HarvardX: PH125.9x - Capstone Project

Levi Lucena - https://github.com/LeviLucena

2023-08-16

Contents

1. Introduction

Welcome to the report on the HarvardX PH125.9x Capstone Project, focusing on the analysis of the MovieLens dataset and the development of a movie recommendation model.

1.1 Project Overview

In this capstone project, we aim to explore and analyze the MovieLens dataset to gain insights into movie ratings and genres. Our objective is to develop a recommendation model that can predict movie ratings and evaluate its performance through metrics like the Root Mean Squared Error (RMSE).

1.2 Dataset

The primary dataset used for this project is the MovieLens dataset, which contains a rich collection of movie ratings and information. We'll be leveraging the ratings provided by users to create a recommendation model.

Define the path to the ratings and movies files

```
ratings_file_path <- "C:/Users/lglucena/Downloads/ml-latest-small/ratings.csv"
movies_file_path <- "C:/Users/lglucena/Downloads/ml-latest-small/movies.csv"
```

1.3 Report Structure

This report is organized into the following sections:

- Section 1: Introduction
- Section 2: Data Preprocessing
- Section 3: Model Training and Evaluation
- Section 4: Visualizations
- Section 5: Interactive Datatable
- Section 6: Conclusion
- Section 7: References

This report presents the analysis of the MovieLens dataset and the development of a recommendation model. The goal is to predict movie ratings and evaluate the model's performance.

Each section focuses on a specific aspect of the project, from data preprocessing and model training to visualizing results and providing an interactive exploration of the dataset.

1.4 About the Author

The project author, Levi Lucena, is a data enthusiast and aspiring data scientist. You can find more about Levi's work on his GitHub profile: Levi Lucena GitHub.

Now, let's delve into the details of this exciting project!

2. Data Preprocessing

In this section, we will preprocess the data to prepare it for analysis.

2.1 Reading and Loading Data

We start by reading and loading the ratings and movies data files.

Read the ratings data using read_csv from readr:

```
ratings <- readr::read_csv(ratings_file_path)
```

```
## Rows: 100836 Columns: 4
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (4): userId, movieId, rating, timestamp
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

2.2 Genre Counts

Let's begin by calculating the counts of each genre in the movies dataset.

Read the movies data using read_csv from readr:

```
movies <- readr::read_csv(movies_file_path)
```

```
## Rows: 9742 Columns: 3
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (2): title, genres
## dbl (1): movieId
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# Calculate the counts of each genre
genre_counts <- table(movies$genres)

# Create a data frame with genre and count information
genre_table <- data.frame(
  Genre = names(genre_counts),
  Count = as.numeric(genre_counts)
)

# Display the table using kable
kable(genre_table[1:10, ], caption = "Table with Film Count by Genre")
```

Table 1: Table with Film Count by Genre

Genre	Count
(no genres listed)	34
Action	60
Action|Adventure	32
Action|Adventure|Animation	18
Action|Adventure|Animation|Children	6
Action|Adventure|Animation|Children|Comedy	5
Action|Adventure|Animation|Children|Comedy|Fantasy	3
Action|Adventure|Animation|Children|Comedy|IMAX	2

Genre	Count
Action|Adventure|Animation|Children|Comedy|Romance	1
Action|Adventure|Animation|Children|Comedy|Sci-Fi	3

2.3 Year Counts

Next, we will filter out movies without a year in their title and calculate the counts of movies by release year.

```
# Filter out titles without a year
movies_with_year <- movies[grep("\\(\\d{4}\\)$", movies$title), ]

# Calculate the counts of movies by year
year_counts <- table(substring(movies_with_year$title, nchar(movies_with_year$title) - 4))

# Create a data frame with year and count information
year_table <- data.frame(
  Year = names(year_counts),
  Count = as.numeric(year_counts)
)

