MovieLens Project

Adolfo Muñoz Macho @dradolfomunoz

2024-06-03

Introduction

This study provides a comprehensive examination of the Movie Rating Prediction project, utilising sophisticated Machine Learning algorithms. The main goal of the research is to forecast movie ratings by analysing user behaviour and specific movie attributes. This report offers a comprehensive summary of the dataset utilised and methodically discusses the goals and essential methodological actions made during the study.

Project Overview: MovieLens

It was created a movie recommendation engine for this project, utilising the MovieLens dataset. The dslabs package's version of MovieLens is a limited portion of the complete dataset, which comprises millions of ratings. In order to simplify the calculation process, it was employed the 10M version of the MovieLens dataset. This project necessitated the use of the numerous tools and concepts that were covered in the previous courses in this series.

Methods and Analysis

The relevant datasets were produced using the given code after obtaining the MovieLens data. The machine learning method was trained using the edx set and then tested using the final_holdout_test set by predicting movie ratings, treating them as unknowns. The Root Mean Squared Error (RMSE) was used to assess the precision of my predictions compared to the actual ratings in the final_holdout_test set.

The analysis approach encompassed data cleansing, exploration, and modelling methodologies. The missing values in the dataset were addressed by imputing them with the mean value of the corresponding columns. The dataset was partitioned into several training and testing sets to enable the construction of the model. The modelling technique involved developing a baseline model to forecast average movie ratings, as well as an advanced model that integrated regularised movie and user impacts.

Code provided by the course:

```
############################################################
# Create edx and final_holdout_test sets
############################################################

# Note: this process could take a couple of minutes

if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
```

```r
library(tidyverse)
library(caret)

# MovieLens 10M dataset:
# https://grouplens.org/datasets/movielens/10m/
# http://files.grouplens.org/datasets/movielens/ml-10m.zip

options(timeout = 120)

dl <- "ml-10M100K.zip"
if(!file.exists(dl))
  download.file("https://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

ratings_file <- "ml-10M100K/ratings.dat"
if(!file.exists(ratings_file))
  unzip(dl, ratings_file)

movies_file <- "ml-10M100K/movies.dat"
if(!file.exists(movies_file))
  unzip(dl, movies_file)

ratings <- as.data.frame(str_split(read_lines(ratings_file), fixed("::"), simplify = TRUE),
                         stringsAsFactors = FALSE)
colnames(ratings) <- c("userId", "movieId", "rating", "timestamp")
ratings <- ratings %>%
  mutate(userId = as.integer(userId),
         movieId = as.integer(movieId),
         rating = as.numeric(rating),
         timestamp = as.integer(timestamp))

movies <- as.data.frame(str_split(read_lines(movies_file), fixed("::"), simplify = TRUE),
                        stringsAsFactors = FALSE)
colnames(movies) <- c("movieId", "title", "genres")
movies <- movies %>%
  mutate(movieId = as.integer(movieId))

movielens <- left_join(ratings, movies, by = "movieId")

# Final hold-out test set will be 10% of MovieLens data
set.seed(1, sample.kind="Rounding") # if using R 3.6 or later
# set.seed(1) # if using R 3.5 or earlier
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
edx <- movielens[-test_index,]
temp <- movielens[test_index,]

# Make sure userId and movieId in final hold-out test set are also in edx set
final_holdout_test <- temp %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")

# Add rows removed from final hold-out test set back into edx set
removed <- anti_join(temp, final_holdout_test)
edx <- rbind(edx, removed)
```

```
rm(dl, ratings, movies, test_index, temp, movielens, removed)
```

Data handling and Preprocessing

Refers to the process of managing and manipulating data in a systematic manner. Imputation: The missing values in the dataset were imputed by replacing them with the mean value of their respective columns. This phase guarantees that the data is coherent and prepared for subsequent analysis.

Splitting of Data

The dataset was partitioned into separate training and testing sets to ease the creation and evaluation of the model. This procedure of splitting guarantees that the model is trained on a subset of the data and evaluated on another subset in order to validate its performance.

