Machine Learning

R and Hadoop

Introduction

- Machine learning concepts are used to
  - enable applications to take a decision from
    - the available datasets.
- used to develop
  - **spam mail detectors,**
  - **self-driven cars,**
  - **speech recognition,**
  - **face recognition, and**
  - **online transactional fraud-activity detection.**

Introduction..

- popular organizations are using
  - machine-learning algorithms
  - to make their service or product understand the need of their users
  - and provide services as per their behavior.

  - **Google intelligent web search engine**
  - **Spam classification in Google Mail**
  - **News labeling in Google News and**
  - **Amazon for recommender systems**.

Introduction.....

- There are many open source frameworks available for developing these types of applications/frameworks:
  - **R,**
  - **Python,**
  - **Apache Mahout, and**
  - **Weka.**

Introduction........

- **Three different types** of machine-learning algorithms for intelligent system development:

  - **Supervised machine-learning algorithms**
  - **Unsupervised machine-learning algorithms**
  - **Recommender systems**

Supervised machine-learning algorithms

Linear Regression(1)

- Linear regression is mainly used for
  - predicting and forecasting values
  - based on historical information

- Regression is a technique to identify the linear relationship between
  - target variables (that are going to be predicted)

    and
  - explanatory variables(going to help predict the target variables).

Linear Regression(2)

- In mathematics, regression can be formulated as follows:
  - y = ax +e   (simple linear regression)
    - a: slope; e: error
  - The slope of the regression line is given by:
    $a= (N\Sigma xy - (\Sigma x)(\Sigma y)) / (N\Sigma x^2 - (\Sigma x)^2)$
  - the intercept point of regression is given by:
    $e = (\Sigma y - b(\Sigma x)) / N$
- Suppose we have the data shown in the following table:

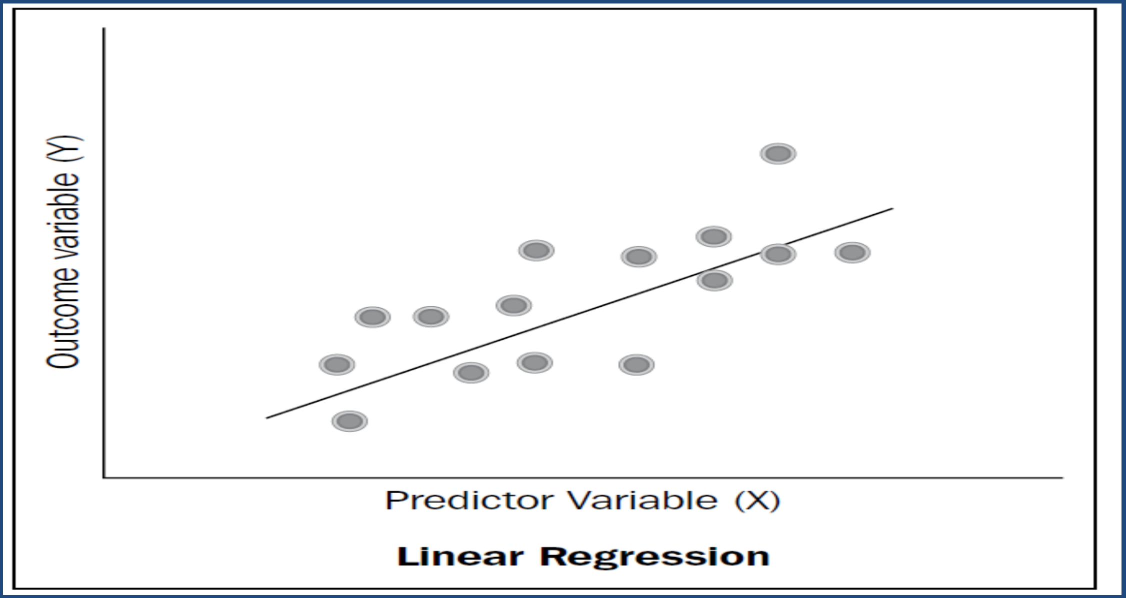
x	**y**
63	3.1
64	3.6
65	3.8
66	4

- If we have a new value of *x*,
  - we can get the value of *y* with it with the help of the regression formula/model.

Linear Regression(3)

Multiple Linear Regression

- $Y = e_0 + a_0x_0 + a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4$
- Here, *Y* is the target variable (response variable), *xi* are explanatory variables,
- and e_0 is the sum of the squared error term



Linear Regression(4)

Applications:

- Sales forecasting
- Predicting optimum product price
- Predicting the next online purchase from various sources and campaigns

Linear Regression with R(1)

- **Model <-lm(target ~ ex_var,**
  data=train_dataset)
- It will build a regression model based on the property of the provided dataset

  and
- store all of the variables' coefficients and model parameters

  used for
- predicting and identifying of data pattern from the model variable values.

