202001103042
Arjan Singh Johar
Btech – 1

Practical No – 01

Aim – Introduction to R tool for data analytics science.

Code –

Program 1 –

```
42+18
Arjan = 3042
class(Arjan)
print(Arjan)
print("Hello")
class(Arjan)
b=100L
class(b)
s="Afaf loves to do programming"
print(s)
class(s)
Arjan=as.integer()
class(Arjan)
x=c(42,38,52,44)
print(sqrt(x))
print(mean(x))
print(median(x))
x1=c("Arjan","Afaf","Saim","Abdul")
print(x1)
y=c(42,NA,38,NA,52,NA,44)
print(mean(y,na.rm=TRUE))
```

202001103042
Arjan Singh Johar
Btech – 1
Output –

```
> 42+18
[1] 60
> Arjan = 3042
> class(Arjan)
[1] "numeric"
> print(Arjan)
[1] 3042
> print("Hello")
[1] "Hello"
> class(Arjan)
[1] "numeric"
> b=100L
> class(b)
[1] "integer"
> s="Afaf loves to do programming"
> print(s)
[1] "Afaf loves to do programming"
> class(s)
[1] "character"
> Arjan=as.integer()
> class(Arjan)
[1] "integer"
```

```
> s="Afaf loves to do programming"
> print(s)
[1] "Afaf loves to do programming"
> class(s)
[1] "character"
> Arjan=as.integer()
> class(Arjan)
[1] "integer"
> x=c(42,38,52,44)
> print(sqrt(x))
[1] 6.480741 6.164414 7.211103 6.633250
> print(mean(x))
[1] 44
> print(median(x))
[1] 43
> x1=c("Arjan","Afaf","Saim","Abdul")
> print(x1)
[1] "Arjan" "Afaf"  "Saim"  "Abdul"
> y=c(42,NA,38,NA,52,NA,44)
> print(mean(y,na.rm=TRUE))
[1] 44
```

202001103042
Arjan Singh Johar
Btech – 1
Program 2 –

```
stud_data =
data.frame(Name=c("Afaf","Arjan","Abdul"),PRN=c(38,42,44),Marks=c(100,89,89))

stud_data

a=20

b=40

if (a>b){

  print("a is greater")

}else

  print("b is greater")

x=c(1,2,3,4,5)

sum=0

for (i in x){

  sum=sum+i

}

print(sum)

print(sum(x))
```

Output –

```
> stud_data = data.frame(Name=c("Afaf","Arjan","Abdul"),PRN=c(38,42,44),Marks=c(1
00,89,89))
> stud_data
   Name PRN Marks
1  Afaf  38   100
2 Arjan  42    89
3 Abdul  44    89
> a=20
> b=40
> if (a>b){
+   print("a is greater")
+ }else
+   print("b is greater")
[1] "b is greater"
> x=c(1,2,3,4,5)
> sum=0
> for (i in x){
+   sum=sum+i
+ }
> print(sum)
[1] 15
> print(sum(x))
[1] 15
>
```

202001103042
Arjan Singh Johar
Btech – 1

Practical No – 02

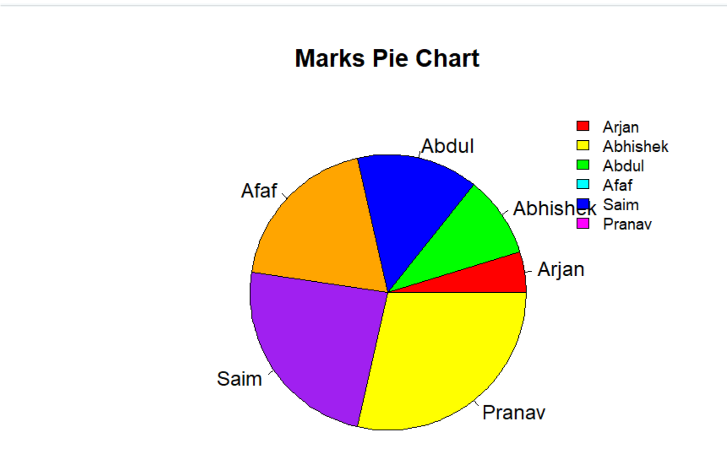
Aim – Programs for Basic Statistics and Visualization in R

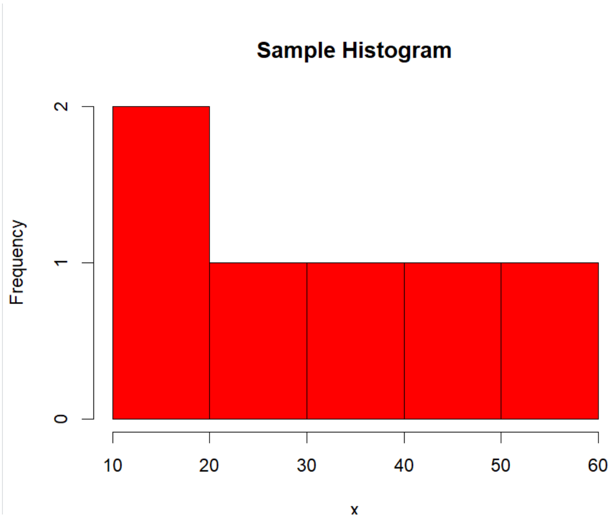
Code –

Program 1 –

```
x=c(10,20,30,40,50,60)
labels=c("Arjan","Abhishek","Abdul","Afaf","Saim","Pranav")
color=c("red","green","blue","orange","purple","yellow")
pie(x,labels,col = color,main="Marks Pie Chart")
legend("topright", labels, cex = 0.8, bty = "n", fill = rainbow(length(labels)))
hist(x,col="red",main="Sample Histogram")
```

