DS PRACTICALS

DATA SCIENCE

Data science is a field of study that involves using scientific methods, processes, algorithms, and systems to extract insights and knowledge from data. It combines principles from statistics, computer science, and domain-specific fields to analyze, interpret, and draw conclusions from complex and large-scale data.

Data science involves various stages, such as data collection, cleaning, processing, analysis, visualization, and interpretation. It also includes machine learning, which is a branch of artificial intelligence that enables computers to learn from data and make predictions or decisions based on that learning.

Data science is used in a wide range of industries, such as finance, healthcare, marketing, and transportation, to extract value from data and gain insights into customer behavior, operational performance, risk analysis, and other critical areas. The applications of data science are vast and continue to expand as more data is generated and collected in various domains.

DATA

Data refers to any information or facts that can be collected, recorded, and analyzed. It can be in various formats, such as text, numbers, images, audio, or video.

Data can be categorized into two types: structured and unstructured. Structured data is organized in a predefined format, such as tables or spreadsheets, and can be easily processed by machines. Unstructured data, on the other hand, does not have a fixed format and is more difficult to process, such as text documents, images, and videos.

Data is generated from a wide range of sources, such as sensors, social media, transactions, surveys, and various other sources. With the growth of digital technologies and the Internet, the amount of data being generated and collected is increasing rapidly. This has led to the emergence of data science as a field of study, as organizations seek to extract insights and value from the vast amounts of data available to them.

Practical 1

Aim : Practical of data collection, curation and management using couch DB

Theory:

Data management refers to the process of organizing, storing, protecting, and maintaining data throughout its lifecycle. It involves the collection, processing, and analysis of data to provide insights for decision-making and other purposes.

Data collection is the process of gathering information and data from various sources, such as surveys, sensors, social media, and other data sources. This data can be either structured or unstructured.

Data curation is the process of organizing, cleaning, and transforming data to ensure its accuracy, completeness, and consistency. This includes tasks such as data cleaning, data normalization, data integration, and data enrichment.

CouchDB is a type of NoSQL database that stores data in a document-oriented format, rather than in tables with rows and columns. It is designed to handle unstructured data and is highly scalable, making it an ideal choice for modern applications.

Unstructured data refers to data that is not organized in a predefined format, such as text, images, audio, and video. It is often stored in its original form and requires specialized tools for processing and analysis.

NoSQL refers to a class of database management systems that do not use traditional relational database structures. Instead, they use a variety of non-tabular structures, such as key-value pairs, document-oriented databases, and graph databases, to store and retrieve data. NoSQL databases are often used for big data and real-time applications, where speed and scalability are important factors.

Code:

```
install.packages("sofa")

library(sofa)

x<-Cushion$new()

x$ping()
```

```
db_create(x,dbname = 'ty')

db_list(x)

doc1<-'{"rollno":"01","name":"ABC","GRADE":"A"}'

doc_create(x,doc1,dbname = "ty",docid = "a_1")

doc2<-'{"rollno":"02","name":"PQR","GRADE":"A"}'

doc_create(x,doc2,dbname = "ty",docid = "a_2")

doc3<-'{"rollno":"03","name":"xyz","GRADE":"B","REMARK":"PASS"}'

doc_create(x,doc3,dbname = "ty",docid = "a_3")

db_changes(x,"ty")

db_query(x,dbname = "ty", selector = list('_id'=list('$gt'=NULL)))$docs

db_query(x,dbname = "ty",selector = list(GRADE="A"))$docs

db_query(x,dbname = "ty",selector = list(REMARK="PASS"))$docs

db_query(x,dbname = "ty",selector = list(rollno=list('$gt'='02')),fields=c("name
","GRADE"))$docs


library("jsonlite")

res<-db_query(x,dbname = "ty",selector = list('_id'=list('$gt'=NULL)),fiel

ds=c("name","rollno","GRADE","REMARK"),as="json")


fromJSON(res)$docs


doc_delete(x,dbname = "ty",docid = "a_2")


doc_update(x,dbname = "ty",doc=doc2,docid="a_3",rev = "1-
f13d2a583fc7fd0d645d421014a295b2
```

Practical 2:

Aim : Data collection , curation and management using Mongo DB

Theory :

Data management refers to the process of organizing, storing, protecting, and maintaining data throughout its lifecycle. It involves the collection, processing, and analysis of data to provide insights for decision-making and other purposes.