Linear Regression with R(2)

Simple Linear Regression

- X1 = rnorm(20)
- y1 = rnorm(20)
- lm(X1~y1)

- Call:
- lm(formula = X1 ~ y1)

- Coefficients:
- (Intercept)   y1
- 0.01192   -0.10478

Linear Regression with R(3)

Predicting New Values

- m<- lm(y~ u+v+w)
- preds<- data.frame(u=3.1, v=4.0, w=5.5)
- predict(m, newdata = preds)

Example:

y <- x + rnorm(15)

x <- rnorm(15)

m<-lm(y~x)

p<-data.frame(x=0.8)

predict(m,newdata=p)

predict(m,newdata=p, interval='prediction‘)

Linear Regression with R(9)

Multiple Linear Regression

 # Defining data variables

```
X = matrix(rnorm(2000), ncol = 10)
y = as.matrix(rnorm(200))
```

 # Bundling data variables into dataframe

```
train_data <- data.frame(X,y)
```

 # Training model for generating prediction

```
lmodel<- lm(y~ train_data $X1 + train_data $X2 + train_data $X3 +
train_data $X4 + train_data $X5 + train_data $X6 + train_data $X7 +
train_data $X8 + train_data $X9 + train_data $X10, data= train_data)
```

Linear Regression in R(10)

```
> summary(lmodel)

Call:
lm(formula = y ~ train_data$x1 + train_data$x2 + train_data$x3 +
    train_data$x4 + train_data$x5 + train_data$x6 + train_data$x7 +
    train_data$x8 + train_data$x9 + train_data$x10, data = train_data)

Residuals:
     Min       1Q   Median       3Q      Max 
-2.63032 -0.63309 -0.07399  0.62334  2.83372 

Coefficients:
                Estimate Std. Error t value Pr(>|t|)   
(Intercept)    -0.166414   0.070605  -2.357  0.01945 * 
train_data$x1   0.031970   0.071050   0.450  0.65325   
train_data$x2  -0.089957   0.072481  -1.241  0.21611   
train_data$x3   0.067545   0.069906   0.966  0.33517   
train_data$x4   0.187189   0.071434   2.620  0.00949 **
train_data$x5  -0.049948   0.072221  -0.692  0.49004   
train_data$x6   0.019923   0.071427   0.279  0.78060   
train_data$x7   0.013168   0.074747   0.176  0.86035   
train_data$x8   0.079554   0.074907   1.062  0.28957   
train_data$x9  -0.008961   0.068948  -0.130  0.89674   
train_data$x10 -0.110755   0.067407  -1.643  0.10203   
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.9841 on 189 degrees of freedom
Multiple R-squared:  0.06692,	Adjusted R-squared:  0.01755 
F-statistic: 1.355 on 10 and 189 DF,  p-value: 0.204
```

Linear Regression in R(11)

Residuals: Observed – Predicted. Both the sum and the mean of the residuals are equal to zero

Ideally normal distribution with median =0.   Deviation indicates skew

Estimate:

Estimated regression coefficients. A zero value → variable is worthless

t value and P(>|t|): How likely true coefficient is zero?

t value is coefficient divided by its *standard error*

Variables with large p value(>0.05, likely to be insignificant) are candidate for elimination [Note: ***, **, *, .]

Residual standard error:

sample standard deviation of Ɛ

Linear Regression in R(12)

Degrees of Freedom (**DOF**):

This is used for identifying the degree of fit for the prediction model, which should be as small as possible (logically, the value 0 means perfect prediction).

Multiple R-Squared; Adjusted R-squared(accounts for no. of variables):

Measure of model's quality. Bigger is better

F Statistic:

