

Technical Appendix

Catch the Pink Flamingo Analysis

Produced by: Eduardo Cabezas A.

Acquiring, Exploring and Preparing the Data

Data Exploration

Data Set Overview

The table below lists each of the files available for analysis with a short description of what is found in each one.

File Name	Description	Fields
ad-clicks.csv	Contains player clicks on advertisements in the Flamingo App	adCategory: the category/type of ad clicked on adId: the id of the ad clicked on teamId: the current team id of the user who made the click timestamp: when the click occurred. txId: a unique id for the click userId: the user id of the user who made the click userSessionId: the id of the user session for the user who made the click
buy-clicks.csv	A line is added to this file when a player makes an in-app purchase in the Flamingo app.	buyId: Id of the item purchased Price: price of the item purchased team: current team of the user who made the purchase timestamp: when the purchase was made txId: Unique id for the purchase userId: user id of the user who made the purchase userSessionId: id of the user session for the user who made the purchase
game-clicks.csv	File contains a line for each time a user performs a click in the game	clickId: unique id for the click isHit: if the click was on a flamingo value is 1, for a missed flamingo value is 0. teamId: Id of the team of the user

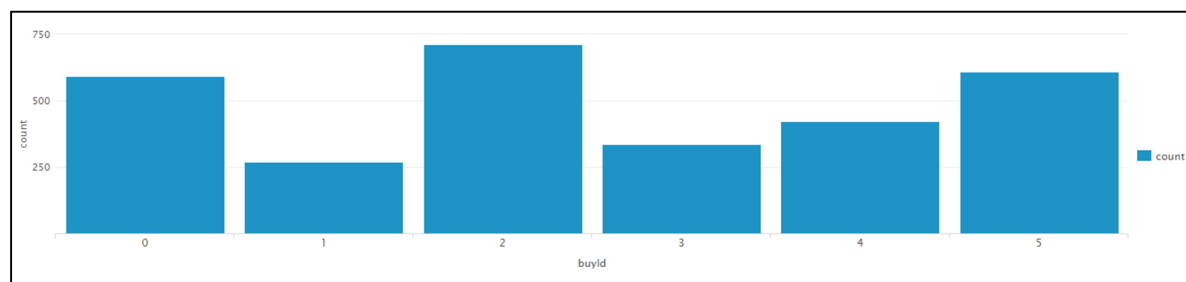
		teamLevel: current level of the team of the user timestamp: when the click occurred userId: id of the user performing the click userSessionId: id of the session of the user when the click is performed.
level-events.csv	Contains a line for each time a team starts or finishes a level in the game	eventId: unique id for the event extracted_eventType: type of the event, either start or end teamId: id of the team teamLevel: level started or completed timestamp: when the event occurred.
team.csv	Contains a line for each team terminated in the game	currentLevel: current level of the team name: name of the team strength: strength of the team, roughly corresponding to the success of a team. teamCreationTime: timestamp when the team was created. teamEndTime: timestamp when the last member left the team. teamId: Id of the team.
team-assignments.csv	Contains a line for each time a user joins a team. A user can be in at most a single team at a time.	assignmentId: unique id for this assignment. team: id of the team timestamp: when the user joined the team userId: id of the user
users.csv	File contains a line for each user playing the game	country: two letter country code where the user lives dob: date of birth of the user nick: nickname of the user timestamp: when user played the game for the first time twitter: twitter handle of the user userId: User id assigned to the user
user-session.csv	Each line describes a user session. When the user starts and stop playing the game. Also when a team goes to the next level, the session is ended for each user in the team and a new one started.	assignmentId: team assignment id for the user of the team platformType: type of platform of the user during this session. sessionType: whether the event is the start or the end of a session. teamId: current user's team teamLevel: level of the team during this

		session timestamp: when the event occurred userId: current user's Id userSessionId: unique id for the session
--	--	--

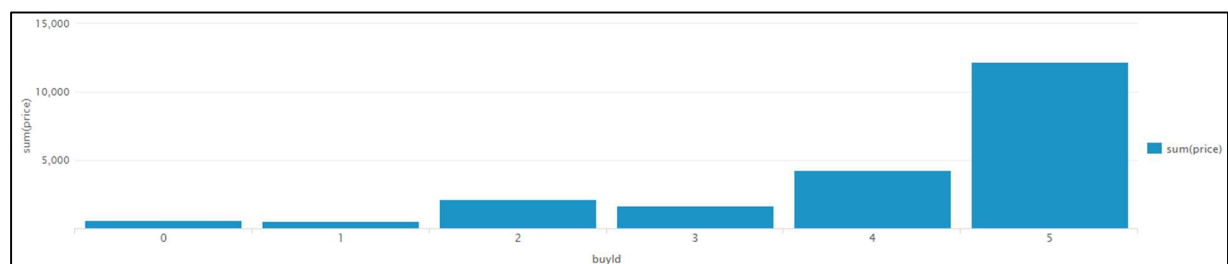
Aggregation

Amount spent buying items	USD 21407.-
Number of unique items available to be purchased	6