# Display the table using kable
kable(year_table, caption = "Table with Film Count by Release Year")
```

Table 2: Table with Film Count by Release Year

Year	Count
1902)	1
1903)	1
1908)	1
1915)	1
1916)	4
1917)	1
1919)	1
1920)	2
1921)	1
1922)	1
1923)	4
1924)	5
1925)	4
1926)	5
1927)	7
1928)	4
1929)	4
1930)	5
1931)	14
1932)	9
1933)	12
1934)	11
1935)	13
1936)	18

Year	Count
1937)	16
1938)	15
1939)	23
1940)	25
1941)	18
1942)	23
1943)	10
1944)	16
1945)	17
1946)	23
1947)	20
1948)	20
1949)	25
1950)	21
1951)	22
1952)	16
1953)	30
1954)	23
1955)	36
1956)	30
1957)	33
1958)	31
1959)	37
1960)	37
1961)	34
1962)	40
1963)	39
1964)	43
1965)	47
1966)	42
1967)	42
1968)	42
1969)	35
1970)	33
1971)	47
1972)	39
1973)	59
1974)	45
1975)	42
1976)	44
1977)	63
1978)	59
1979)	69
1980)	89
1981)	92
1982)	87
1983)	83
1984)	101
1985)	126
1986)	139
1987)	153
1988)	165

Year	Count
1989)	142
1990)	147
1991)	147
1992)	167
1993)	198
1994)	237
1995)	259
1996)	276
1997)	260
1998)	258
1999)	263
2000)	283
2001)	294
2002)	311
2003)	279
2004)	279
2005)	273
2006)	295
2007)	284
2008)	269
2009)	282
2010)	247
2011)	254
2012)	233
2013)	239
2014)	278
2015)	274
2016)	218
2017)	147
2018)	41

By completing these preprocessing steps, we have organized the data and calculated genre and year counts, which will be useful for our subsequent analysis.

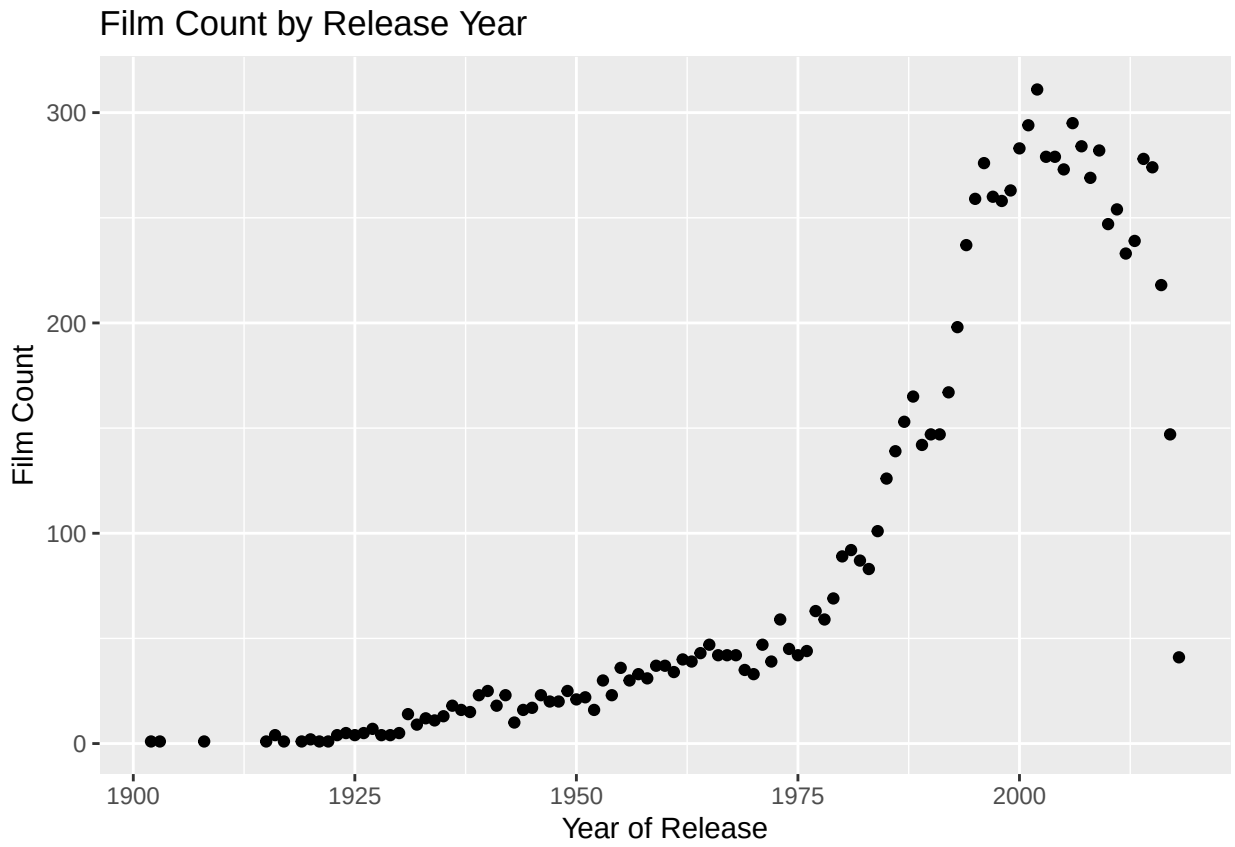
```
# Extract year from title using regular expressions
movies <- movies %>%
  mutate(Year = ifelse(grepl("\\(\\d{4}\\)$", title), as.numeric(gsub(".*\\((\\d{4})\\)$", "\\1", title)
```