Data collection is the process of gathering information and data from various sources, such as surveys, sensors, social media, and other data sources. This data can be either structured or unstructured.

Data curation is the process of organizing, cleaning, and transforming data to ensure its accuracy, completeness, and consistency. This includes tasks such as data cleaning, data normalization, data integration, and data enrichment.

MongoDB is a popular open-source NoSQL database management system. It is designed to handle large amounts of unstructured data and is particularly well-suited for handling big data and real-time applications.

MongoDB uses a document-oriented data model, which means that data is stored in flexible, JSON-like documents that can vary in structure and content. This makes it easy to store and retrieve data without having to define a fixed schema beforehand. MongoDB is also highly scalable and can run on a cluster of servers to handle high volumes of data and traffic.

Unstructured data refers to data that is not organized in a predefined format, such as text, images, audio, and video. It is often stored in its original form and requires specialized tools for processing and analysis.

NoSQL refers to a class of database management systems that do not use traditional relational database structures. Instead, they use a variety of non-tabular structures, such as key-value pairs, document-oriented databases, and graph databases, to store and retrieve data. NoSQL databases are often used for big data and real-time applications, where speed and scalability are important factors.

A database in MongoDB is a logical container for collections. It is a storage system that contains one or more collections, which in turn contain individual documents. A database is identified by its name, which is unique within the MongoDB instance.

A collection in MongoDB is a group of documents that are stored together within a database. Collections are the equivalent of tables in a traditional relational database, but unlike tables, collections can store documents with different fields and structures. Each document in a collection has a unique identifier (_id), which is used to retrieve, update or delete individual documents.

In MongoDB, databases and collections are created dynamically. This means that you do not have to define a schema before inserting data into a collection. MongoDB allows for a flexible schema design where documents in the same collection can have different fields or structures. This makes MongoDB particularly well-suited for handling unstructured data and allows for easy scaling and performance optimization.

Code:

To display anything, name.pretty()

Using pretty() method

Create database :

db command will show all the databases

use db_name to use or switch the the database(It will create it the db does not exists)

show dbs to show all the databases

db.dropDatabase()

Creating a collection :

db.colllection_name.insert({name:"john", age:19})

To check whether the collection is created successfully, use the following command.

```
show collections
```

Method 2: Creating collection with options before inserting the documents

We can also create collection before we actually insert data in it. This method provides you the options that you can set while creating a collection.

Syntax:

```
db.createCollection(name, options)
```

4.MongoDB Drop collection:

To drop a collection , first connect to the database in which you want to delete collection and then type the following command to delete the collection:

```
db.collection_name.drop()
```

5.MongoDB Insert Document:

Syntax to insert a document into the collection:

```
db.collection_name.insert({name:”John”,age:19})
```

6.MongoDB Query Document:

Find():

Querying all the documents in JSON format

Lets say we have a collection `students` in a database named `beginnersbookdb`. To get all the documents we use this command:

```
db.students.find()
```

8.Delete document in MongoDB:

MongoDB Delete Document from a Collection

In this tutorial we will learn how to delete documents from a collection. The remove() method is used for removing the documents from a collection in MongoDB.

Syntax of remove() method:

```
db.collection_name.remove(delete_criteria)
```

If you want to remove all the documents from a collection but does not want to remove the collection itself then you can use remove() method like this:

```
db.collection_name.remove({})
```

10.limit() and skip() method in MongoDB:

The limit() method in MongoDB

This method limits the number of documents returned in response to a particular query. Syntax:

```
db.collection_name.find().limit(number_of_documents)
```

Practical 3: Practical of PCA

Theory

PCA, or Principal Component Analysis, is a technique in data science used for reducing the dimensionality of a large dataset. It works by identifying the most important features, or principal components, of the data, and then projecting the data onto a new coordinate system that is defined by these principal components.

