"),xlab = "No oF Clusters",
   ylab = "WCSS")

#model build
set.seed(22)
km=kmeans(X,5,iter.max = 300,nstart = 10)

#visualising the clusters
library(cluster)
clusplot(X, km$cluster,
       lines=0, shade=TRUE, color=TRUE, labels=2, plotchar=FALSE
       span=TRUE,
       main=paste("KMeans Cluster Alg"),
       xlab="Annaul Income",
       ylab="SpendingScore")
```

Hierarchical_Clustering

```
#import data
data=read.csv("Mall_Customers.csv")
X=data[,4:5]

#Finding no of clusters using dendrograms
dendro=hclust(dist(X,method="euclidean"),method = 'ward.D')
plot(dendro,
    main = paste("DendroGrams"),
    xlab = "Customers",
    ylab = "Euclidean Distance")

#Build the model
hc=hclust(dist(X,method="euclidean"),method = 'ward.D')
y_hc=cutree(hc,5)

#visualising the clusters
library(cluster)
clusplot(X,
        y_hc,
        lines=0,
        shade=TRUE,
        color=TRUE,
        labels=2,
        plotchar=FALSE,
        span=TRUE,
        main=paste("KMeans Cluster Alg"),
        xlab="Annaul Income",
        ylab="SpendingScore")
```

APRIORI-Association Rule Mapping

```
#Data Preprocessing
dataset=read.csv("Market_Basket_Optimisation.csv",header = FALSE)

#install.packages('arules')
library(arules)
dataset=read.transactions("Market_Basket_Optimisation.csv",sep = ",",
                    rm.duplicates = TRUE)
summary(dataset)
itemFrequencyPlot(dataset,topN=10)

#train the model
rules=apriori(data = dataset,parameter = list(support=0.005, confidence=0.2))

#visualize
inspect(sort(rules,by='lift')[0:10])
```

ECLAT- Association Rule Mapping

```
#Data Preprocessing
dataset=read.csv("Market_Basket_Optimisation.csv",header = FALSE)

#install.packages('arules')
library(arules)
dataset=read.transactions("Market_Basket_Optimisation.csv",sep = ",",
                          rm.duplicates = TRUE)
summary(dataset)
itemFrequencyPlot(dataset,topN=10)

#train the model
rules=eclat(data = dataset,parameter = list(support=0.005, minlen=2))

#visualize
inspect(sort(rules,by='support')[0:1
```

Upper Confidence Bound(UCB)

```
# Upper Confidence Bound
# Importing the dataset
dataset = read.csv('Ads_CTR_Optimisation.csv')
# Implementing UCB
N = 10000
d = 10
ads_selected = integer(0)
numbers_of_selections = integer(d)
sums_of_rewards = integer(d)
total_reward = 0
for (n in 1:N) {
  ad = 0
  max_upper_bound = 0
  for (i in 1:d) {
    if (numbers_of_selections[i] > 0) {
      average_reward = sums_of_rewards[i] / numbers_of_selections[i]
      delta_i = sqrt(3/2 * log(n) / numbers_of_selections[i])
      upper_bound = average_reward + delta_i
    } else {
      upper_bound = 1e400
    }
    if (upper_bound > max_upper_bound) {
      max_upper_bound = upper_bound
      ad = i
    }
  }
  ads_selected = append(ads_selected, ad)
  numbers_of_selections[ad] = numbers_of_selections[ad] + 1
  reward = dataset[n, ad]
  sums_of_rewards[ad] = sums_of_rewards[ad] + reward
  total_reward = total_reward + reward
}
# Visualising the results
hist(ads_selected,
    col = 'blue',
    main = 'Histogram of ads selections',
    xlab = 'Ads',
    ylab = 'Number of times each ad was selected')
```

Thompson Sampling

```
# Importing the dataset
dataset = read.csv('Ads_CTR_Optimisation.csv')
# Implementing Thompson Sampling
N = 10000
d = 10
ads_selected = integer(0)
numbers_of_rewards_1 = integer(d)
numbers_of_rewards_0 = integer(d)
total_reward = 0
for (n in 1:N) {
  ad = 0
  max_random = 0
  for (i in 1:d) {
    random_beta = rbeta(n = 1,
                        shape1 = numbers_of_rewards_1[i] + 1,
                        shape2 = numbers_of_rewards_0[i] + 1)
    if (random_beta > max_random) {
      max_random = random_beta
      ad = i
    }
  }
  ads_selected = append(ads_selected, ad)
  reward = dataset[n, ad]
  if (reward == 1) {
    numbers_of_rewards_1[ad] = numbers_of_rewards_1[ad] + 1
  } else {
    numbers_of_rewards_0[ad] = numbers_of_rewards_0[ad] + 1
  }
  total_reward = total_reward + reward
}
# Visualising the results
hist(ads_selected,
     col = 'blue',
     main = 'Histogram of ads selections',
     xlab = 'Ads',
     ylab = 'Number of times each ad was selected')
```

Natural language processing (NLP)

