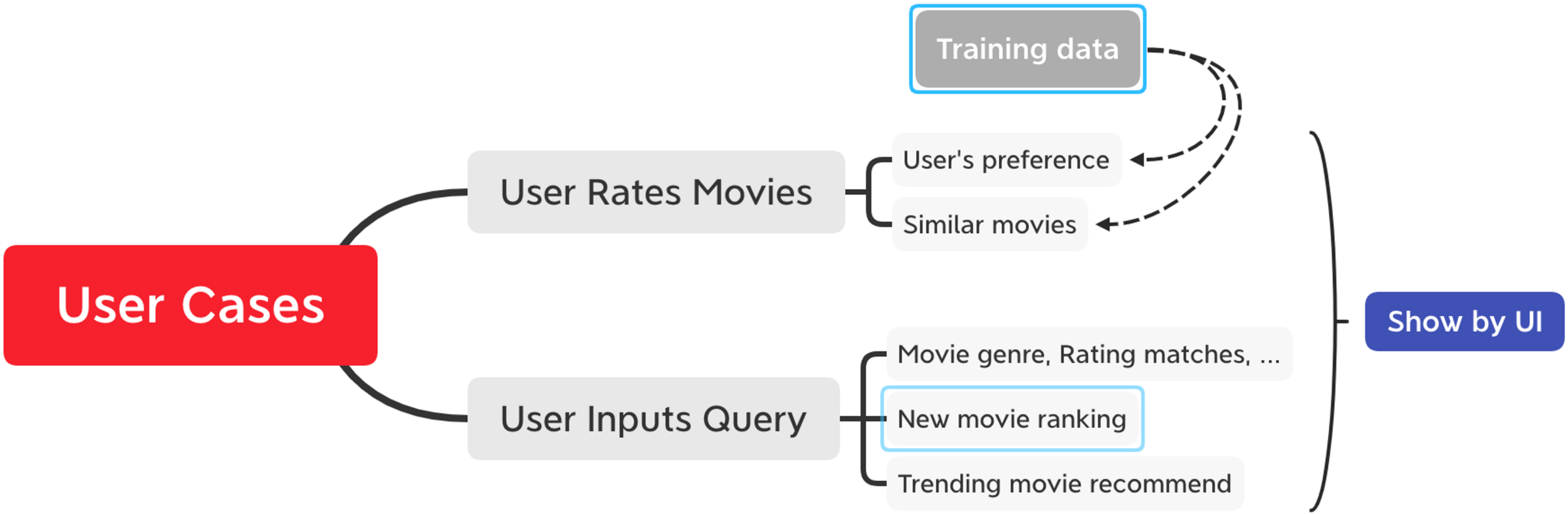
Movie Recommendation

Team III

Mengzhe ZHANG

Rongqi SUN

Yue FANG

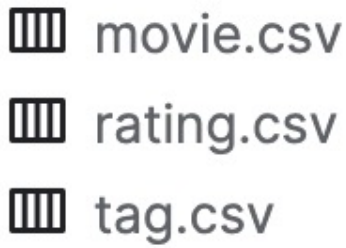

User Cases
User Rates Movies
User Inputs Query
Training data
User's preference
Similar movies
Movie genre, Rating matches, ...
New movie ranking
Trending movie recommend
Show by UI

Methodology

- **Statistics:**
  - Use Spark Core + Spark SQL deal dataset
    - Load dataset with MongoDB through Spark SQL
- **Recommendation:**
  - Algorithms use ALS to summarize user's preference and similar movies
  - Use Spark Core + Spark MLlib to implement recommendation methods

Data Source

- **MovieLens 20M Dataset**
  - https://www.kaggle.com/grouplens/movielens-20m-dataset
  - **It contains 20000263 ratings and 465564 tag applications across 27278 movies.**
  - **These data were created by 138493 users between January 09, 1995 and March 31, 2015.**
  - **Users were selected at random for inclusion. All selected users had rated at least 20 movies.**



