

# Technical Appendix

## Catch the Pink Flamingo Analysis

Produced by: Natalya Patrikeeva

### 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	A line is added to this file when a player clicks on an advertisement in the Flamingo app.	timestamp: when the click occurred. txID: a unique id (within adclicks.log) for the click. userSessionid: the id of the user session for the user who made the click. teamid: the current team id of the user who made the click. userid: the user id of the user who made the click. adID: the id of the ad clicked on adCategory: the category/type of ad clicked on.
buy-clicks.csv	A line is added to this file when a player makes an inapp purchase in the Flamingo app.	timestamp: when the purchase was made. txID: a unique id (within buyclicks.log) for the purchase. userSessionid: the id of the user session for the user who made the purchase. team: the current team id of the user who made the purchase. userId: the user id of the user who made the purchase. buyId: the id of the item purchased price: the price of the item.

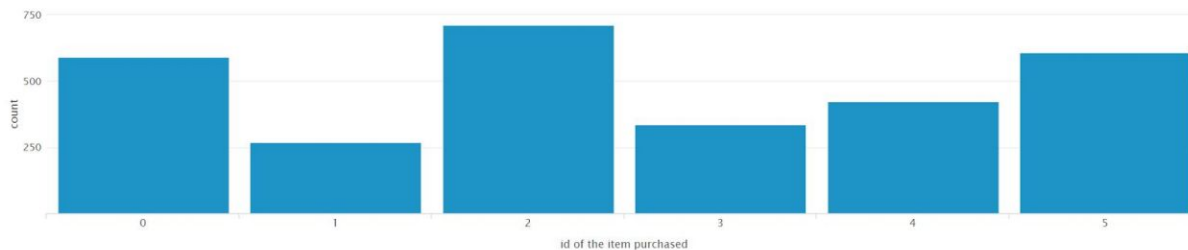
		purchased
users.csv	This file contains a line for each user playing the game.	timestamp: when user first played the game. id: the user id assigned to the user. nick: the nickname chosen by the user. twitter: the twitter handle of the user. dob: the date of birth of the user. country: the two letter country code where the user lives.
team.csv	This file contains a line for each team terminated in the game.	teamid: the id of the team. name: the name of the team. teamCreationTime: the timestamp when the team was created. teamEndTime: the timestamp when the last member left the team. strength: a measure of team strength, roughly corresponding to the success of a team. currentLevel: the current level of the team.
team-assignments.csv	A line is added to this file each time a user joins a team. A user can be in at most a single team at a time.	time: when the user joined the team. team: the id of the team. userid: the id of the user. assignmentid: a unique id for this assignment.
level-events.csv	A line is added to this file each time a team starts or finishes a level in the game.	time: when the event occurred. eventid: a unique id for the event. teamid: the id of the team level: the level started or completed. eventType: the type of event, either start or end.
user-session.csv	Each line in this file describes a user session, which denotes when a user starts and stops playing the game. Additionally, when a team goes to the next	timeStamp: a timestamp denoting when the event occurred. userSessionId: a unique id for the session. userId: the current user's ID.

	level in the game, the session is ended for each user in the team and a new one started.	<p>teamId: the current user's team.</p> <p>assignmentId: the team assignment id for the user to the team.</p> <p>sessionType: whether the event is the start or end of a session.</p> <p>teamLevel: the level of the team during this session.</p> <p>platformType: the type of platform of the user during this session.</p>
game-clicks.csv	A line is added to this file each time a user performs a click in the game.	<p>time: when the click occurred.</p> <p>clickid: a unique id for the click.</p> <p>userid: the id of the user performing the click.</p> <p>usersessionid: the id of the session of the user when the click is performed.</p> <p>isHit: denotes if the click was on a flamingo (value is 1) or missed the flamingo (value is 0).</p> <p>teamId: the id of the team of the user.</p> <p>teamLevel: the current level of the team of the user.</p>