Output –





202001103042
Arjan Singh Johar
Btech – 1
Program 2 –

```
#histogram
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
samples=np.random.randint(1,100,50)
plt.hist(samples)
plt.show()
```

```
#Pie chart
slices=[70,40,50,69,35]
lang=["JAVA","C++","PYTHON","C","R"]
cols=["Red","Green","Blue","Orange","Yellow"]
plt.pie(slices,labels=lang,colors=cols,autopct="%0.2f%%",explode=[0.5,0,0,0,0])
plt.show()
```

```
#histogram
import matplotlib.pyplot as plt
import numpy as np

x = (10,20,15,14,12,25,22,11,30,35)
plt.hist(x,color="Blue")
plt.show()
```

```
import matplotlib.pyplot as plt
categories = ['Category 1', 'Category 2', 'Category 3', 'Category 4']
values = [25, 40, 30, 50]
plt.bar(categories, values)
plt.xlabel('Categories')
plt.ylabel('Values')
plt.title('Bar Plot Example')
plt.show()
```

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.get_dataset_names()
tips=sns.load_dataset('tips')
tips.head()
x=[1,2,3,4,5]
plt.boxplot(x)
```

```
import seaborn as sns
```

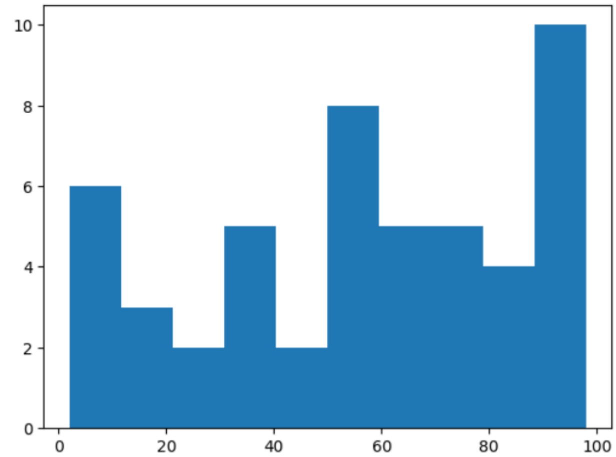
202001103042
Arjan Singh Johar
Btech – 1

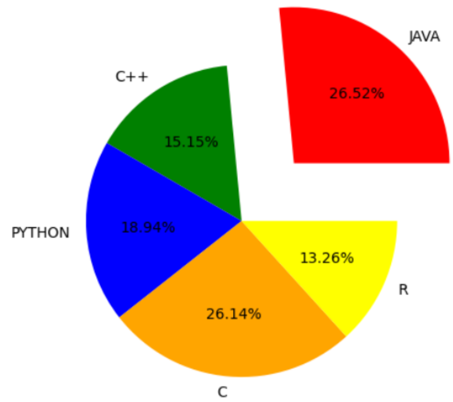
```
import matplotlib.pyplot as plt
sns.get_dataset_names()
titanic=sns.load_dataset('titanic')
titanic.head()
sns.scatterplot(x='sex',y='age',data=titanic,hue='alone')
```

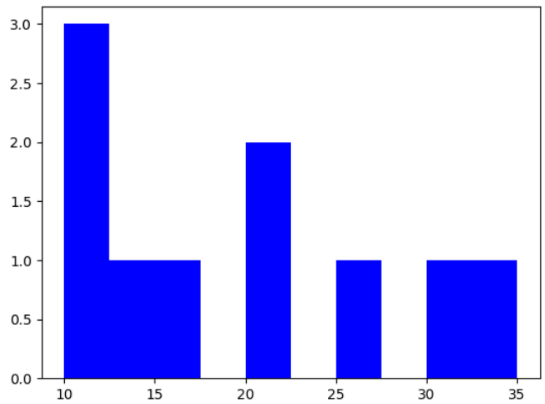
```
import matplotlib.pyplot as plt
import seaborn as sns
x=[1,2,3,4,5]
y=[10,20,30,40,50]
sns.pointplot(x=x,y=y)
```

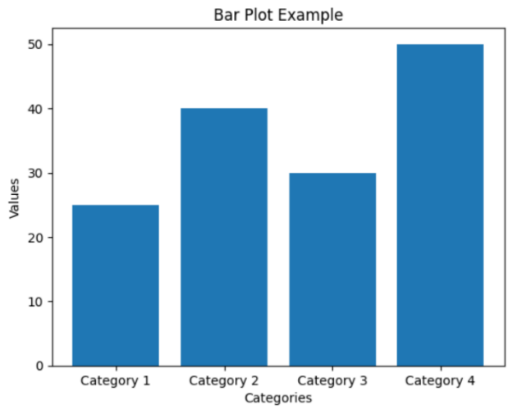
```
import sklearn
from sklearn.datasets import load_diabetes
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
diabetes = load_diabetes()
column_name = diabetes.feature_names
df_diabetes = pd.DataFrame(diabetes.data)
df_diabetes .columns = column_name
df_diabetes .head()
print("Old Shape: ", df_diabetes.shape)
Q1 = df_diabetes['bmi'].quantile(0.25)
Q3 = df_diabetes['bmi'].quantile(0.75)
IQR = Q3 - Q1
lower = Q1 - 1.5*IQR
upper = Q3 + 1.5*IQR
upper_array = np.where(df_diabetes['bmi']>=upper)[0]
lower_array = np.where(df_diabetes['bmi']<=lower)[0]
df_diabetes.drop(index=upper_array, inplace=True)
df_diabetes.drop(index=lower_array, inplace=True)
print("New Shape: ", df_diabetes.shape)
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.boxplot(df_diabetes['bmi'], vert=False)
plt.title('Before Removing Outliers')
plt.xlabel('BMI')
plt.subplot(1, 2, 2)
plt.boxplot(diabetes.data[:, np.where(df_diabetes.columns == 'bmi')[0]], vert=False)
plt.title('After Removing Outliers')
plt.xlabel('BMI')
plt.tight_layout()
plt.show()
```

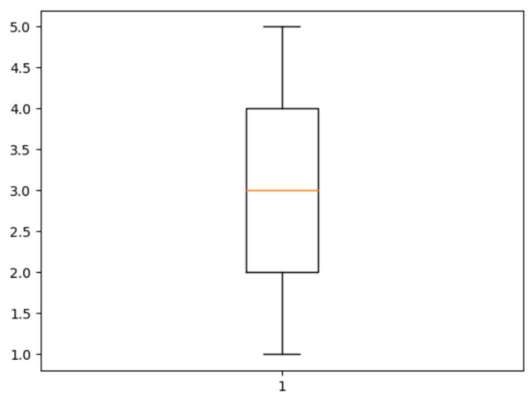
202001103042
Arjan Singh Johar
Btech – 1
Output –



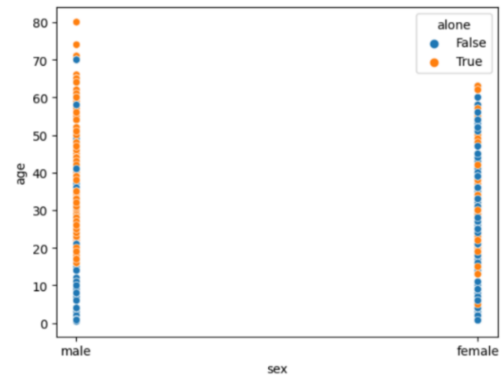




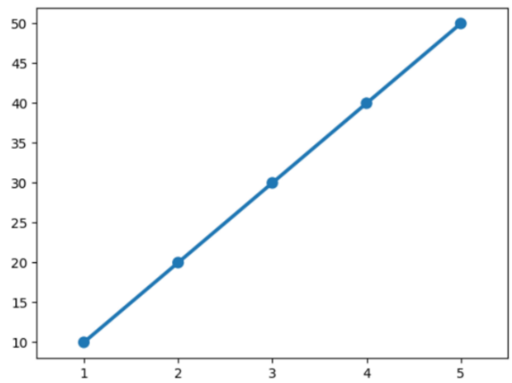




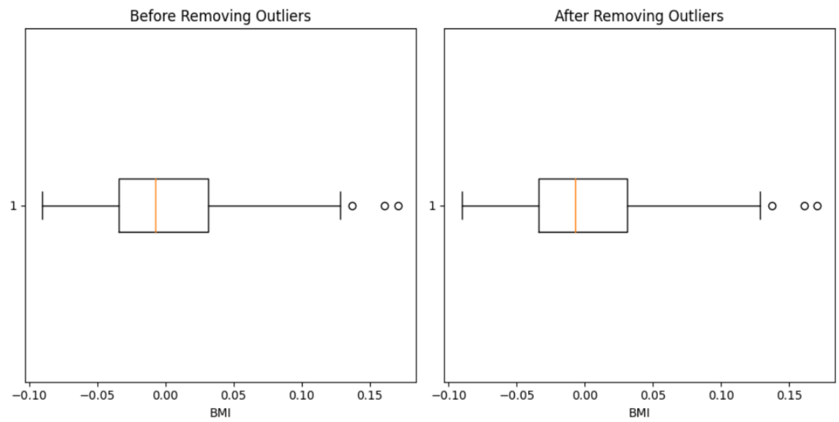
`<Axes: xlabel='sex', ylabel='age'>`



`<Axes: >`