A histogram showing how many times each item is purchased:

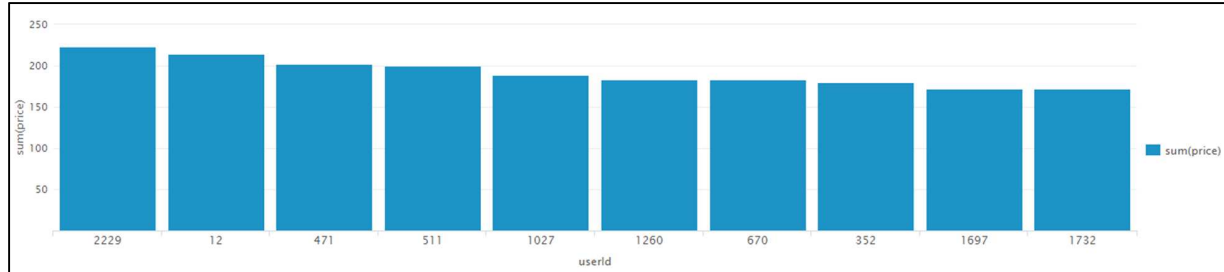


A histogram showing how much money was made from each item:



Filtering

A histogram showing how many times each category of advertisement was clicked-on:



The following table shows the user id, platform, and hit-ratio percentage for the top three buying users:

Rank	User Id	Platform	Hit-Ratio (%)
1	2229	iphone	11.6
2	12	iphone	13.1
3	471	iphone	14.5

Data Classification Analysis

Data Preparation

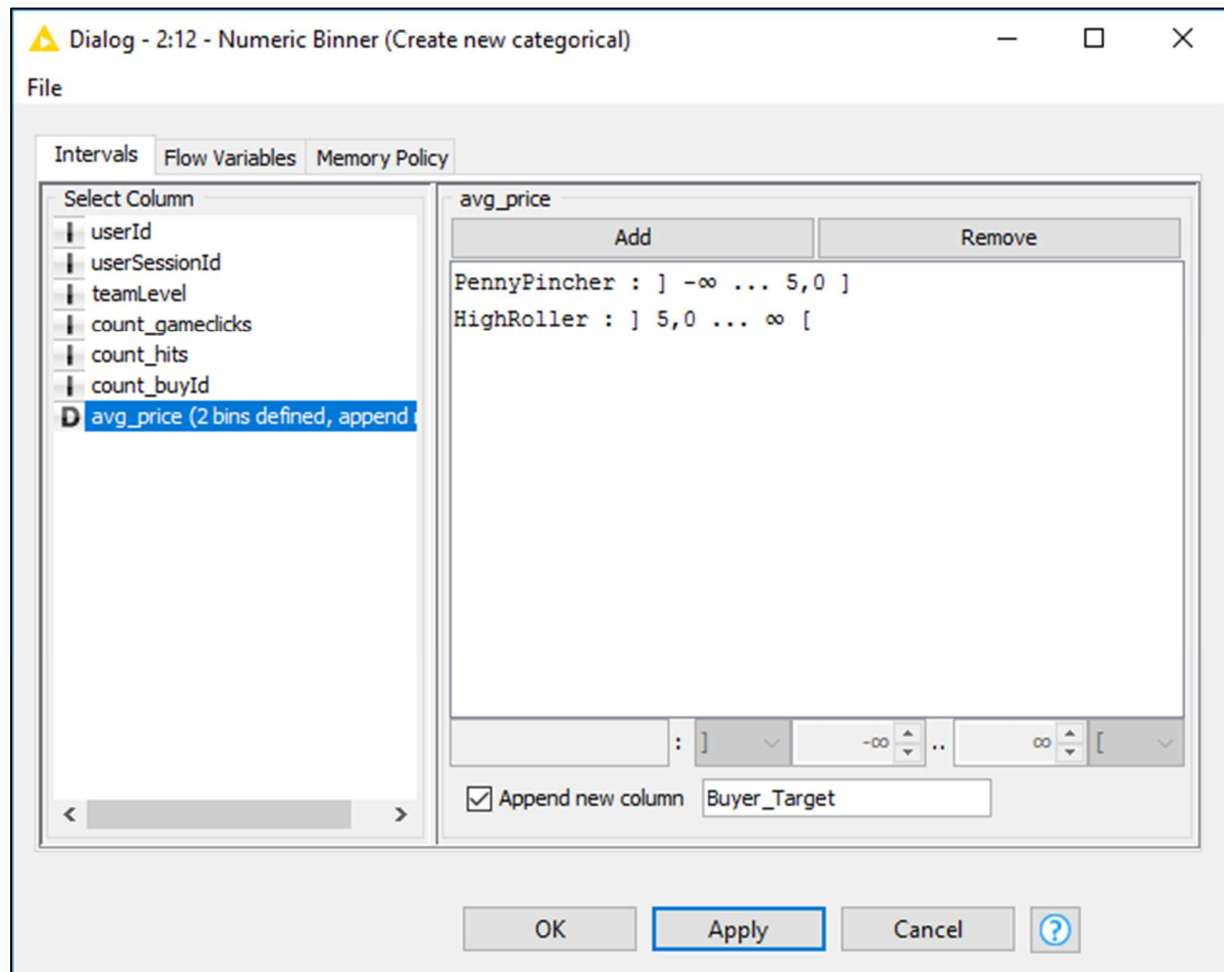
Analysis of combined_data.csv

Sample Selection

Item	Amount
# of Samples	4619
# of Samples with Purchases	1411

Attribute Creation

A new categorial attribute was created to enable analysis of players as broken into 2 categories (HighRollers and PennyPinchers). A screenshot of the attribute follows:



The new categorical attribute was defined as “buyer target” defining two categories: PennyPincher for buyers of items that cost more than \$5.00.- and HighRoller for buyers of items that cost \$5.00 or less.

The creation of this new categorical attribute was necessary because we need to classify users who buy big-ticket items vs buyers of inexpensive items in order to predict which user is likely to spend more money while playing Catch the Pink Flamingo.

Attribute Selection

The following attributes were filtered from the dataset for the following reasons:

Attribute	Rationale for Filtering
userId	It has a large number of different values with high intrinsic info, which may result in over fitting.
userSessionId	Same as above

Data Partitioning and Modeling

The data was partitioned into train and test datasets.

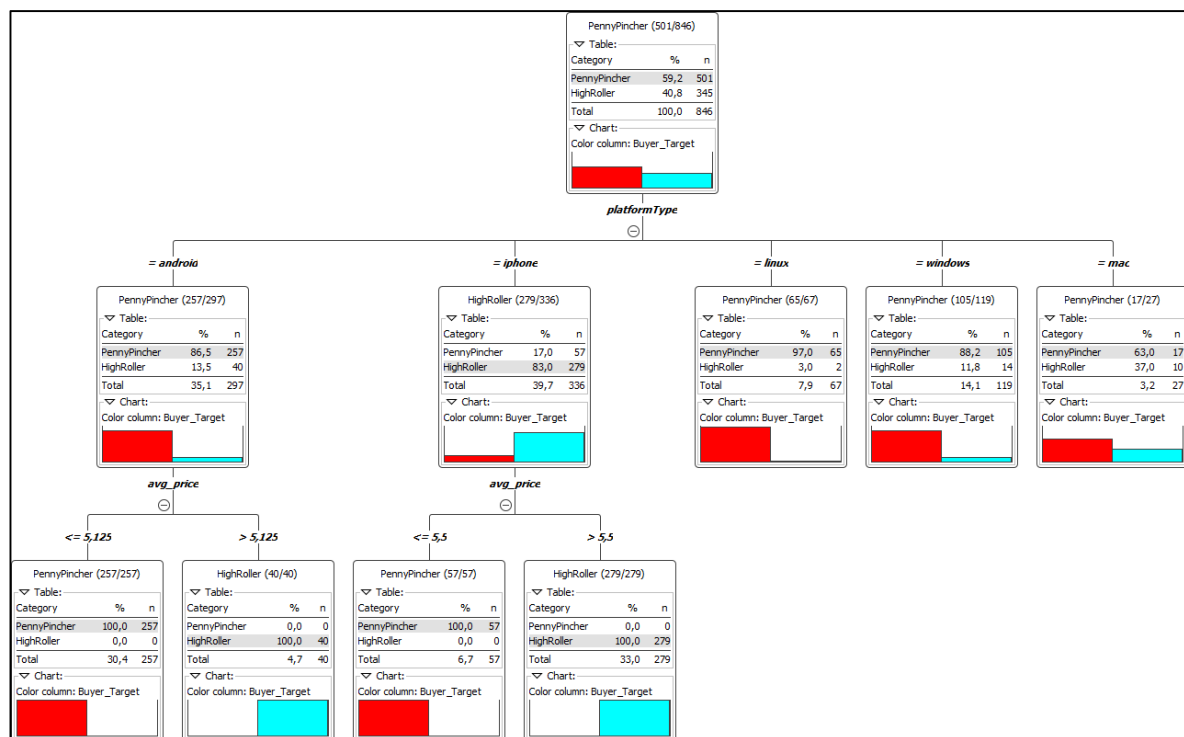
The Train data set was used to create the decision tree model.

The trained model was then applied to the Test dataset.

This is important because the train data is used for the purpose of training the model while the test data set will be used for the purpose of evaluating the performance of the model. Test set is kept separated from the training process because not doing so would bias the model causing the error estimate to be smaller than the true error rate.