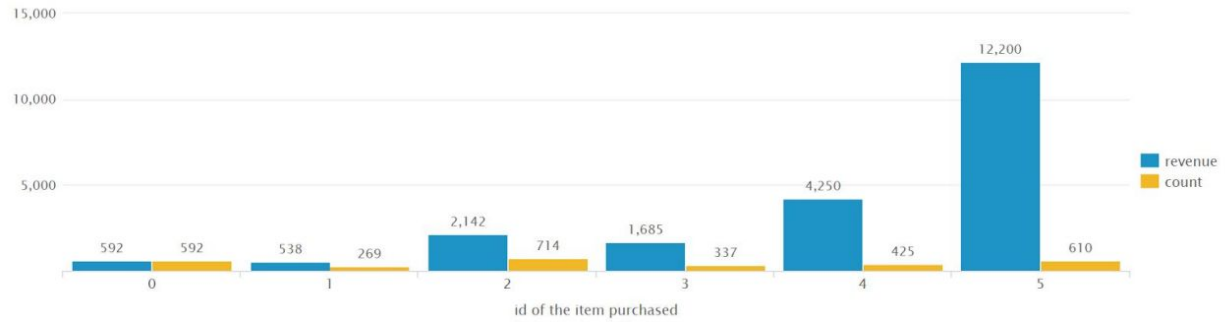
## Aggregation

Amount spent buying items	21407
# Unique items available to be purchased	6

A histogram showing how many times each item is purchased:

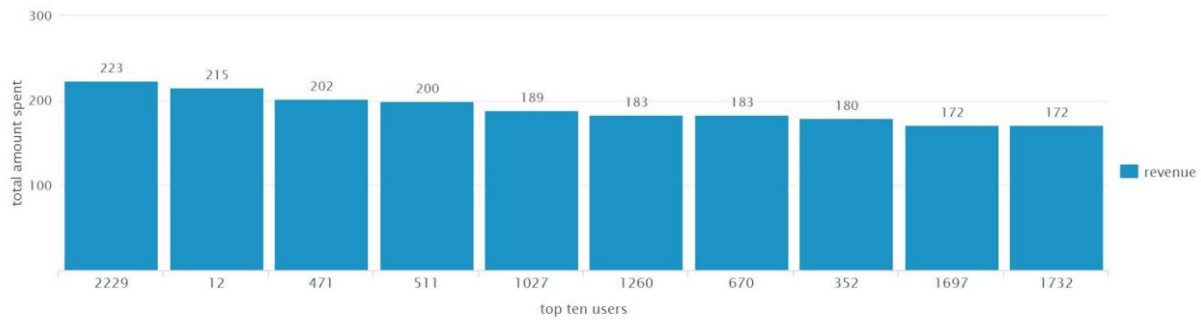


A histogram showing how much money was made from each item:



## Filtering

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



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

Rank	User Id	Platform	Hit-Ratio (%)
1	2229	iphone	0.115970
2	12	iphone	0.130682
3	471	iphone	0.145038

