→ YARN

Resource manager
Application Manager
Containers

7/07/2023

* Rprogramming

- Statistical Measures

⇒ Descriptive : Inferential

1. central tendancy or moments (eg. mean, mode, median)
2. Measures of positions → Quantiles (fractions)
3. " " Distribution → skewness and kurtosis
4. " " Dispersion (Spread) → variance, std. deviation, co variance, corelation coeff

⇒ Inferential

- Probability

eg: population, sample

→ P-test
→ t-test
→ z-test
→ chi-square test

* installing packages in R

```
install.packages ("moments")
library (moments)
```

```
> x = c (4, 1, 8, 9, 10)
> min (x)
> max (x)
> sort (x)
> range (x)
> var (x)
> sd (x)
> cov (x, y)
> cor (x, y)
> Skewness (x)
> kurtosis (x)
> mean (x)
> median (x)
```

Std. devi ⇒ $\sigma = \sqrt{\frac{\Sigma(x-\bar{x})^2}{n}}$

Var $= \sigma^2 = \frac{\Sigma(x-\bar{x})^2}{n}$

Population ⇒ /n

Sample ⇒ /n-1

⇒ mode has no predefined function in 'R'

```
> quantile (x, probs = 0.10)
Q1 = quantile (x, probs = 0.25)  → quartile
     quantile (x, probs = 0.50)  → median
Q3 = quantile (x, probs = 0.75)  → quartile
> IQR (Q3 - Q1)
```

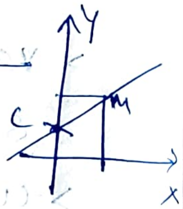
Quantile $= \frac{\text{fractionper}}{100}(n+1)$

```
> data ()           // display all predefined datasets
> data (iris)       // iris dataset
> str (iris)        // gives attributes
> summary (iris)    // gives summary
> iris [1:150,]          // entire dataset

> iris [1:10,]           // 10 rows

> iris [1:150, -5]       // displays only 4 columns

> iris [1:150,c(-4,-5)]  // removes last two col,
                         // more than one value shd
                            give in vector
```

Programs

1. Linear Regression

x	y
1	2
2	4
3	6
4	8



```
> x = c (1,2,3,4)

> y = c (2,4,6,8)

> z = data.frame (x, y)

> lr = lm (y ~ x,                 // linear model is predefined func
           data = z)
                                  // multiple => lm (y~(x1+x2)
> print (lr)   // gives summary                    , data = )

> new = data.frame (x = 5)

> P = predict (lr, new)

> print (P)
```

2. Multiple Regression

3. Decision Tree Model (DTM)

→ recursive partition regression trees

```
> install . packages (rpart)
> library ( class)
> library ( rpart.plot)

> x = iris [1:140, ]
> y = iris [141:150, ]
> dtm = rpart (Specces ~ (sepal.length + sepal. width +
                    petal. length + petal. width)),
                              data = x , method = "class")
                (or)
  dtm = rpart (Specces ~ ., data = x , method = "class")

> plot (dtm)
> text (dtm)   // gives summary based on branch
> P = predict (dtm, y[,-5] , type = "class")
> print (P)
> table (y[,5], P)
```

	S	VC	V
S	0	0	0
VC	0	0	0
V	0	0	10

- Randomforest : iterative of multiple DTM's

14/07

Scala

⇒ Scalable language

Hadoop 1.0	Hadoop 2.0	Hadoop 3.0
Hdfs + MapReduce	Hdfs + MapReduce + YARN	Hdfs + MapReduce + Spark → load balancing

$\underbrace{\text{Hadoop 1.0, Hadoop 2.0}}$ Modules developed by R/python

$\underbrace{\text{Hadoop 3.0}}$ Modules developed by: Scala

Scala programs

Syntax

```
 object mca
 {
     def main (args: Array[String])
     {
      //
     }
 }
Class classname
{
   def f1()
   {
   }//
  def f2()
  { //
  }
```

→ Val : Value which is constant

→ Var : Variable which is not fixed

Syntax : Val x: Int =95; → Immutable

Var x: Int =100; → mutable

→ object declaration

Syntax : Var obj = new classname();

```
* object mca
  {
     def main (args: Array[String])
     {
         var x: Int =50 ;
         Var y: Double= 49.99 ;
         Var z: String = "Rani";
     }   Print (x+" "+y+" "+z);
  }
```

* Reading i/p directly from keyboard

```
var x: Int = Scala.io.StdIn.readLine.toInt;

var y: String = Scala.io.StdIn.readLine;
```

```
Object peuledu
{
   def main (args: Array[String])
   {
      Var z: Double = Scala.io.StdIn.readLine.toDouble;
      Print(z);
}  }
```