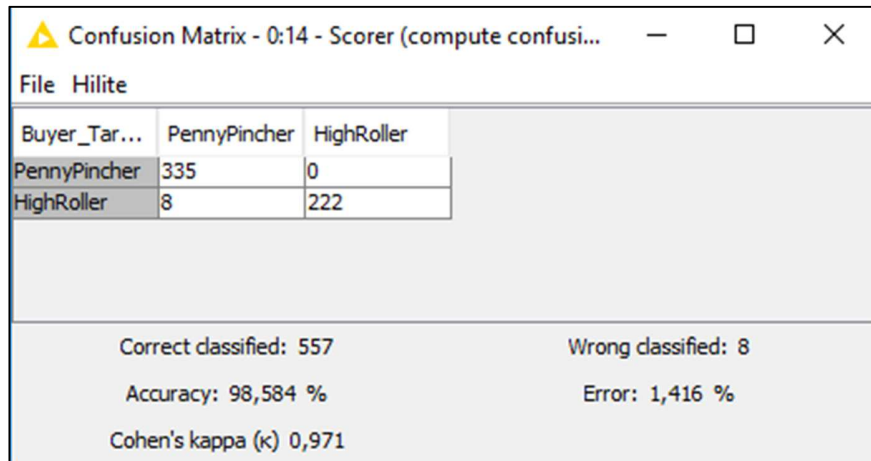
When partitioning the data using sampling, it is important to set the random seed in order to get reproducible results upon re-execution. If a random seed is not specified, a new random seed is taken for each execution.

A screenshot of the resulting decision tree can be seen below:



Evaluation

A screenshot of the confusion matrix can be seen below:



Buyer_Tar...	PennyPincher	HighRoller
PennyPincher	335	0
HighRoller	8	222

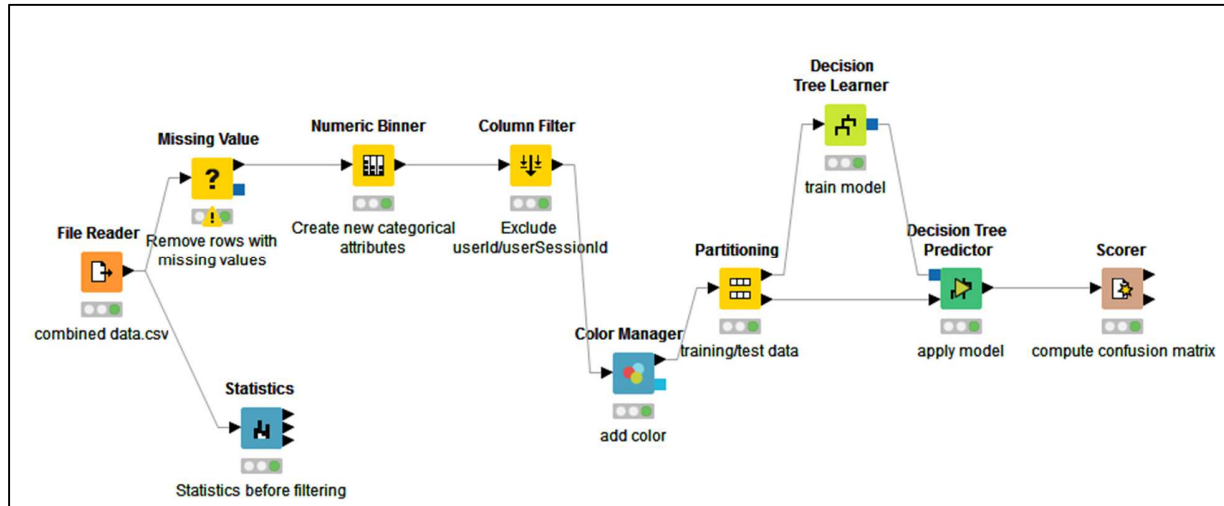
Correct classified: 557	Wrong classified: 8
Accuracy: 98,584 %	Error: 1,416 %
Cohen's kappa (κ) 0,971	

As seen in the screenshot above, the overall accuracy of the model is 98.6%

There are 565 samples in the test data set, 335 PennyPincher samples were correctly classified, no PennyPincher samples were incorrectly classified. 222 HighRoller samples were correctly classified, 8 HighRoller samples were incorrectly classified.

Analysis Conclusions

The final KNIME workflow is shown below:



What makes a HighRoller vs. a PennyPincher?

Analysing the model, we found that iphone users are more likely to be HighRollers while Android and Windows users are more likely to be PennyPinchers.

Specific Recommendations to Increase Revenue
1. Offer more in game purchases to iphone users.
2. Advertise more expensive products to iphone users.

Clustering Analysis

Attribute Selection

Attribute	Rationale for Selection
Price	Clustering based on revenue should be helpful to visualize users more profitable
TotalAdClicks	Analyze ad clicks by user to identify frequent clickers
HitRate	Look for any correlation between user success in the game and purchasing or ad clicking by such user.

Training Data Set Creation

The training data set used for this analysis is shown below (first 5 lines):

	totalAdClicks	revenue	HitRate
0	44	21.0	96
1	10	53.0	38
2	37	80.0	62
3	19	11.0	340
4	46	215.0	92

Dimensions of the training data set (rows x columns): (543 x 3)

of clusters created: 3

Cluster Centers

Cluster #	Cluster Center
1	40.23863636, 111.76136364, 86.65909091
2	26.65491184, 23.29722922, 56.33753149
3	31.5, 39.36206897, 238.03448276

These clusters can be differentiated from each other as follows:

Cluster 1 is different from the others in that revenue is the highest (~2.8 to 4.8 times higher) while also ad clicking is the highest (~1.3 to 1.5 times higher) and hit rate is the second highest.

