Data 612

# Final Project

## Presentation

Zhi Ying Chen
Sin Ying Wong
Fan Xu

# Data 612 Final Project

## Introduction

In this project, we are going to implement a recommender system using different algorithms for movie recommendations by using the ***MovieLens*** dataset, which can be found at [https://grouplens.org/datasets/movielens/latest/] or [http://grouplens.org/datasets/].

We will implement User-Based Collaborative Filtering (**UBCF**) model, Item-Based Collaborative Filtering (**IBCF**) model, singular value decomposition (**SVD**) model, alternating least square (**ALS**) model, and **Spark ALS** model to our datasets and compare their performance.

# Data 612 Final Project

## Note

To develop an efficient program of this project in PC environment but yet to effectively demonstrate building recommender systems using R studio, **we will be covering two MovieLens datasets**.

In the first section, a smaller MovieLens dataset will be used when building recommender systems using **`Recommenderlab`**.

In the second section, a larger MovieLens dataset with 27M+ ratings that is shrinked to around **12,000 users and 12,000 movies** will be used when building a recommender system **using `sparklyr`**.

# Data 612 Final Project

## Project Flow

**IMPORT DATA & EXPLORATION**

Load the MovieLens datasets to our system and create our sample datasets.

**BUILD MODELS w/ RECOMMENDERLAB**

Build various models with recommenderlab.

**COMPARE PERFORMANCES**

Compare model performances with ROC Curves and RMSE metrics.

**BUILD MODEL w/ SPARKLYR**

Build ALS model with sparklyr and calculate its performance.

**BUILD A UI w/ SHINY**

Use shiny app to design a simple user interface.

# Data 612 Final Project

## Data Exploration

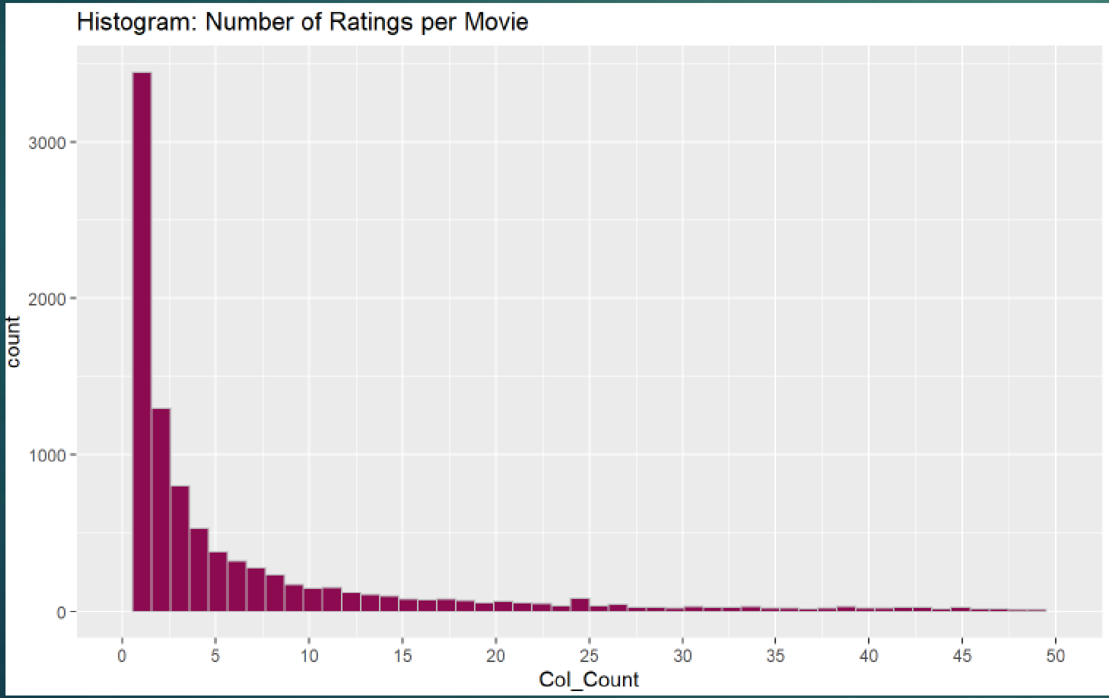
- The `ratings` dataset is very sparse.

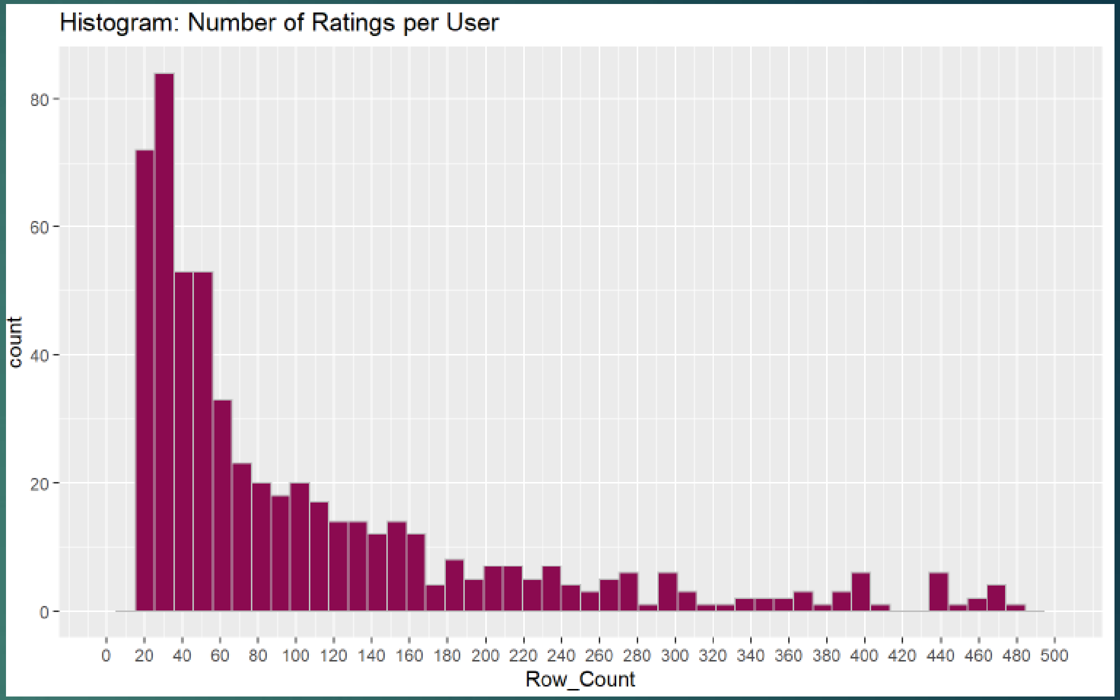

*heatmap of partial dataset*

# Data 612 Final Project

## Data Exploration

- Most movies are rated by no more than 5 users.