PCA is useful in data analysis and machine learning because it can help to identify patterns and relationships in high-dimensional data that may not be apparent using other methods. By reducing the number of features in a dataset, PCA can make it easier to visualize and understand the data, and can also help to improve the accuracy and efficiency of machine learning models.

In simple terms, PCA can be thought of as a way to simplify complex data by identifying the most important features and creating a new, more compact representation of the data that captures the essential information. This can be useful for a wide range of applications, from image and signal processing to finance and marketing.

Code:

```
data_iris <- iris[1:4]
Cov_data <- cov(data_iris )
Eigen_data <- eigen(Cov_data)
PCA_data <- princomp(data_iris ,cor="False")
Eigen_data$values
PCA_data$sdev^2
PCA_data$loadings[,1:4]
Eigen_data$vectors
summary(PCA_data)
biplot (PCA_data)
screeplot(PCA_data, type="lines")
model2 = PCA_data$loadings[,1]
model2_scores <- as.matrix(data_iris) %*% model2
library(class)
install.packages("e1071")
library(e1071)
mod1<-naiveBayes(iris[,1:4], iris[,5])
mod2<-naiveBayes(model2_scores, iris[,5])
table(predict(mod1, iris[,1:4]), iris[,5])
table(predict(mod2, model2_scores), iris[,5])
```

Practical 4

Aim : Clustering

Theory : Clustering is a technique in data science used to group a set of objects in such a way that objects in the same group, called a cluster, are more similar to each other than to those in other groups or clusters. Clustering is an unsupervised learning technique, which means that it does not require labeled data to perform its analysis.

The goal of clustering is to identify patterns or relationships in the data that may not be immediately apparent using other methods. Clustering can be used for a variety of tasks, such as identifying customer segments based on their behavior, grouping documents based on their content, or detecting anomalies in network traffic.

There are several algorithms used for clustering, such as K-Means, Hierarchical clustering, DBSCAN, and others. These algorithms differ in their approach to grouping objects into clusters, but they all rely on some measure of similarity or distance between the objects.

Clustering is a powerful technique in data science because it allows for the discovery of hidden structures or patterns in the data that can inform decision-making, optimization, and other applications.

CODE:

```
install.packages("ggplot2")
library(ggplot2)

scatter <- ggplot(data=iris, aes(x = Sepal.Length, y = Sepal.Width)) scatter
+ geom_point(aes(color=Species, shape=Species)) + theme_bw()+

  xlab("Sepal Length") + ylab("Sepal Width") +
  ggtitle("Sepal Length-Width")


ggplot(data=iris, aes(Sepal.Length, fill = Species))+
  theme_bw()+

  geom_density(alpha=0.25)+

  labs(x = "Sepal.Length", title="Species vs Sepal Length") vol <-

ggplot(data=iris, aes(x = Sepal.Length))


vol + stat_density(aes(ymax = ..density.., ymin = -..density.., fill =
                       Species, color = Species),

                   geom = "ribbon", position = "identity") +

  facet_grid(. ~ Species) + coord_flip() + theme_bw()+labs(x = "Sepal Length",
title="Species vs Sepal Length")


vol <- ggplot(data=iris, aes(x = Sepal.Width))


vol + stat_density(aes(ymax = ..density.., ymin = -..density.., fill =
                       Species, color = Species),

                   geom = "ribbon", position = "identity") +

  facet_grid(. ~ Species) + coord_flip() + theme_bw()+labs(x = "Sepal Width",
title="Species vs Sepal Width")


irisData <- iris[,1:4]
totalwSS<-c()
```

```
for (i in 1:15)

{

  clusterIRIS <- kmeans(irisData, centers=i)
  totalwSS[i]<-clusterIRIS$tot.withinss

}


plot(x=1:15,                          # x= No of clusters, 1 to 15
     y=totalwSS,                        # tot_wss for each

     type="b",                        # Draw both points as also connect them
     xlab="Number of Clusters",

     ylab="Within groups sum-of-squares")



install.packages("NbClust")
library(NbClust)

par(mar = c(2,2,2,2))

nb <- NbClust(irisData, method = "kmeans")

hist(nb$Best.nc[1,], breaks = 15, main="Histogram for Number of Clusters")



install.packages("vegan")
library(vegan)


modelData <- cascadeKM(irisData, 1, 10, iter = 100) # Test for clusters 1 to 10 plot(modelData,
sortg = TRUE)


modelData$results[2,]


which.max(modelData$results[2,])

library(cluster)

cl <- kmeans(iris[,-5], 2)

dis <- dist(iris[,-5])^2

sil = silhouette (cl$cluster, dis)

plot(sil, main = "Clustering Data with Silhoutte plot using 2 Clusters", col = c("cyan", "blue"))

library(cluster)
```

```
cl <- kmeans(iris[,-5], 8)

dis <- dist(iris[,-5])^2

sil = silhouette (cl$cluster, dis)


plot(sil, main = "Clustering Data with Silhoutte plot using 8 Clusters", col = c("cyan", "blue",
"orange", "yellow", "red", "gray", "green", "maroon")) install.packages("factoextra")

library(factoextra)
install.packages("clustertend")
library(clustertend)

genx<-function(x){ runif(length(x),
  min(x), (max(x)))

}

random_df <- apply(iris[,-5], 2, genx)


random_df <- as.data.frame(random_df)


iris[,-5] <- scale(iris[,-5]) random_df
<- scale(random_df)


res <- get_clust_tendency(iris[,-5],

                          n = nrow(iris) -1 ,
                          graph = FALSE)

res$hopkins_stat


hopkins(iris[,-5], n = nrow(iris) -1)

res <- get_clust_tendency(random_df, n = nrow(random_df)-1, graph
                          = FALSE)


res$hopkins_stat
```

