Ex No: 2 Date:	**Use Matrix Transformation and Linear Algebra to process the data using R**

AIM:

To process data using matrix transformations and linear algebra techniques in R.

ALGORITHM:

1. Prints the original data matrix.
2. Centers the data by subtracting the mean of each column.
3. Prints the centered data matrix.
4. Scales the data by scaling each column to have mean 0 and standard deviation 1.
5. Prints the scaled data matrix.
6. Performs matrix multiplication by multiplying the data matrix by its transpose.
7. Prints the result of the matrix multiplication.

PROGRAM:

```
# Sample data (5 data points with 2 features) data <- matrix(c(1,
2, 3, 4, 5, 6, 7, 8, 9, 10), ncol = 2, byrow = TRUE)
print("Original Data:") print(data)


# Centering the data (subtracting mean of each column)
centered_data <- scale(data, center = TRUE, scale =
FALSE) print("Centered Data:")
print(centered_data)
# Scaling the data (scaling each column to have mean 0 and standard
deviation 1)
```

```
scaled_data <- scale(data)

print("Scaled Data:")

print(scaled_data)

# Performing matrix multiplication
# For example, let's multiply the data matrix by its
transpose matrix_multiplication <- data %*% t(data)
print("Matrix
Multiplication:")
print(matrix_multiplication)
```

OUTPUT:

```
Output
Rscript /tmp/CDoh9ICkwO.r
[1] "Original Data:"
     [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
[4,]    7    8
[5,]    9   10
[1] "Centered Data:"
     [,1] [,2]
[1,]   -4   -4
[2,]   -2   -2
[3,]    0    0
[4,]    2    2
[5,]    4    4
attr(,"scaled:center")
[1] 5 6
[1] "Scaled Data:"
          [,1]       [,2]
[1,] -1.2649111 -1.2649111
[2,] -0.6324555 -0.6324555
[3,]  0.0000000  0.0000000
[4,]  0.6324555  0.6324555
[5,]  1.2649111  1.2649111
attr(,"scaled:center")
[1] 5 6
attr(,"scaled:scale")
[1] 3.162278 3.162278
[1] "Matrix Multiplication:"
     [,1] [,2] [,3] [,4] [,5]
[1,]    5   11   17   23   29
[2,]   11   25   39   53   67
[3,]   17   39   61   83  105
[4,]   23   53   83  113  143
[5,]   29   67  105  143  181
```

RESULT:

Thus the above program is executed successfully.

Ex No: 3 Date:	**Randomly split a large dataset into training and test sets using R**

AIM:

To Randomly split a large dataset into training and test sets using R.

ALGORITHM:

1. Loads the 'mtcars' dataset.
2. Assigns the dataset to a variable named my_data.
3. Sets a seed for reproducibility using set.seed(123).
4. Generates random indices for the training and test sets. 80% of the data are randomly selected for the training set, and the remaining are used for the test set.
5. Creates the training set (train_data) by subsetting my_data based on the generated training indices.
6. Creates the test set (test_data) by subsetting my_data based on the generated test indices.
7. Prints the dimensions of the training and test sets

PROGRAM:

```
data(mtcars) # Load the 'mtcars' dataset my_
data<- mtcars
my_data <- iris # Load the 'iris' dataset from the datasets package


set.seed(123)

# Generate random indices for training and
test sets n <- nrow(my_data) # Number of
rows in the dataset
train_indices <- sample(1:n, size = 0.8*n, replace = FALSE) # 80% for training
test_indices <- setdiff(1:n, train_indices) # Remaining for testing
```

```
# Create training and test sets using the generated indices
train_data<- my_data[train_indices, ]
test_data<-my_data[test_indices, ]

# Print dimensions of training and test sets
cat("Training set dimensions:",
dim(train_data), "\n")
cat("Test set dimensions:", dim(test_data), "\n"
```

OUTPUT:

```
Training set dimensions: 120 5
Test set dimensions: 30 5
```

RESULT :

Thus the above program is executed successfully

Ex No: 4 Date:	**Apply various statistical and machine learning functions Using R**

AIM:

To Apply various statistical and machine learning functions Using R.

ALGORITHM:

1. Loads the mtcars dataset.
2. Checks the structure of the dataset using str() to understand its variables.
3. Fits a linear regression model to predict mpg (miles per gallon) using cyl (number of cylinders) as the predictor variable.
4. Summarizes the fitted regression model using summary().
5. Makes predictions on the mtcars dataset using the fitted model.
6. Prints the first few predicted values.