```
Old Shape:  (442, 10)
New Shape:  (439, 10)
```



202001103042
Arjan Singh Johar
Btech – 1

Practical No – 03

Aim – Program for simple linear regression is an approach for predicting a response using a single multiple feature.

Code –

Program 1 –

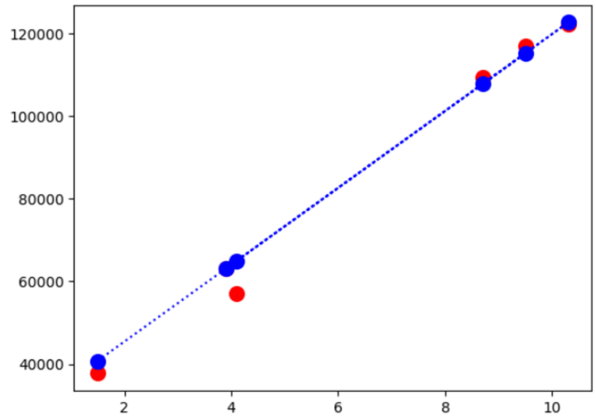
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
data=pd.read_csv("Salary_Data.csv")
data.head()
data.info()
model = LinearRegression()
x = data[['YearsExperience']]
y = data['Salary']
x_train,x_test,y_train,y_test = train_test_split(x,y,train_size=0.8,random_state=0)
len(x_train)
len(x_test)
len(y_train)
len(y_test)
model.fit(x_train,y=y_train)
y_pred = model.predict(x_test)
plt.scatter(x_test,y_test,color='red',s=100)
plt.scatter(x_test,y_pred,color='blue',s=100)
plt.plot(x_test,y_pred,color='blue',linestyle="dotted")
from sklearn.metrics import r2_score
score=r2_score(y_test,y_pred)
score*100
```

Output –

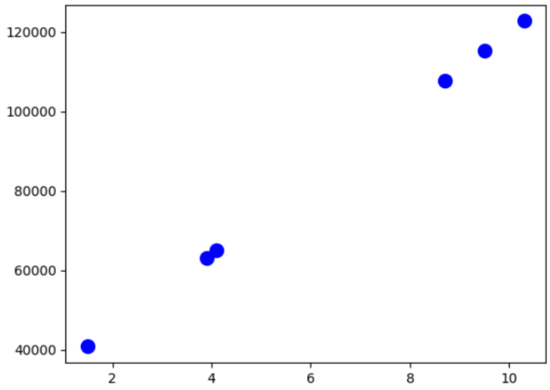
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 2 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   YearsExperience  30 non-null     float64
 1   Salary           30 non-null     float64
dtypes: float64(2)
memory usage: 608.0 bytes
```

202001103042
Arjan Singh Johar
Btech – 1