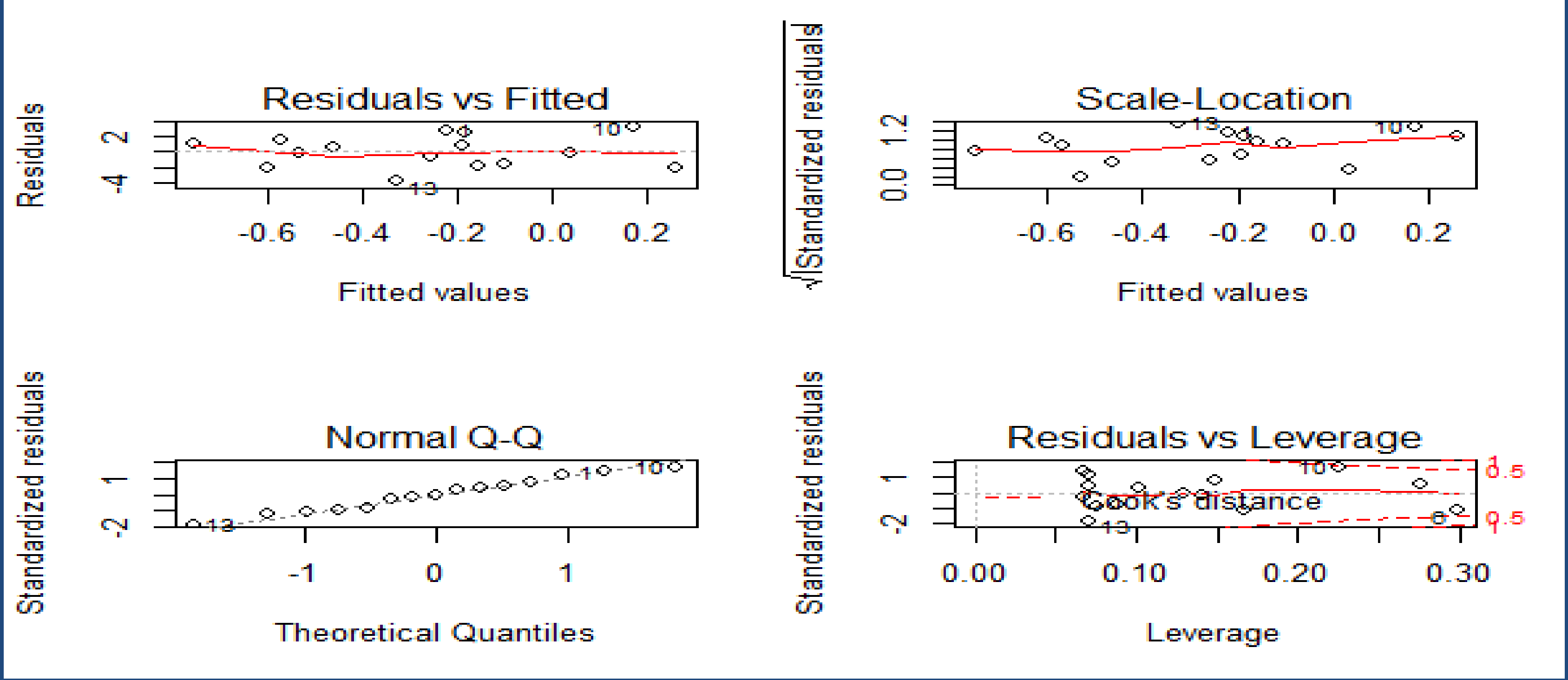
tells whether model is significant or insignificant(if all coefficients are zero). P-value <0.05 indicates model is likely significant

Linear Regression with R(4)

Results of linear Regression

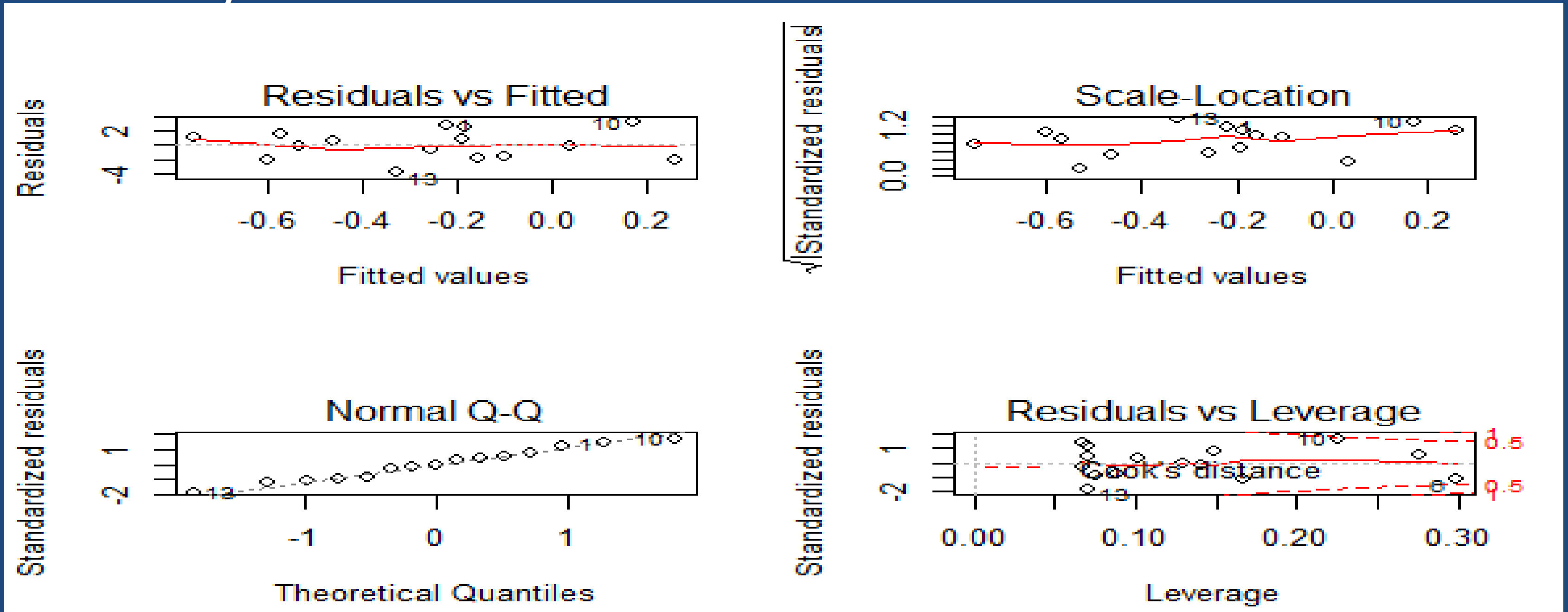
> layout(matrix(1:4,2,2))