# Data Classification Analysis

## Data Preparation

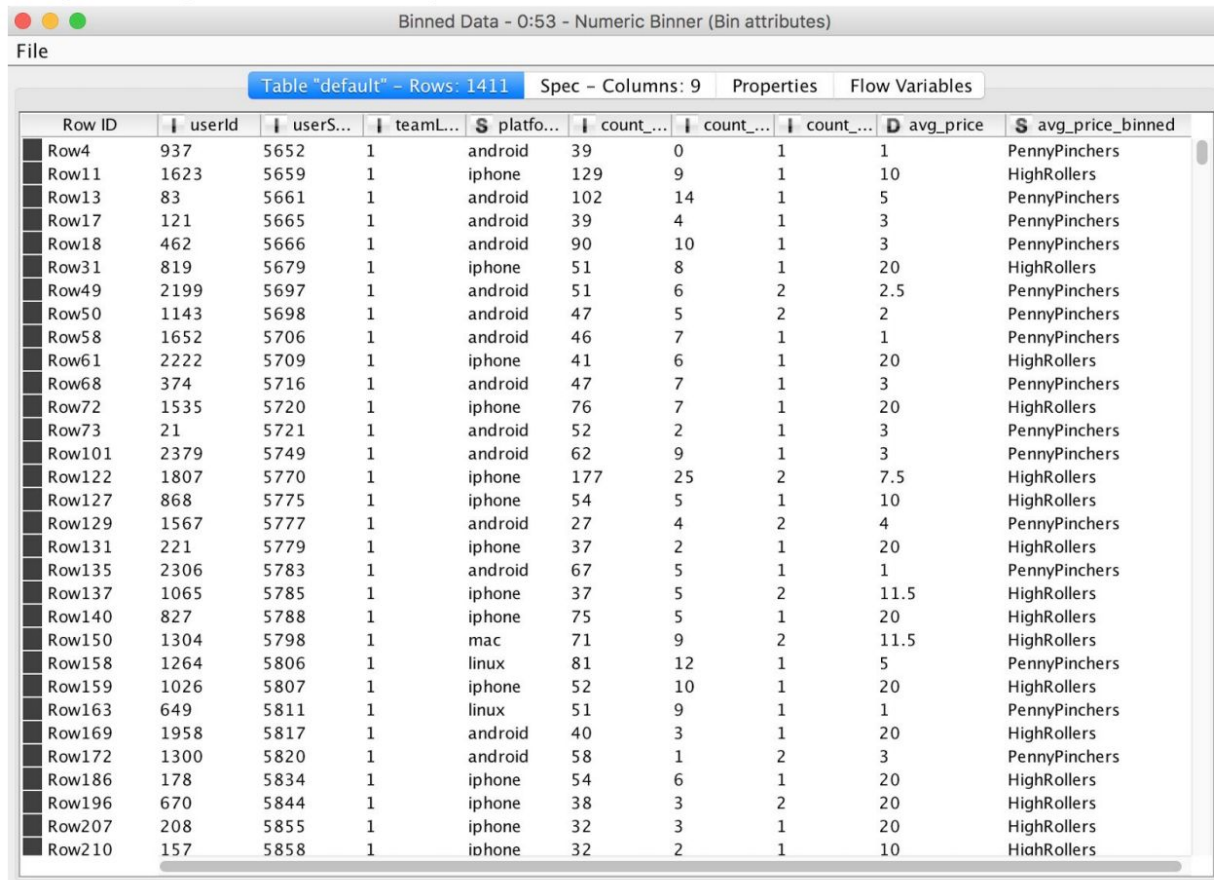
Analysis of combined\_data.csv

### Sample Selection

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

### Attribute Creation

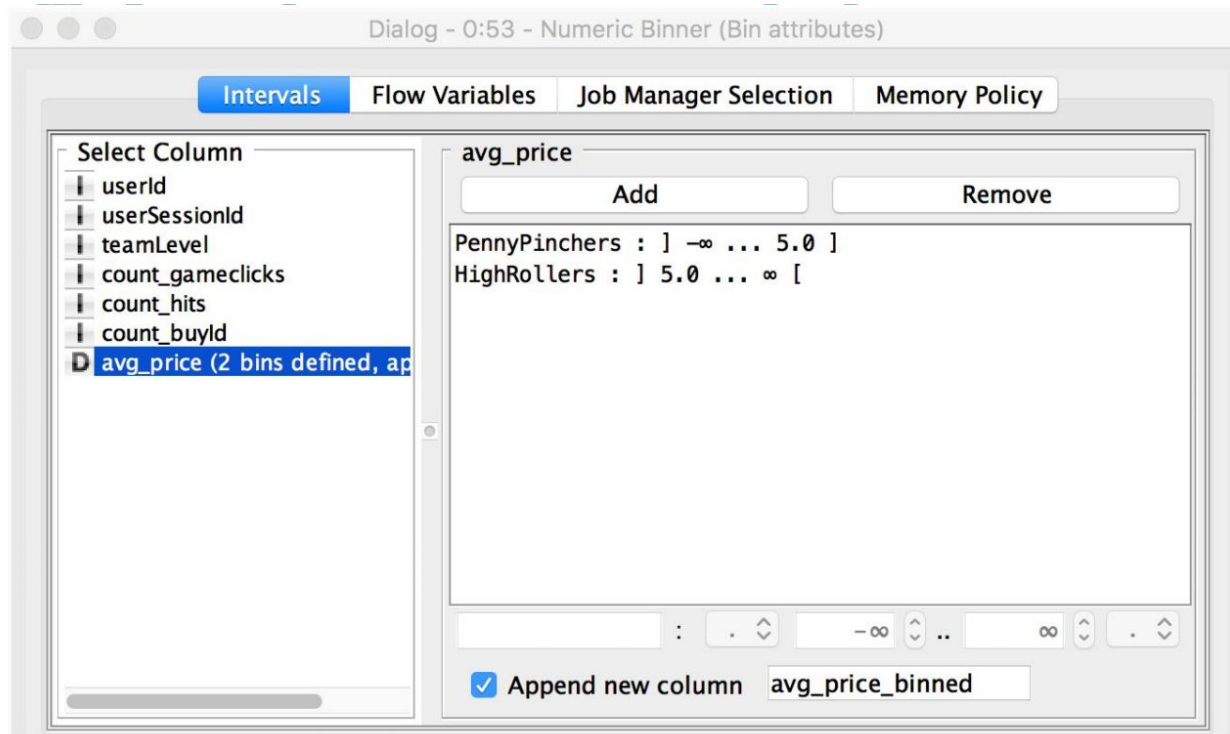
A new categorical attribute was created to enable analysis of players as broken into 2 categories (HighRollers and PennyPinchers). A screenshot of the attribute follows:



The screenshot shows a data table titled "Binned Data - 0:53 - Numeric Binner (Bin attributes)". The table has 10 columns: Row ID, user ID, user score, team level, platform, count, count, count, average price, and average price binned. The data is sorted by average price binned, showing two categories: PennyPinchers and HighRollers. The table is displayed in a software interface with a menu bar (File, Table, Spec, Properties, Flow Variables) and a toolbar.