```
[<matplotlib.lines.Line2D at 0x7e3708c2a8f0>]
```




```
<matplotlib.collections.PathCollection at 0x7e36fe7d9b70>
```



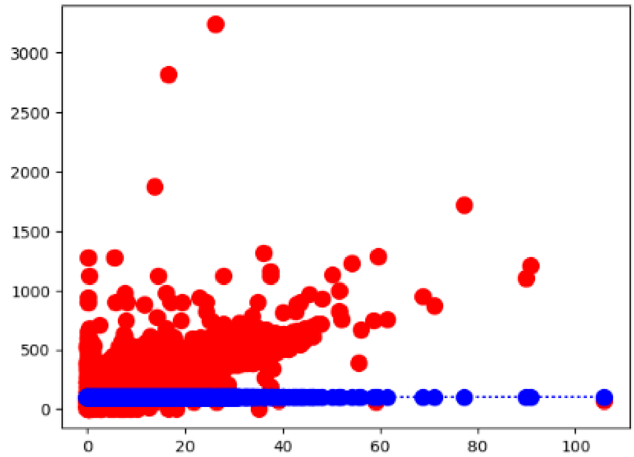

```
98.8169515729126
```

Program 2 –

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
data=pd.read_csv("train.csv")
data.head()
data.info()
model = LinearRegression()
x = data[['distance_traveled']]
y = data['fare']
x_train,x_test,y_train,y_test = train_test_split(x,y,train_size=0.8,random_state=0)
len(x_train)
len(x_test)
len(y_train)
len(y_test)
model.fit(x_train,y=y_train)
y_pred = model.predict(x_test)
y_pred
plt.scatter(x_test,y_test,color='red',s=100)
plt.scatter(x_test,y_pred,color='blue',s=100)
plt.plot(x_test,y_pred,color='blue',linestyle="dotted")
from sklearn.metrics import r2_score
score=r2_score(y_test,y_pred)
score*100
```

202001103042
Arjan Singh Johar
Btech – 1
Output –

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 209673 entries, 0 to 209672
Data columns (total 8 columns):
 #   Column              Non-Null Count   Dtype
---  ------              --------------   -----
 0   trip_duration       209673 non-null  float64
 1   distance_traveled   209673 non-null  float64
 2   num_of_passengers   209673 non-null  float64
 3   fare                209673 non-null  float64
 4   tip                 209673 non-null  int64
 5   miscellaneous_fees  209673 non-null  float64
 6   total_fare          209673 non-null  float64
 7   surge_applied       209673 non-null  int64
dtypes: float64(6), int64(2)
memory usage: 12.8 MB
0.19829883706034002
```



202001103042
Arjan Singh Johar
Btech – 1

Practical No – 04

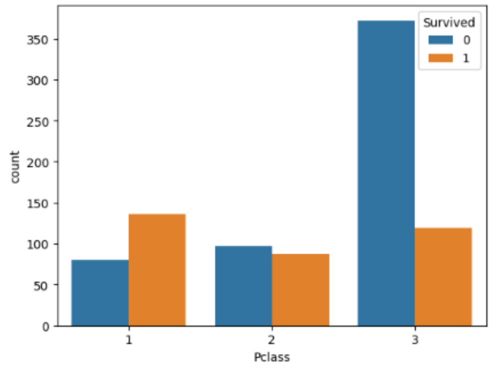
Aim – Program for predicting whether a user will purchase the product or not, using logistic Regression.

Code –

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
data=pd.read_csv('Titanic-Dataset.csv')
data.head(5)
sns.countplot(x=data['Pclass'],hue=data['Survived'])
data.info()
mean_age = round((data['Age'].mean()),2)
data['Age'] = data['Age'].fillna(mean_age)
data.info()
col = ['PassengerId','Name','Cabin','Ticket','Fare']
data = data.drop(col , axis = 1)
data.head(5)
data.isnull().sum()
data = data.drop('Embarked' , axis = 1)
from sklearn.preprocessing import LabelEncoder
Encoder = LabelEncoder()
data['Sex'] = Encoder.fit_transform(data['Sex'])
data.head()
from sklearn.model_selection import train_test_split
x_train , x_test , y_train , y_test = train_test_split(x , y , train_size = 0.8 ,
random_state = 0)
len(x_train)
len(x_test)
x_test
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(x_train,y_train)
y_pred = model.predict(x_test)
y_pred
y_test
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test , y_pred)
accuracy*100
model.predict([[1,0,31.0,1,0]])
```

202001103042
Arjan Singh Johar
Btech – 1
Output –

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```




```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          891 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

	Survived	Pclass	Sex	Age	SibSp	Parch
0	0	3	1	22.0	1	0
1	1	1	0	38.0	1	0
2	1	3	0	26.0	0	0
3	1	1	0	35.0	1	0
4	0	3	1	35.0	0	0

	Pclass	Sex	Age	SibSp	Parch
495	3	1	29.7	0	0
648	3	1	29.7	0	0
278	3	1	7.0	4	1
31	1	0	29.7	1	0
255	3	0	29.0	0	2
...	...	...	...	...	...
780	3	0	13.0	0	0
837	3	1	29.7	0	0
215	1	0	31.0	1	0
833	3	1	23.0	0	0
372	3	1	19.0	0	0

179 rows × 5 columns

```
495    0
648    0
278    0
31     1
255    1
      ..
780    1
837    0
215    1
833    0
372    0
Name: Survived, Length: 179, dtype: int64
```

```
array([1])
```

```
80.44692737430168
```

202001103042
Arjan Singh Johar
Btech – 1

Practical No – 05

Aim – Write a program to implement k-Nearest Neighbour algorithm to classify the iris data set. Print both correct and wrong predictions

Program 1 – Titanic Dataset

Code –

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
data=pd.read_csv('Titanic-Dataset.csv')
data.head(5)
sns.countplot(x=data['Pclass'],hue=data['Survived'])
data.info()
mean_age = round((data['Age'].mean()),2)
data['Age'] = data['Age'].fillna(mean_age)
data.info()
col = ['PassengerId','Name','Cabin','Ticket','Fare']
data = data.drop(col , axis = 1)
data.head(5)
data.isnull().sum()
data = data.drop('Embarked' , axis = 1)
from sklearn.preprocessing import LabelEncoder
Encoder = LabelEncoder()
data['Sex'] = Encoder.fit_transform(data['Sex'])
data.head()
from sklearn.model_selection import train_test_split
x_train , x_test , y_train , y_test = train_test_split(x , y , train_size = 0.8 ,
random_state = 0)
len(x_train)
len(x_test)
x_test
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors = 3)
model.fit(x_train , y_train)
y_pred = model.predict(x_test)
y_pred
y_test
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test , y_pred)
accuracy*100
```