PROGRAM:

```
# Load the built-in mtcars dataset
data(mtcars)

# Check the structure of the dataset
str(mtcars)

# Fit a linear regression model to predict mpg using the number of cylinders
model <- lm(mpg ~ cyl, data = mtcars)

# Summary of the regression model
summary(model)

# Make predictions using the fitted model
predictions <- predict(model, newdata = mtcars)
```

```
# Print the first few predicted values
cat("First few predicted mpg values:\n")
head(predictions)

# Model evaluation: R-squared and RMSE
r_squared <- summary(model)$r.squared
rmse <- sqrt(mean((mtcars$mpg - predictions)^2))

cat("\nR-squared:", r_squared, "\n")
cat("Root Mean Squared Error (RMSE):", rmse, "\n")

# Visualization of the regression line
plot(mtcars$cyl, mtcars$mpg,
     main = "Linear Regression: MPG vs Cylinders",
     xlab = "Number of Cylinders", ylab = "Miles Per Gallon (mpg)",
     pch = 19, col = "blue")

# Add regression line
abline(model, col = "red", lwd = 2)

# Add predicted values as points
points(mtcars$cyl, predictions, col = "green", pch = 4)
legend("topright", legend = c("Actual", "Predicted", "Regression Line"),
col = c("blue", "green", "red"), pch = c(19, 4, NA), lty = c(NA, NA, 1),
lwd = c(NA, NA, 2))
```

OUTPUT:

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  37.8846     2.0738   18.27  < 2e-16 ***
cyl          -2.8758     0.3224   -8.92 6.11e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.206 on 30 degrees of freedom
Multiple R-squared:  0.7262,    Adjusted R-squared:  0.7171
F-statistic: 79.56 on 1 and 30 DF,  p-value: 6.113e-10
```

RESULT :

Thus the above program is executed successfully.

Ex No: 5 Date:	**Provide an analysis of the reliability and goodness of fit of the results Using R**

AIM:

To Provide an analysis of the reliability and goodness of fit of the results Using R

ALGORITHM:

1. Load the 'mtcars' dataset.
2. Create training and test sets by randomly splitting the dataset.
3. Fit a linear regression model using the training set.
4. Make predictions on both the training and test sets.
5. Compute Mean Squared Error (MSE) and R-squared for both sets.
6. Create visualizations to analyze the distribution of variables and relationships between them.

PROGRAM:

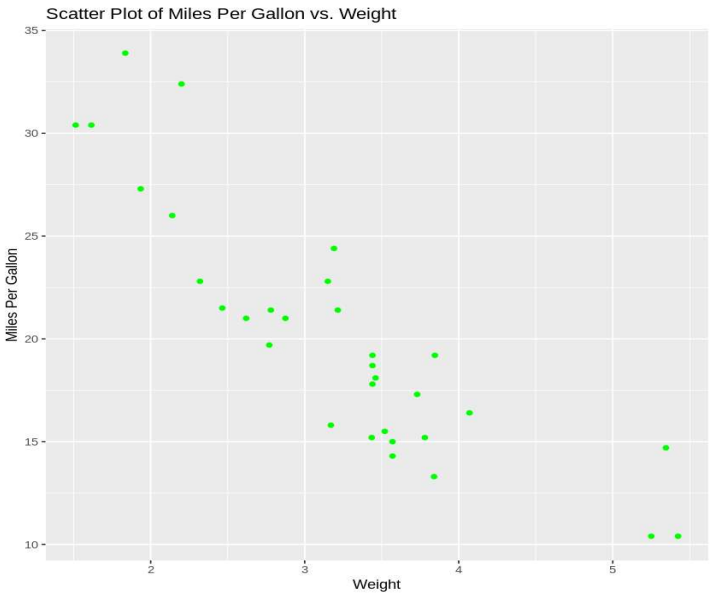
```
# Load the 'mtcars' dataset data(mtcars)

my_data <- mtcars

# Set seed for reproducibility set.seed(123)

# Generate random indices for training and test sets

n <- nrow(my_data) # Number of rows in the

dataset
```

```
train_indices <- sample(1:n, size = 0.8*n, replace = FALSE) # 80% for training
test_indices <- setdiff(1:n, train_indices) # Remaining for testing