PRACTICAL 5:

Aim : Practical of time series analysis

Theory : Time series analysis is a statistical technique used to analyze data that is collected over time. It involves studying patterns and trends in the data to identify underlying structures and relationships.

A time series is a set of observations or measurements taken at regular intervals over time. Time series analysis is used to model and forecast these observations based on their past behavior.

Trend refers to the long-term movement of the data in a particular direction, either up or down. A trend can be positive (increasing) or negative (decreasing), or it can be flat (remaining constant). Trends can be linear, meaning they increase or decrease at a constant rate, or nonlinear, meaning they increase or decrease at an increasing or decreasing rate.

Seasonality refers to the pattern of repeating fluctuations in the data that occur at regular intervals, such as daily, weekly, monthly, or yearly. Seasonality can be caused by various factors, such as weather patterns, holidays, or other seasonal events.

Cyclicity refers to longer-term patterns in the data that repeat over a longer period than seasonality, such as business cycles or economic cycles. Cyclical patterns can be caused by a variety of factors, such as changes in consumer behavior or shifts in market demand.

Time series analysis is useful in many applications, such as finance, economics, and weather forecasting, to make predictions about future behavior based on historical trends and patterns. By understanding the trend, seasonality, and cyclicity of a time series, we can make informed decisions and better allocate resources for a range of applications.

CODE:

```
data(AirPassengers)

class(AirPassengers)

start(AirPassengers)

end(AirPassengers)

frequency(AirPassengers)

summary(AirPassengers)

plot(AirPassengers)

abline(reg=lm(AirPassengers~time(AirPassengers)))

cycle(AirPassengers)
```

```
plot(aggregate(AirPassengers,FUN=mean))

boxplot(AirPassengers~cycle(AirPassengers))

acf(log(AirPassengers))

acf(diff(log(AirPassengers)))

(fit <- arima(log(AirPassengers), c(0, 1, 1),seasonal = list(order = c(0, 1, 1), period = 12)))

pred <- predict(fit, n.ahead = 10*12)

ts.plot(AirPassengers,2.718^pred$pred, log = "y", lty = c(1,3))

 frequency(AirPassengers)

summary(AirPassengers)

plot(AirPassengers)

abline(reg=lm(AirPassengers~time(AirPassengers)))

cycle(AirPassengers)

plot(aggregate(AirPassengers,FUN=mean))

boxplot(AirPassengers~cycle(AirPassengers))

acf(log(AirPassengers))

acf(diff(log(AirPassengers)))

(fit <- arima(log(AirPassengers), c(0, 1, 1),seasonal = list(order = c(0, 1, 1), period = 12)))

pred <- predict(fit, n.ahead = 10*12)

ts.plot(AirPassengers,2.718^pred$pred, log = "y", lty = c(1,3))
```

Practical 6

Aim: Simple multiple regression

Theory: Simple linear regression and multiple linear regression are statistical techniques used to model the relationship between a dependent variable and one or more independent variables.

