Steam Users & Reviews Analysis

- Group Member: Zhipeng Hong, Zhiyi Ren, Xinyue Wang, Haotian Gong, Zihao Ren

Overview

Steam is a PC(personal computer) video games platform, and it is used to distribute games for games companies. For each game in Steam, players can post their reviews of the game to inform others who want to buy this game, and other players and game developers can reply to these reviews. The reviews in steam have a binary attitude: recommend or not. In our analysis, we analysis and model the Steam's game reviews data.

Data Set 1: Steam Reviews Dataset

- 6,976,390 unique users
- 15,437,471 reviews
- 8,183 games reviewed

This dataset is scraped from the STEAM API and contains user reviews of video games in the Steam store, as well as a wealth of information about each review, such as steamid, appid, playtime, the review text, likes/dislike, timestamp of review created, timestamp of review updated, etc.

Data size: 5.32GB

Factor	Datatype	Detail
steamid	Numeric	The user's ID
appid	Numeric	the ID of the game being reviewed
voted_up	Boolean	True/False - true means it was a positive recommendation
votes_up	Numeric	the number of other users who found this review helpful
votes_funny	Numeric	the number of other users who found this review funny
weighted_vote_score	Numeric	helpfulness score
playtime_forever	Numeric	how many hours the user had played this game at retrieval
playtime_at_review	Numeric	how many hours the user had played the game when writing this review
num_games_owned	Numeric	the number of games this user owns
num_reviews	Numeric	the number of reviews this user has written
review	Text	the text of the written review
unix_timestamp_created	Timestamp	date the review was created (unix timestamp)
unix_timestamp_updated	Timestamp	date the review was last updated (unix timestamp)

Data Set 2: Steam Store Games Dataset

This dataset provides information about **Steam Store Games** about various aspects of video games in the Steam store, such as appid, name, release date, english/not english, platforms, categories, genres, positive rating, negative rating, average playtime, median playtime and price, etc.

Data size: 5.83MB

Factor	Datatype	Detail
`appid`	Numeric	the ID of the video game
`name`	Text	video game name
`release_date`	Timestamp	release date of the game in format YYYY-MM-DD
`english`	Binary	1 if is in English, 0 otherwise
`developer`	Text	name of developer(s)
`publisher`	Text	name of publisher(s)
`platforms`	Text	semicolon delimited list of supported platforms. At most includes: windows;mac;linux
`required_age`	Numeric	minimum required age according to PEGI UK standards.
`categories`	Text	semicolon delimited list of game categories, e.g. single-player; multi-player
`genres`	Text	semicolon delimited list of game genres, e.g. action; adventure
`achievements`	Numeric	number of in-games achievements
`positive_ratings`	Numeric	number of positive ratings
`negative_ratings`	Numeric	number of negative ratings
`average_playtime`	Numeric	average user playtime
`median_playtime`	Numeric	median user playtime
`owners`	Categorical	estimated number of owners. Contains lower and upper bound (like 20000-50000)
`price`	Numeric	current full price of title in GBP(Great Britain Pound)

Why Chose these data sets?

- On Kaggle, the first dataset (Steam Reviews Dataset) has not yet been used for analysis so we feel this dataset is a good place to start
- Two datasets have a wide range of information we can utilize and get insights from them
- We are all steamers!!!

Data Analytics Goals

- How we can segment the steam reviewers and what are their traits using the K-means cluster.
- Trying to figure out what features are important to decide whether the steam players are more likely to 'voted up' the steam game and compare the decision tree, logistic regression model and random forest performance.
- Provide game recommendations to each validation users using collaborative filtering by ALS model.

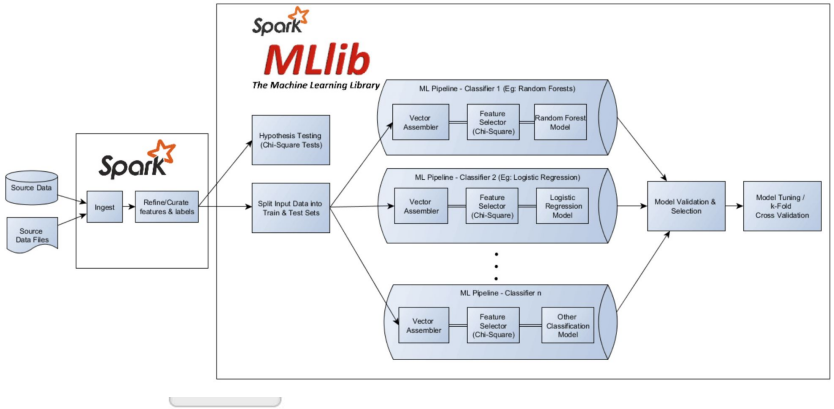
Data Pipeline

mongoDB. Atlas

M30 (8 GB RAM, 80 GB Storage per Shard)
3,000 IOPS per Shard, Encrypted, Auto-expand Storage

