Name-Vedashree Bhalerao

Roll No-2213191

Batch-B

Practical No: 7

Aim: Analytical representation of Linear Regression using Movie recommendation dataset

Theory:

Regression shows a line or curve that passes through all the data points on the target-predictor graph in such a way that the vertical distance between the data points and the regression line is minimum.
There are two types of linear regression.

Simple linear regression: uses only one independent variable
Multiple linear regression: uses two or more independent variables

Linear Regression is a commonly used type of predictive analysis. Linear Regression is a statistical approach for modeling the relationship between a dependent variable and a given set of independent variables. It is predicted that a straight line can be used to approximate the relationship. The goal of linear regression is to identify the line that minimizes the discrepancies between the observed data points and the line's anticipated values. In Machine Learning Linear regression is one of the easiest and most popular Machine Learning algorithms.

- It is a statistical method that is used for predictive analysis.
- Linear regression makes predictions for continuous/real or numeric variables such as sales, salary, age, product price, etc.
- Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (x) variables, hence called as linear regression. Since linear regression shows the linear relationship, which means it *finds how the value of the dependent variable changes according to the value of the independent variable.*

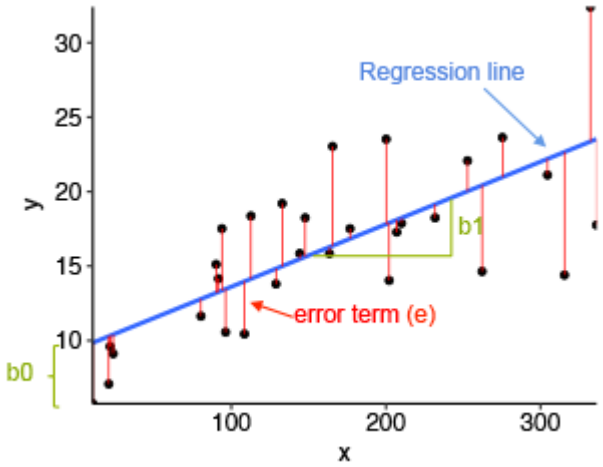
It is a statistical method that allows us to summarize and study relationships between two continuous (quantitative) variables. One variable denoted x is regarded as an independent variable and the other one denoted y is regarded as a dependent variable. It is assumed that the two variables are linearly related. Hence, we try to find a linear function that predicts the response value(y) as accurately as possible as a function of the feature or independent variable(x).

$$\mathbf{Y = \beta_0 + \beta_1 X + \varepsilon}$$

- The dependent variable, also known as the response or outcome variable is represented by the letter Y.
- The independent variable, often known as the predictor or explanatory variable, is denoted by the letter X.
- The intercept, or value of Y when X is zero, is represented by the β_0.
- The slope or change in Y resulting from a one-unit change in X is represented by the β_1.
- The error term or the unexplained variation in Y is represented by the ε.

The figure below illustrates the linear regression model, where:

the best-fit regression line is in blue
the intercept (b0) and the slope (b1) are shown in green
the error terms (e) are represented by vertical red lines



From the scatter plot above, it can be seen that not all the data points fall exactly on the fitted regression line. Some of the points are above the blue curve and some are below it; overall, the residual errors (e) have approximately mean zero.

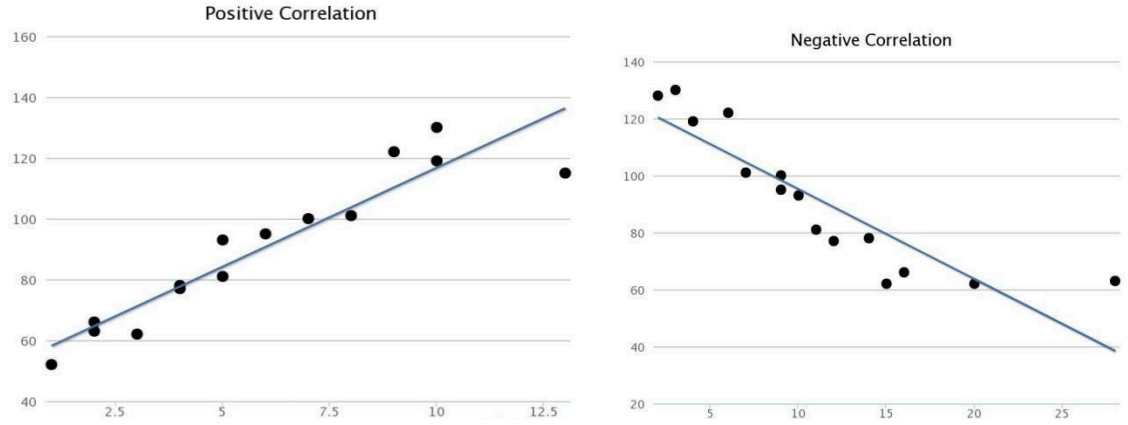
The sum of the squares of the residual errors are called the Residual Sum of Squares or RSS.