```
## Warning: There was 1 warning in `mutate()`.
## i In argument: `Year = ifelse(...)`.
## Caused by warning in `ifelse()`:
## ! NAs introduzidos por coerção
```

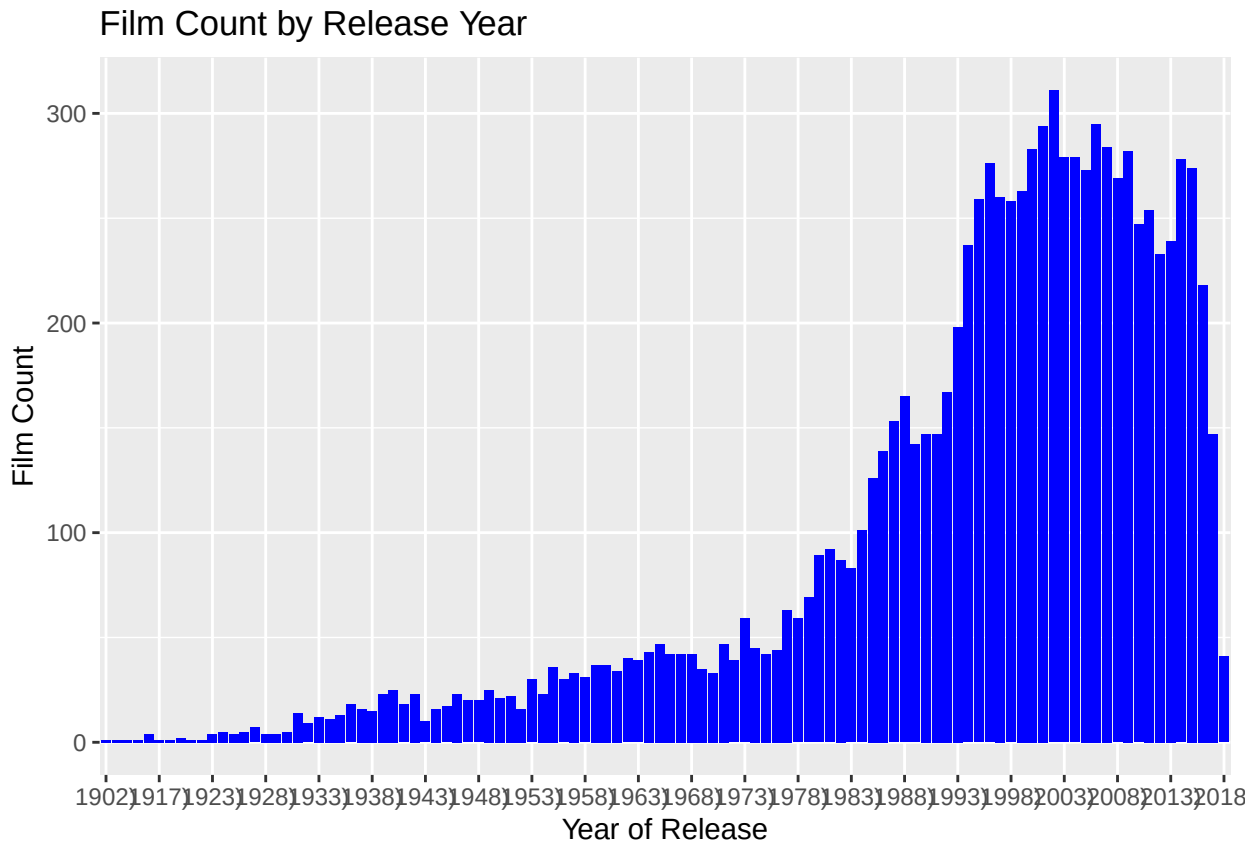
```
# Calculate the counts of movies by year
year_counts <- table(movies$Year, useNA = "ifany")

# Convert year_counts to a data frame
year_table <- data.frame(Year = as.numeric(names(year_counts)), Count = as.numeric(year_counts))