> plot(m)



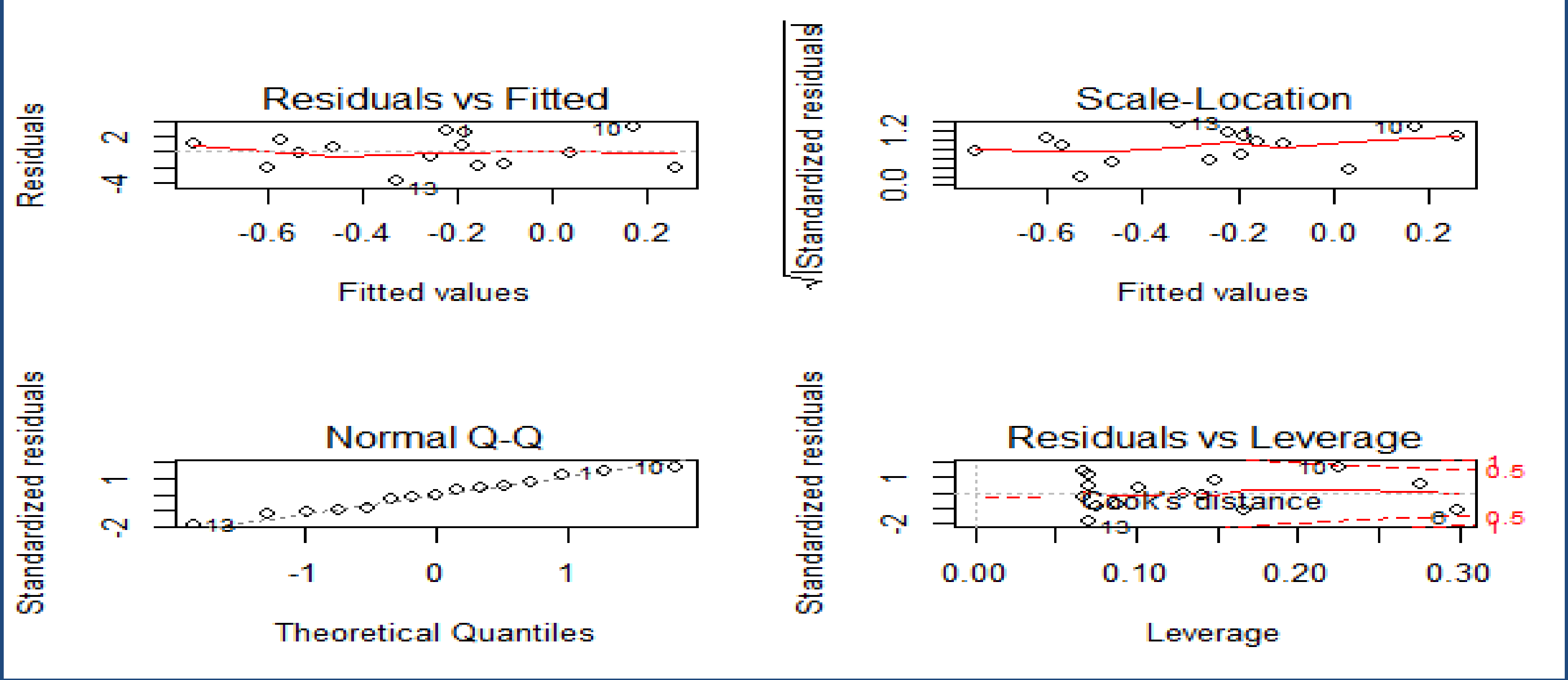
Linear Regression in R(5)

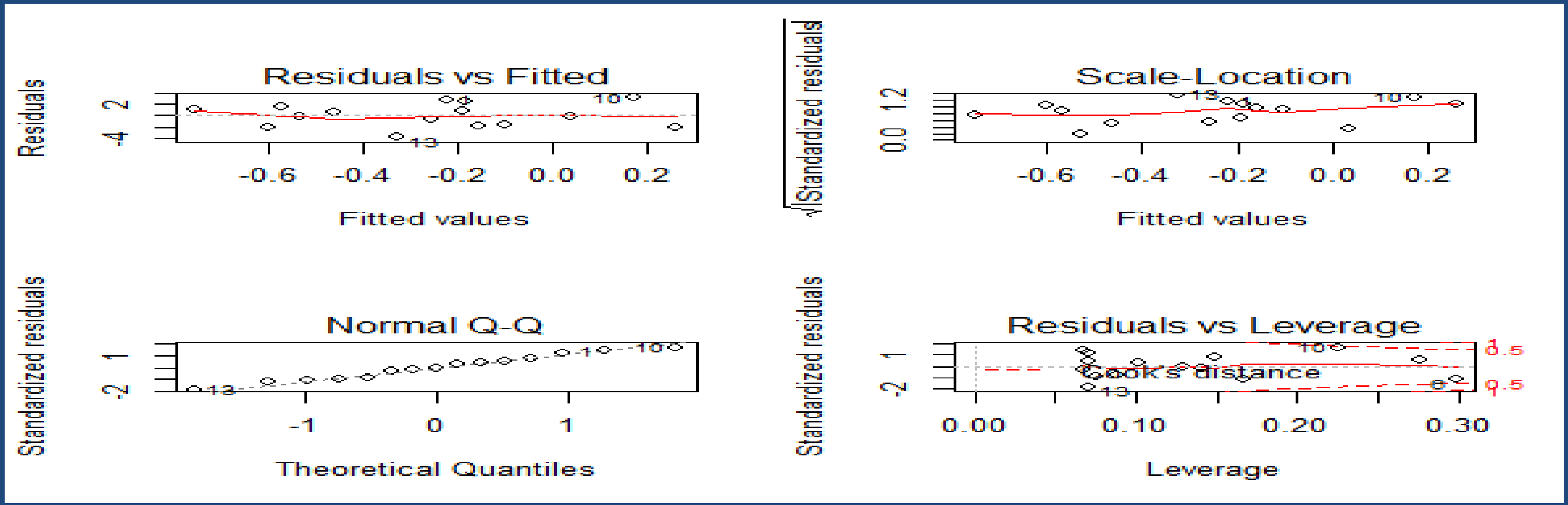
- Residual errors plotted versus their fitted values. The residuals should be randomly distributed around the horizontal line representing a residual error of zero; that is, there should not be a distinct trend in the distribution of points.
- Q-Q plot, which should suggest that the residual errors are normally distributed.