Cluster 2 is different from the others in that users have the lowest hit rate (~1.5 to 4.2 times lower) and also the lowest revenue also ad clicking is the lowest.

Cluster 3 is different from the others in that users have the best hit success of the three clusters and the second-best revenue also ad clicking is the second highest.

Recommended Actions

Action Recommended	Rationale for the action
Offer higher price in app purchases to group of second best hit rate players	Since second best flamingo hitters spend more, higher end apps should be offered more oftenly to such group of users.
Increase number of ads	Ad clicking and revenue are directly correlated on all of the three clusters
Provide points or basic app bonuses for ad clicking (lower hit success users)	Users with less hit success spend less than other users, providing bonuses for ad clicking may make them more interested in the game and get them to the next level of hit success and start spending more on better in app purchases

Graph Analytics Analysis

Modeling Chat Data using a Graph Data Model

The nodes and relationships created in Neo4j from the chat data loaded is presented and explained in the following table.

Entity	Description	Relationships
User	Users of the game	(:user)-[:InteractsWith]->(:user) (:user)-[:CreateChat]->(:ChatItem) (:user)-[:Joins]->(:TeamChatSession) (:user)-[:Leaves]->(:TeamChatSession) (:ChatItem)-[:Mentioned]->(:user) (:user)-[:CreateSession]->(:TeamChatSession)
Team	Teams in Catch the pink Flamingo	(:TeamChatSession)-[:OwnedBy]->(:Team)
TeamChatSession	Session for team members to see each other's chats	(:user)-[:Joins]->(:TeamChatSession) (:user)-[:Leaves]->(:TeamChatSession) (:user)-[:CreateSession]->(:TeamChatSession) (:TeamChatSession)-[:OwnedBy]->(:Team)
ChatItem	One instance of a chat text	(:user)-[:CreateChat]->(:ChatItem) (:ChatItem)-[:PartOf]->(:TeamChatSession) (:ChatItem)-[:Mentioned]->(:user) (:ChatItem)-[:RespondsTo]->(: ChatItem)

Creation of the Graph Database for Chats

Describe the steps you took for creating the graph database. As part of these steps

- i) The following table describes the schema of the 6 csv files containing the chat data.

File	Description	Fields
chat_create_team_chat.csv	A line is added to this file when a player creates a new chat with their team. Each record creates a node labeled, "TeamChatSession", and edges labeled, "CreateSession" from user and "OwnedBy" to Team	Userid: id for the user starting the session Teamid: id for the team TeamChatSessionId: session id Timestamp: time of creation of the session
chat_join_team_chat.csv	A line is added when a user joins a team chat session. Creates an edge labeled "Joins" from user to teamchatsession	Userid: id for the user joining the chat session TeamChatSessionId: session id Timestamp: time when the user joined the session
chat_leave_team_chat.csv	A line is added when a user leaves a team chat session.	Userid: id for the user leaving the chat session

	Creates an edge labeled "Leaves" from user to teamchatsession	TeamChatSessionId: session id Timestamp: time when the user left the session
chat_item_team_chat.csv	A line is added to this file when a user adds a chat item to a team chat session. Creates the node labeled "ChatItem" and the edges labeled "CreateChat" from user to ChatItem and "PartOf" from ChatItem to TeamChatSession	Userid: id for the user creating the chat item TeamChatSessionId: session id ChatItemId: id of the chat item Timestamp: time of creation for the chat item.
chat_mention_team_chat.csv	A line is added when a user mentions another user in a chat session. Creates an edge labeled "Mentioned" from chatitem to user.	Userid: id of mentioned user ChatItemId: id of the chat item Timestamp: time of mention
chat_respond_team_chat.csv	A line is added when a user responds to a chat post. Creates an edge labeled "ResponseTo" from ChatItem to ChatItem	ChatItemId: id of source chat item ChatItemId: id of chat item responding source chat item Timestamp: time of response

ii) To load the chat data the following steps were executed:

- Download the game data from the link:
https://d3c33hcgiv3v3.cloudfront.net/de5ad40e9a4965ee4b6aad31eea4ac7a_big_data_capstone_datasets_and_scripts.zip?Expires=1493942400&Signature=WZ2umIVGT3ecmWbF0OIPoVd-HTCFiHx7KKE-qm~PHWvkiKW7eluWN02Nf3Htzdpd8UsMd4LUYvYGMg7sK4HdH3HuyYn~MkgNBCKHuNnUuNOwlggHmnhZqF7d6fTJs1sDWwr3Xyx5mH5AhLXHnc3-9fb8BJwrOivptjqQvmiii-Q_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A
- Unzip the file "chat-data.zip" containing the 6 CSV files described previously in i) to the folder Documents\Neo4j\default.graphdb\Import
- Load the data to neo4j using the following commands:

```
LOAD CSV FROM "file: /chat_create_team_chat.csv" AS row
MERGE (u:User {id: toInteger(row[0])}) MERGE (t:Team {id: toInteger(row[1])})
MERGE (c:TeamChatSession {id: toInteger(row[2])})
MERGE (u)-[:CreatesSession{timeStamp: row[3]}]->(c)
MERGE (c)-[:OwnedBy{timeStamp: row[3]}]->(t)
```

```
LOAD CSV FROM "file: /chat_join_team_chat.csv" AS row
MERGE (u:User {id: toInteger(row[0])}) MERGE (c:TeamChatSession {id: toInteger(row[1])})
MERGE (u)-[:Joins{timeStamp: row[2]}]->(c)
```

```
LOAD CSV FROM "file: /chat_leave_team_chat.csv" AS row
MERGE (u:User {id: toInteger(row[0])}) MERGE (c:TeamChatSession {id: toInteger(row[1])})
```

```
MERGE (u)-[:Leaves{timeStamp: row[2]]->(c)
```

```
LOAD CSV FROM "file: /chat_item_team_chat.csv" AS row
```

```
MERGE (u:User {id: toInteger(row[0])}) MERGE (c:TeamChatSession{id: toInteger(row[1])})
```

```
MERGE (i:ChatItem{id: toInteger(row[2])})
```

```
MERGE (u)-[:CreateChat{timeStamp: row[3]]->(i)
```

```
MERGE (i)-[:PartOf{timeStamp: row[3]]->(c)
```

```
LOAD CSV FROM "file: /chat_mention_team_chat.csv" AS row
```

```
MERGE (i:ChatItem{id: toInteger(row[0])}) MERGE (u:User {id: toInteger(row[1])})
```

```
MERGE (i)-[:Mentioned{timeStamp: row[2]]->(u)
```

```
LOAD CSV FROM "file: /chat_respond_team_chat.csv" AS row
```

```
MERGE (i:ChatItem{id: toInteger(row[0])}) MERGE (j:ChatItem {id: toInteger(row[1])})
```

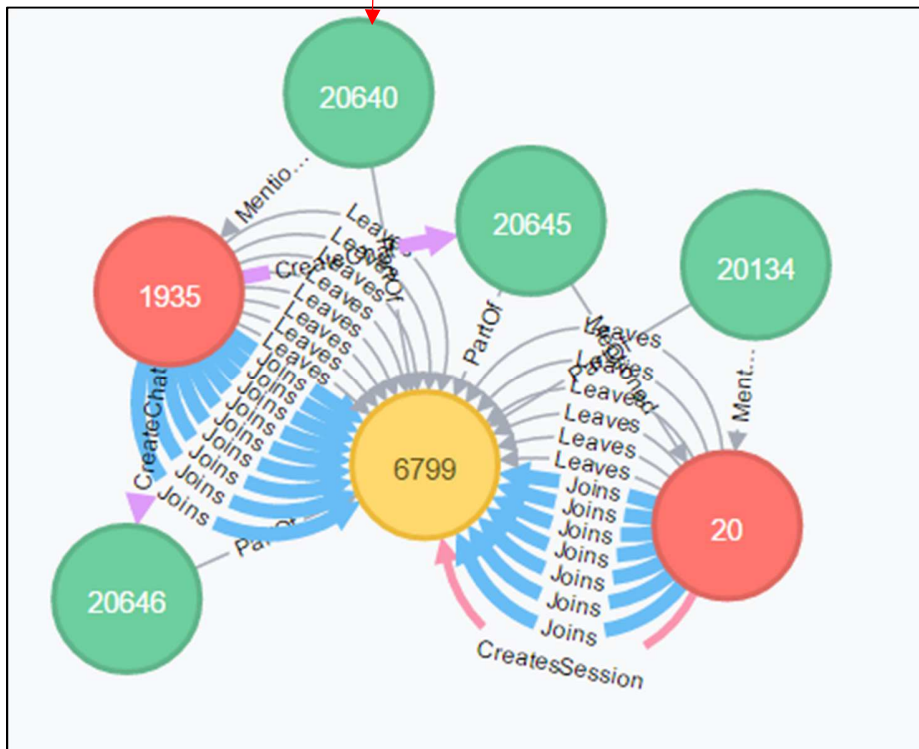
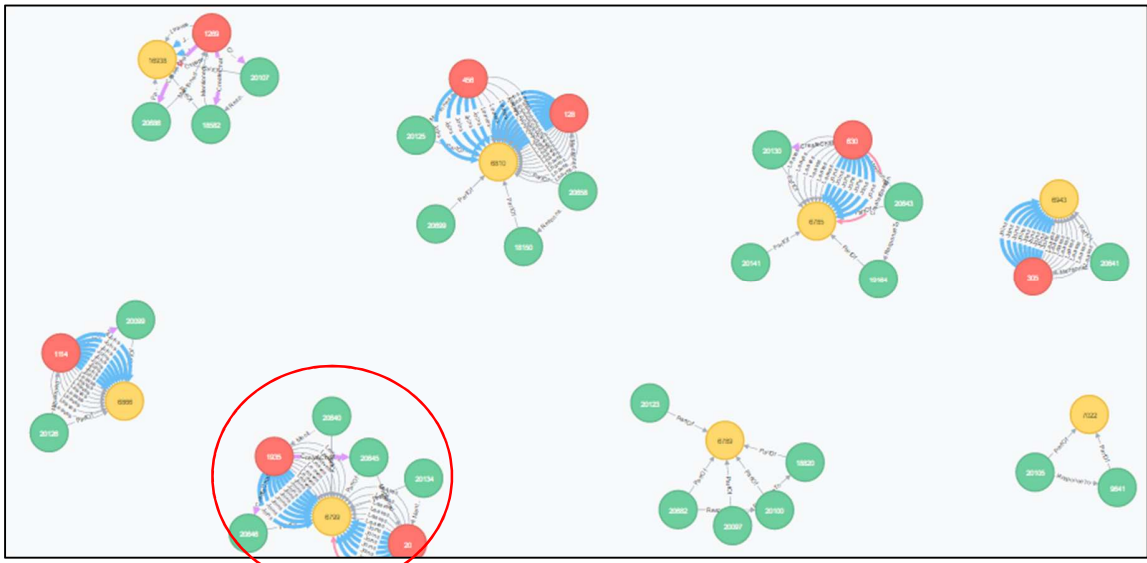
```
MERGE (i)-[:ResponseTo{timeStamp: row[2]]->(j)
```