- Users rated at least 20 movies.

# Data 612 Final Project

## Data Exploration

The `movie` dataset contains
- movie IDs,
- movie titles, and
- genres.

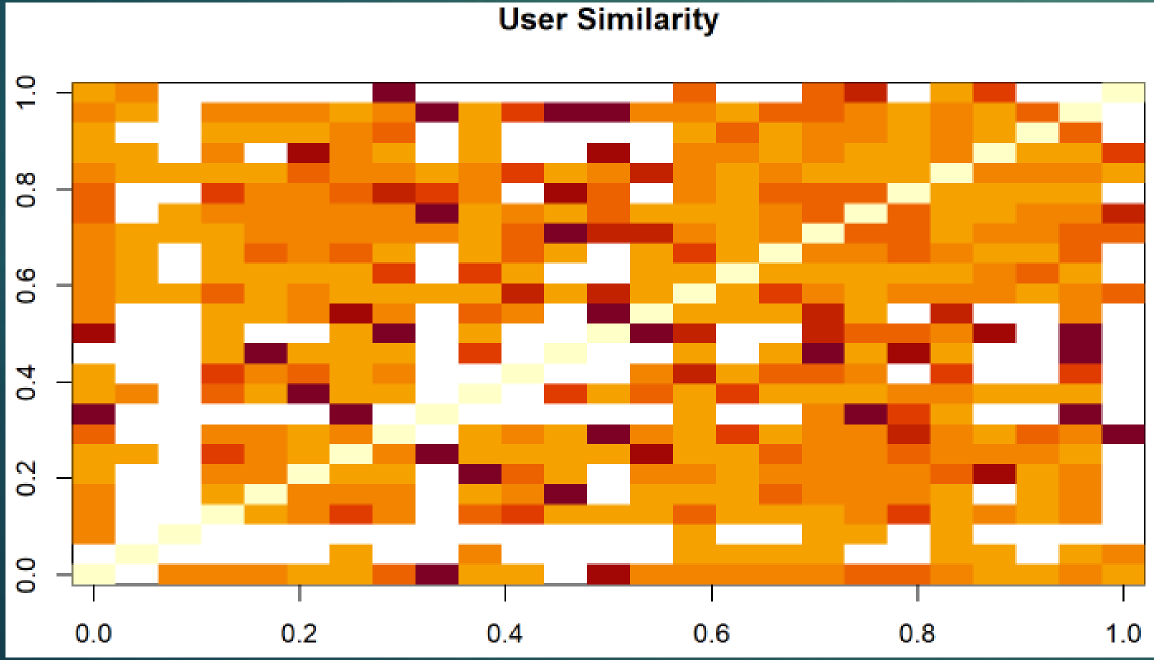
| | movieId <int> | title <fctr> | genres <fctr> |
|---|---|---|---|
| 1 | 1 | Toy Story (1995) | Adventure\|Animation\|Children\|Comedy\|Fantasy |
| 2 | 2 | Jumanji (1995) | Adventure\|Children\|Fantasy |
| 3 | 3 | Grumpier Old Men (1995) | Comedy\|Romance |
| 4 | 4 | Waiting to Exhale (1995) | Comedy\|Drama\|Romance |
| 5 | 5 | Father of the Bride Part II (1995) | Comedy |
| 6 | 6 | Heat (1995) | Action\|Crime\|Thriller |