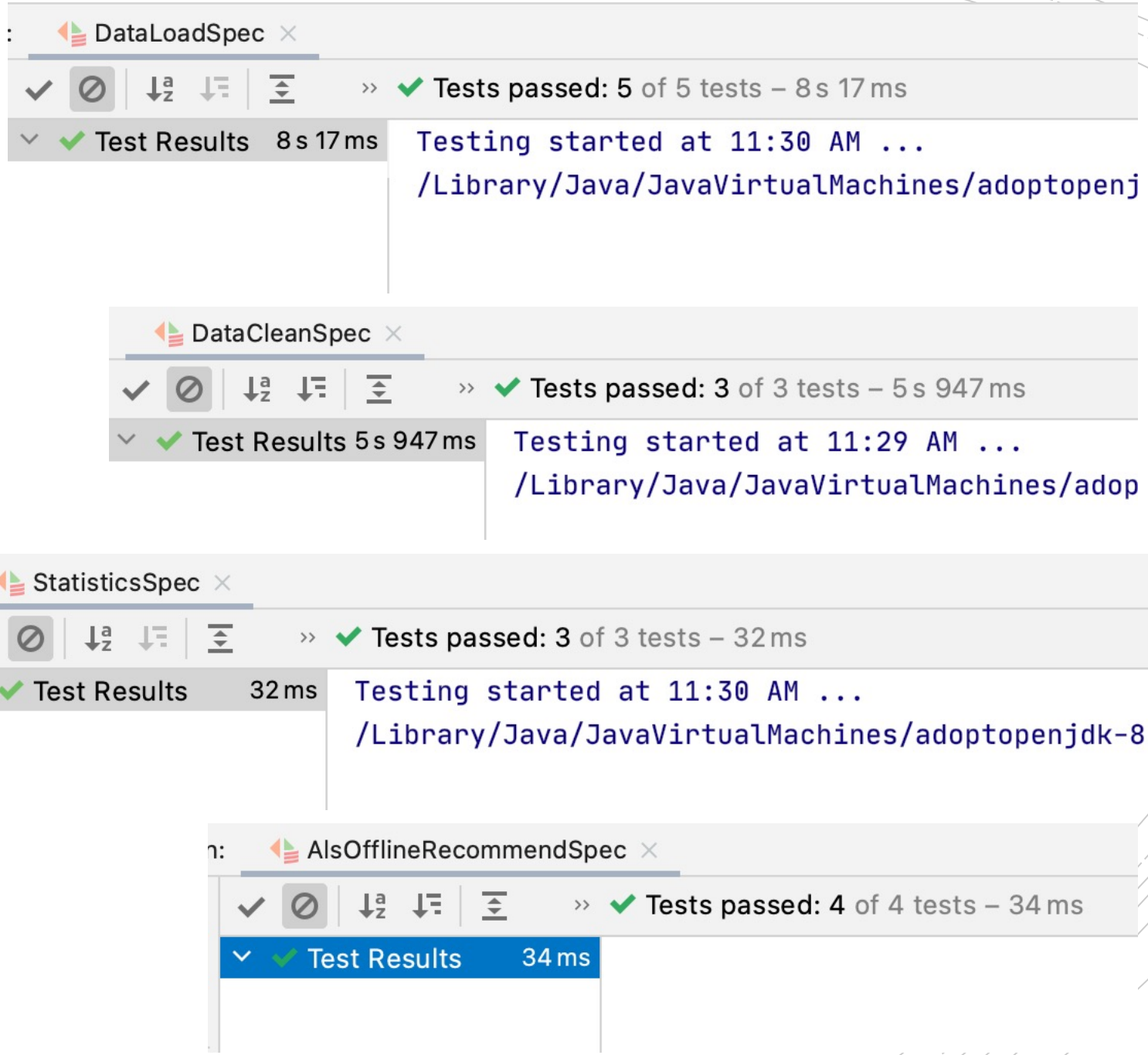
Milestones

Milestones	Time
Data cleaning and processing Unit Test	3.22 – 3.28
Recommendation methods implementing Unit Test	3.29 – 4.7
Setup UI Implement visualization	4.8 – 4.14
Final model and use cases testing System Test	4.15 – 4.21

Repository

- Scala:
- Recommendation Part
- User Interface Use Play
- MongDB for dataset using SparkSQL
- Repo: https://github.com/tracy626/CSYE7200_FinalProj_Team3

Unit Test


DataLoadSpec
Tests passed: 5 of 5 tests – 8 s 17 ms
Test Results 8 s 17 ms
Testing started at 11:30 AM ...
/Library/Java/JavaVirtualMachines/adoptopenj
DataCleanSpec
Tests passed: 3 of 3 tests – 5 s 947 ms
Test Results 5 s 947 ms
Testing started at 11:29 AM ...
/Library/Java/JavaVirtualMachines/adop
StatisticsSpec
Tests passed: 3 of 3 tests – 32 ms
Test Results 32 ms
Testing started at 11:30 AM ...
/Library/Java/JavaVirtualMachines/adoptopenjdk-8.
AlsOfflineRecommendSpec
Tests passed: 4 of 4 tests – 34 ms
Test Results 34 ms