The average variation of points around the fitted regression line is called the Residual Standard Error (RSE). This is one the metrics used to evaluate the overall quality of the fitted regression model. The lower the RSE, the better it is.

Linear Regression Line

A linear line showing the relationship between the dependent and independent variables is called a regression line. A regression line can show two types of relationship:

Positive Linear Relationship: If the dependent variable increases on the Y-axis and the independent variable increases on the X-axis, then such a relationship is termed as a Positive linear relationship.



Negative Linear Relationship: If the dependent variable decreases on the Y-axis and independent variable increases on the X-axis, then such a relationship is called a negative linear relationship.

Example:

For understanding the concept let's consider a salary dataset where it is given the value of the dependent variable (salary) for every independent variable (years experienced).

Defined for general purposes:
x as a feature vector, i.e x = [x_1, x_2, …., x_n],
y as a response vector, i.e y = [y_1, y_2, …., y_n]
for n observations (in the example, n=10).

Years experienced	Salary
1.1	39343.00
1.3	46205.00
1.5	37731.00
2.0	43525.00
2.2	39891.00
2.9	56642.00
3.0	60150.00
3.2	54445.00
3.2	64445.00
3.7	57189.00

R CODE

```
# 1 Create the data frame
data <- data.frame(
  Years_Exp = c(1.1, 1.3, 1.5, 2.0, 2.2, 2.9, 3.0, 3.2, 3.2, 3.7),
  Salary = c(39343.00, 46205.00, 37731.00, 43525.00,
             39891.00, 56642.00, 60150.00, 54445.00, 64445.00, 57189.00))

# 2  Create the scatter plot
plot(data$Years_Exp, data$Salary,
     xlab = "Years Experienced",
     ylab = "Salary",
     main = "Scatter Plot of Years Experienced vs Salary")
```

Now, let's find a line that fits the above scatter plot through which we can predict any value of y or response for any value of x
The line which best fits is called the Regression line.

The equation of the regression line is given by: **y = a + bx**
Where y is the predicted response value, a is the y-intercept,
x is the feature value and b is the slope.
To create the model, evaluate the values of regression coefficients a and b. And as soon as the estimation of these coefficients is done, the response model can be predicted using the Least Square Technique.

The basic syntax for regression analysis in R is
lm(Y ~ model)

 # 3 implement Simple Linear Regression:
install.packages('caTools')*
library(caTools)
split = sample.split(data$Salary, SplitRatio = 0.7)
trainingset = subset(data, split == TRUE)
testset = subset(data, split == FALSE)

# Fitting Simple Linear Regression to the Training set
lm.r= lm(formula = Salary ~ Years_Exp,
data = trainingset)
#Summary of the model
summary(lm.r)

*The caTools package in R Programming Language is a versatile and widely used package that provides a collection of tools for data analysis, including functions for splitting data, running moving averages, and performing various mathematical and statistical operations.

Output:

```
Call:
lm(formula = Salary ~ Years_Exp, data = trainingset)
Residuals:
      1       2       3       5       6       8      10
  463.1  5879.1 -4041.0 -6942.0  4748.0   381.9  -489.1
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    30927       4877   6.341  0.00144 **
Years_Exp       7230       1983   3.645  0.01482 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
Residual standard error: 4944 on 5 degrees of freedom
Multiple R-squared:  0.7266,    Adjusted R-squared:  0.6719
F-statistic: 13.29 on 1 and 5 DF,  p-value: 0.01482
```

Call: Using the “lm” function, we will be performing a regression analysis of “Salary” against “Years_Exp” according to the formula displayed on this line.

Residuals: Each residual in the “Residuals” section denotes the difference between the actual salaries and predicted values. These values are unique to each observation in the data set. For instance, observation 1 has a residual of 463.1.