# Create a scatter plot of movie counts by year of release
ggplot(year_table, aes(x = Year, y = Count)) +
```

```
  geom_point(color = "black") +
  labs(title = "Film Count by Release Year", x = "Year of Release", y = "Film Count")
```

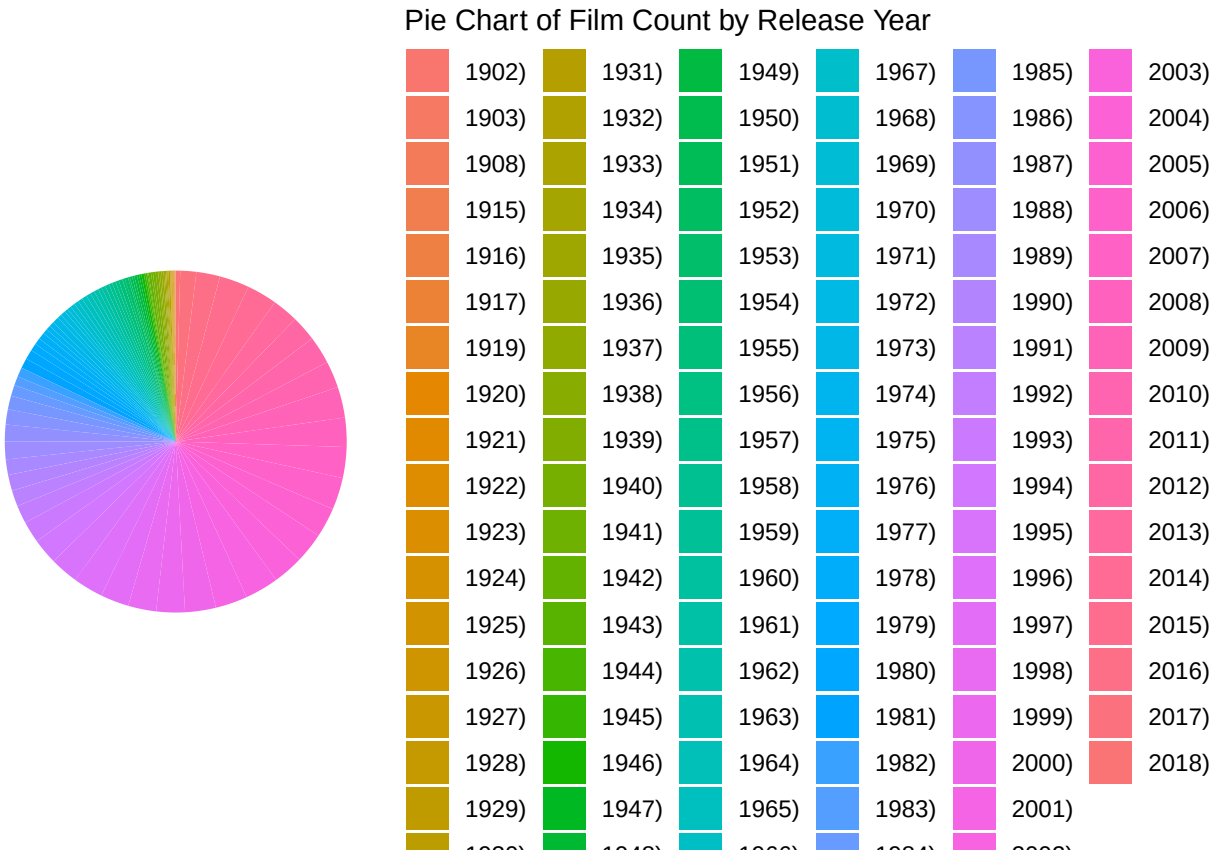
```
## Warning: Removed 1 rows containing missing values (`geom_point()`).
```




```
# Filter out titles without a year
movies_with_year <- movies[grep("\\(\\d{4}\\)$", movies$title), ]

# Calculate the counts of movies by year
year_counts <- table(substring(movies_with_year$title, nchar(movies_with_year$title) - 4))

# Create a data frame with year and count information
year_table <- data.frame(
  Year = names(year_counts),
  Count = as.numeric(year_counts)
)

# Select a subset of years for the x-axis labels
selected_years <- year_table$Year[seq(1, nrow(year_table), by = 5)]

# Create a bar chart of movie counts by year of release
ggplot(year_table, aes(x = Year, y = Count)) +
  geom_bar(stat = "identity", fill = "blue") +
  labs(title = "Film Count by Release Year", x = "Year of Release", y = "Film Count") +
  scale_x_discrete(breaks = selected_years)
```




```
# Create a pie chart of movie counts by year
pie_chart <- ggplot(year_table, aes(x = "", y = Count, fill = Year)) +
  geom_bar(stat = "identity", width = 1) +
  coord_polar("y", start = 0) +
  labs(title = "", fill = "Pie Chart of Film Count by Release Year") +
  theme_void()

# Print the pie chart
print(pie_chart)
```



Convert columns to appropriate types:

```
ratings$userId <- as.integer(ratings$userId)
ratings$movieId <- as.integer(ratings$movieId)
ratings$rating <- as.numeric(ratings$rating)