Implementation

Statistics

- Most rated movies
- Movies of highest average score
- Movies of highest score in genre

```
  storeInMongoDB(genresTopMoviesDF, GENRES_TOP_MOVIES)
}

def storeInMongoDB(df: DataFrame, collection_name: String)(implicit mongoConfig: MongoConfig): Unit = {
  df.write
    .option("uri", mongoConfig.uri)
    .option("collection", collection_name)
    .mode( saveMode = "overwrite")
    .format( source = "com.mongodb.spark.sql")
    .save()
}
```

Implementation

ALS

- The Cartesian product of UserID and MovieID produces a empty tuple of (uid, mid)
- The tuple of (uid, mid) predicted by the model.
- Sort the prediction results by prediction score.
- Return the K movies with the highest scores as the recommendation of the current user.
- Parameters (rank, iterations, lambda)

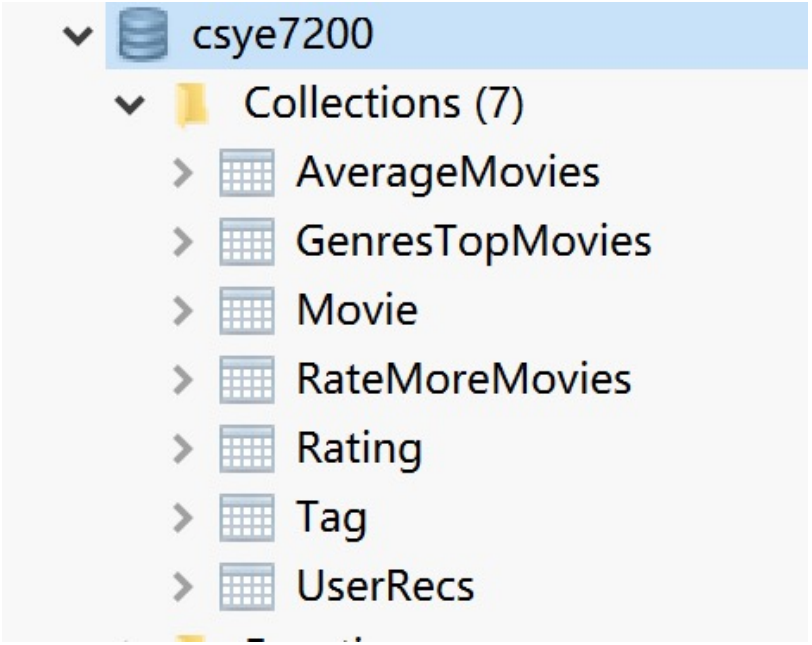
We need to evaluate the model. The usual approach is to calculate the root mean square error (RMSE) and examine the error between the predicted score and the actual score.

- With RMSE, we can select the group with the smallest RMSE as the optimal choice for our model by adjusting the parameter values multiple times.

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{t=1}^{N}(observed_t - predicted_t)^2}$$