Coefficients: Linear regression coefficients are revealed within the contents of this section.
(Intercept): The estimated salary when Years_Exp is zero is 30927, which represents the intercept for this case.
Years_Exp: For every year of experience gained, the expected salary is estimated to increase by 7230 units according to the coefficient for “Years_Exp”. This coefficient value suggests that each year of experience has a significant impact on the estimated salary.
Estimate:The model’s estimated coefficients can be found in this column.
Std. Error: “More precise estimates” can be deduced from smaller standard errors that are a gauge of the ambiguity that comes along with coefficient estimates.
t value: The coefficient estimate’s standard error distance from zero is measured by the t-value. Its purpose is to examine the likelihood of the coefficient being zero by testing the null hypothesis. A higher t-value’s absolute value indicates a higher possibility of statistical significance pertaining to the coefficient.
Pr(>|t|): This column provides the p-value associated with the t-value. The p-value indicates the probability of observing the t-statistic (or more extreme) under the null hypothesis that the coefficient is zero. In this case, the p-value for the intercept is 0.00144, and for “Years_Exp,” it is 0.01482.
Signif. codes: These codes indicate the level of significance of the coefficients.
Residual standard error: This is a measure of the variability of the residuals. In this case, it’s 4944, which represents the typical difference between the actual salaries and the predicted salaries.
Multiple R-squared: R-squared (R^2) is a measure of the goodness of fit of the model. It represents the proportion of the variance in the dependent variable that is explained by the independent variable(s). In this case, the R-squared is 0.7266, which means that approximately 72.66% of the variation in salaries can be explained by years of experience.
Adjusted R-squared: The adjusted R-squared adjusts the R-squared value based on the number of predictors in the model. It accounts for the complexity of the model. In this case, the adjusted R-squared is 0.6719.

F-statistic: The F-statistic is used to test the overall significance of the model. In this case, the F-statistic is 13.29 with 1 and 5 degrees of freedom, and the associated p-value is 0.01482. This p-value suggests that the model as a whole is statistically significant.
In summary, this linear regression analysis suggests that there is a significant relationship between years of experience (Years_Exp) and salary (Salary). The model explains approximately 72.66% of the variance in salaries, and both the intercept and the coefficient for "Years_Exp" are statistically significant at the 0.01 and 0.05 significance levels, respectively.

Recommendation System

A recommendation system is an artificial intelligence or AI algorithm, usually associated with machine learning that uses Big Data to suggest or recommend additional products to consumers. These can be based on various criteria, including past purchases, search history, demographic information, and other factors. Recommender systems are highly useful as they help users discover products and services they might otherwise have not found on their own.

Types of Recommendation Systems:

- Collaborative filtering algorithms recommend items (this is the filtering part) based on preference information from many users (this is the collaborative part).
- Content filtering, by contrast, uses the attributes or features of an item (this is the content part) to recommend other items similar to the user's preferences.
- Hybrid recommender systems combine the advantages of the types above to create a more comprehensive recommending system.
- Context filtering includes users' contextual information in the recommendation process.

R Code for Analytical representation of linear regression using Movie recommendation dataset

1. **Load Required Libraries:**

   Make sure you have the necessary libraries installed `ggplot2` for visualization and `dplyr` for data manipulation.

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

library(ggplot2)
library(dplyr)
```

2. **Load the Dataset:**

   Load dataset into R. Dataset name "`movies_data.csv`".

```
# Load dataset
movies_data <- read.csv("movies_data.csv")

# View the first few rows of the dataset
head(movies_data)
```

3. **Explore and Prepare the Data:**

   Check the structure of your dataset and prepare it for analysis.
   Dataset name "`movies_data.csv`" columns `Rating`, `Genre`, and `Budget`.

```
# Check the structure of the dataset
str(movies_data)

# Convert categorical variables to factors if necessary
movies_data$Genre <- as.factor(movies_data$Genre)

# Summary statistics
summary(movies_data)
```

4. **Create Training and Test Sets:**

   Split the dataset into training and test sets.
   This helps to evaluate the performance of your model.