# Create training and test sets using the generated indices train_data
<- my_data[train_indices, ]
test_data <- my_data[test_indices, ]

# Fit a linear regression model using the training set model
<- lm(mpg ~ ., data = train_data)

# Make predictions on both training and test sets
train_predictions <- predict(model, newdata = train_data)
test_predictions <- predict(model, newdata = test_data)

# Compute Mean Squared Error (MSE)
train_mse <- mean((train_data$mpg - train_predictions)^2)
test_mse<- mean((test_data$mpg - test_predictions)^2)

# Compute R-squared
train_r_squared <- summary(model)$r.squared
test_r_squared<- cor(test_data$mpg, test_predictions)^2
# Print the results cat("Training Set Metrics:\n")
cat("Mean Squared Error (MSE):", train_mse, "\n")
cat("R-squared:", train_r_squared, "\n\n")
```

```
 cat("Test Set Metrics:\n")

cat("Mean Squared Error (MSE):", test_mse, "\n") cat("R-squared:",

test_r_squared, "\n")

# Load required libraries library(ggplot2)
# Histogram of mpg (miles per gallon) ggplot(my_data, aes(x =

mpg)) + geom_histogram(binwidth = 2, fill = "skyblue", color =

"black") + labs(title = "Distribution of Miles Per Gallon", x =

"Miles Per Gallon", y = "Frequency")

# Scatter plot of mpg vs. horsepower (hp) ggplot(my_data, aes(x = hp, y = mpg)) +
geom_point(color = "blue")


labs(title = "Scatter Plot of Miles Per Gallon vs. Horsepower",

x = "Horsepower",

y = "Miles Per Gallon")


# Scatter plot of mpg vs. weight (wt)
ggplot(my_data, aes(x= wt, y = mpg)) + geom_point(color = "green") + labs(title

= "Scatter Plot of Miles Per Gallon vs. Weight", x =

"Weight",y = "Miles Per Gallon")


# Scatter plot of mpg vs. number of cylinders (cyl) ggplot(my_data,

aes(x = cyl, y = mpg)) + geom_point(color = "red") + labs(title =

"Scatter Plot of Miles Per Gallon vs. Number of Cylinders", x =

"Number of Cylinders", y = "Miles Per
```

OUTPUT:

Test Set Metrics:
Mean Squared Error (MSE): 19.13392
R-squared: 0.2879197
$x[1] "Horsepower"
$y[1] "Miles Per Gallon"
$title
[1] "Scatter Plot of Miles Per Gallon vs. Horsepower"
attr(,"class")
[1] "labels"



RESULT :

Thus the above program is executed successfully.

Ex No: 6 Date:	**Resume Parser Using AI**

AIM:

To Build a Resume Parser Using AI

ALGORITHM:

1. Input:
2. Receive a resume document as input.
3. Text Extraction:
4. Extract text from the resume document.
5. Information Extraction:
6. Use regular expressions to extract basic information such as name, email, phone number, education, and work experience.
7. Print or store the extracted information.

PROGRAM:

```
import re

# Sample resume text resume_text
= """
John Doe
123 Main St, Anytown, USA john.doe@email.com
(123) 456-7890

Education:
Bachelor of Science in Computer Science
University of Anytown, Anytown, USA
Graduated: May 2015
Work Experience:
Software Engineer
XYZ Tech, Anytown, USA
```

June 2015 - Present

- Developed web applications using Django framework
- Implemented RESTful APIs for data exchange """

```
# Define regular expressions for extracting information name_regex =
re.compile(r'^([A-Z][a-z]+)\s([A-Z][a-z]+)$')          email_regex          =
re.compile(r'\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a- z]{2,}\b')
phone_regex = re.compile(r'\(?\d{3}\)?[-.\s]?\d{3}[-.\s]?\d{4}')

# Extract name
match_name = name_regex.search(resume_text)
if match_name:
    full_name = match_name.group() print("Name:",
    full_name)
# Extract email
match_email = email_regex.search(resume_text)
if match_email:
    email = match_email.group()
    print("Email:", email)

# Extract phone number
match_phone = phone_regex.search(resume_text)

if match_phone:
    phone_number = match_phone.group() print("Phone
    Number:", phone_number)
#   Extract   education
education_match=re.search(r'Education:(.*?)Work Experience:', resume_text,
re.DOTALL)
```