Exploratory Data Analysis

Baseline Model: A preliminary model was developed to forecast the mean rating of movies. This model functions as a basic standard for assessing the effectiveness of more intricate models.

Advanced Model Development

The sophisticated model employed regularisation techniques to account for the effects of movies and users, utilising a parameter lambda. This methodology aids in mitigating overfitting and enhancing the resilience of the model.

Cross-Validation:

Sectional cross-validation was employed to select the ideal lambda value, aiming to enhance model generalisation and mitigate model bias. This approach divides the data into separate parts for training and validation, enabling a more accurate assessment of the model's performance.

Package Installation

The "magrittr" package: The installation of the "magrittr" package was essential in order to utilise the %>% operator for data processing. This operator enhances code simplicity and enhances legibility by linking operations in a sequence.

The "data.table" package was installed to utilise its efficient data processing capabilities. This package offers effective tools for rapid data manipulation, enhancing the efficiency of data processing activities.

Data normalisation and missing value check

Data normalisation is a process that maintains consistency and prevents bias towards specific features in a model. By imputing missing values with the mean of the columns, the dataset is made ready for subsequent analysis.

Dealing with Missing Values

It is crucial to perform a check for any remaining missing values in the data to guarantee that the data is complete and accurate. Through the process of examining and resolving any instances of missing data, the accuracy and dependability of the model's performance and predictions are

Developed Code

```
############################################################
# Movie Rating Prediction using Machine Learning
############################################################

# Install necessary package data.table if not already installed
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
if(!require(magrittr)) install.packages("magrittr", repos = "http://cran.us.r-project.org")

# Load the required library
library(data.table)
library(magrittr)

# Handle missing values by replacing them with the mean of the columns
edx <- edx %>%
  mutate(rating = ifelse(is.na(rating), mean(rating, na.rm = TRUE), rating),
         userId = ifelse(is.na(userId), mean(userId, na.rm = TRUE), userId),
         movieId = ifelse(is.na(movieId), mean(movieId, na.rm = TRUE), movieId))

# Check for remaining missing values
if(any(is.na(edx))) {
  cat("There are remaining missing values in the data. Please investigate.\n")
} else {
  cat("No missing values found after normalization.\n")
}

# Further split the edx dataset into training and testing sets for model development
set.seed(1, sample.kind = "Rounding")
train_index <- createDataPartition(y = edx$rating, times = 1, p = 0.9, list = FALSE)
train_set <- edx[train_index,]
test_set <- edx[-train_index,]

# Ensure userId and movieId in the test set are also in the train set
test_set <- test_set %>%
  semi_join(train_set, by = "movieId") %>%
  semi_join(train_set, by = "userId")

# Baseline Model: Predict the average movie rating
mu_hat <- mean(train_set$rating)

# Compute RMSE for the baseline model
rmse_baseline <- sqrt(mean((test_set$rating - mu_hat)^2))
cat("Baseline RMSE:", rmse_baseline, "\n")

# Advanced Model: Regularized Movie and User Effects Model
```

```r
# 1. Calculate overall average rating
mu <- mean(train_set$rating)

# 2. Regularization parameter (lambda) - determine through cross-validation
lambdas <- seq(0, 10, 0.25)
calculate_rmse <- function(lambda) {
  # Compute movie effects
  b_i <- train_set %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu) / (n() + lambda))

  # Compute user effects
  b_u <- train_set %>%
    left_join(b_i, by = "movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - mu - b_i) / (n() + lambda))

  # Predict ratings based on movie and user effects
  predicted_ratings <- test_set %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(pred = mu + b_i + b_u) %>%
    pull(pred)

  # Calculate RMSE
  return(sqrt(mean((test_set$rating - predicted_ratings)^2)))
}

# Find optimal lambda
rmse_results <- sapply(lambdas, calculate_rmse)
best_lambda <- lambdas[which.min(rmse_results)]
best_rmse <- min(rmse_results)

cat("Optimal lambda:", best_lambda, "\n")
cat("Best RMSE (Cross-Validation):", best_rmse, "\n")

# Final Model: Train on the entire edx dataset and evaluate on final_holdout_test

# Compute movie and user effects using the optimal lambda
b_i_final <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu) / (n() + best_lambda))

b_u_final <- edx %>%
  left_join(b_i_final, by = "movieId") %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - mu - b_i) / (n() + best_lambda))

# Predict ratings for final_holdout_test
predicted_final <- final_holdout_test %>%
  left_join(b_i_final, by = "movieId") %>%
  left_join(b_u_final, by = "userId") %>%
  mutate(pred = mu + b_i + b_u) %>%
```

```
  pull(pred)

# Compute RMSE on the final_holdout_test set
rmse_final <- sqrt(mean((final_holdout_test$rating - predicted_final)^2))
cat("Final RMSE on final hold-out test set:", rmse_final, "\n")
```