```
# Set seed for reproducibility
set.seed(123)

# Create a training set (70%) and test set (30%)
sample_indices <- sample(seq_len(nrow(movies_data)), size = 0.7 * nrow(movies_data))
train_set <- movies_data[sample_indices, ]
test_set <- movies_data[-sample_indices, ]
```

5. **Fit the Linear Regression Model:**

   Fit a linear regression model to the training set. Let's predict `Rating` based on `Genre` and `Budget`.

MovieID	Genre	Budget	Rating
1	Action	15000000	7.8
2	Comedy	5000000	6.5
3	Drama	10000000	8.2
4	Action	20000000	7.9
5	Comedy	6000000	6.9
6	Drama	12000000	8
7	Action	18000000	7.5
8	Comedy	5500000	6.7
9	Drama	11000000	8.1
10	Action	16000000	7.7

```
# Fit linear regression model
model <- lm(Rating ~ Genre + Budget, data = train_set)

# Summary of the model
```

```
summary(model)
```

6. **Evaluate the Model:**

   Use the test set to evaluate the model's performance.

```
# Predict on the test set
predictions <- predict(model, newdata = test_set)

# Combine predictions with actual values
results <- data.frame(Actual = test_set$Rating, Predicted = predictions)

# Calculate Mean Squared Error
mse <- mean((results$Actual - results$Predicted)^2)
print(paste("Mean Squared Error:", mse))
```

Formula for MSE
The formula for Mean Squared Error is:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Where:

- *n is the number of observations.*
- *y_i is the actual value of the i-th observation.*
- *$\hat{y}_i$ is the predicted value of the i-th observation.*

```
# Plot Actual vs Predicted values
ggplot(results, aes(x = Actual, y = Predicted)) +
  geom_point() +
  geom_abline(intercept = 0, slope = 1, color = "red") +
  labs(title = "Actual vs Predicted Ratings", x = "Actual Rating", y = "Predicted Rating")
```

7. **Visualize the Results:**

   Visualize the results to better understand the model's performance.

```
# Plot residuals
residuals <- results$Actual - results$Predicted

ggplot(data.frame(Residuals = residuals), aes(x = Residuals)) +
  geom_histogram(bins = 30, fill = "blue", color = "black") +
  labs(title = "Histogram of Residuals", x = "Residuals", y = "Frequency")
```

*/*If you have other features or a different structure, modify the `lm()` function accordingly.*/*

Movie Recommendation Dataset

```
MovieID,Genre,Budget,Rating
1,Action,15000000,7.8
2,Comedy,5000000,6.5
3,Drama,10000000,8.2
4,Action,20000000,7.9
5,Comedy,6000000,6.9
6,Drama,12000000,8.0
7,Action,18000000,7.5
8,Comedy,5500000,6.7
9,Drama,11000000,8.1
```

```
10,Action,16000000,7.7
```

Create the Dataset in R

```
# Create a data frame with the synthetic data
movies_data <- data.frame(
  MovieID = 1:10,
  Genre = factor(c("Action", "Comedy", "Drama", "Action", "Comedy", "Drama", "Action",
"Comedy", "Drama", "Action")),
  Budget = c(15000000, 5000000, 10000000, 20000000, 6000000, 12000000, 18000000, 5500000,
11000000, 16000000),
  Rating = c(7.8, 6.5, 8.2, 7.9, 6.9, 8.0, 7.5, 6.7, 8.1, 7.7)
)
```

Where:

MovieID: Unique identifier for each movie.
Genre: Genre of the movie (e.g., Action, Comedy, Drama).
Budget: Production budget of the movie.
Rating: Movie rating (e.g., from 1 to 10)

```
# Save the data frame to a CSV file
write.csv(movies_data, file = "movies_data.csv", row.names = FALSE)