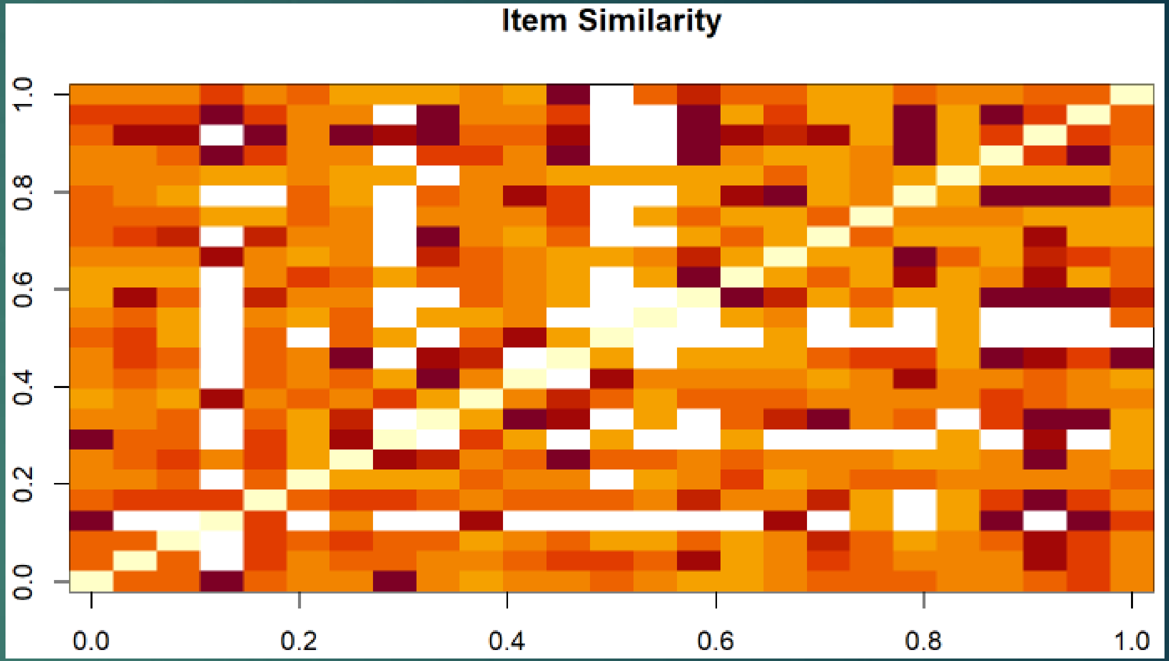
6 rows

# Data 612 Final Project

## Data Exploration

- User Similarity

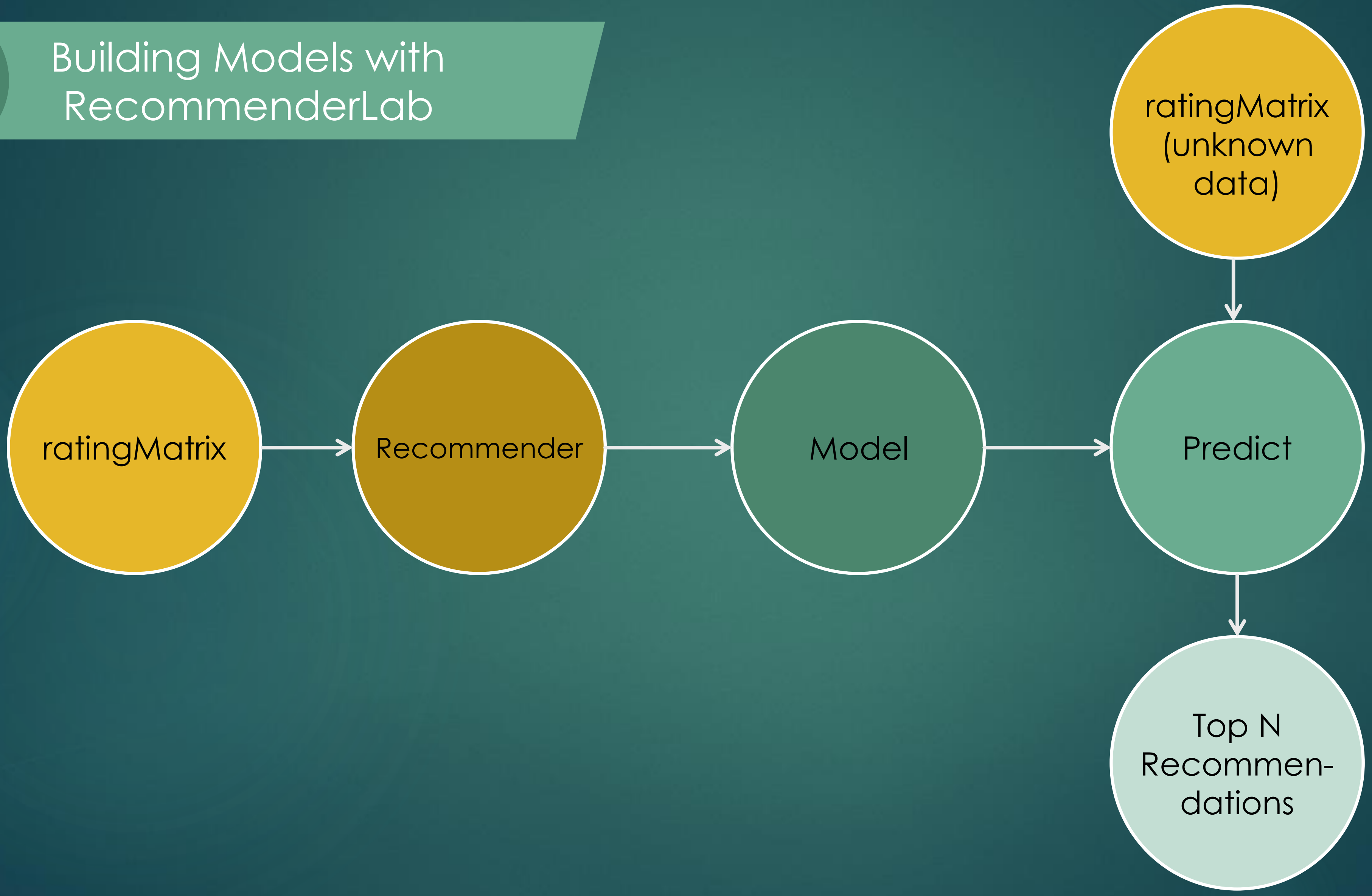


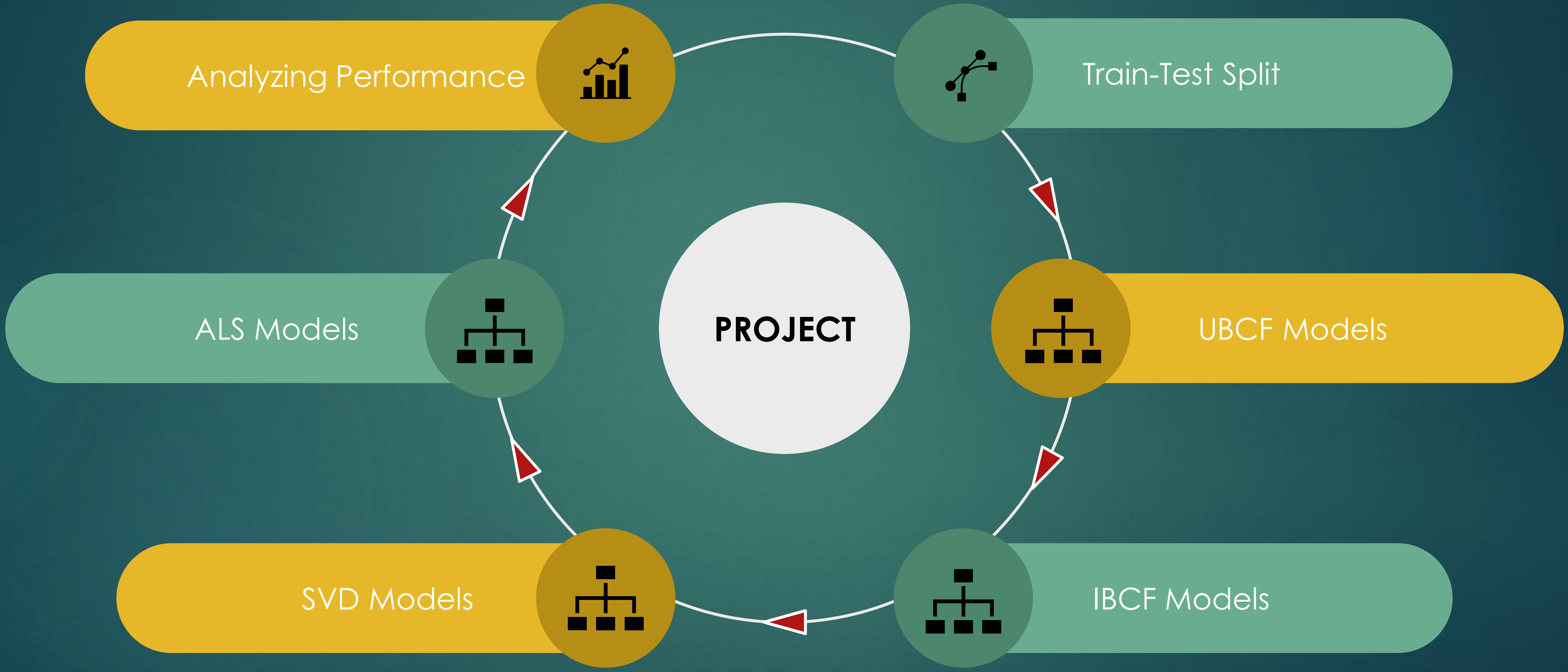

- Item Similarity

# Data 612 Final Project

## Building Models with RecommenderLab



# Data 612 Final Project

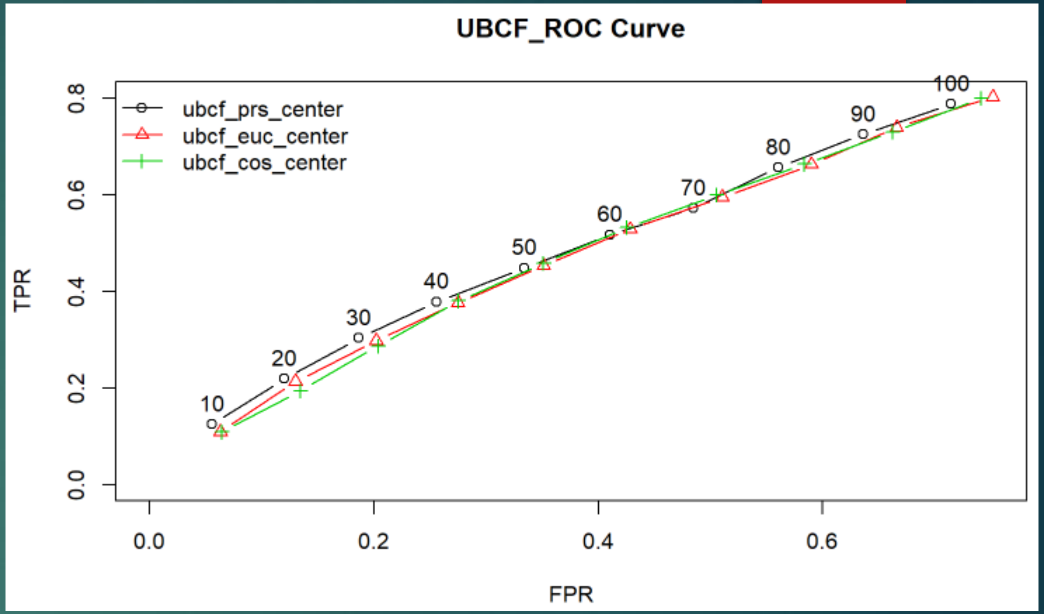


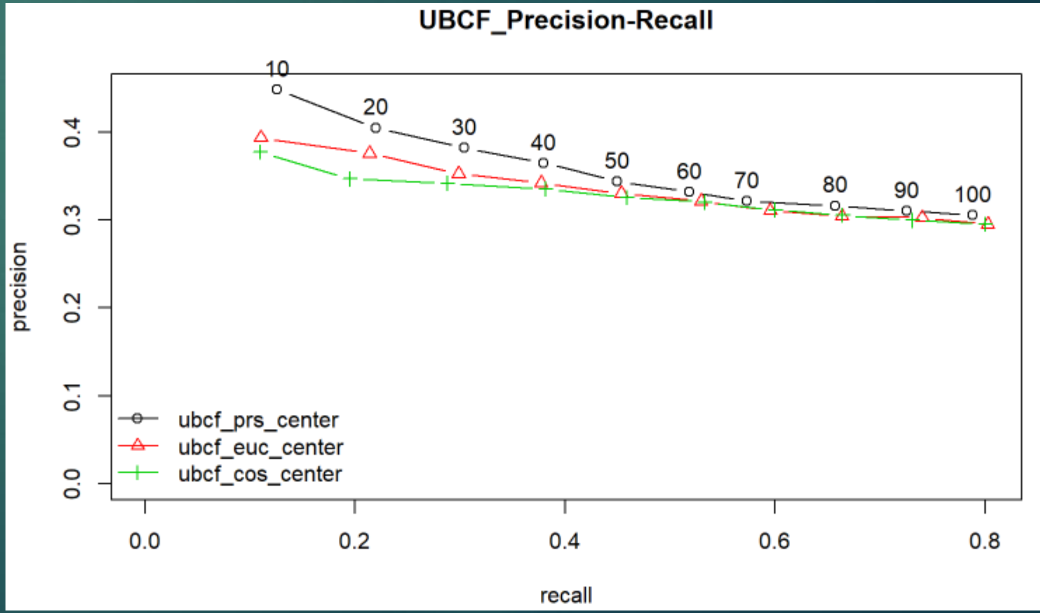
# Data 612 Final Project

## Building Models with RecommenderLab

### User-Based Collaborative Filtering (UBCF)

User-Based Collaborative Filtering (UBCF) assumes that users with similar preferences will rate items similarly. Thus missing ratings for a user can be predicted by first finding a Movie of similar users and then aggregate the ratings of these users to form a prediction.