Linear Regression in R(6)

- The scale-location plot in the upper right shows the square root of the standardized residuals (sort of a square root of relative error) as a function of the fitted values. Again, there should be no obvious trend in this plot.





- Finally, the plot in the lower right shows each points leverage, which is a measure of its importance in determining the regression result.
- Superimposed on the plot are contour lines for the Cook's distance, which is another measure of the importance of each observation to the regression.
- Smaller distances means that removing the observation has little affect on the regression results. Distances larger than 1 are suspicious and suggest the presence of a possible outlier or a poor model.

Linear Regression: R and Hadoop(1)

- Parallel Linear Regression – Map and Reduce
- The outline of the linear regression algorithm is as follows:
  1. Calculating the Xtx value with MapReduce job1.
  2. Calculating the Xty value with MapReduce job2.
  3. Deriving the coefficient values with

     Solve (Xtx, Xty) #b = $(X^T X)^{-1} X^T y$

Linear Regression: R and Hadoop(2)

- Data Set:

```
# Defining the datasets with Big Data matrix X
    X = matrix(rnorm(20000), ncol = 10)
    X.index = to.dfs(cbind(1:nrow(X), X))
    y = as.matrix(rnorm(2000))

# Function defined to be used as reducers
    Sum =
    function(., YY)
    keyval(1, list(Reduce('+', YY)))
```

Linear Regression: R and Hadoop(3)

1. Calculating the Xtx value with MapReduce job1.

```
# XtX =
values(
# For loading hdfs data in to R
from.dfs(
# MapReduce Job to produce XT*X
mapreduce(
input = X.index,
```

Linear Regression: R and Hadoop(4)

1. Calculating the Xtx value with MapReduce job1…..

```
# Mapper – To calculate and emitting XT*X
map =
function(., Xi) {
yi = y[Xi[,1],]
Xi = Xi[,-1]
keyval(1, list(t(Xi) %*% Xi))},
# Reducer – To reduce the Mapper output by performing sum
operation over them
reduce = Sum,
combine = TRUE)))[[1]]
```

Linear Regression: R and Hadoop(5)

2. Calculating the Xty value with MapReduce job2.

```
Xty = values(
# For loading hdfs data
from.dfs(
# MapReduce job to produce XT * y
mapreduce(
input = X.index,
# Mapper – To calculate and emitting XT*y
map = function(., Xi) {
yi = y[Xi[,1],]
Xi = Xi[,-1]
keyval(1, list(t(Xi) %*% yi))},
```

Linear Regression: R and Hadoop(6)

2. Calculating the Xty value with MapReduce job2.

Reducer – To reducer the Mapper output by performing # sum

operation over them

reduce = Sum,

combine = TRUE))) [[1]]

Linear Regression: R and Hadoop(7)

3. Deriving the coefficient values with

solve (Xtx, Xty).

- Using Calculus rules for matrices, it can be derived that the ordinary least squares estimates of the coefficients are calculated using the matrix formula
- $b = (X^T X)^{-1} X^T y$,

Linear Regression: R and Hadoop(8)

- Output:

```
> solve(XtX, Xty)
              [,1]
 [1,]  0.038845121
 [2,]  0.015100617
 [3,]  0.012841903
 [4,] -0.033987022
 [5,] -0.004162355
 [6,] -0.175773152
 [7,] -0.080512728
 [8,]  0.036393052
 [9,] -0.063170450
[10,]  0.073065252
```

Unsupervised machine-learning algorithms

Clustering(1)