```
#import data
dataset_ori=read.delim("Restaurant_Reviews.tsv",quote = "",stringsAsFactors = FALSE)

#datacleaning
# install.packages('NLP')
library(tm)
corpus=VCorpus(VectorSource(dataset_ori$Review))
corpus=tm_map(corpus,content_transformer(tolower))
corpus=tm_map(corpus,removeNumbers)
corpus=tm_map(corpus,removePunctuation)

#install.packages("SnowballC")
library(SnowballC)
corpus=tm_map(corpus,removeWords,stopwords())
corpus=tm_map(corpus,stemDocument)
corpus=tm_map(corpus,stripWhitespace)

#create a bag of words
dtm=DocumentTermMatrix(corpus)
dtm=removeSparseTerms(dtm,0.999)

#dataset
dataset=as.data.frame(as.matrix(dtm))
dataset$liked=dataset_ori$Liked
```

```
#encoding target var
dataset$liked=factor(dataset$liked,levels = c(0,1))

#splitting data
set.seed(123)
library(caTools)
split=sample.split(dataset$liked,SplitRatio = 0.80)
train_set=subset(dataset,split==TRUE)
test_set=subset(dataset,split==FALSE)

#build the model and test the model
#install.packages('randomForest')
library(randomForest)
classifier=randomForest(x=train_set[-692],
              y=train_set$liked,
              ntree = 10)
#prediction
y_pred=predict(classifier,newdata =test_set[-692])
y_pred = ifelse(prob_pred > 0.5, 1, 0)
#confusion matrix
cm=table(test_set[,692],y_pred)
```

Principal Component Analysis

```
# Importing the dataset
dataset = read.csv('Wine.csv')

# Splitting the dataset into the Training set and Test set
# install.packages('caTools')
library(caTools)
set.seed(123)
split = sample.split(dataset$Customer_Segment, SplitRatio = 0.8)
training_set = subset(dataset, split == TRUE)
test_set = subset(dataset, split == FALSE)

# Feature Scaling
training_set[-14] = scale(training_set[-14])
test_set[-14] = scale(test_set[-14])

# Applying PCA
install.packages('caret')
library(caret)
# install.packages('e1071')
library(e1071)
pca = preProcess(x = training_set[-14], method = 'pca', pcaComp = 2)
training_set = predict(pca, training_set)
training_set = training_set[c(2, 3, 1)]
test_set = predict(pca, test_set)
test_set = test_set[c(2, 3, 1)]
```

```
# Fitting SVM to the Training set
# install.packages('e1071')
library(e1071)
classifier = svm(formula = Customer_Segment ~ .,
          data = training_set,
          type = 'C-classification',
          kernel = 'linear')

# Predicting the Test set results
y_pred = predict(classifier, newdata = test_set[-3])

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

# Visualising the Training set results
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('PC1', 'PC2')
y_grid = predict(classifier, newdata = grid_set)
plot(set[, -3],
   main = 'SVM (Training set)',
   xlab = 'PC1', ylab = 'PC2',
    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 == 2, 'deepskyblue', ifelse(y_grid == 1, 'springgreen3', 'tomato')))
points(set, pch = 21, bg = ifelse(set[, 3] == 2, 'blue3', ifelse(set[, 3] == 1, 'green4', 'red3')))

# Visualising the Test set results
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('PC1', 'PC2')
y_grid = predict(classifier, newdata = grid_set)
plot(set[, -3], main = 'SVM (Test set)',
    xlab = 'PC1', ylab = 'PC2',
    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 == 2, 'deepskyblue', ifelse(y_grid == 1, 'springgreen3', 'tomato')))
points(set, pch = 21, bg = ifelse(set[, 3] == 2, 'blue3', ifelse(set[, 3] == 1, 'green4', 'red3')))
```

LinearDiscriminantAnalysis

```
# Importing the dataset
dataset = read.csv('Wine.csv')


# Splitting the dataset into the Training set and Test set
# install.packages('caTools')
library(caTools)
set.seed(123)
split = sample.split(dataset$Customer_Segment, SplitRatio = 0.8)
training_set = subset(dataset, split == TRUE)
test_set = subset(dataset, split == FALSE)