202001103042
Arjan Singh Johar
Btech – 1
Output –

```
495    0
648    0
278    0
31     1
255    1
      ..
780    1
837    0
215    1
833    0
372    0
Name: Survived, Length: 179, dtype: int64
```

```
75.41899441340783
```

Program 2 – Iris Dataset

Code –

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
data=pd.read_csv('Iris.csv')
data.head(5)
data.info()
data.isnull().sum()
x = data.iloc[:, :-1].values
y = data.iloc[:, -1].values
from sklearn.model_selection import train_test_split
x_train , x_test , y_train , y_test = train_test_split(x , y ,
train_size = 0.8 , random_state = 0)
# len(x_train)
len(x_test)
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors = 3)
model.fit(x_train , y_train)
y_pred = model.predict(x_test)
y_pred
y_test
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test , y_pred)
accuracy*100
model.predict()
```

202001103042
Arjan Singh Johar
Btech – 1
Output –

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

```
array(['Iris-virginica', 'Iris-versicolor', 'Iris-setosa',
       'Iris-virginica', 'Iris-setosa', 'Iris-virginica', 'Iris-setosa',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-virginica', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-setosa',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-setosa', 'Iris-setosa',
       'Iris-virginica', 'Iris-versicolor', 'Iris-setosa', 'Iris-setosa',
       'Iris-virginica', 'Iris-setosa', 'Iris-setosa', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-setosa'], dtype=object)
```

```
100.0
```

202001103042
Arjan Singh Johar
Btech – 1

Practical No – 06

Aim – For the given data perform ANOVA and tell the difference of variance between groups.

Program –

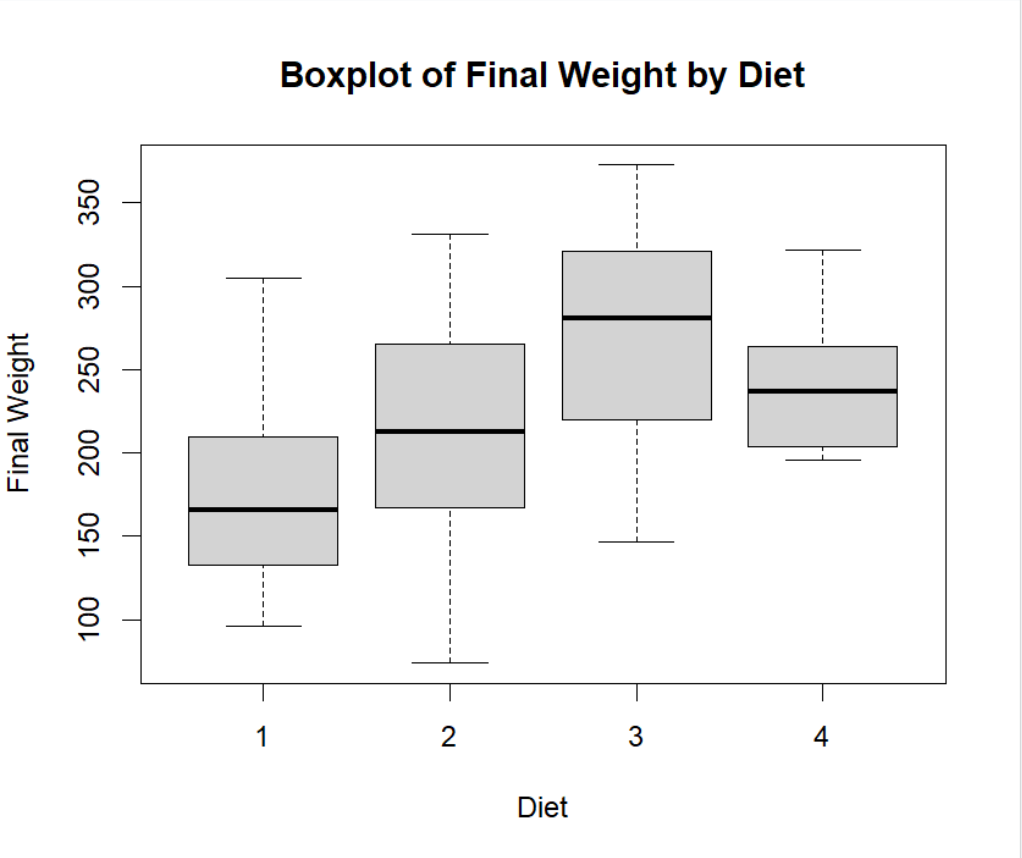
Code –

```
# Load the ChickWeight dataset

data(ChickWeight)

# Subset the data to get the final weights

final_weight_data <- ChickWeight[ChickWeight$Time == max(ChickWeight$Time), ]

# Calculate the mean of final weights for each diet group

mean_weight <- tapply(final_weight_data$weight, final_weight_data$Diet, mean)

# Convert 'Diet' to a factor for ANOVA

final_weight_data$Diet <- factor(final_weight_data$Diet)

# Perform one-way ANOVA

anova_result <- aov(weight ~ Diet, data = final_weight_data)

# Post hoc test (Tukey's HSD)

tukey_result <- TukeyHSD(anova_result)

# Equality of variance (Levene's test)

library(car)

levene_test <- leveneTest(weight ~ Diet, data = final_weight_data)

  print(levene_test)

# Create a boxplot

boxplot(weight ~ Diet, data = final_weight_data, main = "Boxplot of Final Weight by Diet", xlab = "Diet", ylab = "Final Weight")

# Check the p-value from ANOVA

p_value <- summary(anova_result)[[1]][["Pr(>F)"]][1]