MongoDB 5.0, Backup, 3 Shards
Cloud Backup

databricks



Worker type | Min workers | Max workers

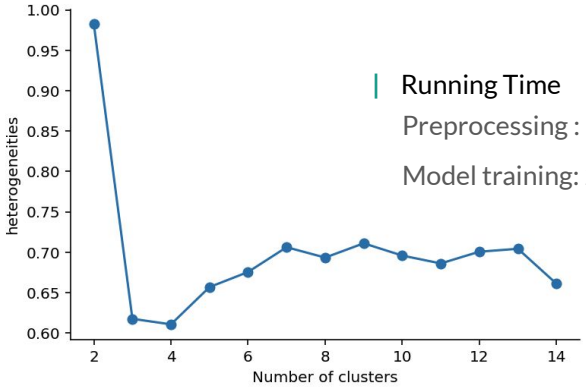
i3.xlarge — 30.5 GB Memory, 4 Cores | 2 | 8

Driver type

i3.xlarge — 30.5 GB Memory, 4 Cores

Steam Reviewer segmentation

We segment the reviewer into 4 categories using the ‘elbow rule’ after applying the K-means++ algorithm



Running Time

Preprocessing : 1 min 15 seconds

Model training: 2 mins 4 seconds

Group(%) Average Performance	Hours Played	Price	Games Owned	Number of Reviews	Other ‘s Votes up	
1 (19%) “Addictive Steamer”	3,270	16.55	209	18	1.6	Highly involved in the community which provides a good review and is very lucrative for steam
2 (7%) “Free Steamer”	22,494	0	65	4	0.5	Only play free games and give poor reviews for games but love play these games
3 (49%) “Normal Steamer”	2,468	15.93	98	7	0.84	Normal gamers with some game and review
4 (25%) “Golden Steamer”	5,229	20.88	139	10	1.31	Play high price games and love to buy new games, also a good reviewer group

Game Reviews - Voted up

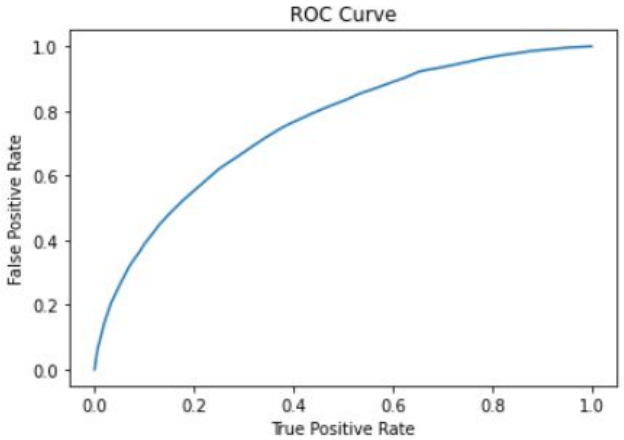
Using logistic regression, decision tree, and random forest model to explore what **features** are important to decide whether the steam player will likely to '**voted up**' the steam game.