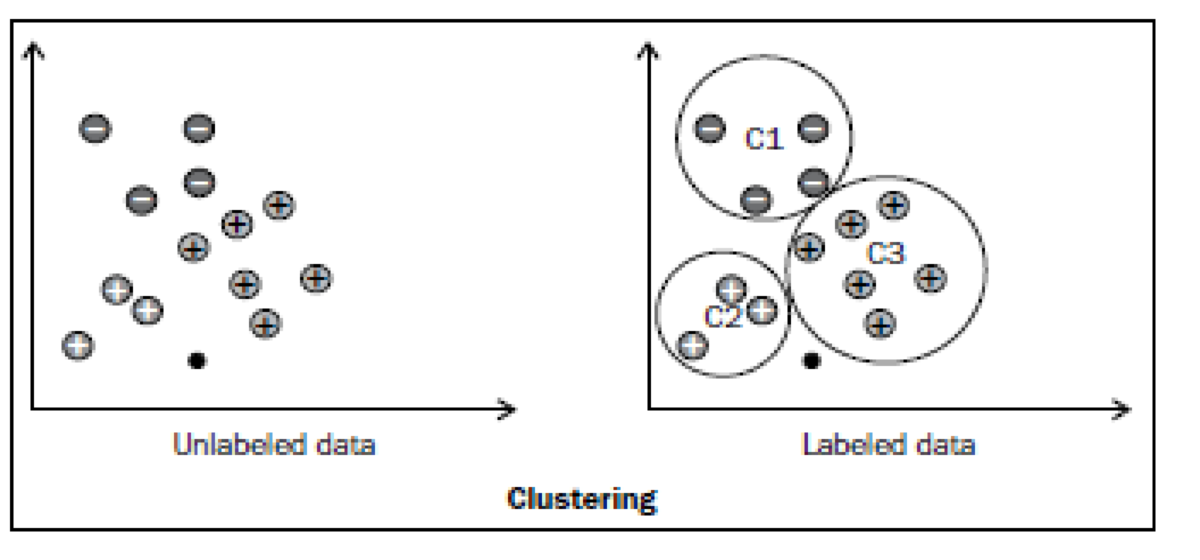
- unsupervised learning is used for
  - finding the hidden structure
    - from the unlabeled dataset.

Clustering(2)

- Clustering is the task of grouping a set of object in such a way that
  - similar objects with similar characteristics are grouped in the same category,
  - but other objects are grouped in other categories

Clustering(3)

From the following figure, we can identify clustering as grouping objects based on their similarity:



Clustering(4)

Clustering techniques available within R libraries:

- k-means,
- k-medoids,
- hierarchical, and
- density-based clustering.

Among them, k-means is widely used

Clustering(5)

Applications of clustering are as follows:

- Market segmentation
- Social network analysis
- Organizing computer network
- Astronomical data analysis

Clustering(6)

```
# Loading iris flower dataset
data("iris")

# generating clusters for iris dataset
kmeans <- kmeans(iris[, -5], 3, iter.max = 1000)

#cluster centroids
kmeans$centers


# comparing iris$Species with generated cluster points
>kmeans$cluster
>Iris[,5]

>table(iris[, 5], kmeans$cluster)
```

Clustering (7)

- plot(iris, col=iris$Species)
- plot(iris, col=kmeans$cluster)

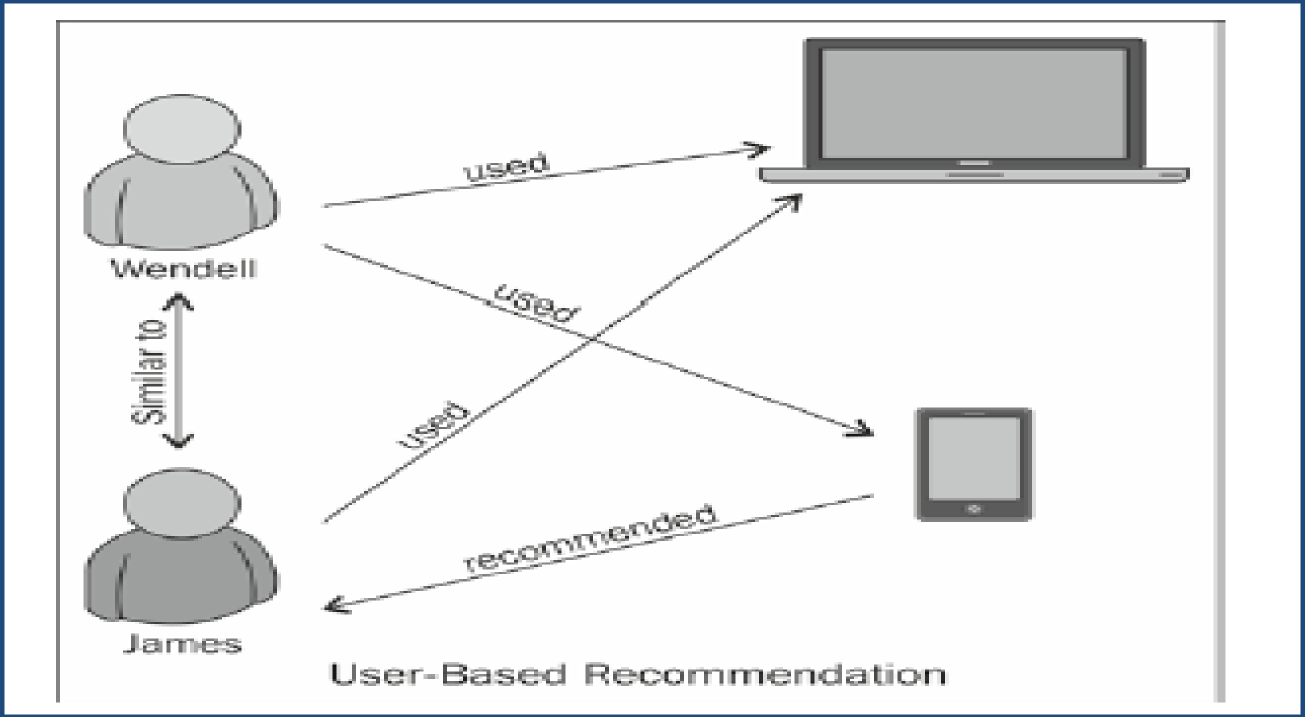
Recommendation Algorithms(1)