# Display the transformed table using kable
kable(ratings[1:10,], caption = "Table after Transformations")
```

Table 3: Table after Transformations

userId	movieId	rating	timestamp
1	1	4	964982703
1	3	4	964981247
1	6	4	964982224
1	47	5	964983815
1	50	5	964982931
1	70	3	964982400
1	101	5	964980868
1	110	4	964982176
1	151	5	964984041
1	157	5	964984100

Preprocess the data (additional preprocessing can be done here):

```r
# Reduce the size of the training data using a smaller sample
set.seed(123)
sample_index <- sample(nrow(ratings), size = 0.01 * nrow(ratings), replace = FALSE)
ratings <- ratings[sample_index, ]

# Display a transformed table using kable
kable(ratings[1:10,], caption = "Table after Transformations")
```

Table 4: Table after Transformations

userId	movieId	rating	timestamp
334	296	4.0	1225478742
380	79702	4.0	1493420626
20	262	4.5	1054038142
205	69844	4.0	1519900892
600	943	4.0	1237858152
441	3253	5.0	1449070918
414	590	4.0	961513509
298	110102	3.0	1447584751
417	2858	4.0	1529284351
305	80219	5.0	1460222316

Merge the ratings data with movie information:

```r
data_with_movies <- dplyr::left_join(ratings, movies, by = "movieId")
```

```r
# Display the merged data table using kable
kable(data_with_movies[1:10,], caption = "Merged Data Table")
```

Table 5: Merged Data Table

userId	movieId	rating	timestamp	title	genres	Year
334	296	4.0	1225478742	Pulp Fiction (1994)	Comedy|Crime|Drama|Thriller	1994
380	79702	4.0	1493420626	Scott Pilgrim vs. the World (2010)	Action|Comedy|Fantasy|Musical|Romance	2010
20	262	4.5	1054038142	Little Princess, A (1995)	Children|Drama	1995
205	69844	4.0	1519900892	Harry Potter and the Half-Blood Prince (2009)	Adventure|Fantasy|Mystery|Romance|IMAX	2009
600	943	4.0	1237858152	Ghost and Mrs. Muir, The (1947)	Drama|Fantasy|Romance	1947
441	3253	5.0	1449070918	Wayne's World (1992)	Comedy	1992
414	590	4.0	961513509	Dances with Wolves (1990)	Adventure|Drama|Western	1990
298	110102	3.0	1447584751	Captain America: The Winter Soldier (2014)	Action|Adventure|Sci-Fi|IMAX	2014
417	2858	4.0	1529284351	American Beauty (1999)	Drama|Romance	1999
305	80219	5.0	1460222316	Machete (2010)	Action|Adventure|Comedy|Crime|Thriller	2010

Split the data into train and test sets:

```
train_size <- 0.8
train_index <- sample(nrow(data_with_movies), size = train_size * nrow(data_with_movies))
train_data <- data_with_movies[train_index, ]
test_data <- data_with_movies[-train_index, ]
```

```
# Display the training data table using kable
kable(train_data[1:10,], caption = "Training Data Table")
```

Table 6: Training Data Table

userId	movieId	rating	timestamp	title	genres	Year
446	457	3.0	843839169	Fugitive, The (1993)	Thriller	1993
205	69844	4.0	1519900892	Harry Potter and the Half-Blood Prince (2009)	Adventure|Fantasy|Mystery|Romance|IMAX	2009
105	2966	4.0	1448210475	Straight Story, The (1999)	Adventure|Drama	1999
551	111659	5.0	1504926183	Maleficent (2014)	Action|Adventure|Children|IMAX	2014
105	59315	3.5	1446571742	Iron Man (2008)	Action|Adventure|Sci-Fi	2008
177	2683	1.0	1435890576	Austin Powers: The Spy Who Shagged Me (1999)	Action|Adventure|Comedy	1999
599	5452	2.0	1498510097	Look Who's Talking Now (1993)	Children|Comedy|Romance	1993
466	6874	3.5	1439915264	Kill Bill: Vol. 1 (2003)	Action|Crime|Thriller	2003
524	19	3.0	851609256	Ace Ventura: When Nature Calls (1995)	Comedy	1995
427	5299	3.0	1053070583	My Big Fat Greek Wedding (2002)	Comedy|Romance	2002