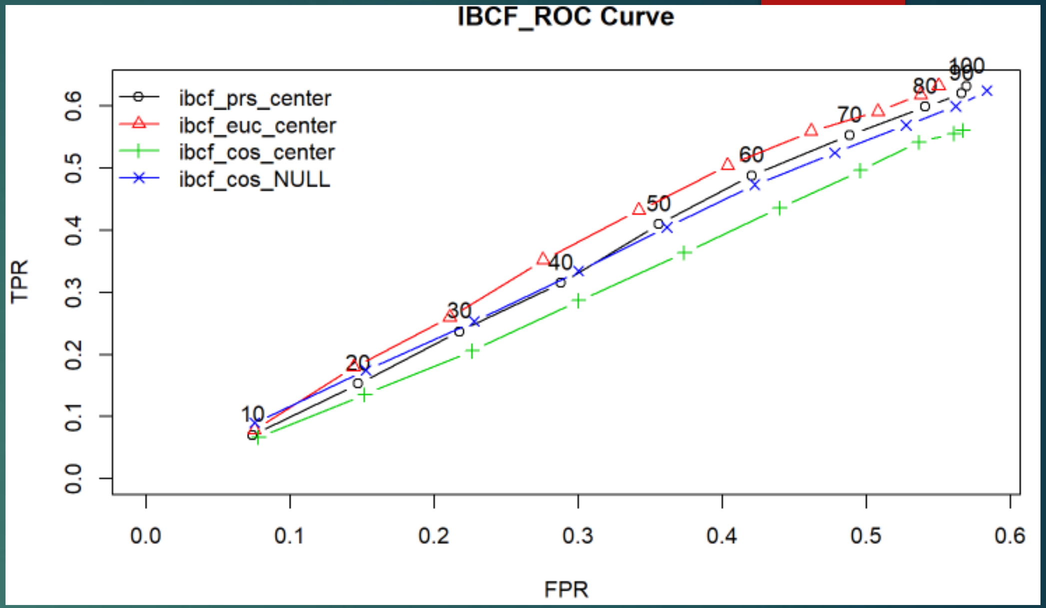
# Data 612 Final Project

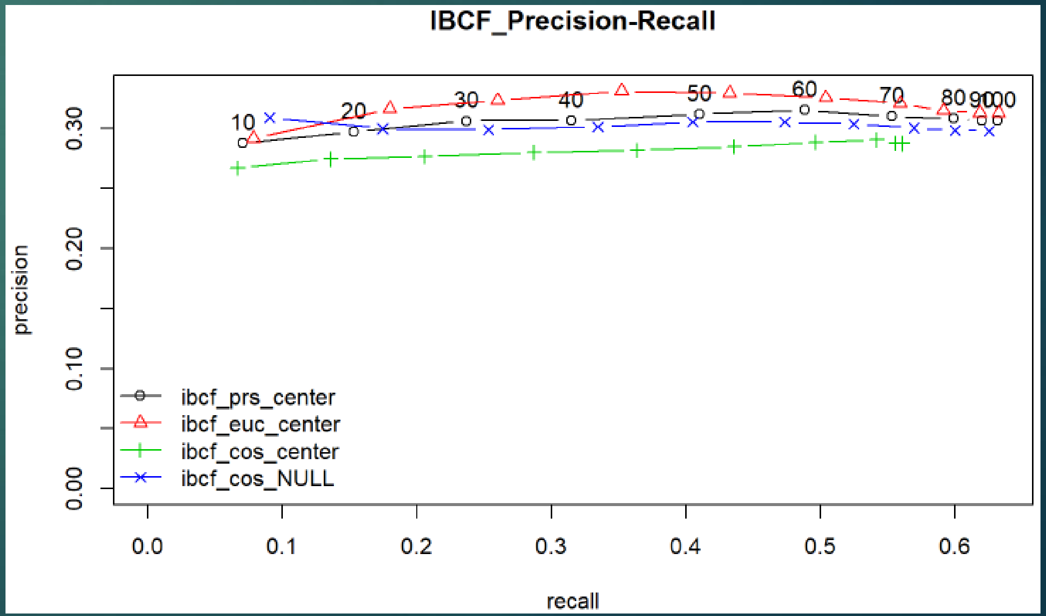
## Building Models with RecommenderLab

### Item-Based Collaborative Filtering (IBCF)

Item-Based Collaborative Filtering (IBCF) is a model-based approach which produces recommendations based on the relationship between items inferred from the rating matrix.

The assumption behind this approach is that users will prefer items that are similar to other items they like. The model-building step consists of calculating a similarity matrix containing all item-to-item similarities using a given similarity measure. Popular measures are Pearson correlation and Cosine similarity, which we have applied to our models.