Simple linear regression is used when there is only one independent variable and one dependent variable. It involves fitting a straight line to the data in order to model the relationship between the variables. The equation of the line is typically of the form Y = b0 + b1*X, where Y is the dependent variable, X is the independent variable, b0 is the intercept, and b1 is the slope of the line.

Multiple linear regression is used when there are multiple independent variables and one dependent variable. It involves fitting a linear equation to the data that includes all of the independent variables. The equation is typically of the form Y = b0 + b1*X1 + b2*X2 + ... + bn*Xn, where Y is the dependent variable, X1, X2, ..., Xn are the independent variables, b0 is the intercept, and b1, b2, ..., bn are the slopes of the lines for each independent variable.

In both simple and multiple linear regression, the goal is to find the best fit line that minimizes the sum of the squared errors between the predicted values and the actual values of the dependent variable. These techniques are commonly used in various fields, such as finance, economics, and social sciences, to make predictions and gain insights into the relationships between variables.

Code:

```
lsfit(iris$Petal.Length, iris$Petal.Width)$coefficients

plot(iris$Petal.Length, iris$Petal.Width, pch=21,
bg=c("red","green3","blue")[unclass(iris$Species)], main="Iris Data", xlab="Petal length",
ylab="Petal width")

abline(lsfit(iris$Petal.Length, iris$Petal.Width)$coefficients, col="black")

lm(Petal.Width ~ Petal.Length, data=iris)$coefficients

plot(iris$Petal.Length, iris$Petal.Width, pch=21,
bg=c("red","green3","blue")[unclass(iris$Species)], main="Iris Data", xlab="Petal length",
ylab="Petal width")

abline(lm(Petal.Width ~ Petal.Length, data=iris)$coefficients, col="black")

summary(lm(Petal.Width ~ Petal.Length, data=iris))

plot(iris$Sepal.Width, iris$Sepal.Length, pch=21,
bg=c("red","green3","blue")[unclass(iris$Species)], main="Iris Data", xlab="Sepal Width",
ylab="Sepal Length")

abline(lm(Sepal.Length ~ Sepal.Width, data=iris)$coefficients, col="black")

summary(lm(Sepal.Length ~ Sepal.Width, data=iris))

plot(iris$Sepal.Width, iris$Sepal.Length, pch=21,
bg=c("red","green3","blue")[unclass(iris$Species)], main="Iris Data", xlab="Petal length",
ylab="Sepal length")

abline(lm(Sepal.Length ~ Sepal.Width, data=iris)$coefficients, col="black")
abline(lm(Sepal.Length ~ Sepal.Width,
data=iris[which(iris$Species=="setosa"),])$coefficients, col="red")

abline(lm(Sepal.Length ~ Sepal.Width,
data=iris[which(iris$Species=="versicolor"),])$coefficients, col="green3")

abline(lm(Sepal.Length ~ Sepal.Width,
data=iris[which(iris$Species=="virginica"),])$coefficients, col="blue")

lm(Sepal.Length ~ Sepal.Width, data=iris[which(iris$Species=="setosa"),])$coefficients

lm(Sepal.Length ~ Sepal.Width, data=iris[which(iris$Species=="versicolor"),])$coefficients
lm(Sepal.Length ~ Sepal.Width, data=iris[which(iris$Species=="virginica"),])$coefficients

lm(Sepal.Length ~ Sepal.Width:Species + Species - 1, data=iris)$coefficients
```

```
summary(lm(Sepal.Length ~ Sepal.Width:Species + Species - 1, data=iris))

summary(step(lm(Sepal.Length ~ Sepal.Width * Species, data=iris))) lm(Sepal.Length ~ Sepal.Width:Species +
Species - 1, data=iris)$coefficients lm(Sepal.Length ~ Sepal.Width:Species + Species, data=iris)$coefficients
```

PRACTICAL 7 :

Logistic regression

Logistic regression is a statistical technique used to model the relationship between a binary dependent variable and one or more independent variables. In binary classification problems, the dependent variable can only take on one of two possible values, typically represented as 0 or 1.