- Recommendation is a
  - machine-learning technique to predict what new items a user would like
  - based on associations with the user's previous items.

- Recommendations are widely used in the field of e-commerce applications.
  - **'Customers Who Bought This Item Also Bought'** window

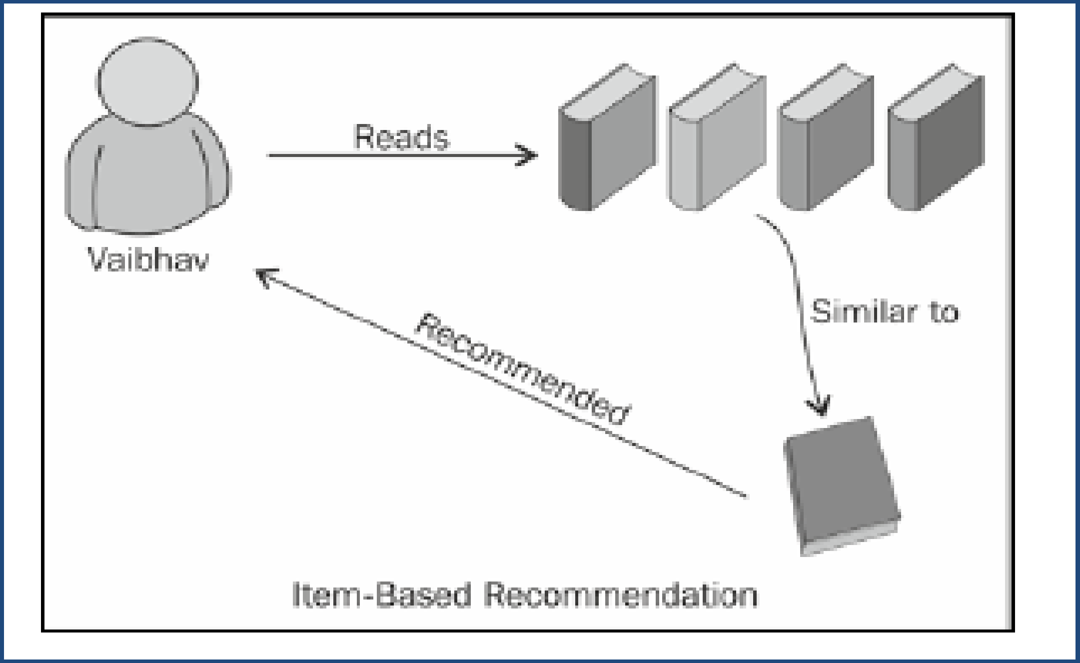
Recommendation Algorithms(2)

- Types of recommendations: User based



Recommendation Algorithms(3)

- Types of recommendations: Item based



Recommendation Algorithms(4)

Collaborative Filtering Algorithm:

Recommendations can be derived from the matrix-factorization technique as follows:

- Recommended Results =
  Co-occurrence matrix * scoring matrix

To generate the recommenders, we will follow the given steps:

1. Computing the co-occurrence matrix.
2. Establishing the user-scoring matrix.
3. Generating recommendations

Recommendation Algorithms(5)

Data set:

----

	user	item	pref
1	1	101	5.0
2	1	102	3.0
3	1	103	2.5
4	2	101	2.0
5	2	102	2.5
6	2	103	5.0
7	2	104	2.0
8	3	101	2.0
9	3	104	4.0
10	3	105	4.5
11	3	107	5.0
12	4	101	5.0
13	4	103	3.0
14	4	104	4.5
15	4	106	4.0
16	5	101	4.0
17	5	102	3.0
18	5	103	2.0
19	5	104	4.0
20	5	105	3.5
21	5	106	4.0