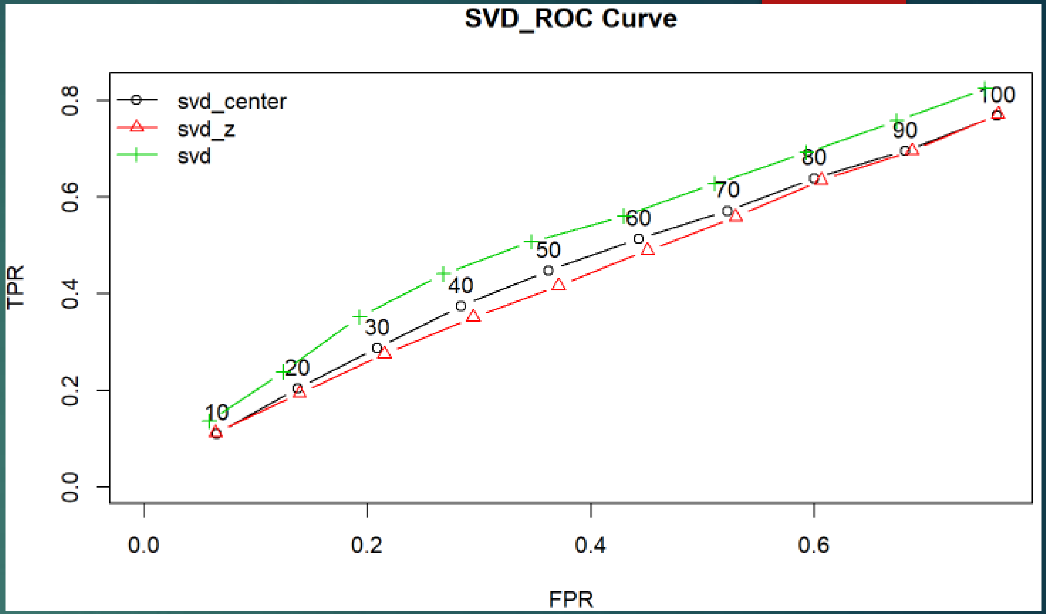



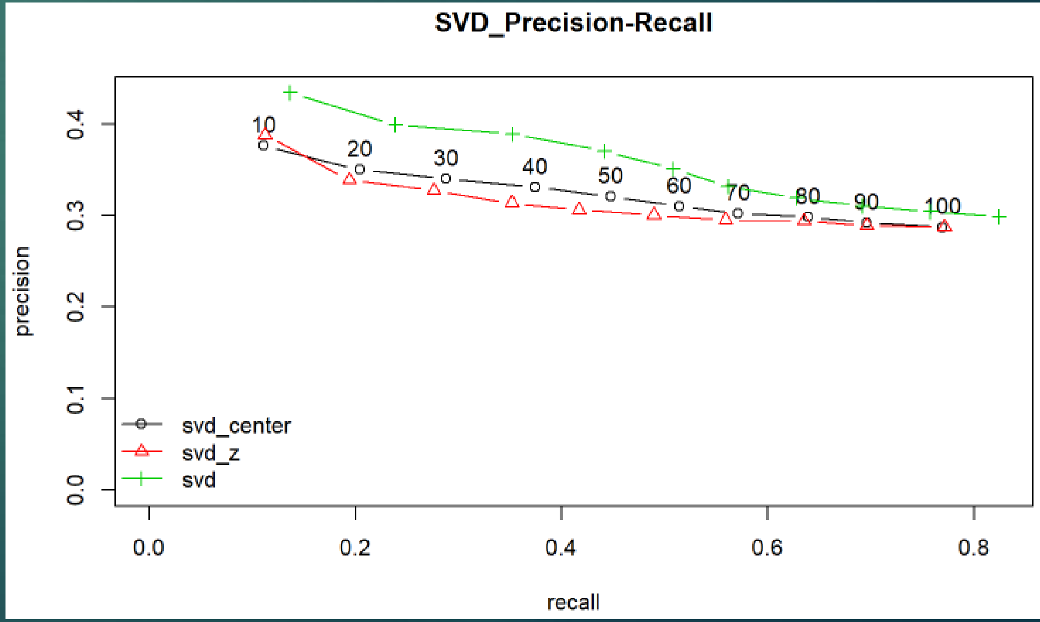
# Data 612 Final Project

## Building Models with RecommenderLab

### Singular Value Decomposition (SVD)

Singular Value Decomposition (SVD) is a matrix factorization technique, which reduces the number of features of a dataset by reducing the space dimension from N-dimension to K-dimension (where K<N). In the context of the recommender system, the SVD is used as a collaborative filtering technique.



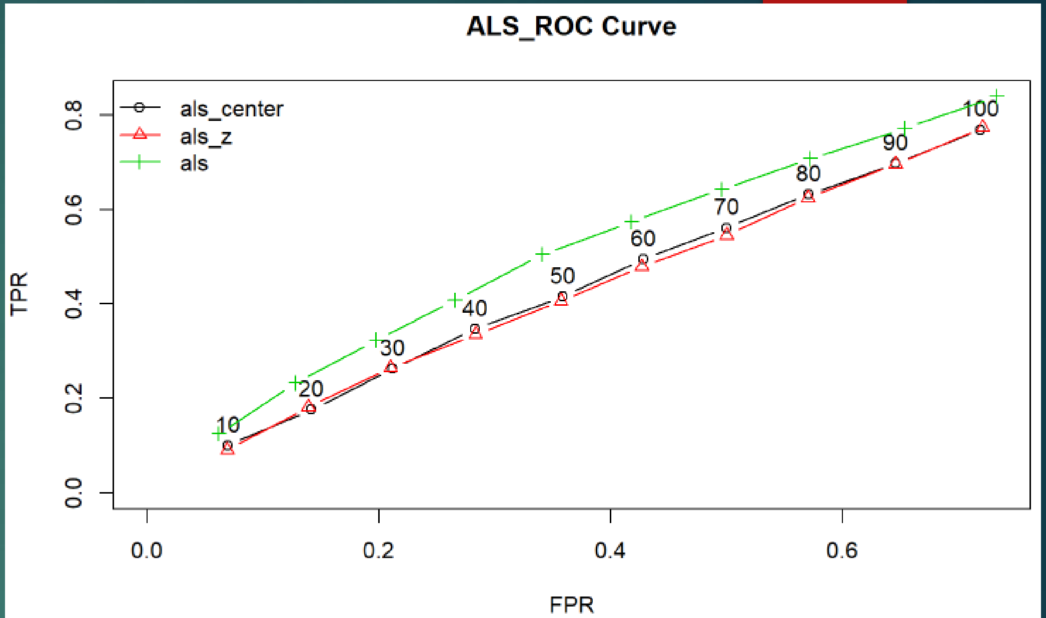



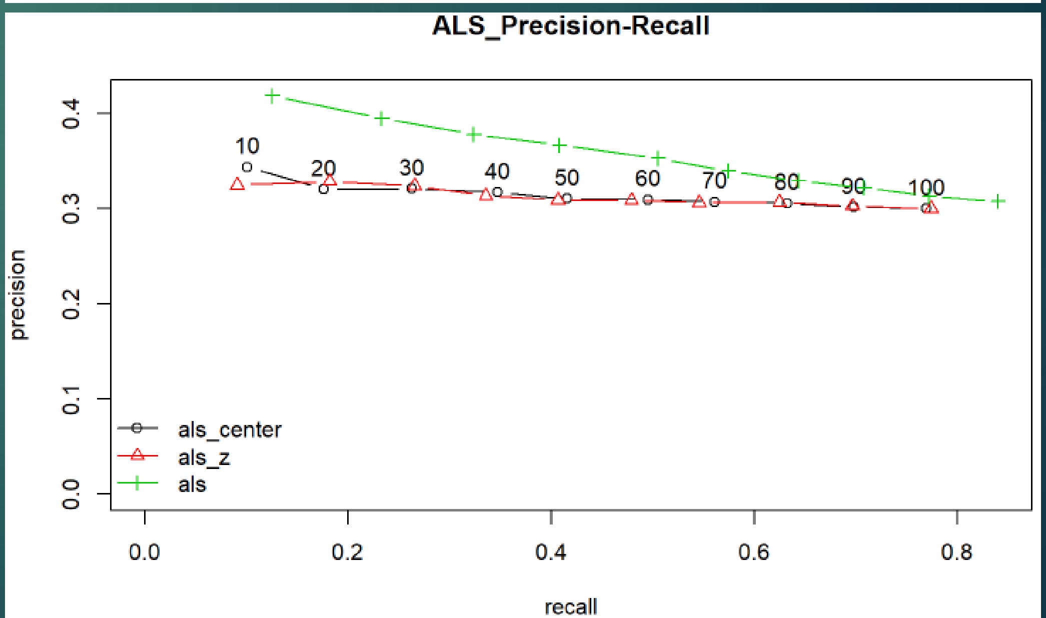
# Data 612 Final Project

## Building Models with RecommenderLab

### Alternating Least Squares(ALS)

ALS recommender is a matrix factorization algorithm that uses Alternating Least Squares with Weighted-Lamda-Regularization (ALS-WR). It factors the user to item matrix A into the user-to-feature matrix U and the item-to-feature matrix M.