Database

Recommendation Results



- **Statistics Recommendation:**

  AverageMovies: Movies with average scores

  GenresTopMovies: Different genres with top 10 ranking movies

  RateMoreMovies: Movies with counts of ranking

- **ALS Recommendation:**

  UserRecs: 20 pieces of movie recommendation to each user

- Play

GUI

Statistics Recommendation

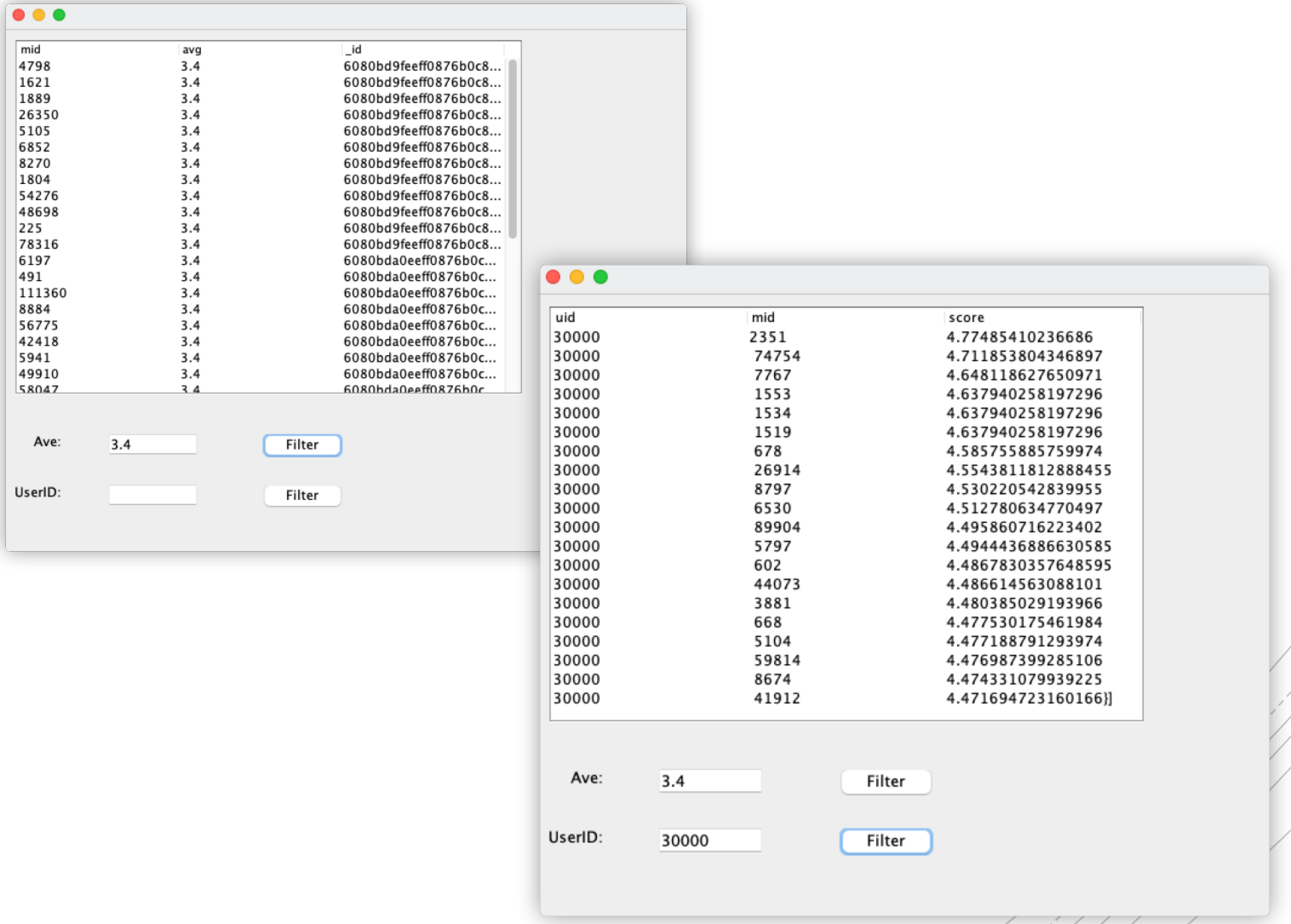
GET	`/rate/{id}`	Get a Movie
GET	`/rate`	Find all Movies in Most Rating List
GET	`/avg/score/{avg}`	Find all Movies with Request Average Score
GET	`/avg/{id}`	Get a Movie
GET	`/avg`	Find all Movies in Average Score Recommendation List
GET	`/gtop/{genre}`	Get Movies from Chosen Genres
GET	`/gtop`	Find all Movies in Genres Recommendation

User Preference

GET	`/user/{uid}`	Get Movies Recommend for User
GET	`/user`	Find all Movies in Genres Recommendation

GUI

- Swing (only for presentation demo)



Acceptance criteria

- User is able to:
  - Get movie recommendation according to this user's rating history (predict user's preference)
  - Ask for recommendation of movies and get list by ranking and genres
  - Get trending movie recommendation and check information with average score, ranking count…
  - Get User-Based Recommend result in less than 1 seconds on UI
  - Get Statistics Recommend result in less than 1 second on UI

- Recommendation system has:
  - Statistic results about movie recommendation related to rating, score, counts, genres…
  - ALS results about movie recommendation based on user's previous preference and prediction analysis
  - All records and results are available in MongoDB database
  - An UI constructed with Play framework

Goals of the Project

- Clean and process raw dataset
- Analyze movies rating with other features
- Input recommendation and analysis results to database
- Create UI for recommendation system
- Create page for user to filter recommendation results