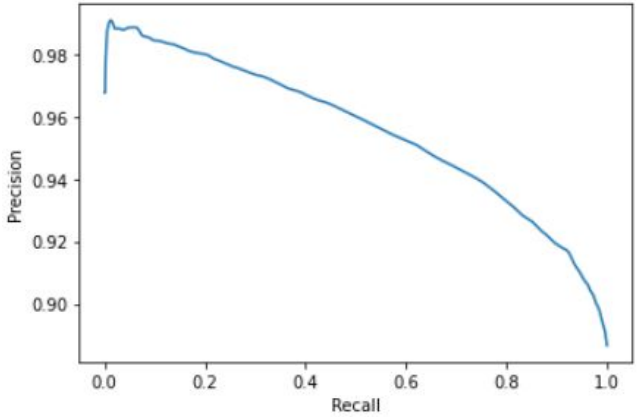
Logistic Regression Model

Model Parameters

Regularization : 0.01

Intercept : True



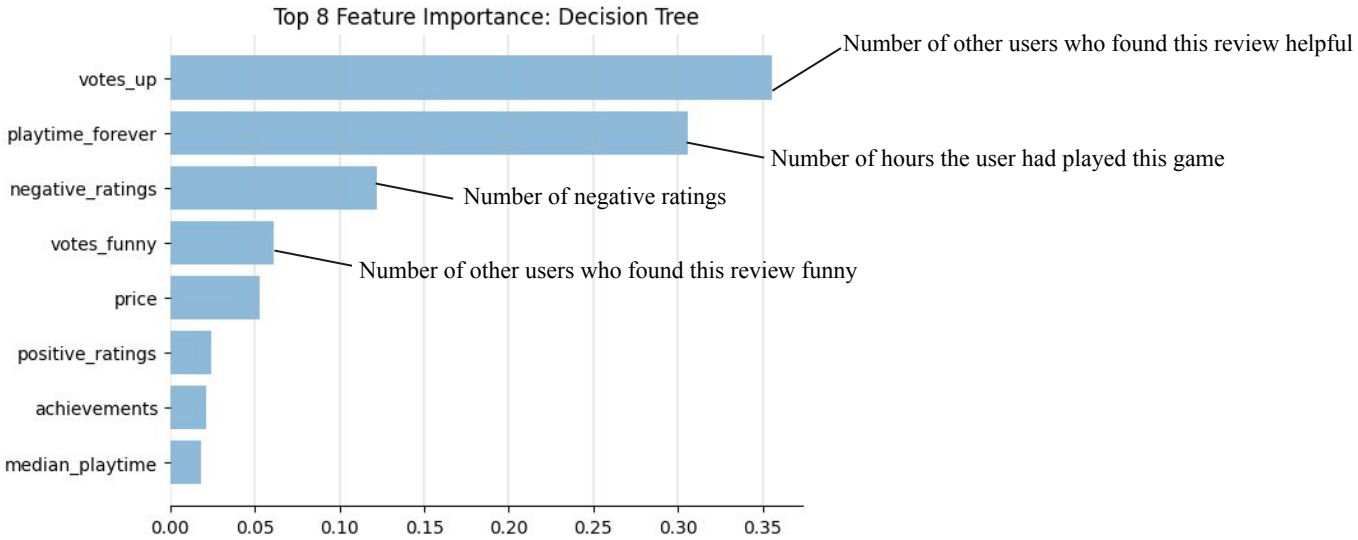


Evaluate Logistic Regression Model

- Running Time
  - Preprocessing : 1 min 32 seconds
  - Model training: 1 mins 25 seconds
  - Evaluation: 3 mins 37 seconds
- Evaluation Metrics
  - **Accuracy**: 0.84240
  - **F1 Score**: 0.84244
- Model Parameters
  - **Regularization** : 0.01
  - **Intercept** : True

Confusion Matrix		
	Actually Positive(1)	Actually Negative(0)
Predicted Positive(1)	1,998,915	6,792
Predicted Negative(0)	246,068	10257

Decision Tree model



Evaluate our Decision Tree model

- Running Time
  - Preprocessing : 1 min 49 seconds
  - Model training: 8 mins 5 seconds
  - Cross Validation: 2 hours 14 mins 16 seconds
- Evaluation Metrics
  - **Accuracy**: 0.7912
  - **F1 Score**: 0.8852
- Cross Validation Check
  - **Accuracy** : 0.8923

Set maxDepth = [10]

Set maxDepth = [5,10,**15**,20]

Best MaxDepth

Confusion Matrix		
	Actually Positive(1)	Actually Negative(0)
Predicted Positive(1)	1,963,084	39,070
Predicted Negative(0)	171,116	85,214

Random Forest model - Adding Review Sentiments

Using feature engineering to convert review into the sentimental score.