# Print the data frame to verify
print(movies_data)
```

Assignment:

1. Linear Regression Analysis on House Price.
2. Using the Simple Linear Regression predict the Happiness rate based on the Income.
3. Consider to evaluate the impact of advertising budgets of three Medias (youtube, facebook and newspaper) on future sales. Is this example of a problem that can be modeled with linear regression?
4. Advantages and Drawbacks of the Liner Regression model.
5. Difference between Linear and Nonlinear regression models.

CASE STUDY:

how Netflix recommendation engine works

Conclusion:

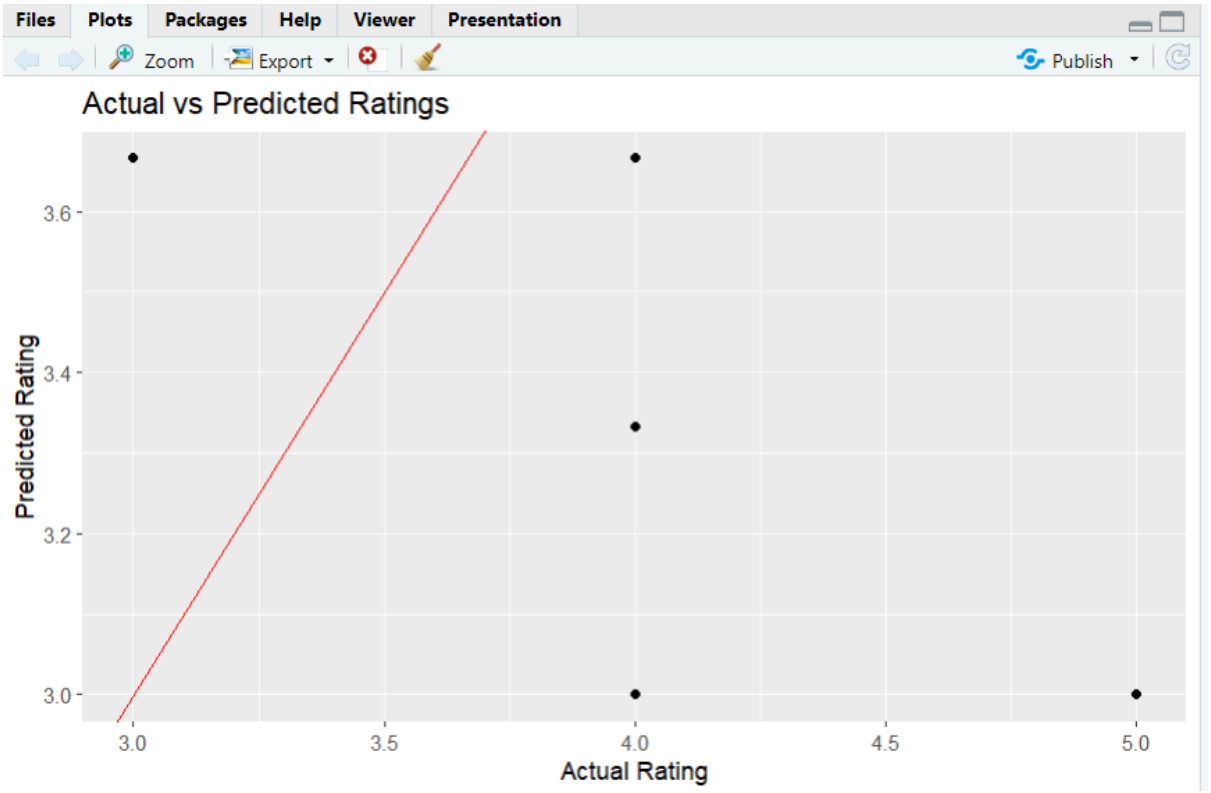
Code:

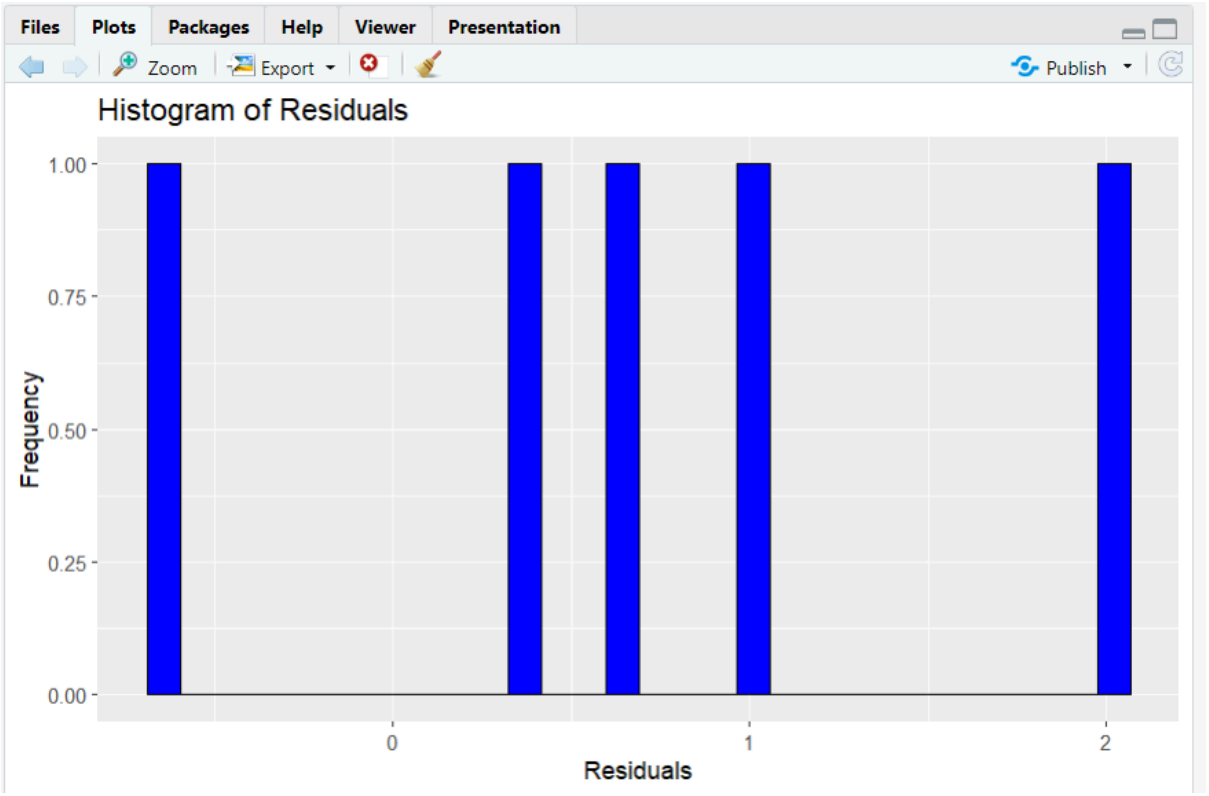
```
# Load necessary libraries