Logistic regression is used to predict the probability of an event occurring based on the values of the independent variables. The output of a logistic regression model is a probability between 0 and 1, which represents the likelihood of the dependent variable being 1.

The logistic regression model is based on the logistic function, which transforms a linear equation into a probability. The logistic function is of the form: $P = 1 / (1 + \exp(-z))$, where P is the probability of the dependent variable being 1, exp is the exponential function, and z is the linear combination of the independent variables.

Logistic regression can be used for both binary and multiclass classification problems, by using different variations of the logistic function, such as softmax regression.

Logistic regression is commonly used in various fields, such as healthcare, marketing, and social sciences, to make predictions and gain insights into the relationships between variables.

Code:

```
library(datasets)

ir_data<- iris

head(ir_data)

str(ir_data)

levels(ir_data$Species)

sum(is.na(ir_data))

ir_data<-ir_data[1:100,]

set.seed(100)

samp<-sample(1:100,80)

ir_test<-ir_data[samp,]

ir_ctrl<-ir_data[-samp,]

install.packages("ggplot2")

library(ggplot2)

 install.packages("GGally")

library(GGally)

ggpairs(ir_test)

y<-ir_test$Species; x<-ir_test$Sepal.Length

glfit<-glm(y~x, family = 'binomial')

summary(glfit)

newdata<- data.frame(x=ir_ctrl$Sepal.Length)

predicted_val<-predict(glfit, newdata, type="response")

prediction<-data.frame(ir_ctrl$Sepal.Length, ir_ctrl$Species,predicted_val)

prediction

qplot(prediction[,1], round(prediction[,3]), col=prediction[,2], xlab = 'Sepal Length', ylab
= 'Prediction using Logistic Reg.')
```

Practical 8

Hypothesis Testing

Hypothesis testing is a statistical method used to determine whether a hypothesis about a population parameter is supported by the sample data. The hypothesis is typically stated in terms of a null hypothesis (H0) and an alternative hypothesis (Ha).

The null hypothesis is the hypothesis of no difference or no effect, and it is assumed to be true unless there is sufficient evidence to reject it. The alternative hypothesis is the hypothesis that the researcher is interested in testing, and it is the hypothesis that is accepted if the null hypothesis is rejected.

If the test statistic falls within the rejection region (i.e., the critical value or the p-value is less than alpha), then the null hypothesis is rejected in favor of the alternative hypothesis. If the test statistic does not fall within the rejection region, then the null hypothesis is not rejected.

Hypothesis testing is used in many fields, such as business, healthcare, and social sciences, to make decisions and draw conclusions based on sample data.

Code :

```
x= c(6.2, 6.6, 7.1, 7.4, 7.6, 7.9, 8, 8.3, 8.4, 8.5, 8.6,
    8.8, 8.8, 9.1, 9.2, 9.4, 9.4, 9.7, 9.9, 10.2, 10.4, 10.8,
    11.3, 11.9)

t.test(x-9,alternative="two.sided",conf.level=0.95)

x=c(418,421,421,422,425,427,431,434,437,439,446,447,448,453,454,463,465)

y=c(429,430,430,431,36,437,440,441,445,446,447)

test2<-t.test(x,y,alternative="two.sided",mu=0,var.equal=F,conf.level=0.95)

test2
```

Practical 9:

Analysis of variance

Analysis of variance (ANOVA) is a statistical technique used to compare the means of two or more groups to determine whether there is a statistically significant difference between them. ANOVA is used to test the null hypothesis that the means of the groups are equal.