# Feature Scaling
training_set[-14] = scale(training_set[-14])
test_set[-14] = scale(test_set[-14])


# Applying LDA
library(MASS)
lda = lda(formula = Customer_Segment ~ ., data = training_set)
training_set = as.data.frame(predict(lda, training_set))
training_set = training_set[c(5, 6, 1)]
test_set = as.data.frame(predict(lda, test_set))
test_set = test_set[c(5, 6, 1)]


# Fitting SVM to the Training set
# install.packages('e1071')
library(e1071)
classifier = svm(formula = class ~ .,
                 data = training_set,
                 type = 'C-classification',
                 kernel = 'linear')

# Predicting the Test set results
y_pred = predict(classifier, newdata = test_set[-3])

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

# Visualising the Training set results
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('x.LD1', 'x.LD2')
y_grid = predict(classifier, newdata = grid_set)
plot(set[, -3],
     main = 'SVM (Training set)',
     xlab = 'LD1', ylab = 'LD2',
     xlim = range(X1), ylim = range(X2))
contour(X1, X2, matrix(as.numeric(y_grid), length(X1), length(X2)), add = TRUE)
```

```r
points(grid_set, pch = '.', col = ifelse(y_grid == 2, 'deepskyblue', ifelse(y_grid == 1, 'springgreen3', 'tomato')))

points(set, pch = 21, bg = ifelse(set[, 3] == 2, 'blue3', ifelse(set[, 3] == 1, 'green4', 'red3')))


# Visualising the Test set results

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('x.LD1', 'x.LD2')

y_grid = predict(classifier, newdata = grid_set)

plot(set[, -3], main = 'SVM (Test set)',

    xlab = 'LD1', ylab = 'LD2',

    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 == 2, 'deepskyblue', ifelse(y_grid == 1, 'springgreen3', 'tomato')))

points(set, pch = 21, bg = ifelse(set[, 3] == 2, 'blue3', ifelse(set[, 3] == 1, 'green4', 'red3’)))
```

Kernel PCA

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

# Splitting the dataset into the Training set and Test set
# install.packages('caTools')
library(caTools)
set.seed(123)
split = sample.split(dataset$Purchased, SplitRatio = 0.75)
training_set = subset(dataset, split == TRUE)
test_set = subset(dataset, split == FALSE)

# Feature Scaling
training_set[, 1:2] = scale(training_set[, 1:2])
test_set[, 1:2] = scale(test_set[, 1:2])

# Applying Kernel PCA
install.packages('kernlab')
library(kernlab)
kpca = kpca(~., data = training_set[-3], kernel = 'rbfdot', features = 2)
training_set_pca = as.data.frame(predict(kpca, training_set))
training_set_pca$Purchased = training_set$Purchased
test_set_pca = as.data.frame(predict(kpca, test_set))
test_set_pca$Purchased = test_set$Purchased
```

```
# Fitting Logistic Regression to the Training set
classifier = glm(formula = Purchased ~ .,
                 family = binomial,
                 data = training_set_pca)

# Predicting the Test set results
prob_pred = predict(classifier, type = 'response', newdata = test_set_pca[-3])
y_pred = ifelse(prob_pred > 0.5, 1, 0)

# Making the Confusion Matrix
cm = table(test_set_pca[, 3], y_pred)

# Visualising the Training set results
install.packages('ElemStatLearn')
library(ElemStatLearn)
set = training_set_pca
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('V1', 'V2')
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 = 'PC1', ylab = 'PC2',
     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'))
```

```
# Visualising the Test set results
# install.packages('ElemStatLearn')
library(ElemStatLearn)
set = test_set_pca
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('V1', 'V2')
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'))
```

K-Fold Cross Validation

```r
# Importing the dataset
dataset = read.csv('Social_Network_Ads.csv')
dataset = dataset[3:5]
# Encoding the target feature as factor
dataset$Purchased = factor(dataset$Purchased, levels = c(0, 1))

# Splitting the dataset into the Training set and Test set
# install.packages('caTools')
library(caTools)
set.seed(123)
split = sample.split(dataset$Purchased, SplitRatio = 0.75)
training_set = subset(dataset, split == TRUE)
test_set = subset(dataset, split == FALSE)

# Feature Scaling
training_set[-3] = scale(training_set[-3])
test_set[-3] = scale(test_set[-3])

# Fitting Kernel SVM to the Training set
# install.packages('e1071')
library(e1071)
classifier = svm(formula = Purchased ~ .,
                 data = training_set,
                 type = 'C-classification',
                 kernel = 'radial')
```

```
# Predicting the Test set results
y_pred = predict(classifier, newdata = test_set[-3])

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

# Applying k-Fold Cross Validation
# install.packages('caret')
library(caret)
folds = createFolds(training_set$Purchased, k = 10)
cv = lapply(folds, function(x) {
  training_fold = training_set[-x, ]
  test_fold = training_set[x, ]
  classifier = svm(formula = Purchased ~ .,
                   data = training_fold,
                   type = 'C-classification',
                   kernel = 'radial')
  y_pred = predict(classifier, newdata = test_fold[-3])
  cm = table(test_fold[, 3], y_pred)
  accuracy = (cm[1,1] + cm[2,2]) / (cm[1,1] + cm[2,2] + cm[1,2] + cm[2,1])
  return(accuracy)
})
accuracy = mean(as.numeric(cv))
```