```
if education_match:
education_details= education_match.group(1).strip()
print("Education:")
print(education_details)
# Extract work experience
work_experience_match = re.search(r'Work Experience:(.*?)$', resume_text,
re.DOTALL)
ifwork_experience_match:
work_experience_details=work_experience_match.group(1).strip()
print("Work Experience:")
 print(work_experience_details)
```

OUTPUT:

```
Email: john.doe@email.com
Phone Number: (123) 456-7890
Education:
Bachelor of Science in Computer Science
University of Anytown, Anytown, USA
Graduated: May 2015
Work Experience:
Software Engineer
XYZ Tech, Anytown, USA
June 2015 - Present
- Developed web applications using Django framework
- Implemented RESTful APIs for data exchange
```

RESULT:

Thus the above program is executed successfully.

Ex No: 7 Date:	**Fake news detector using AI**

AIM:

To Build a Fake news detector using AI

ALGORITHM:

1. Data Collection: Gather labeled news articles dataset.
2. Preprocessing: Clean, tokenize, and normalize text data.
3. Feature Extraction: Convert text into numerical features (TF-IDF, word embeddings).
4. Model Training: Train a machine learning model (e.g., logistic regression).
5. Evaluation: Assess model performance using accuracy metrics.
6. Deployment: Deploy the trained model for real-time detection.
7. Feedback Loop: Continuously improve the model based on feedback.

PROGRAM:

```
import pandas as pd
from sklearn.model_selection
import train_test_split
from sklearn.feature_extraction.text
import TfidfVectorizer
from sklearn.linear_model
import LogisticRegression
from sklearn.metrics import classification_report,
accuracy_score
import nltk
from nltk.corpus import stopwords
```

```
nltk.download('stopwords')
stop_words = set(stopwords.words('english'))
data = {
   'text': [
      'Climate change is a hoax.',
      'The Earth revolves around the Sun.',
      '5G technology spreads viruses.',
         'Scientists discover a new species in the
Amazon.',
      'Eating carrots improves night vision.',
      'Electric cars are better for the environment.'
   ],
   'label': [1, 0, 1, 0, 1, 0]
}
df = pd.DataFrame(data)
df['text'] = df['text'].str.lower().str.replace('[^\w\s]',
'', regex=True)
df['text'] = df['text'].apply(lambda x: ' '.join([word
for word in x.split() if word not in stop_words]))
X_train, X_test, y_train, y_test =
train_test_split(df['text'], df['label'], test_size=0.3,
random_state=42)
tfidf = TfidfVectorizer()
X_train_vec = tfidf.fit_transform(X_train)
X_test_vec = tfidf.transform(X_test)
model = LogisticRegression()
model.fit(X_train_vec, y_train)
y_pred = model.predict(X_test_vec)
print("Accuracy:", accuracy_score(y_test, y_pred))
```

```
print("ClassificationReport:\n",
classification_report(y_test, y_pred))
sample = ["Vaccines contain microchips"]
sample_vec = tfidf.transform(sample)
prediction = model.predict(sample_vec)
print(f"Prediction for sample: {'Fake' if
prediction[0]==1 else 'Real'}")
```

OUTPUT:

Classification Report:

Label	Precision	Recall	F1-Score	Support
0	0.50	1.00	0.67	1
1	0.00	0.00	0.00	1
Accuracy			0.50	2
Macro Avg	0.25	0.50	0.33	2
Weighted Avg	0.25	0.50	0.33	2

Prediction for sample: Real

RESULT :

Thus the above program is executed successfully.

Ex No: 8 Date:	**Instagram Spam Detector using AI**

AIM:

To Build a Instagram Spam Detector using AI

ALGORITHM:

1. Data Collection: Gather a dataset of Instagram comments labeled as spam or not spam.
2. Preprocessing: Clean, tokenize, and normalize text data.
3. Feature Extraction: Convert text into numerical features (TF-IDF, word embeddings).
4. Model Training: Train a machine learning model (e.g., logistic regression).
5. Evaluation: Assess model performance using accuracy metrics.
6. Deployment: Integrate the trained model for real-time detection.
7. Feedback Loop: Continuously improve the model based on feedback

PROGRAM:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, accuracy_score
import nltk
from nltk.corpus import stopwords
nltk.download('stopwords')
stop_words = set(stopwords.words('english'))
```

```
data = {
   'message': [

      'Get rich quick with this one simple trick!',
      'Your profile is amazing, keep it up!',
      'Claim your free gift card now!',
      'Let’s work together on a project!',
      'You’ve won a lottery! Click here to claim.',
      'Great post! Looking forward to more.',
      'Exclusive offer just for you! Act now!',
      'Thanks for sharing such valuable content!'
   ],
   'label': [1, 0, 1, 0, 1, 0, 1, 0]
}
df = pd.DataFrame(data)
df['message'] = df['message'].str.lower().str.replace('[^\w\s]', '', regex=True)
df['message'] = df['message'].apply(lambda x: ' '.join([word for word in x.split() if
word not in stop_words]))
X_train, X_test, y_train, y_test = train_test_split(df['message'], df['label'],
test_size=0.3, random_state=42)
tfidf = TfidfVectorizer()
X_train_vec = tfidf.fit_transform(X_train)
X_test_vec = tfidf.transform(X_test)
model = LogisticRegression()
model.fit(X_train_vec, y_train)
y_pred = model.predict(X_test_vec)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
sample = ["Hey! You just won a free vacation. DM us now!"]
sample_vec = tfidf.transform(sample)
prediction = model.predict(sample_vec)
print(f"Prediction for sample: {'Spam' if prediction[0]==1 else 'Not Spam'}")
```