* Static Single dimensional array

```
Object areyKoushik
{
    def main (args : Array [String])
    {
        var x = Array (3, 4, 5, 8, 6);
        var y = Array (Array (1,2,3,4),
                              Array (4, 3, 2,1));
        //var x: Array[Int] = Array (1,2,3,4);
        ~~print~~
        for (i ← 0 to 4)
        {
            print (x(i));
        }
        for (i ← 0 to 1)
        {
            for (j ← 0 to 4)
            {
                print (y(i)(j));
            }
        }
    }
}
```

* Dynamic Single D & Two D array

```
object nasavnensasta
{
    def main (args: Array [String])
    {
        var x = Array [Int] (5);
        for (i <- 0 to 4)
        {
            x(i) = Scala. io. StdIn. readLine. toInt;
        }

        for (i <- 0 to 4)
        {
            println(x(i));
        }

        var y = Array.ofDim [Int] (2,5);
        for (i <- 0 to 1)
        {
            for (j <- 0 to 4)
            {
                y(i)(j) = Scala.io.StdIn.readLine.toInt;
            }
        }
        for (i <- 0 to 1)
        {
            for (j <- 0 to 4)
            {
                Println (y(i)(j));
        }   }
    }
}
```

```
for (i <- 0 to 1;
       j <- 0 to 4)
{
    Print ( y(i)(j))
}
```

Exercise

→ Create method overloading

→ constructor overloading

→ simple Inheritance

→ method overriding

28/7

→ Real time appl. for method overloading & constr. overloading

4/08

* Spark

→ Hadoop 3.0

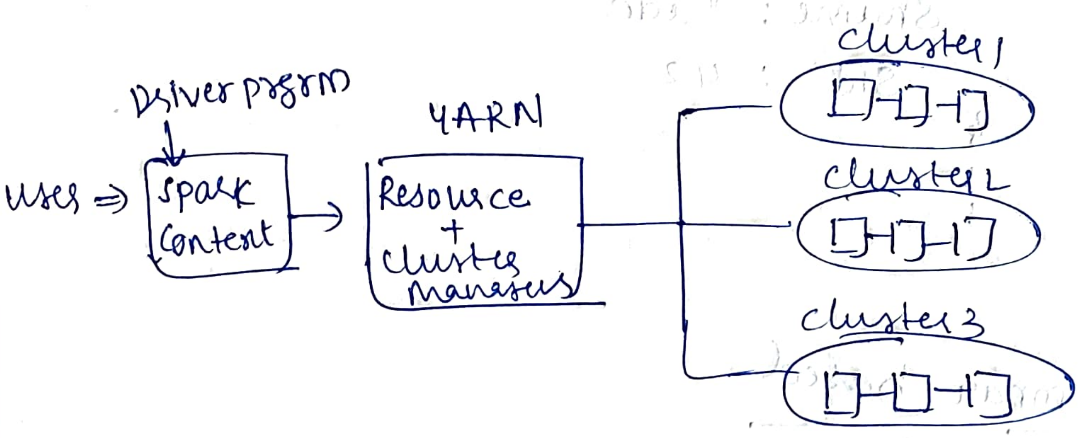
• Spark Ecosystem :-

Data Mangement	Ozzie	Zookeeper	R,Java Python/Scala
Data Accessing	Spark MLLib	Spark Streaming	Spark SQL
Data Processing	Map Reduce	Spark → RDD Core	
Data Storage	Rdbms	Spark HBASE	

- RDD : Re-Silent Distributed Databases
  ↳ Spark Context

  - Re-Silent ⇒ fault-tolerance

→ In-memory processing is done in Spark



* NOSQL

→ Not Only SQL Databases

1. Key-value based
2. columnar - based
3. Graph - based
4. Document - based

11/03

→ BASE properties

BA → Basically Available.
S → Soft State
E → Eventually consistency

1. Key-value based

relation : attributekey = "value"

eg Student : "sname" = "xyz"
Student : "sid" = 45

2. **Document - based**

→ Json / BSON

Student

{

{

Sname : "adc"

Sid : 42

}

:

}

Bson
↓
either 1 or 0

3. **columnar - based**

→ Dynamically create column for the row we needed instead of creating a column for entire table like in RDBMS

↙
row-based seggregation

abc	48	98458	
xyz	49	85698	9.6

column-based

abc	48	9858
xyz	49	8568

- row based takes more traversal time than column based

- row based → can increase as many columns as we need