## Analyzing Performance

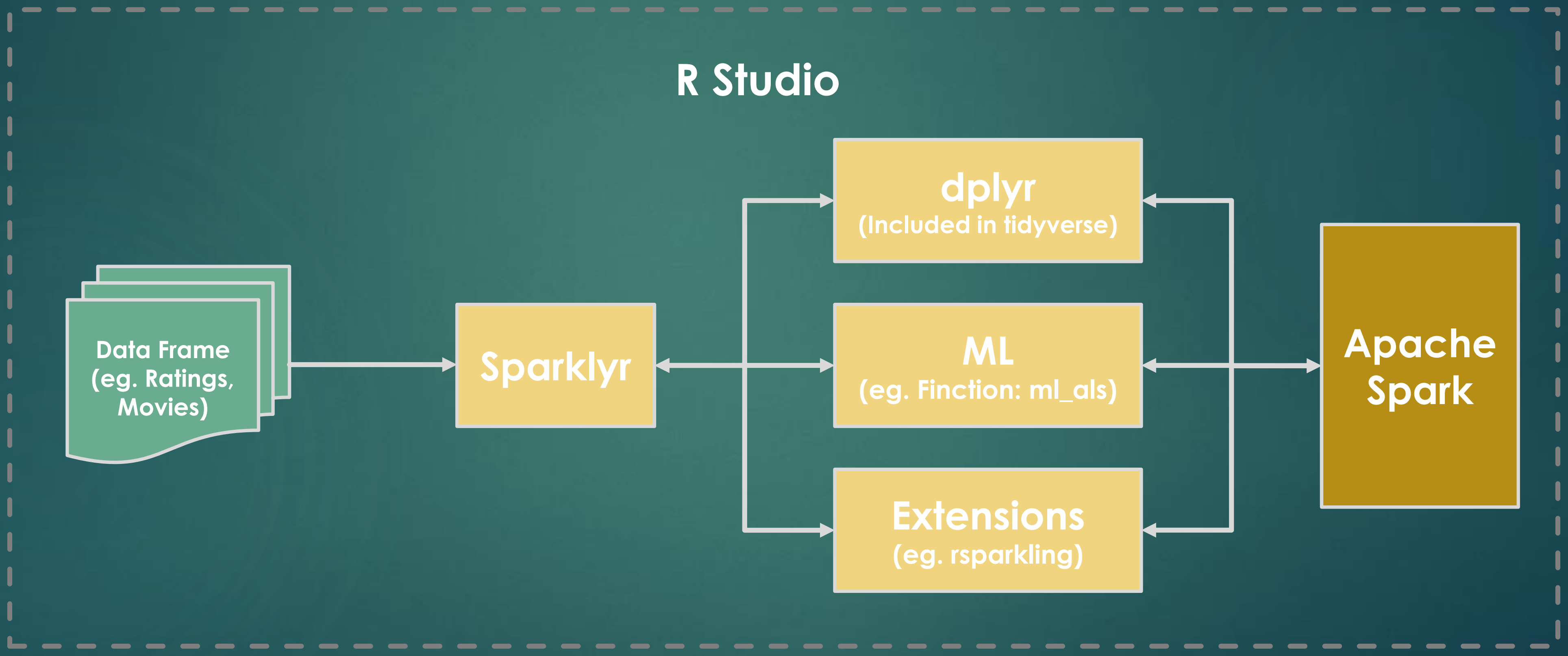
### Accuracy Metrics

Among each algorithm, we have
UBCF with Pearson correlation,
IBCF with Euclidean distance,
non-normalized SVD, and
non-normalized ALS as the best model.

By comparing the accuracy of these four models, we have the non-normalized ALS model as the best model as it has the lowest RMSE value.

| Metrics Comparison | | | |
|---|---|---|---|
| Model | RMSE | MSE | MAE |
| ALS_error | 0.8946856 | 0.8004623 | 0.6858065 |
| SVD_error | 0.9308717 | 0.8665221 | 0.7042577 |
| IBCF_error | 1.0076847 | 1.0154284 | 0.7302223 |
| UBCF_error | 1.0593320 | 1.1221843 | 0.7790488 |

## Building Recommender System in Spark



## Building Recommender System in Spark

### ***Load a Large Dataset***

To meet the requirement of project but also make the it executable in PC, in the following part of this project, a `ratings` dataset that is shrinked to around 12,000+ users and 12,000+ movies with over 1 million ratings is loaded into R.

```
```{r dimension}
ratings %>%
  select(user, item, rating) %>%
  spread(key=item, value = rating) %>%
  select(-user) %>%
  as.matrix() %>%
  as('realRatingMatrix')
```

12924 x 12057 rating matrix of class 'realRatingMatrix' with 1151103 ratings.
```

# Building Recommender System in Spark

## *Create Local Spark Connection*

- spark_config()
- spark_ connect(master, config)

Config Spark local server. Set 50% of our system(PC)'s accessible memory to Spark.

```{r set spark config}
conf <- spark_config()
conf$spark.memory.fraction <- 0.5

sc <- spark_connect(master = 'local', config = conf)
```

## *Copy Data to Spark*

- sdf_copy_to()

```{r load data movielense}
sdf_movies <- sdf_copy_to(sc, movies, 'movies', overwrite = TRUE)
sdf_ratings <- sdf_copy_to(sc, ratings, 'ratings', overwrite = TRUE)

```

# Data 612 Final Project

## Building Recommender System in Spark

### Train-Test-Split

- sdf_random_split()

```
```{r train test split 2}
movie_split <- sdf_ratings %>%
  sdf_random_split(training = 0.8, testing = 0.2)

movie_train <- movie_split$training
movie_test <- movie_split$testing
```
```

### Train Model

- ml_als()

```
```{r train model in spark}
model_formula = rating ~ user + item
rec_als_spark <- ml_als(movie_train, model_formula, max_iter = 5)
```
```