OUTPUT:

Classification Report:

Label	Precision	Recall	F1-Score	Support
0	0.00	0.00	0.00	2
1	0.33	1.00	0.50	1
Accuracy			0.33	3
Macro Avg	0.17	0.50	0.25	3
Weighted Avg	0.11	0.33	0.17	3

Prediction for sample: Spam

RESULT:

Thus the above program is executed successfully

Ex No: 9 Date:	**ANIMAL SPECIES PREDICTION USING AI**

AIM:

To Build a Animal Species Prediction using AI.

ALGORITHM:

1. Collect a dataset with text descriptions and corresponding animal species labels.
2. Clean and preprocess the text by lowering case, removing punctuation, and eliminating stopwords.
3. Split the dataset into training and testing sets for evaluation.
4. Convert text data into numerical form using TF-IDF vectorization.
5. Train a Logistic Regression model on the TF-IDF features and species labels.
6. Evaluate the model using accuracy and classification metrics on the test set.
7. Use the trained model to predict species from new text descriptions.

PROGRAM:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import
TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report,
accuracy_score
import nltk
from nltk.corpus import stopwords
nltk.download('stopwords')
stop_words = set(stopwords.words('english'))
```

```
data = {
    'description': [
        'Fast-running bird that cannot fly',
        'Large mammal with antlers',
        'Small amphibian that hops and croaks',
        'Marine creature with eight legs',
        'Striped horse-like animal found in Africa',
        'Large bird of prey with sharp talons',
        'Reptile with a hard shell',
        'Small insect that produces honey'
    ],
    'species': [
        'Ostrich', 'Deer', 'Frog', 'Octopus', 'Zebra', 'Eagle',
'Turtle', 'Bee'
    ]
}
df = pd.DataFrame(data)
df['description'] =
df['description'].str.lower().str.replace('[^\w\s]', '',
regex=True)
df['description'] = df['description'].apply(lambda x: '
'.join([word for word in x.split() if word not in
stop_words]))
X_train, X_test, y_train, y_test =
train_test_split(df['description'], df['species'],
test_size=0.3, random_state=42)
tfidf = TfidfVectorizer()
X_train_vec = tfidf.fit_transform(X_train)
X_test_vec = tfidf.transform(X_test)
model = LogisticRegression(max_iter=1000)
```

```
model.fit(X_train_vec, y_train)
y_pred = model.predict(X_test_vec)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n",
classification_report(y_test, y_pred))
sample = ["Striped horse-like animal found in Africa'"]
sample_vec = tfidf.transform(sample)
prediction = model.predict(sample_vec)
print(f"Prediction for sample: {prediction[0]}")
```

OUTPUT:

Classification Report:

Label	Precision	Recall	F1-Score	Support
Deer	0.00	0.00	0.00	1.0
Eagle	0.00	0.00	0.00	1.0
Ostrich	0.00	0.00	0.00	1.0
Zebra	0.00	0.00	0.00	0.0
Accuracy			0.00	3.0
Macro Avg	0.00	0.00	0.00	3.0
Weighted Avg	0.00	0.00	0.00	3.0

Prediction for sample: Zebra

RESULT :

Thus the above program is executed successfully.