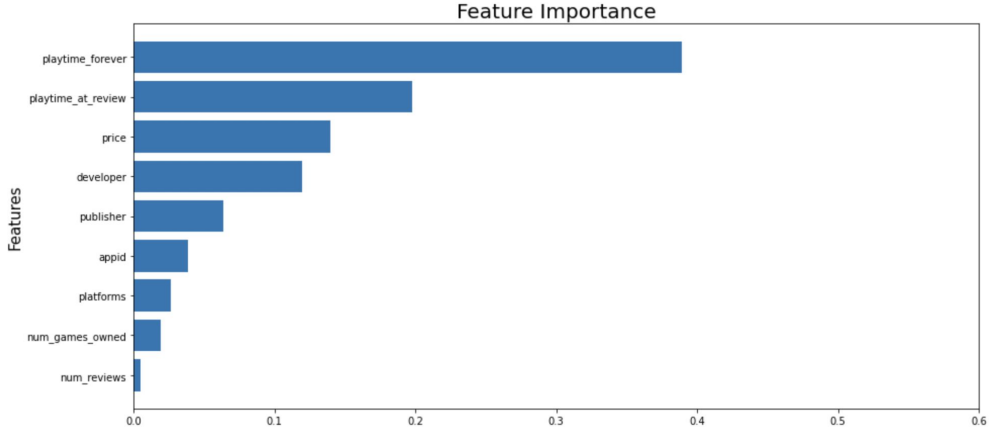
Feature Engineering

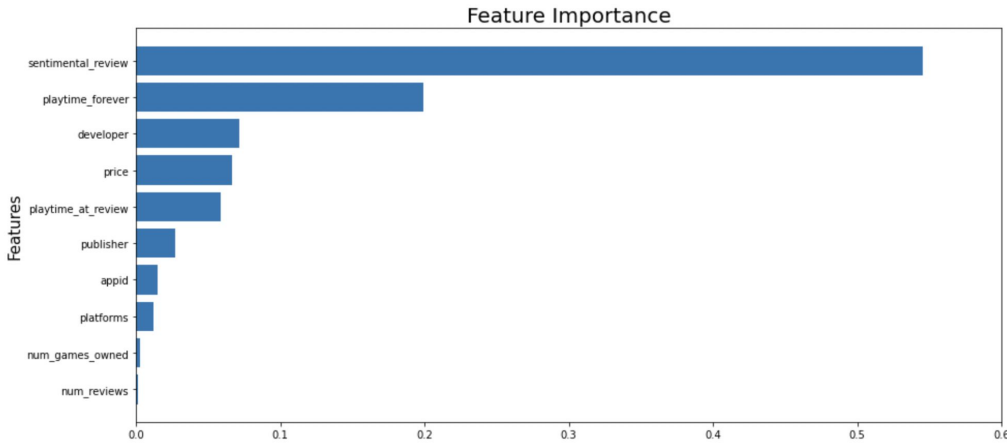
As review contains lots of information, we need to transform the ‘review’ column and use vaderSentiment package to convert review content into a sentimental score between [-1,1].

Then we use the review content to fit and predict the ‘voted up’ (response variable).

Random Forest model

Comparing the 2 different models, the feature importance shows that sentimental score is vital important to our prediction.





Evaluate Random Forest

- Evaluation Metrics

  Model (with sentimental score) : **Accuracy**,0.974 **Best Model** : numTree = 15, maxDepth = 10, maxBin = 6000

  Model (without sentimental score) : **Accuracy**,0.956 **Best Model** : numTree = 15, maxDepth = 10, maxBin = 6000

- Running Time

  Model1 : **Preprocessing** : 4.6mins, **Cross Validation**: 39min 8s **Evaluation:** 4min 22s

  Model2: **Preprocessing**: 3.7 mins, **Cross Validation**: 27min 34s **Evaluation** 1min1s

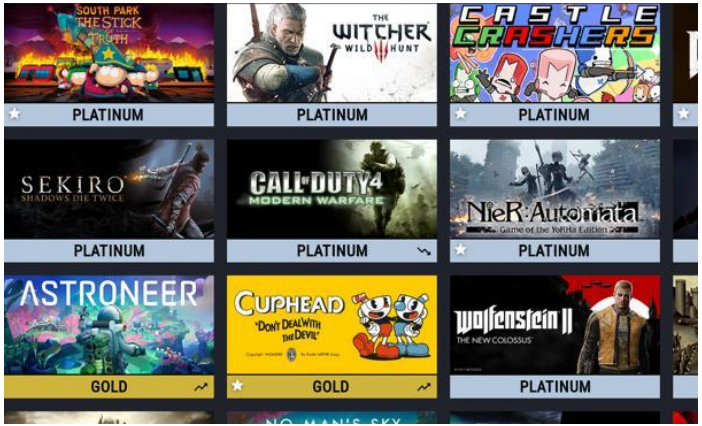
Model Comparison

Model	Accuracy	Total Running Time
Logistic Regression	0.8424	6 min 32 secs
Decision Tree	0.8923	2 hours 23 min 15 secs (4 iterations of grid search)
Random Forest	0.974	48 min 22 secs

- The Cluster setting is the same as mentioned above

Game Recommendation

Commercial success of modern games hinges on player satisfaction and retention. In particular, we are using **collaborative filtering** model and **alternative least squares (ALS)** algorithm to learn the user-item association matrix, and recommend 10 games to each user in the validation set.