Row ID	user ID	user score	team level	platform	count	count	count	avg_price	avg_price_binned
Row4	937	5652	1	android	39	0	1	1	PennyPinchers
Row11	1623	5659	1	iphone	129	9	1	10	HighRollers
Row13	83	5661	1	android	102	14	1	5	PennyPinchers
Row17	121	5665	1	android	39	4	1	3	PennyPinchers
Row18	462	5666	1	android	90	10	1	3	PennyPinchers
Row31	819	5679	1	iphone	51	8	1	20	HighRollers
Row49	2199	5697	1	android	51	6	2	2.5	PennyPinchers
Row50	1143	5698	1	android	47	5	2	2	PennyPinchers
Row58	1652	5706	1	android	46	7	1	1	PennyPinchers
Row61	2222	5709	1	iphone	41	6	1	20	HighRollers
Row68	374	5716	1	android	47	7	1	3	PennyPinchers
Row72	1535	5720	1	iphone	76	7	1	20	HighRollers
Row73	21	5721	1	android	52	2	1	3	PennyPinchers
Row101	2379	5749	1	android	62	9	1	3	PennyPinchers
Row122	1807	5770	1	iphone	177	25	2	7.5	HighRollers
Row127	868	5775	1	iphone	54	5	1	10	HighRollers
Row129	1567	5777	1	android	27	4	2	4	PennyPinchers
Row131	221	5779	1	iphone	37	2	1	20	HighRollers
Row135	2306	5783	1	android	67	5	1	1	PennyPinchers
Row137	1065	5785	1	iphone	37	5	2	11.5	HighRollers
Row140	827	5788	1	iphone	75	5	1	20	HighRollers
Row150	1304	5798	1	mac	71	9	2	11.5	HighRollers
Row158	1264	5806	1	linux	81	12	1	5	PennyPinchers
Row159	1026	5807	1	iphone	52	10	1	20	HighRollers
Row163	649	5811	1	linux	51	9	1	1	PennyPinchers
Row169	1958	5817	1	android	40	3	1	20	HighRollers
Row172	1300	5820	1	android	58	1	2	3	PennyPinchers
Row186	178	5834	1	iphone	54	6	1	20	HighRollers
Row196	670	5844	1	iphone	38	3	2	20	HighRollers
Row207	208	5855	1	iphone	32	3	1	20	HighRollers
Row210	157	5858	1	iphone	32	2	1	10	HighRollers

I bin the data into two categories based on average purchase price for user session. PennyPinchers are buyers of items that cost \$5.00 or less, while HighRollers are buyers of items that cost more than \$5.00. I use existing attribute avg\_price to create a new categorical attribute avg\_price\_binned using Numeric Binner node with the following configuration.



The creation of this new categorical attribute was necessary because our prediction task is to classify users into one of these two categories, PennyPinchers or HighRollers using available data. First, I needed to bin the data into these categories before I can build a decision tree model to classify users.

### Attribute Selection

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

Attribute	Rationale for Filtering
userId	User ID is a unique ID for the current user playing the game, so I exclude it because it is a random number and will not be predictive of which user is likely to purchase big-ticket items.
userSessionId	User session ID is a unique ID for the session when a user starts and stops playing the game, so I exclude it because it is a random number and will not be predictive of which user is likely to purchase big-ticket items.
avg_price	We exclude the average price because we used this attribute to

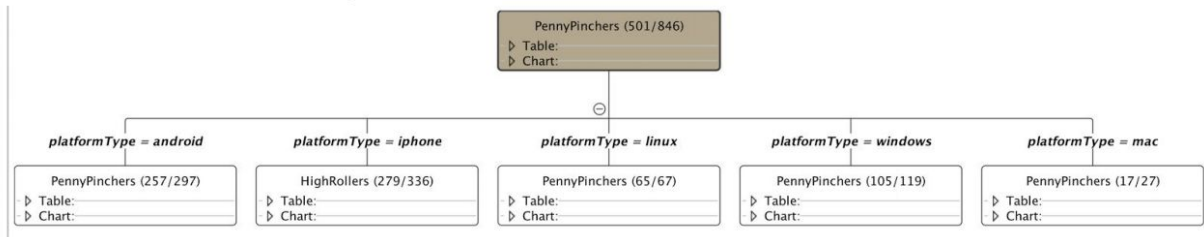
	generate the categorical attribute avg_price_binned that our model is going to predict so it cannot be the used in the model.
--	---