# Define the significance level (alpha)

alpha <- 0.05
```

202001103042
Arjan Singh Johar
Btech – 1

```
# Print the decision based on the p-value
if (p_value < alpha) {
  cat("Reject the null hypothesis. There is a significant difference in means.\n")
} else {
  cat("Accept the null hypothesis. There is no significant difference in means.\n")
}
```

Output –




```
Levene's Test for Homogeneity of Variance (center = median)
      Df F value Pr(>F)
group  3  1.1597 0.3367
      41
Reject the null hypothesis. There is a significant difference in means.
```

202001103042
Arjan Singh Johar
Btech – 1

Practical No – 07

Aim – Implementation of Naive Bayesian Classifier.

Code and Output –

```
library(e1071)

# Load the Iris dataset

data(iris)

library(caret)

set.seed(123)  # For reproducibility

train_index <- createDataPartition(iris$Species, p = 0.7, list = FALSE)

train_data <- iris[train_index, ]

test_data <- iris[-train_index, ]

# Train the Naive Bayes classifier

nb_classifier <- naiveBayes(Species ~ ., data = train_data)

# Predict on the test set

test_predictions <- predict(nb_classifier, newdata = test_data)

# Create a confusion matrix

confusion_matrix <- table(Actual = test_data$Species, Predicted = test_predictions)

confusion_matrix

# Calculate accuracy

accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)

accuracy
```

Output –

```
            Predicted
Actual       setosa versicolor virginica
  setosa         15          0         0
  versicolor      0         13         2
  virginica       0          2        13
>
> # Calculate accuracy
> accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)
> accuracy
[1] 0.9111111
>
```

202001103042
Arjan Singh Johar
Btech – 1

Practical No – 08

Aim – Program for market basket analysis using Association Rules.

Program –

```
# Load the libraries
install.packages("arulesViz")
library(arules)
library(arulesViz)
library(datasets)
# Load the data set
data(Groceries)
# Create an item frequency plot for the top 20 items
itemFrequencyPlot(Groceries,topN=20,type="absolute")
# Get the rules
rules <- apriori(Groceries, parameter = list(supp = 0.001, conf = 0.8))
# Show the top 5 rules, but only 2 digits
options(digits=2)
inspect(rules[1:5])
rules<-sort(rules, by="confidence", decreasing=TRUE)
rules <- apriori(Groceries, parameter = list(supp = 0.001, conf = 0.8,maxlen=3))
subset.matrix <- is.subset(rules, rules)
subset.matrix[lower.tri(subset.matrix, diag=T)] <- NA
redundant <- colSums(subset.matrix, na.rm=T) >= 1
rules.pruned <- rules[!redundant]
rules<-rules.pruned
rules<-apriori(data=Groceries, parameter=list(supp=0.001,conf = 0.08),
        appearance = list(default="lhs",rhs="whole milk"),
        control = list(verbose=F))
```

202001103042
Arjan Singh Johar
Btech – 1

rules<-sort(rules, decreasing=TRUE,by="confidence")

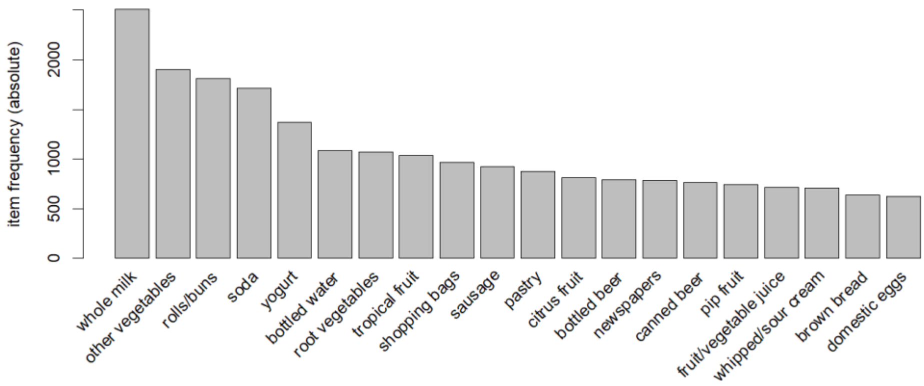
inspect(rules[1:5])

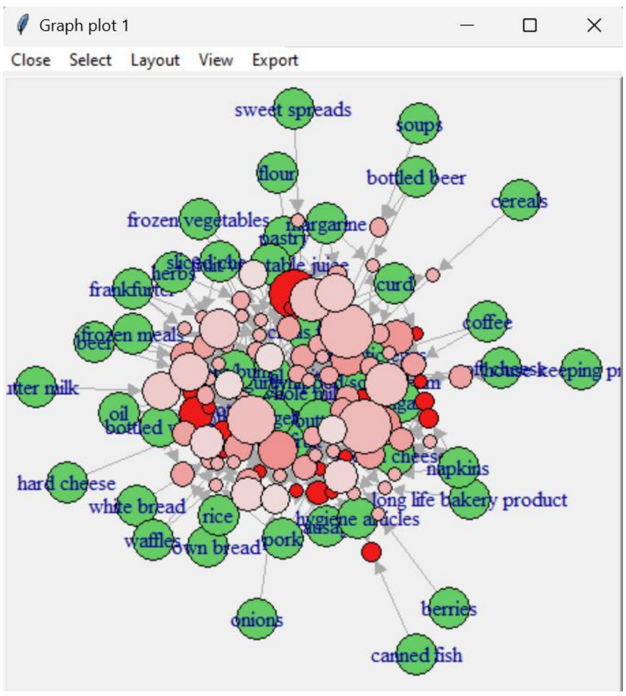
library(arulesViz)

plot(rules,method="graph",interactive=TRUE)

Output –