Input Data

Features: SteamID, AppID
Label: Rating (Explicit, shown as voted_up)

Game Recommendation - Data Preprocessing

Collaborative filtering generally works bad on cord starter problems. Thus, we set up threshold for both users and games, and only those complying with the standard are subject to feed the model. Also, as the ALS model setting, we converted the data type to integer.

01 | Games

Take off games that have less than 100 user rating.

02 | Users

Take of users that have rated less than 10 games.

Game Recommendation - Key Indicator

- **Data Size**
  150 millions of user, item pairs after preprocessing
- **Running Time**
  Preprocessing: 2 mins 40 seconds
  Model training: 13 seconds
- **Evaluation Metrics**
  RMSE: 0.4016
- **Game recommendations**
  Generated top 10 recommended games for each user. Sample is shown on the right.

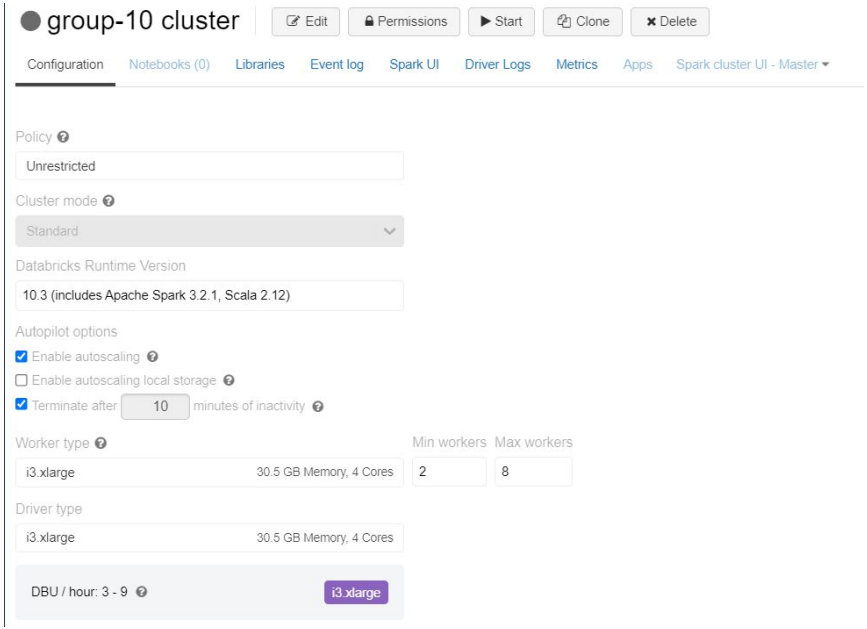
```
+-------+--------------------+
|steamid|     recommendations|
+-------+--------------------+
|     12|[{3689, 1.6259432...|
|     13|[{3689, 1.5246562...|
|     14|[{3391, 1.5112182...|
|     18|[{4081, 1.5102459...|
|     38|[{4014, 1.6496912...|
|     46|[{3689, 1.6301169...|
|     67|[{3614, 1.6708444...|
|     70|[{3830, 1.9571373...|
|     93|[{3908, 1.4536773...|
|    107|[{4014, 2.0046563...|
|    148|[{4082, 1.6894785...|
```

Conclusion

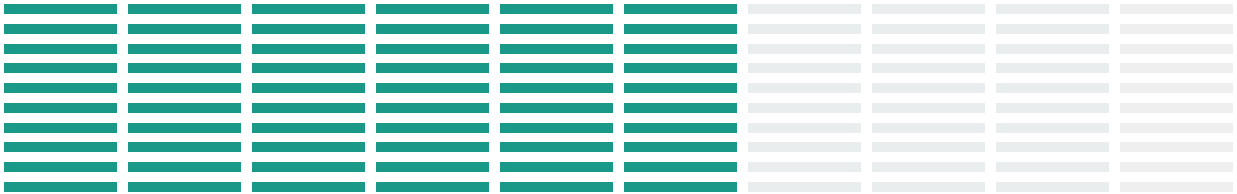
- K-means is a fast and reasonable way to segment the steamer and thus help us generate many insights we can not get at the initial look.
- Price and playtime are the most important features that can decide the user attitude.
- Convert review into sentiment score also helps increase model's performance.
- Collaborative filtering can learn the user-item matrix to generate related games for similar users.

Thank you.

Appendix I - Cluster Setting



Appendix II - Efficiency Improvement



60%

Each member achieved an average 60% improvement on running time with persisting on memory

Appendix III - Lesson Learned

- We need to strike a balance between model efficiency and accuracy
  a. Simple model saving a lot of time
  b. Spark ML is powerful
- Large Dataset
  a. Data may not be as clean as we expect
  b. Select feature is the key