## Data Partitioning and Modeling

The data was partitioned into train and test datasets.  
The training data set was used to create the decision tree model.  
The trained model was then applied to the test dataset.  
This is important because we want to test our model and predictions on unseen data so we split the input dataset into training and testing set. We use training dataset to create a model and a test dataset to test the model.

When partitioning the data using sampling, it is important to set the random seed because we want to be able to reproduce our results. Setting a random seed allows us to split the data into the same partitions every time the Partitioning node is executed.

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



## Evaluation

A screenshot of the confusion matrix can be seen below:

Confusion Matrix - 0:65 - Scorer (Compute confusion matrix)			
File Hilite			
avg_price_binned \ Prediction (avg_price_binned)		PennyPinchers	HighRollers
PennyPinchers		308	27
HighRollers		38	192
Correct classified: 500		Wrong classified: 65	
Accuracy: 88.496 %		Error: 11.504 %	
Cohen's kappa (κ) 0.76			

As seen in the screenshot above, the overall accuracy of the model is 88.496%.

The columns of the confusion matrix show the predictions the model made (“PennyPinchers” and “HighRollers”) and the rows of the confusion matrix show the actual values in the test data. Top left value of 308 is true positives, the number the model correctly predicted the true value of “PennyPinchers” class.

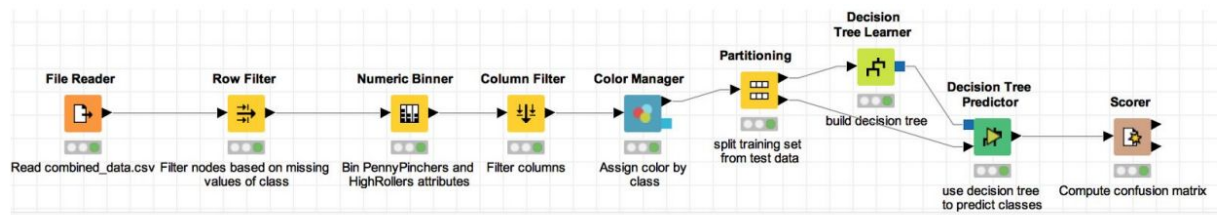
Bottom left value of 38 is false positive value, which is the number the model incorrectly predicted the classification to be “PennyPinchers” when the value was really “HighRollers”.

Top right value of 27 is false negatives, which is incorrectly predicted “HighRollers” but the actual value in the test dataset was “PennyPinchers”.

Bottom right value of 192 is true negatives, which is correctly predicted “HighRollers” class.

## Analysis Conclusions

The final KNIME workflow is shown below:



What makes a HighRoller?

The main attribute used to make classification decision is platformType. If the user platform is iPhone, then the model predicts “HighRollers”. For the platform types Android, Linux, Windows and Mac, the model predicts “PennyPinchers” class.

Specific Recommendations to Increase Revenue
1. We can advertise in-app more expensive items to iPhone users since they tend to purchase big-ticket items based on our model and advertise inexpensive items to users of other platforms.
2. We can upcharge iPhone in-app purchases to increase revenue because iphone users tend to purchase big items of than \$5.00. So if we make inexpensive items that usually cost less, cost more in the app for iphone users, they are more likely to purchase these items for more.



# Clustering Analysis

## Attribute Selection

Attribute	Rationale for Selection
revenue	This attribute is the total revenue per user. Sum of money spent by each user in the game can help us cluster users who spent a lot of money on the game and generate big revenue and users who do not.
AverageHitRate	Average hit rate per user can help us to cluster users who are good at the game and users who are not.
TotalClicks	Total number of clicks for each user in the game can help us to cluster users who are prone to clicking while playing the game and users who are not.