# Install these packages if you haven't already

install.packages("ggplot2")

install.packages("dplyr")

install.packages("reshape2")


# Load the libraries

library(ggplot2)

library(dplyr)

library(reshape2)


# Simulated ratings dataset

ratings_data <- data.frame(

  UserID = rep(1:5, each = 4),

  MovieID = rep(1:4, times = 5),

  Rating = c(4, 5, NA, 3, 5, 3, 4, NA, 2, NA, 4, 4, NA, 3, 5, 2, 3, 4, 2, 5)

)


# Reshape data to wide format for matrix structure

ratings_matrix <- dcast(ratings_data, UserID ~ MovieID, value.var = "Rating", fill = NA)

ratings_matrix <- as.matrix(ratings_matrix[,-1])


# Convert matrix to data frame for linear regression model

# Generate simulated Movie Data for Genre and Budget

movies_data <- data.frame(

  MovieID = 1:4,

  Genre = factor(c("Action", "Comedy", "Drama", "Action")),

  Budget = c(15000000, 5000000, 10000000, 20000000)
```

```
)

# Merge movie info with ratings to build a model dataset
rating_info <- na.omit(merge(ratings_data, movies_data, by = "MovieID"))

# Split the data into training (70%) and test (30%) sets
set.seed(123)  # Set seed for reproducibility
sample_indices <- sample(seq_len(nrow(rating_info)), size = 0.7 * nrow(rating_info))
train_set <- rating_info[sample_indices, ]
test_set <- rating_info[-sample_indices, ]

# Fit a linear regression model to predict Rating based on Genre and Budget
model <- lm(Rating ~ Genre + Budget, data = train_set)

# Display the summary of the model
summary(model)

# Predict the ratings on the test set
predictions <- predict(model, newdata = test_set)

# Combine predictions with actual values for comparison
results <- data.frame(Actual = test_set$Rating, Predicted = predictions)

# Calculate Mean Squared Error (MSE) for the model
mse <- mean((results$Actual - results$Predicted)^2)
print(paste("Mean Squared Error:", mse))

# Plot Actual vs. Predicted values
ggplot(results, aes(x = Actual, y = Predicted)) +
  geom_point() +
```

```
  geom_abline(intercept = 0, slope = 1, color = "red") +
  labs(title = "Actual vs Predicted Ratings", x = "Actual Rating", y = "Predicted Rating")

# Visualize the distribution of residuals to assess model fit
residuals <- results$Actual - results$Predicted
ggplot(data.frame(Residuals = residuals), aes(x = Residuals)) +
  geom_histogram(bins = 30, fill = "blue", color = "black") +
  labs(title = "Histogram of Residuals", x = "Residuals", y = "Frequency")
```

Output:




Files
Plots
Packages
Help
Viewer
Presentation
Zoom
Export
Publish
Histogram of Residuals
Frequency
1.00
0.75
0.50
0.25
0.00
0
1
2
Residuals