- iii) The following screenshot presents the nodes and relationships in the database, using the query:

```
start n=node(*) match (n)-[r]->(m) return n,r,m limit 1000
```

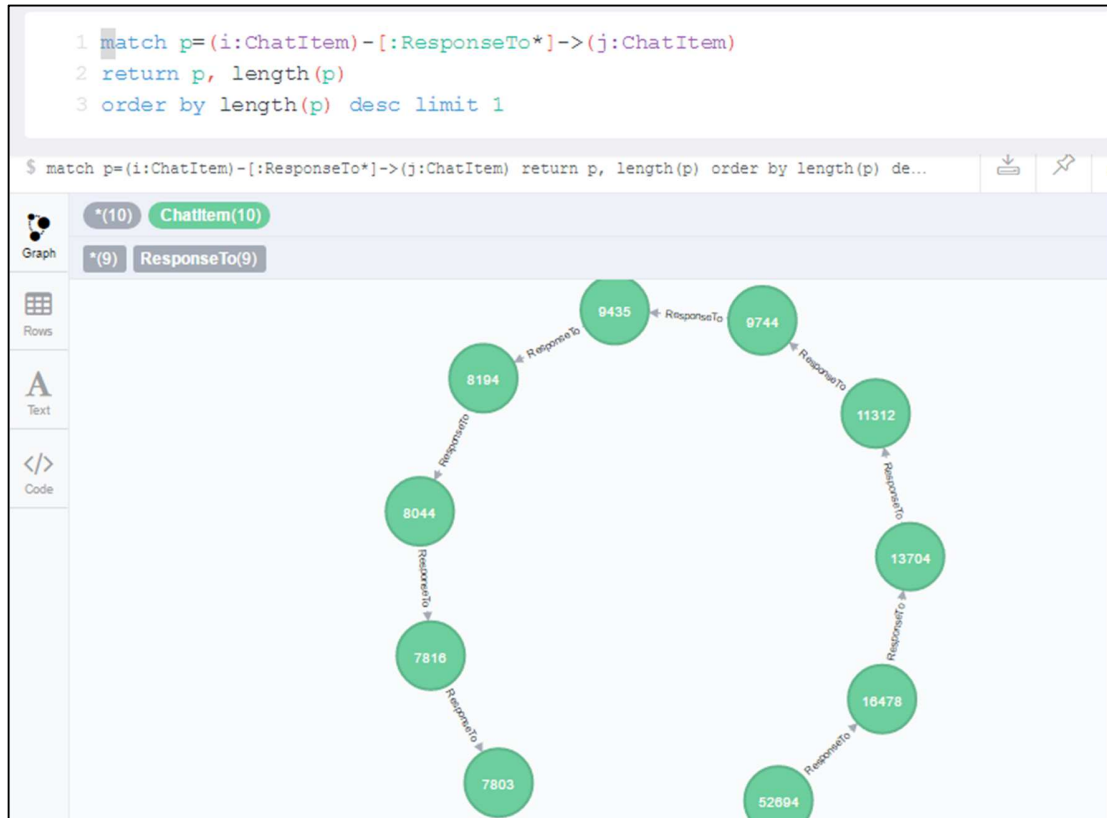
Red colored dots represent user nodes, Green colored dots represent chatitems and yellow colored dots represent TeamChatSessions. Cyan arrows represent Joins edges, purple arrows are createchat edges and pink arrows represent CreatesSession edges.

The zoomed image displayed shows the same query with a different scale to see in detail the nodes and relationships.