```
    lhs                                               rhs          support confidence coverage lift
[1] {rice, sugar}                                  => {whole milk} 0.0012  1          0.0012   3.9
[2] {canned fish, hygiene articles}                => {whole milk} 0.0011  1          0.0011   3.9
[3] {root vegetables, butter, rice}                => {whole milk} 0.0010  1          0.0010   3.9
[4] {root vegetables, whipped/sour cream, flour}   => {whole milk} 0.0017  1          0.0017   3.9
[5] {butter, soft cheese, domestic eggs}           => {whole milk} 0.0010  1          0.0010   3.9
    count
[1] 12
[2] 11
[3] 10
[4] 17
[5] 10
```





202001103042
Arjan Singh Johar
Btech – 1

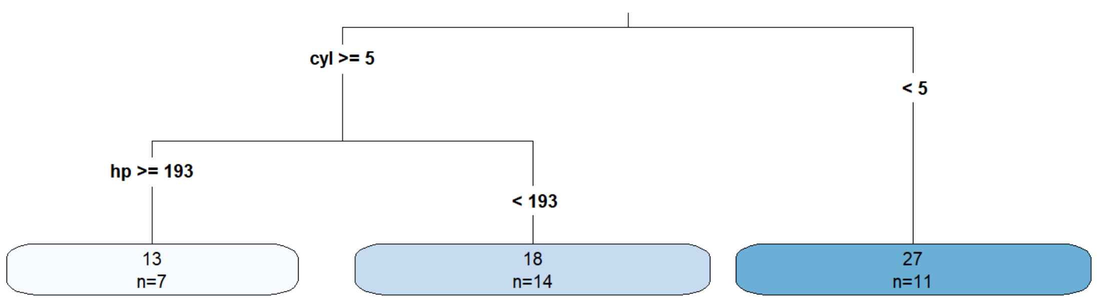
Practical No – 09

Aim – Program to implement a Decision tree algorithm for any application.

Code –

```
# Load the required libraries
library(rpart)
library(rpart.plot)
# Load the "mtcars" dataset (built-in dataset in R)
data(mtcars)
# Split the dataset into features (X) and target (y)
X <- mtcars[, -1]  # Features (excluding the first column, which is the car model)
y <- mtcars$mpg   # Target (miles per gallon)
# Build the decision tree model
model <- rpart(y ~ ., data = data.frame(X, y), method = "anova")
# Visualize the decision tree
rpart.plot(model, box.palette = "auto", type = 3, extra = 1)
```

Output –



202001103042
Arjan Singh Johar
Btech – 1

Practical No – 10

Aim – Program to Simulate Principal component analysis.

Code –

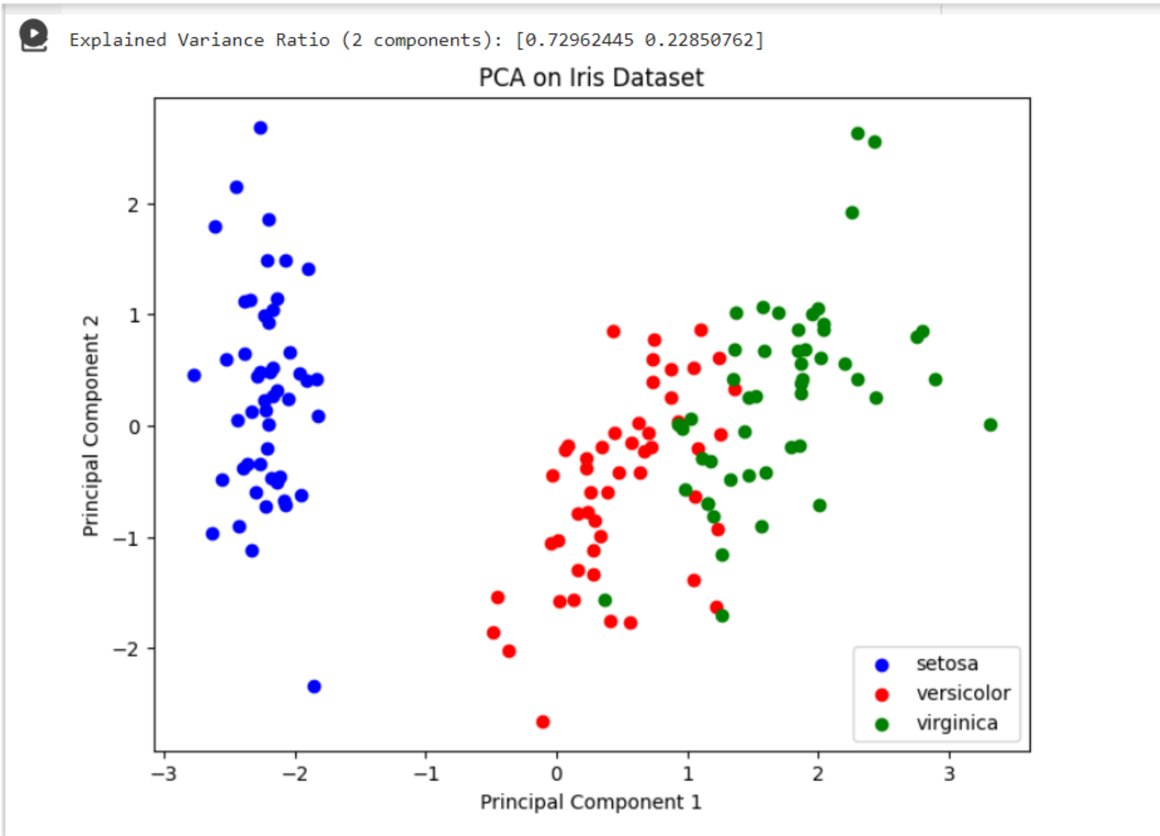
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
# Load the Iris dataset
iris = load_iris()
X = iris.data
y = iris.target
# Standardize the features (mean=0 and variance=1)
X_standardized = StandardScaler().fit_transform(X)
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_standardized, y,
test_size=0.4, random_state=42)
# Create a Naive Bayes classifier
clf = GaussianNB()
# Train the classifier
clf.fit(X_train, y_train)
# Make predictions on the test data
predictions = clf.predict(X_test)
from sklearn.metrics import accuracy_score
accuracy_score(y_test, predictions)
pca = PCA(n_components=1)
X_pca = pca.fit_transform(X_standardized)
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_pca, y,
test_size=0.4, random_state=42)
# Create a Naive Bayes classifier
clf = GaussianNB()
# Train the classifier
clf.fit(X_train, y_train)
# Make predictions on the test data
predictions = clf.predict(X_test)
from sklearn.metrics import accuracy_score
accuracy_score(y_test, predictions)
```