Grid Search

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

# Encoding the target feature as factor
dataset$Purchased = factor(dataset$Purchased, levels = c(0, 1))

# Splitting the dataset into the Training set and Test set
# install.packages('caTools')
library(caTools)
set.seed(123)
split = sample.split(dataset$Purchased, SplitRatio = 0.75)
training_set = subset(dataset, split == TRUE)
test_set = subset(dataset, split == FALSE)

# Feature Scaling
training_set[-3] = scale(training_set[-3])
test_set[-3] = scale(test_set[-3])

# Fitting Kernel SVM to the Training set
# install.packages('e1071')
library(e1071)
classifier = svm(formula = Purchased ~ .,
                 data = training_set,
                 type = 'C-classification',
                 kernel = 'radial')

# Predicting the Test set results
y_pred = predict(classifier, newdata = test_set[-3])

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

# Applying k-Fold Cross Validation
# install.packages('caret')
library(caret)
folds = createFolds(training_set$Purchased, k = 10)
cv = lapply(folds, function(x) {
  training_fold = training_set[-x, ]
  test_fold = training_set[x, ]
  classifier = svm(formula = Purchased ~ .,
                   data = training_fold,
                   type = 'C-classification',
                   kernel = 'radial')
  y_pred = predict(classifier, newdata = test_fold[-3])
  cm = table(test_fold[, 3], y_pred)
  accuracy = (cm[1,1] + cm[2,2]) / (cm[1,1] + cm[2,2] + cm[1,2] + cm[2,1])
  return(accuracy)
})
accuracy = mean(as.numeric(cv))
```

```
# Applying Grid Search to find the best parameters
# install.packages('caret')
library(caret)
classifier = train(form = Purchased ~ ., data = training_set, method = 'svmRadial')
classifier
classifier$bestTune
```

XGBoost

```
# Importing the dataset
dataset = read.csv('Churn_Modelling.csv')
dataset = dataset[4:14]
# Encoding the categorical variables as factors
dataset$Geography = as.numeric(factor(dataset$Geography,
                                      levels = c('France', 'Spain', 'Germany'),
                                      labels = c(1, 2, 3)))
dataset$Gender = as.numeric(factor(dataset$Gender,
                                   levels = c('Female', 'Male'),
                                   labels = c(1, 2)))
# Splitting the dataset into the Training set and Test set
library(caTools)
set.seed(123)
split = sample.split(dataset$Exited, SplitRatio = 0.8)
training_set = subset(dataset, split == TRUE)
test_set = subset(dataset, split == FALSE)
```

```
# Fitting XGBoost to the Training set
#install.packages('xgboost')
library(xgboost)
classifier = xgboost(data = as.matrix(training_set[-11]), label =
training_set$Exited, nrounds = 10)


# Predicting the Test set results
y_pred = predict(classifier, newdata = as.matrix(test_set[-11]))
y_pred = (y_pred >= 0.5)
# Making the Confusion Matrix
cm = table(test_set[, 11], y_pred)


# Applying k-Fold Cross Validation
library(caret)
folds = createFolds(training_set$Exited, k = 10)
cv = lapply(folds, function(x) {
  training_fold = training_set[-x, ]
  test_fold = training_set[x, ]
  classifier = xgboost(data = as.matrix(training_set[-11]), label =
training_set$Exited, nrounds = 10)
  y_pred = predict(classifier, newdata = as.matrix(test_fold[-11]))
  y_pred = (y_pred >= 0.5)
  cm = table(test_fold[, 11], y_pred)
  accuracy = (cm[1,1] + cm[2,2]) / (cm[1,1] + cm[2,2] + cm[1,2] + cm[2,1])
  return(accuracy)
})
accuracy = mean(as.numeric(cv))
```