Finding the longest conversation chain and its participants

The following images present the query and results for the longest conversation chain and the number of distinct users involved.



Ten chats form the longest conversation chain with 5 users involved in it.

```
1 match p=(i:ChatItem)-[:ResponseTo*]->(j:ChatItem)
2 with p, EXTRACT(x IN NODES(p) | x.id) as iItemsId, length(p) as len order by
  len desc limit 1
3 match (u:User)-[:CreateChat]->(i:ChatItem)
4 where i.id in iItemsId
5 return count(distinct(u)) as usrCNT, u.id as UsrId
```

	usrCNT	UsrId
1	1	1153
1	1	1192
1	1	1978
1	1	1514
1	1	853

Started streaming 5 records after 763 ms and completed after 763 ms.

Analyzing the relationship between top 10 chattiest users and top 10 chattiest teams

In order to find the chattiest users we need to match all users with a createchat edge, return the users id and count them in descending order:

```
match p=(u:User)-[:CreateChat]->()
return u.id, count(u) as cnt
order by cnt desc
limit 10
```

Chattiest Users

Users	Number of Chats
394	115
2067	111
1087	109

In order to find the chattiest teams, we need to match all chatitems with a PartOf connecting to a TeamChatSession node and also the TeamChatSession nodes must have an OwnedBy edge connecting them with any other node:

```
match p=(i:ChatItem)-[:PartOf]->(c:TeamChatSession)
with p, Extract(x IN NODES(p) | x.id) as cSessionId
match (c:TeamChatSession)-[:OwnedBy]->(n)
where c.id in cSessionId
return n.id as TeamId, count(c) as cnt order by cnt desc limit 10
```

Chattiest Teams

Teams	Number of Chats
82	1324
185	1036
112	957

To find whether any of the chattiest users are part of any of the chattiest teams, the following query is used:

```
match p=(u:User)-[:CreateChat]->(i:ChatItem)-[:PartOf]->(c:TeamChatSession)-[:OwnedBy]->(t:Team)
return u.id as UserId, t.id as TeamId, count(u) as cnt
order by cnt desc
limit 10
```

The result shows that user 394 is part of team 63, users 2067 and 209 are both members of team 7 and user 1087 is a member of team 77, therefore none of the top 3 chattiest users are part of any of the top 3 chattiest teams. There is one top 10 user though which is member of one of the top 10 chattiest team, user 999 – team 52.

This kind of analysis is important because we can find users that are more influential than others and teams and users more active on the game.

How Active Are Groups of Users?

Describe your steps for performing this analysis. Be as clear, concise, and as brief as possible. Finally, report the top 3 most active users in the table below.

- a) The first step is to create a neighborhood of users. A user will be a member of the neighborhood if it has an “InteractsWith” relationship with any other user in the neighborhood.

There are two queries used to create this relationship because there is a Mentioned relationship and a ResponseTo relationship between users:

```
Match(u1:User)-[:CreateChat]->(:ChatItem)-[:Mentioned]->(u2:User)
create((u1)-[:InteractsWith]->(u2))
```

```
Match(u1:User)-[:CreateChat]->(:ChatItem)-[:ResponseTo]->(:ChatItem)-[:CreateChat]-
(u2:User)
create((u1)-[:InteractsWith]->(u2))
```

- b) Given the scheme above, it is possible for a user to interact with his/her own chat items. To eliminate all self-loops, the following query is used:

```
$ Match(u1)-[:InteractsWith]->(u1) Delete(r)
```

Deleted 4377 relationships, statement completed in 3360 ms.

- c) The third step is to calculate the “clustering coefficient” which ranges from 0 (disconnected user) to 1 (user in a clique – every node is connected to every other node). The “clustering coefficient” is the ratio $\frac{n}{k \cdot (k-1)}$, where n is the number of edges among the neighbors (not counting the given node) and the denominator represent all the pairwise edges that could possibly exist.

The query used for this task is the following:

```
Match(u1:User)-[:InteractsWith]->(u2:User)
Where u1.id in [394,2067,209,1087,554,516,1627,999,668,461]
With u1 AS user, COLLECT(DISTINCT(u2.id)) AS neighbors
Match(u1:User),(u2:User)
Where u1.id in neighbors AND u2.id in neighbors
Return user.id, TOFLOAT(SUM(CASE WHEN (u1)--(u2) Then 1 Else 0
End))/(Size(neighbors)*(Size(neighbors)-1)) as coefficient
Order BY coefficient Desc
```

Most Active Users (based on Cluster Coefficients)

User ID	Coefficient
394	1
209	0.9523809523809523
554	0.9047619047619048
1087	0.8

User 394 is the most active of the top three chattiest players, based on cluster coefficients.

Recommended Actions

- Considering the data exploration and classification analysis, Eglence Inc should target the game towards iPhone platform users since such users are among the top spenders and also are more likely to spend more money than other platform users
- Also, considering clustering analysis, Eglence Inc should increase the number of ads, and provide bonuses for ad clicking to lower hit success users, since ad clicking and revenue are directly correlated but low success users spend less than the rest.