202001103042
Arjan Singh Johar
Btech – 1

```
# Apply PCA and reduce the data to 2 principal components
pca = PCA(n_components=)
X_pca = pca.fit_transform(X_standardized)
# Print the explained variance ratio
print("Explained Variance Ratio (2 components):",
pca.explained_variance_ratio_)
# Plot the data points in the reduced feature space
plt.figure(figsize=(8, 6))
colors = ['blue', 'red', 'green']
for i, target_name in enumerate(iris.target_names):
    plt.scatter(X_pca[y == i, 0], X_pca[y == i, 1], color=colors[i],
label=target_name)
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('PCA on Iris Dataset')
plt.legend()
plt.show()
```

Output –

```
0.9666666666666667
```



202001103042
Arjan Singh Johar
Btech – 1

Practical No – 11

Aim – Program to Simulate Principal component analysis

Code –

```
# Load the USArrests dataset
data("USArrests")
# Perform SVD on the dataset
svd_result <- svd(USArrests)
# Extract U, Sigma, and V matrices
U <- svd_result$u
Sigma <- diag(svd_result$d)
V <- svd_result$v
# Print the matrices
print("U (left singular vectors):")
print(U)
print("Sigma (singular values):")
print(Sigma)
print("V (right singular vectors):")
print(V)
```

Output –

```
> print("U (left singular vectors):")
[1] "U (left singular vectors):"
> print(U)
             [,1]         [,2]         [,3]         [,4]
 [1,] -0.17162510  0.096325710  0.065154797  0.153695511
 [2,] -0.18911657  0.173452566 -0.426657848 -0.178014378
 [3,] -0.21559302  0.078998111  0.020637399 -0.280707843
 [4,] -0.13902443  0.059889811  0.013922695  0.016104178
 [5,] -0.20677884 -0.009812026 -0.176332443 -0.218674250
 [6,] -0.15587942 -0.064555293 -0.282882796 -0.117974190
 [7,] -0.09086363 -0.196817368  0.177814176 -0.056150268
 [8,] -0.17536307  0.035102548  0.242423936 -0.223770615
 [9,] -0.24315375  0.146502368  0.050477542  0.025718639
[10,] -0.15591071  0.042885364 -0.069631843  0.426192214
[11,] -0.05035785 -0.336841681 -0.093988180  0.169255594
[12,] -0.09273525 -0.071651205  0.048571905 -0.144733647
[13,] -0.18583902 -0.004760115  0.112681109 -0.023621705
[14,] -0.09113246 -0.140219345 -0.077396606  0.106957520
[15,] -0.05057860 -0.189585706  0.028511452 -0.008876337
[16,] -0.09241257 -0.139884238 -0.004741157  0.048135122
[17,] -0.08535772 -0.080906191 -0.029723458  0.262636519
[18,] -0.18215443  0.078717908  0.086540399  0.247322269
[19,] -0.06696147 -0.113964054  0.123029673 -0.066578837
[20,] -0.21660706  0.153849690  0.049568029 -0.114685718
```

```
>
> print("Sigma (singular values):")
[1] "Sigma (singular values):"
> print(Sigma)
         [,1]     [,2]     [,3]     [,4]
[1,] 1419.061   0.0000  0.00000  0.00000
[2,]    0.000 194.8258  0.00000  0.00000
[3,]    0.000   0.0000 45.66134  0.00000
[4,]    0.000   0.0000  0.00000 18.06956
>
> print("V (right singular vectors):")
[1] "V (right singular vectors):"
> print(V)
            [,1]        [,2]        [,3]        [,4]
[1,] -0.04239181  0.01616262 -0.06588426  0.99679535
[2,] -0.94395706  0.32068580  0.06655170 -0.04094568
[3,] -0.30842767 -0.93845891  0.15496743  0.01234261
[4,] -0.10963744 -0.12725666 -0.98347101 -0.06760284
```

202001103042
Arjan Singh Johar
Btech – 1

Practical No – 12

Aim – Using built-in function perform Support Vector machine algorithm

Code –

```
library(e1071)

data(iris)

set.seed(123)  # for reproducibility

sample_indices <- sample(1:nrow(iris), nrow(iris) * 0.7)  # 70% for training

train_data <- iris[sample_indices, ]

test_data <- iris[-sample_indices, ]

svm_model <- svm(Species ~ ., data = train_data, kernel = "linear")

predictions <- predict(svm_model, test_data)

confusion_matrix <- table(predictions, test_data$Species)

print(confusion_matrix)

print(predictions)

accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)

cat("Accuracy:", accuracy, "\n")
```

Output –

```
predictions  setosa versicolor virginica
  setosa         14          0         0
  versicolor      0         17         0
  virginica       0          1        13
> print(predictions)
         1          2          3          5         11         18         19 
    setosa     setosa     setosa     setosa     setosa     setosa     setosa 
        28         29         33         36         45         48         49 
    setosa     setosa     setosa     setosa     setosa     setosa     setosa 
        55         56         57         58         59         61         62 
versicolor versicolor versicolor versicolor versicolor versicolor versicolor 
        65         66         68         70         77         83         84 
versicolor versicolor versicolor versicolor versicolor versicolor  virginica 
        94         95         98        100        101        104        105 
versicolor versicolor versicolor versicolor  virginica  virginica  virginica 
       111        113        116        125        131        133        135 
 virginica  virginica  virginica  virginica  virginica  virginica  virginica 
       140        141        145 
 virginica  virginica  virginica 
Levels: setosa versicolor virginica
> accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)
> cat("Accuracy:", accuracy, "\n")
Accuracy: 0.9777778
>
```