ANOVA is based on the partitioning of the total variation in a dataset into different sources of variation, namely, the variation between the groups (explained variation) and the variation within the groups (unexplained variation). The ratio of the explained variation to the unexplained variation is used to calculate an F-statistic, which is used to determine the statistical significance of the difference between the means of the groups.

ANOVA can be used for different types of designs, such as one-way ANOVA, which compares the means of two or more groups on a single variable, and two-way ANOVA, which compares the means of two or more groups on two variables.

ANOVA is commonly used in various fields, such as psychology, biology, and business, to compare the means of different groups and identify factors that influence the dependent variable.

Code : y1 = c(18.2, 20.1, 17.6, 16.8, 18.8, 19.7, 19.1)

```
y2 = c(17.4, 18.7, 19.1, 16.4, 15.9, 18.4, 17.7)

y3 = c(15.2, 18.8, 17.7, 16.5, 15.9, 17.1, 16.7)

y = c(y1, y2, y3)

n = rep(7, 3)

n

group = rep(1:3, n)

 group

tmp = tapply(y,
group,stem)

stem(y)

 tmpfn = function(x) c(sum = sum(x), mean = mean(x), var = var(x), n =
                    length(x))

tapply(y, group, tmpfn)

tmpfn(y)

data = data.frame(y = y, group = factor(group))
```

```
fit = lm(y ~ group, data)

anova(fit)

df = anova(fit)[, "Df"]

names(df) = c("trt", "err")

df

alpha = c(0.05, 0.01)

qf(alpha, df["trt"], df["err"], lower.tail = FALSE)

 anova(fit)["Residuals", "Sum Sq"]

anova(fit)["Residuals", "Sum Sq"]/qchisq(c(0.025, 0.975), 18,
                                    lower.tail = FALSE)
```

Practical 10

Decision tree

A decision tree is a tree-like model used for making decisions or predictions based on a series of decisions or conditions. It is a graphical representation of all possible solutions to a decision-making problem.

A decision tree consists of nodes and branches. The nodes represent the decisions or conditions, and the branches represent the outcomes or consequences of those decisions or conditions. The tree starts with a single node, known as the root node, and branches out into multiple levels of nodes, known as internal nodes, and finally ends with the leaf nodes, which represent the final outcome or decision.

Decision trees can be used for classification or regression problems. In classification problems, the decision tree is used to predict the class or category of a new data point based on its features. In regression problems, the decision tree is used to predict a continuous output variable based on the input variables.

The decision tree is constructed by recursively splitting the data based on the most important feature at each level of the tree, using a splitting criterion such as information gain or Gini impurity. The goal is to create a tree that is both accurate and interpretable, meaning that it can be easily understood and explained.

Decision trees are commonly used in various fields, such as business, healthcare, and finance, to make decisions and predictions based on data. They are popular because they are easy to understand, visualize, and interpret.

Code:

```
mydata<-data.frame(iris)
attach(mydata)

install.packages("rpart")
library(rpart)
model<-
rpart(Species~Sepal.Length+Sepal.Width+Petal.Length+Petal.Width,data=mydata,method="class")
plot(model)
text(model,use.n=TRUE,all=TRUE,cex=0.8)



install.packages("tree")
library(tree)
model1<-
tree(Species~Sepal.Length+Sepal.Width+Petal.Length+Petal.Width,data=mydata,method="class",sp
lit="gini")
plot(model1)
text(model1,all=TRUE,cex=0.6)
install.packages("party")
library(party)
model2<-ctree(Species~Sepal.Length+Sepal.Width+Petal.Length+Petal.Width,data=mydata)
plot(model2)
library(tree)
mydata<-data.frame(iris)
attach



model1<-
tree(Species~Sepal.Length+Sepal.Width+Petal.Length+Petal.Width,data=mydata,method="class",c
ontrol=tree.control(nobs=150,mincut=10))
plot(model1)
text(model1,all=TRUE,cex=0.6)


predict(model1,iris)(mydata)
```

```
model2<-
ctree(Species~Sepal.Length+Sepal.Width+Petal.Length+Petal.Width,data=mydata,controls=ctree_c
ontrol(maxdepth=2))
plot(model2)
```