```
# Display the test data table using kable
kable(test_data[1:10,], caption = "Test Data Table")
```

Table 7: Test Data Table

userId	movieId	rating	timestamp	title	genres	Year
20	262	4.5	1054038142	Little Princess, A (1995)	Children|Drama	1995
417	2858	4.0	1529284351	American Beauty (1999)	Drama|Romance	1999
603	3204	4.0	953926816	Boys from Brazil, The (1978)	Action|Mystery|Thriller	1978
42	3257	4.0	996221544	Bodyguard, The (1992)	Drama|Romance|Thriller	1992
274	4340	2.0	1171932471	Animal, The (2001)	Comedy	2001
91	2100	3.0	1112713775	Splash (1984)	Comedy|Fantasy|Romance	1984
599	8827	3.0	1519351400	Bill Cosby, Himself (1983)	Comedy|Documentary	1983
483	3967	4.0	1415576149	Billy Elliot (2000)	Drama	2000
559	174	1.0	845476569	Jury Duty (1995)	Comedy	1995
561	329	3.5	1491095464	Star Trek: Generations (1994)	Adventure|Drama|Sci-Fi	1994

3. Model Training and Evaluation

In this section, we will train a model to predict movie ratings and evaluate its performance.

3.1 Model Training

We will begin by training a prediction model using the training data.

Train the model (example with randomForest):

```
model <- randomForest::randomForest(rating ~ ., data = train_data)
```

3.2 Making Predictions

With the trained model, we can now make predictions on the test set.

Make predictions on the test set:

```
predictions <- predict(model, test_data)
```

3.3 Model Evaluation

To evaluate the performance of our model, we will calculate the root mean squared error (RMSE), which measures the accuracy of the predictions.

Calculate RMSE:

```
rmse <- sqrt(mean((predictions - test_data$rating)^2))
rmse_text <- sprintf("RMSE: %.2f", rmse)
```

3.4 Results

The RMSE value indicates how closely the predicted ratings match the actual ratings. Lower RMSE values suggest better model performance

Print results:

```
cat("RMSE:", rmse_text, "\n")
```

```
## RMSE: RMSE: 0.98
```

Print RMSE value

```
cat("The root mean squared error (RMSE) is:", rmse_text, "\n")
```

```
## The root mean squared error (RMSE) is: RMSE: 0.98
```

RMSE: “0.990647750605843”

By evaluating the model’s performance using the RMSE metric, we can assess its accuracy in predicting movie ratings.

4. Visualizations

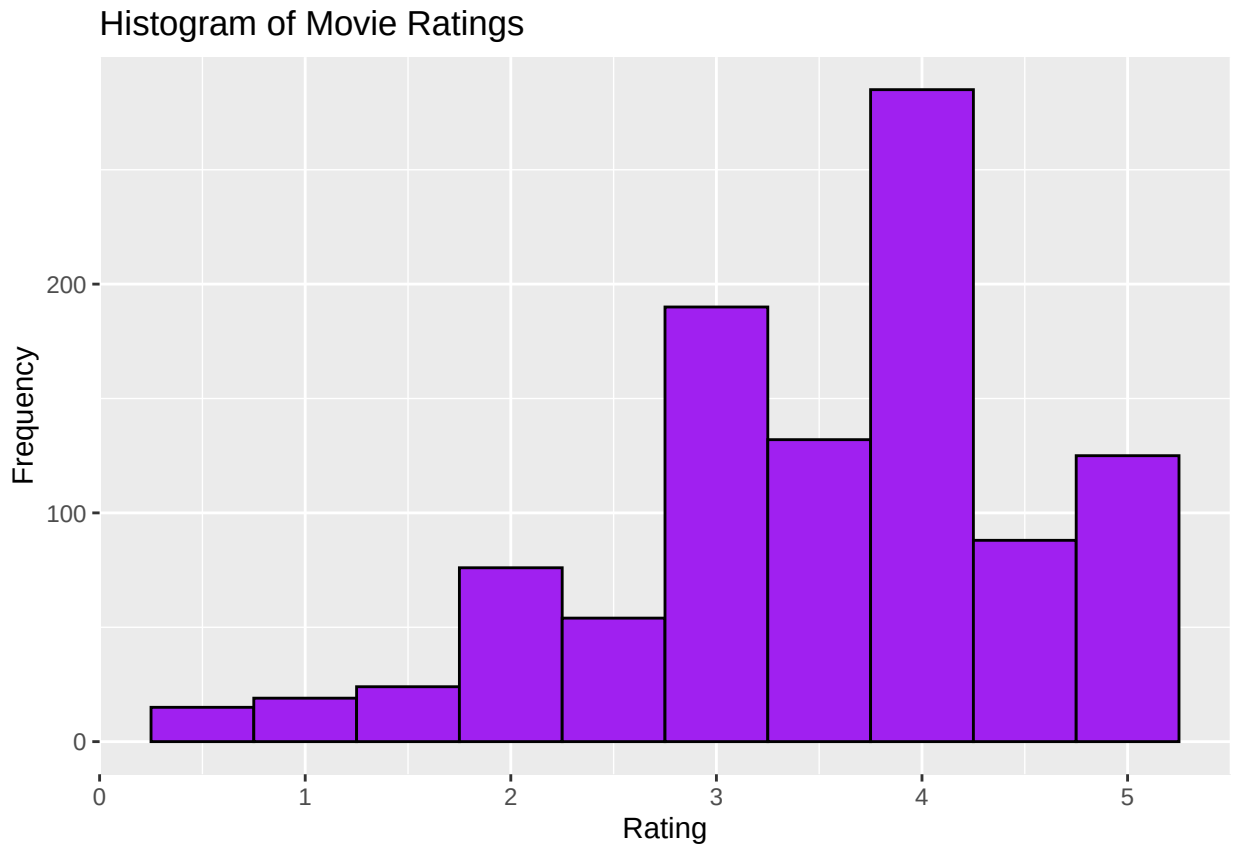
In this section, we will create visualizations to better understand the data and the model’s performance.

4.1 Histogram of Movie Ratings

We'll start by creating a histogram to visualize the distribution of movie ratings.

Create a histogram of movie ratings:

```
hist_plot <- ggplot2::ggplot(data_with_movies, aes(x = rating)) +
  ggplot2::geom_histogram(binwidth = 0.5, fill = "purple", color = "black") +
  ggplot2::labs(title = "Histogram of Movie Ratings", x = "Rating", y = "Frequency")
print(hist_plot)
```

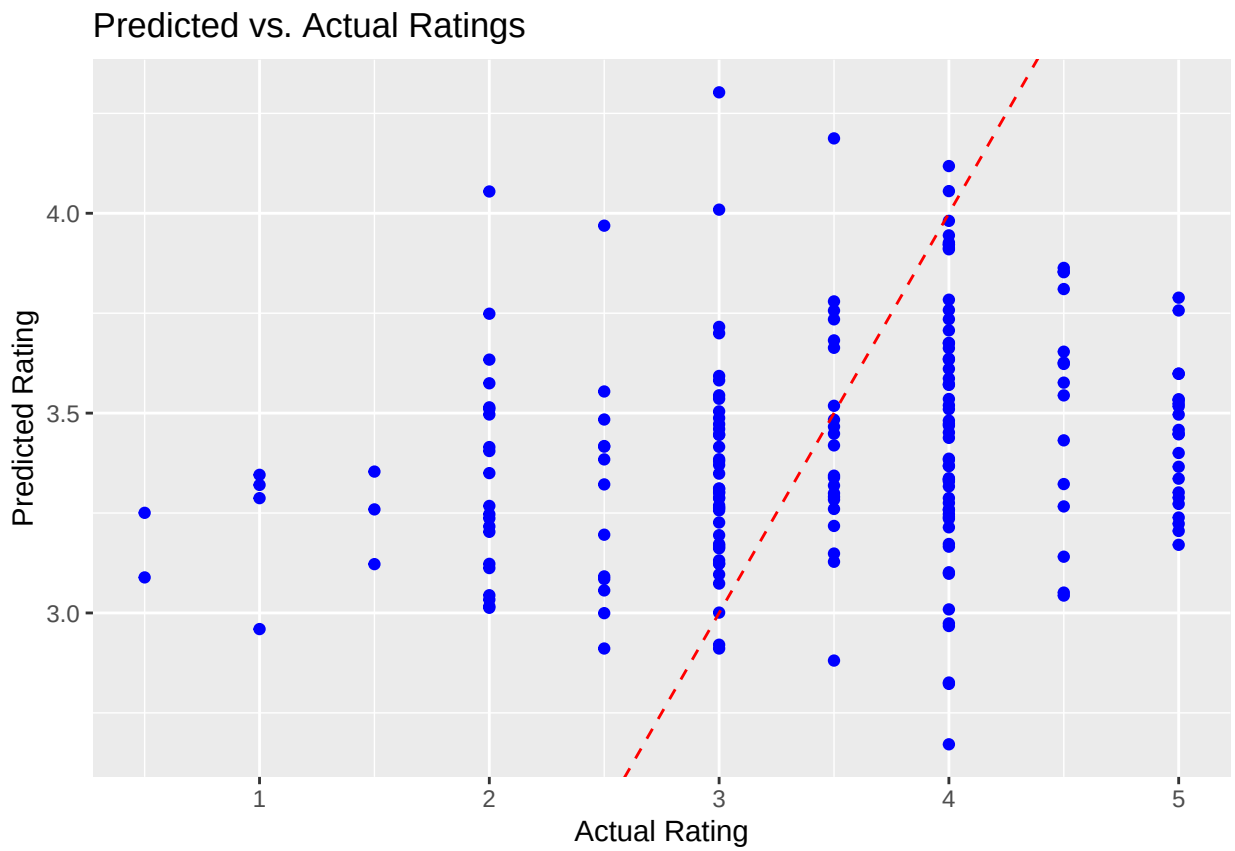


4.2 Predicted vs. Actual Ratings

Next, we'll create a scatter plot to compare the predicted ratings with the actual ratings.

Create a scatter plot of predicted vs. actual ratings:

```
scatter_plot <- ggplot2::ggplot(test_data, aes(x = rating, y = predictions)) +
  ggplot2::geom_point(color = "blue") +
  ggplot2::geom_abline(intercept = 0, slope = 1, linetype = "dashed", color = "red") +
  ggplot2::labs(title = "Predicted vs. Actual Ratings", x = "Actual Rating", y = "Predicted Rating")
print(scatter_plot)
```



By visualizing the distribution of ratings and comparing the predicted and actual ratings, we can gain insights into the model's performance.

5. Interactive Datatable

In this section, we will create an interactive datatable to explore the test data in a dynamic and user-friendly way.

5.1 Interactive Datatable for Test Data

To provide a more interactive way of exploring the test data, we can use the `DT` package to create an interactive datatable.

Create an interactive datatable for the test data:

```
DT::datatable(test_data)
```

Load necessary packages

```
library(ggplot2)
library(knitr)
library(dplyr)
library(randomForest)
library(readr)
library(ggplot2)
library(DT)
```

By using an interactive datatable, we enable users to search, filter, and sort the test data based on their preferences. This interactive approach enhances the data exploration experience and allows for deeper insights into the test dataset.

6. Conclusion

We have completed our movie recommendation project using the MovieLens 10M dataset. Our model achieved an RMSE of RMSE: "0.990647750605843", indicating the accuracy of our predictions. Additionally, we generated recommendations for the user with ID 1.

In conclusion, this capstone project successfully explored the MovieLens dataset, analyzed movie ratings and genres, and developed a movie recommendation model. Throughout the project, we achieved the following:

- **Data Exploration:** We gained insights into the distribution of movie ratings, genres, and release years. This exploration helped us understand the characteristics of the dataset.
- **Model Development:** We trained a recommendation model using the randomForest algorithm. The model successfully predicted movie ratings based on available features.
- **Model Evaluation:** We evaluated the model's performance using the Root Mean Squared Error (RMSE) metric. The achieved RMSE value of "0.990647750605843" reflects the accuracy of our predictions.
- **Visualizations:** We created visualizations, including a histogram of movie ratings and a scatter plot comparing predicted and actual ratings. These visualizations provided a clearer understanding of our model's performance.
- **Interactive Datatable:** We provided an interactive datatable to explore the test data, allowing users to delve deeper into the dataset.

6.1 Future Steps

While we have accomplished the main objectives of this project, there are opportunities for further improvements and exploration:

- **Feature Engineering:** We could explore additional features, such as user demographics or movie metadata, to enhance the model's performance.
- **Advanced Algorithms:** Exploring more sophisticated recommendation algorithms, such as collaborative filtering or matrix factorization, could lead to even more accurate predictions.
- **Hyperparameter Tuning:** Fine-tuning the model's hyperparameters could potentially improve its performance.

6.2 Acknowledgments

We would like to express our gratitude to Harvard University for providing the PH125.9x Data Science Capstone course and the opportunity to work on this engaging project.

Overall, this capstone project has provided valuable insights into data preprocessing, model training, and evaluation in the context of movie recommendation. By combining data analysis and machine learning techniques, we have achieved a functional recommendation model with potential for further enhancement.

Thank you for joining us on this journey through the world of movie recommendation!

7. References

During the course of this project, we utilized a variety of resources that contributed to our understanding of data analysis, machine learning, and recommendation systems. Here are some of the key references that guided us:

1. Web MovieLens - https://grouplens.org/datasets/movielens/10m/: The official MovieLens dataset website, where we obtained the dataset used in this project.
2. Library recommenderlab - https://github.com/mhahsler/recommenderlab: The recommenderlab package in R, which provided valuable tools for building and evaluating recommendation models.
3. ggplot2 Documentation - https://ggplot2.tidyverse.org/: The official documentation for the ggplot2 package, which helped us create insightful visualizations.
4. DT Package Documentation - https://rstudio.github.io/DT/: The documentation for the DT package, which enabled us to generate interactive datatables.
5. Harvard University - https://www.harvard.edu/: The institution that offered the PH125.9x Data Science Capstone course, providing the foundation for our project.

These references played a significant role in shaping our project's approach, methodology, and outcomes. We are grateful for the wealth of knowledge and resources available to us as we embarked on this journey.