### Make Prediction

- ml_predict()

```
```{r predict spark}
predict_spark <- ml_predict(rec_als_spark, movie_test)

```
```

# Data 612 Final Project

## Building Recommender System in Spark

### ***Make Top N Recommendation***

- ml_recommend()

```{r}
ml_recommend(rec_als_spark, type = 'item', 5) %>%
  left_join(sdf_movies, by = 'item') %>%
  select(user, item, title) %>%
  arrange(user)


```

| user <int> | item <int> | title <chr> |
|---|---|---|
| 1 | 790 | Unforgettable Summer, An (Un ÃtÃ inoubliable) (1994) |
| 1 | 27093 | Class Trip, The (La classe de neige) (1998) |
| 1 | 39244 | Leila (1996) |
| 1 | 5395 | Gambler, The (1974) |
| 1 | 26049 | Fires on the Plain (Nobi) (1959) |
| 2 | 26049 | Fires on the Plain (Nobi) (1959) |
| 2 | 27093 | Class Trip, The (La classe de neige) (1998) |
| 2 | 7578 | Midnight (1939) |
| 2 | 69354 | Went the Day Well? (1942) |
| 2 | 39244 | Leila (1996) |

1-10 of 1,000 rows    Previous 1 2 3 4 5 6 ... 100 Next

# Data 612 Final Project

## Building Recommender System in Spark

### ***Calculate RMSE***

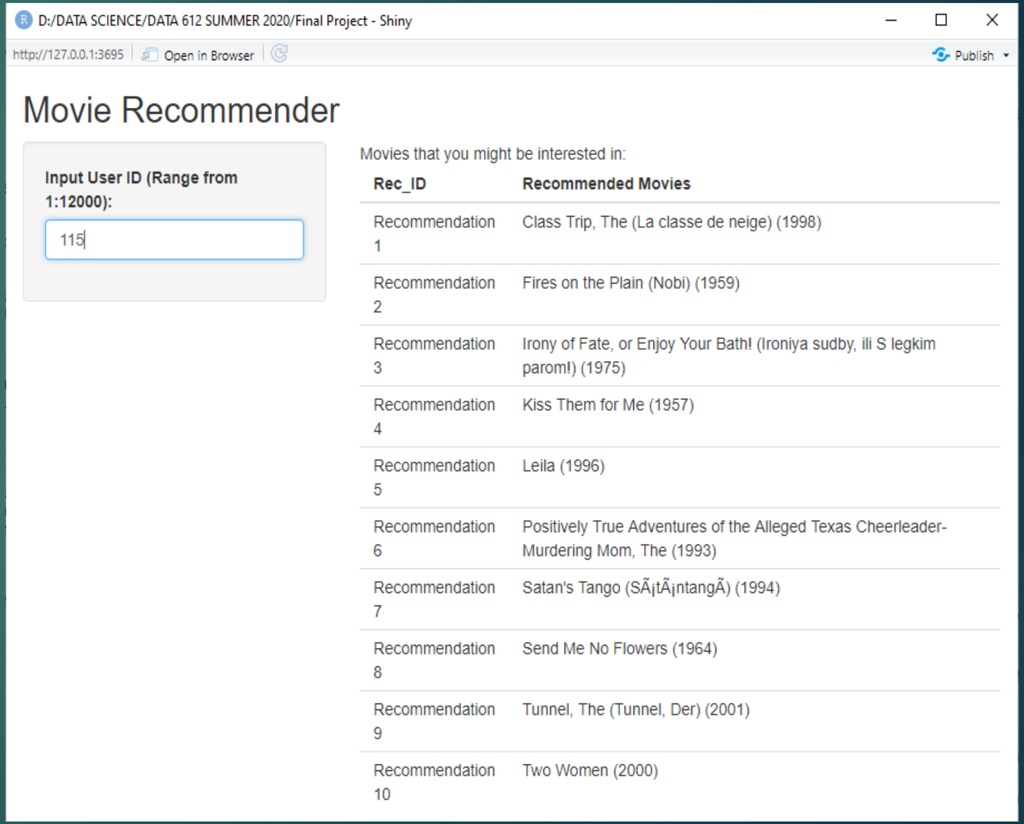
```
```{r spark rmse, message=FALSE, warning=FALSE}
rmse_spark <- predict_spark %>%
  filter(!isnan(prediction)) %>%
  summarise((rating - prediction)^2 %>% mean() %>% sqrt()) %>%
  collect() %>%
  as.numeric()
print(str_c('RMSE of Model Built in Spark: ',rmse_spark %>% as.character()))
```

[1] "RMSE of Model Built in Spark: 0.85406397406834"
```

# Data 612 Final Project

## Building User Interface with Shiny

```
```{r shiny, eval=FALSE}
ui <- fluidPage(
  titlePanel("Movie Recommender"),
  sidebarLayout(
    sidebarPanel(
      textInput("txtInput", "Input User ID (Range from 1:12000):"),

    ),
    mainPanel(
      paste("Movies that you might be interested in:"),
      tableOutput('tblOutput')
    )
  )
)

server <- shinyServer(function(input,output){
  output$tblOutput <- renderTable({
    recommendations <- rec_result %>% filter(User == input$txtInput) %>%
      select(Rec_ID, `Recommended Movies`)
  })
})

shinyApp(ui = ui, server = server)
```
```




شكراً جزيلاً
ngiyabonga
рахмат
danke
謝謝
teşekkür ederim
Баярлалаа
mersi
barka
welalin
tack
спасибо
faafetai lava
vinaka
blagodaram
dank je
misaotra
matondo
paldies
grazzi
gracias
tapadh leat
хвала
kiitos
dankie
dhanyavad
thank you
asante
manana
obrigada
tenki
mochchakkeram
djiere dieuf
dziękuję
hvala
bedankt
enkosi
chnorakaloutioun
gratias ago
gràcies
sulpáy
go raibh maith agat
mamnun
дякую
sobodi
dekuji
obrigado
sukriya
kop khun krap
arigatô
takk
dakujem
trugarez
ありがとう
tanemirt
grazie
mèsi
najis tuke
terima kasih
rahmet
diolch
dhanyavadagalu
shukriya
мерси
kam sah hamnida
xièxie
감사합니다
merci
ευχαριστώ

# Thank You