Results

The initial model calculated the Root Mean Squared Error (RMSE) as a measure of model performance, serving as a reference point for comparison. A sophisticated model was created, and the regularisation parameters were established via cross-validation. The ideal lambda value was determined, and the model's root mean square error (RMSE) was computed. The ultimate model was trained using the complete dataset and assessed on a separate test set, yielding a final Root Mean Square Error (RMSE) number.

Model Comparison and Final Results

After training and evaluating different models for movie rating prediction, the following results were obtained:

Baseline Model Performance Baseline RMSE: **1.059802**

Advanced Model with Regularized Movie and User Effects Optimal lambda: *4.5* Best RMSE (Cross-Validation): **0.8644841**

Final Model Results: Final RMSE on the final hold-out test set: **0.8648242**

Discussion on Model Analysis and Interpretation

The initial model, which forecasted the mean movie rating, exhibited a root mean square error (RMSE) of 1.059802. This model functioned as a basic standard for evaluating the effectiveness of more intricate models.

The advanced model, which includes regularised movie and user effects with an optimal lambda value of 4.5, obtained a lower root mean square error (RMSE) of 0.8644841 through cross-validation. This model exhibited enhanced predicted precision in comparison to the baseline model.

After training the final model on the complete training dataset with the ideal lambda value, the root mean square error (RMSE) on the final hold-out test set was 0.8648242. This suggests that the predictions made by the final model closely matched the actual ratings in the test set.

Conclusion and Model Optimization

The results indicate that the advanced model, which incorporates regularisation, achieved superior predictive accuracy compared to the baseline model. The ideal lambda value improved the model's performance by mitigating overfitting and enhancing generalisation. To enhance the model, one could consider investigating supplementary characteristics or adjusting hyperparameters to attain superior predicted precision and performance.

Through the comparison of several models and the evaluation of their Root Mean Square Error (RMSE) values, we can make well-informed conclusions regarding the efficacy of our predictive models. This allows us to continuously iterate and enhance them, resulting in more accurate forecasts for movie ratings.

To conclude, the paper provides a concise overview of the project's findings and acknowledges its limitations. The investigation demonstrated the efficacy of the machine learning models in forecasting movie ratings. Subsequent research could prioritise augmenting the model's efficacy and investigating supplementary characteristics to enhance predictions. The report highlights the significance of model evaluation and ongoing refinement in order to achieve precise forecasts for movie ratings.

Limitations

An obstacle faced with this project is the difficulty in formatting the PDF output while knitting. Despite diligent attempts to address the formatting issues, specific mistakes remained unresolved, prompting the decision to display the text in this organised way. Potential enhancements could prioritise the resolution of these formatting discrepancies in order to optimise the visual appeal of the ultimate report.

Another issue encountered was the requirement to include the installation of the "data.tables" package in the original code for appropriate functioning of the R file. Failure to do so resulted in an error during compilation.

Adherence to Academic Integrity

Adhering to the edX Honour Code, I guarantee that all submitted work is authentic and created alone by me.

Exclusively, I utilised Chat GPT-4 for this research to enhance my English proficiency, given that I am not a native speaker. In certain highly particular cases, Chat GPT-4 had a role in resolving seemingly impossible problems that arose during the standard procedures used in the process. By following these principles, my objective was to create a resilient and precise movie recommendation system, attaining a low Root Mean Square Error (RMSE) and enhancing my comprehension and use of Machine Learning.