Recommendation Algorithms(6)

```
---------------------------------------------------------
> data
  user item pref idx
1    1  101  5.0   1
2    1  102  3.0   2
3    1  103  2.5   3
4    2  101  2.0   1
5    2  102  2.5   2
6    2  103  5.0   3
7    2  104  2.0   4
8    3  101  2.0   1
9    3  104  4.0   4
10   3  105  4.5   5
11   3  107  5.0   7
12   4  101  5.0   1
13   4  103  3.0   3
14   4  104  4.5   4
15   4  106  4.0   6
16   5  101  4.0   1
17   5  102  3.0   2
18   5  103  2.0   3
19   5  104  4.0   4
20   5  105  3.5   5
21   5  106  4.0   6
---------------------------------------------------------
```

Recommendation Algorithms(7)

Co-occurrence matrix

```
> co
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,]    0    0    0    0    0    0    0
[2,]    0    0    0    0    0    0    0
[3,]    0    0    0    0    0    0    0
[4,]    0    0    0    0    0    0    0
[5,]    0    0    0    0    0    0    0
[6,]    0    0    0    0    0    0    0
[7,]    0    0    0    0    0    0    0
```

Recommendation Algorithms(8)

User 1

```
-------------------------------------------------------------
>u=1
> idx
[1] 1 2 3
-------------------------------------------------------------

-------------------------------------------------------------
> m
  x y
1 1 1
2 2 1
3 3 1
4 1 2
5 2 2
6 3 2
7 1 3
8 2 3
9 3 3
-------------------------------------------------------------
```

Recommendation Algorithms(9)

Co-occurrence matrix with user 1 item combinations:

```
------------------------------------------------------------------
> co
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,]    1    1    1    0    0    0    0
[2,]    1    1    1    0    0    0    0
[3,]    1    1    1    0    0    0    0
[4,]    0    0    0    0    0    0    0
[5,]    0    0    0    0    0    0    0
[6,]    0    0    0    0    0    0    0
[7,]    0    0    0    0    0    0    0
------------------------------------------------------------------
```

Recommendation Algorithms(10)

co_occurrence matrix **# for u = 1 to 5**

```
--------------------------------------------------------

> co
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,]    5    3    4    4    2    2    1
[2,]    3    3    3    2    1    1    0
[3,]    4    3    4    3    1    2    0
[4,]    4    2    3    4    2    2    1
[5,]    2    1    1    2    2    1    1
[6,]    2    1    2    2    1    2    0
[7,]    1    0    0    1    1    0    1
--------------------------------------------------------
```

Recommendation Algorithms(11)

User 1:

```
  user item pref idx
1    1  101  5.0   1
2    1  102  3.0   2
3    1  103  2.5   3
> pref
[1] 5.0 3.0 2.5 0.0 0.0 0.0 0.0
```

Recommendation Algorithms(12)

```
User 1:
------------------------------------------------------------
  user item pref idx
1    1  101  5.0   1
2    1  102  3.0   2
3    1  103  2.5   3
> pref
[1] 5.0 3.0 2.5 0.0 0.0 0.0 0.0
------------------------------------------------------------
```

Recommendation Algorithms(13)

 # User Rating Matrix

```
-------------------------------------------------------
> userx
     [,1]
[1,]  5.0
[2,]  3.0
[3,]  2.5
[4,]  0.0
[5,]  0.0
[6,]  0.0
[7,]  0.0
-------------------------------------------------------
```

Recommendation Algorithms(14)

 # co-occurrence matrix * Scoring matrix

```
------------------------------------------------------
     [,1]
[1,] 44.0
[2,] 31.5
[3,] 39.0
[4,] 33.5
[5,] 15.5
[6,] 18.0
[7,]  5.0
------------------------------------------------------
```

Recommendation Algorithms(15)

 # Recommended Sort

```
------------------------------------------------------------
     [,1]
[1,]  0.0
[2,]  0.0
[3,]  0.0
[4,] 33.5
[5,] 15.5
[6,] 18.0
[7,]  5.0
------------------------------------------------------------
```

get the order of elements, in index

```
------------------------------------------------------------
> idx
[1] 4 6 5 7 1 2 3
------------------------------------------------------------
```

Recommendation Algorithms(16)

Top ‘n’ recommendations   for user 1

> topn

```
  user item  val
1    1  104 33.5
2    1  106 18.0
3    1  105 15.5
4    1  107  5.0
5    1  101  0.0
6    1  102  0.0
7    1  103  0.0
```

Recommendation Algorithms(17)

#recommendation for user 2 and 4

	user	item	val
8	2	106	20.5
9	2	105	15.5
10	2	107	4.0
11	2	101	0.0
12	2	102	0.0
13	2	103	0.0
14	2	104	0.0

15	3	103	24.5
16	3	102	18.5
17	3	106	16.5
18	3	101	0.0
19	3	104	0.0
20	3	105	0.0
21	3	107	0.0

Recommendation Algorithms(18)

#recommendation for user 4 and 5

	user	item	val
22	4	102	37.0
23	4	105	26.0
24	4	107	9.5
25	4	101	0.0
26	4	103	0.0
27	4	104	0.0
28	4	106	0.0

29	5	107	11.5
30	5	101	0.0
31	5	102	0.0
32	5	103	0.0
33	5	104	0.0
34	5	105	0.0
35	5	106	0.0