## Training Data Set Creation

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

```
In [63]: trainingDF.head(n=5)
Out[63]:
```

	revenue	AverageHitRate	TotalClicks
0	21.0	0.134078	716
1	53.0	0.100000	380
2	80.0	0.122047	508
3	11.0	0.109430	3107
4	215.0	0.130682	704

Dimensions of the training data set (rows x columns) : (546, 3)

# of clusters created: 4

## Cluster Centers

Screenshot of the cluster centers for k=4 K-mean model:

```
In [84]: print(kmmodel_4.centers)
[array([ 3.02040000e+01,  1.13652025e-01,  2.93292000e+02]),
 array([ 5.17766990e+01,  1.15574474e-01,  7.32616505e+02]),
 array([ 3.39230769e+01,  1.12722679e-01,  1.40701538e+03]),
 array([ 3.94000000e+01,  1.12110561e-01,  2.73748000e+03])]
```

Converting to a more human readable format, these cluster centers are:

Cluster #	Cluster Center (revenue, average hit rate, total clicks per user)
1	array([30.20, 0.11365, 293.292])
2	array([51.78, 0.115574, 732.62])
3	array([33.92, 0.11272, 1407.02])
4	array([39.40, 0.11211, 2737.48])

These clusters can be differentiated from each other as follows:

Cluster 1 is different from the others in that it has the lowest average revenue and the lowest average total click count per user. It clusters users who spend the least amount of money in the game and click the least while playing the game.

Cluster 2 is different from the others in that it has the highest average revenue and the highest average hit rate per user. It clusters users who generate the most revenue and are the most accurate players in the game.

Cluster 3 is different from the others in that it does not have any extreme values of the cluster centers. All the attributes revenue, average hit rate or total clicks per user has the average values for the center. It has lower total clicks than Cluster 4 but higher total clicks than Cluster 1 and 2. Similarly, its center revenue value is lower than Cluster 2 and Cluster 4 but higher than Cluster 1.

Cluster 4 is different from the others in that it has the highest total clicks per user. It has the lowest average hit rate. Also, it has the second highest average revenue after Cluster 2. It clusters users who click the most in the game and also are the least accurate players.

## Recommended Actions

Action Recommended	Rationale for the action
Increase ad prices shown to Cluster 2.	Cluster 2 players generate the most revenue from ads so we can increase revenue by increasing prices of ads shown to this group.
Increase number of ads shown to Cluster 4.	Cluster 4 players are the most frequent clickers while playing the game and generate the second highest revenue so we can increase revenue by increasing the frequency of ads shown to this group.

# Graph Analytics Analysis

## Modeling Chat Data using a Graph Data Model

The graph model for chat data contains data about chatting patterns among users and teams in the game. User and Team nodes designate users and teams in the game, respectively.

TeamChatSession nodes designate a particular chat session of the team, while ChatItem nodes designate a particular chat sent by a user. Relationships among nodes include CreatesSession edge from User to Team node, OwnedBy edge from TeamChatSession node to Team node, Joins and Leaves edges from User node to TeamChatSession node, CreateChat relationship from User node to ChatItem node, PartOf edge from ChatItem node to TeamChatSession node, Mentioned edge from ChatItem node to User node, and ResponseTo edge from a ChatItem node to another ChatItem node. All edges have a timeStamp property.

## Creation of the Graph Database for Chats

The schema of the 6 chat datasets are:

(1) chat\_create\_team\_chat.csv

UserId, TeamId, TeamChatSessionId, timeStamp

(2) chat\_join\_team\_chat.csv

UserId, TeamChatSessionId, timeStamp

(3) chat\_leave\_team\_chat.csv

UserId, TeamChatSessionId, timeStamp

(4) chat\_item\_team\_chat.csv

UserId, TeamChatSessionId, ChatItemId, timeStamp

(5) chat\_mention\_team\_chat.csv

ChatItemId, UserId, timeStamp

(6) chat\_respond\_team\_chat.csv

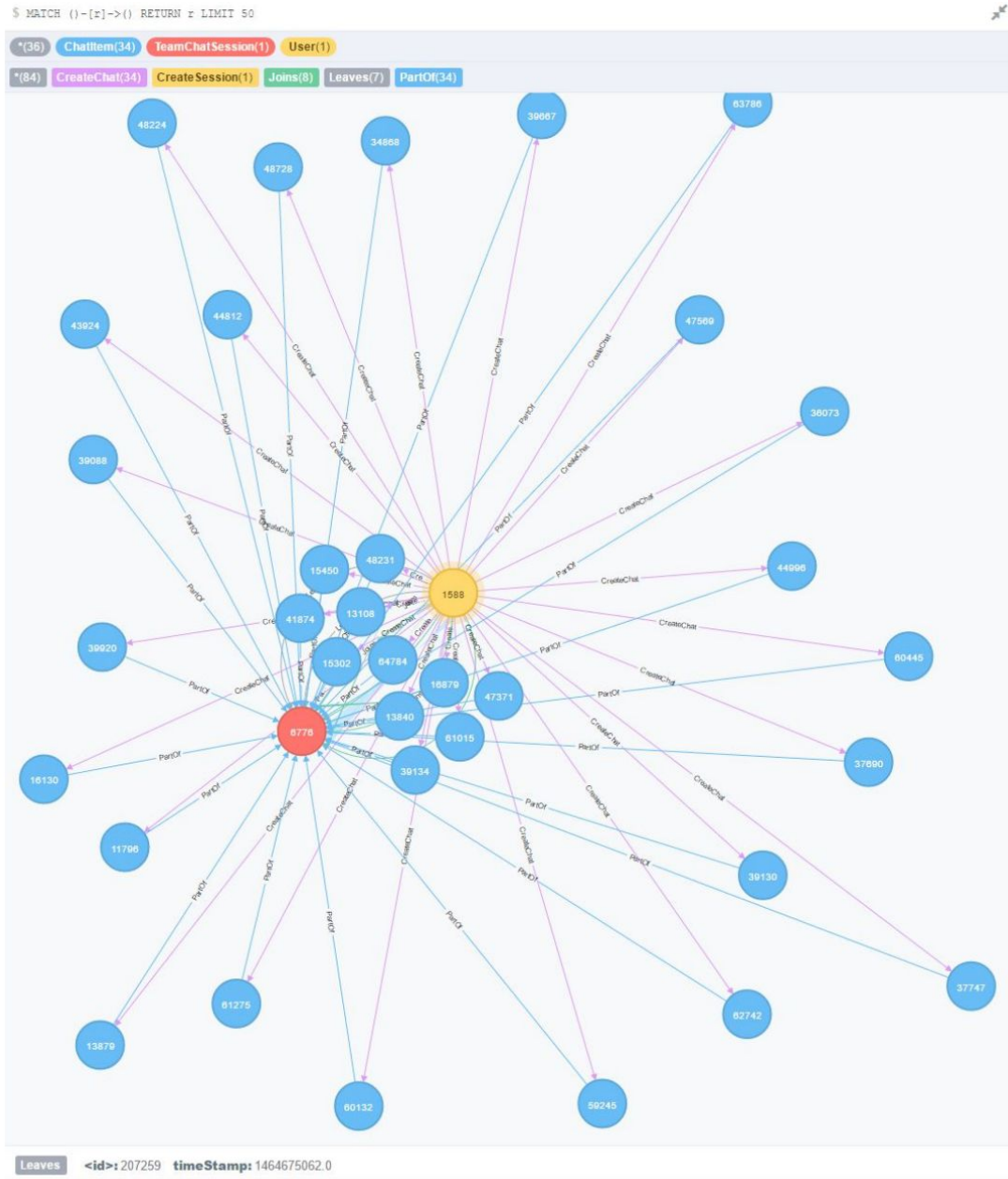
ChatItemId, ChatItemId, timeStamp

We load the data into neo4j using LOAD CSV FROM command, creating nodes and relationships between nodes and their properties. For example, to load chat\_join\_team\_chat.csv, I execute

the following:

```
# (2) chat_join_team_chat.csv
# UserId, TeamChatSessionId, timeStamp
# 1588,6776,146423999.0
# 350,6777,146423400.0
LOAD CSV FROM "file:///C:/Users/Natty/Dropbox/big_data_capstone/big_data_capstone_datasets_and_scripts/chat-data/chat_join_team_chat.csv" AS row
MERGE (u:User {id: toInt(row[0])}) # create nodes User u
MERGE (c:TeamChatSession {id: toInt(row[1])}) # create nodes TeamChatSession c
MERGE (u)-[:Joins{timeStamp: row[2]}->(c) # create edge Joins from u to c with property timeStamp
```

A screenshot of a part of the graph including most node and edge types:



## Finding the longest conversation chain and its participants

Path length is 9 with 10 nodes with 5 users participating in it. We want to find all ChatItem nodes connected by the edge ResponseTo. The query is shown below.

```
match p=(i)-[:ResponseTo*]->(n)
return p,length(p)
order by length(p) desc
limit 1
# 10 chatitems, length is 9
```

```
match p=(i)-[:ResponseTo*]->(n) where length(p)=9
with p
match (u:User)-[:CreateChat]->(i)
where i in nodes(p)
return count(distinct u)
# 5 user nodes
```

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

First, I query users that connect to other user nodes with a CreateChat edge. We can find the chattiest user by counting the outdegree of the node, ordering the results in descending order and limiting to top results.

```
match (u:User)-[:CreateChat*]-()
return u, count(r) as Outdegree
order by Outdegree desc
limit 10
```

### Chattiest Users

Users	Number of Chats
394	115
2067	111
1087 or 209	109

Second, I query teams that connect to TeamChatSession nodes that in order connect to ChatItem nodes. To find the chattiest teams, we need to compute the indegree of the team nodes, order the results in descending order and limit the top results.

```
# 2) identify the top 10 chattiest teams.
match (i:ChatItem)-[r:PartOf*]-(c:TeamChatSession)-[q:OwnedBy*]-(t:Team)
return t, count(q) as Indegree
order by Indegree desc
limit 10
```

### Chattiest Teams

Teams	Number of Chats
82	1324
185	1036
112	957

To analyze if top chattiest users belong to chattiest teams, I queried team nodes that the top chattiest user belonged to and see if these teams matched the chattiest teams with the following command (and changing the userId).

```
MATCH (u:User {id:394})-[r]->(c:TeamChatSession)-[q]->(t:Team)
RETURN distinct t      # return r,q to visualize the path
# user 394 is in team 63 and is not in the top 10 chattiest teams
# user 2067 is in team 7 and is not in the top 10 chattiest teams
# user 1087 is in team 77 and is not in the top 10 chattiest teams
# user 209 is in team 7 and is not in the top 10 chattiest teams
# user 554 is in team 181 and is not in the top 10 chattiest teams
# user 1627 is in team 7 and is not in the top 10 chattiest teams
# user 516 is in team 7 and is not in the top 10 chattiest teams
#--->> user 999 is in team 52 and is in the top 10 chattiest teams
# user 668 is in team 89 and is not in the top 10 chattiest teams
# user 461 is in team 104 and is not in the top 10 chattiest teams
```

From the above, only user 999 is on team 52 who is both the chattiest user and is part of the chattiest team.

### How Active Are Groups of Users?

We create a new edge called InteractsWith between two User nodes when users respond to a chat or mention another user in a chat. Then we eliminate selfloops of InteractsWith edges for users.



```

MATCH (u1:User)-[:CreateChat]-(i:ChatItem)-[:Mentioned]->(u2:User)
CREATE (u1)-[:InteractsWith]->(u2)
# makes 11,084 relationships

MATCH p=(u2:User)-[:CreateChat]-(n:ChatItem)
MATCH (u1:User)-[:CreateChat]-(i:ChatItem)-[:ResponseTo]-(n:ChatItem)
CREATE (u1)-[:InteractsWith]->(u2)
# makes 22,146 relationships

# (b) eliminate self-loops for Users
MATCH (u1:User)-[r:InteractsWith]->(u1) delete r
# deleted 8,754 connections

```

Finally, to find users with dense neighborhoods, we find a neighbors of a node and compute its clustering coefficient based which is defined as the ratio between the number of edges amongst the node's neighbors and all possible pairs of edges.

The code below shows the commands to find user node's neighbors, then neighbor's neighbors for top 10 most chattiest users and tabulates the clustering coefficients.



```

MATCH (u:User {id:394})-[r:InteractsWith]->(m)
RETURN distinct m order by m.id asc

# neighbor's neighbors
MATCH (u:User {id:1782})-[:InteractsWith]->(b:User)
with collect(distinct b.id) as neighbors match (n)-[r:InteractsWith]->(m)
where n.id in (neighbors) and m.id in (neighbors)
return distinct n order by n.id asc

```

\$ match (u)-[r:InteractsWith]->(v) where u.id in [394, 2067, 1087, 209...

	user	clusteringcoeff
Rows	461	1
Code	668	1
	516	0.9523809523809523
	209	0.9523809523809523
	394	0.9166666666666666
	999	0.8194444444444444
	554	0.8095238095238095
	2067	0.7678571428571429
	1627	0.7678571428571429
	1087	0.7666666666666667
	Returned 10 rows in 3720 ms.	

### Most Active Users (based on Cluster Coefficients)

User ID	Coefficient
461	1
668	1
516	0.95238

## Recommended Actions

I recommend to advertise in-app more expensive items to iphone users since they tend to purchase big-ticket items and advertise inexpensive items to users of other platforms. The decision tree model accuracy was high and the classification result is easy to interpret as opposed to K-means modeling when we have to choose the number of clusters or chat data which is harder to interpret and needs a real-time analysis.