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 ad-clicks.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 in-app purchase in the Flamingo app.	➢ timestamp: when the purchase was made. ➢ txId: a unique id (within buy-clicks.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
game-clicks.csv	A line is added to this file each time a user performs a click in the game.	➢ timestamp: when the click occurred. ➢ clickId: a unique id for the click.

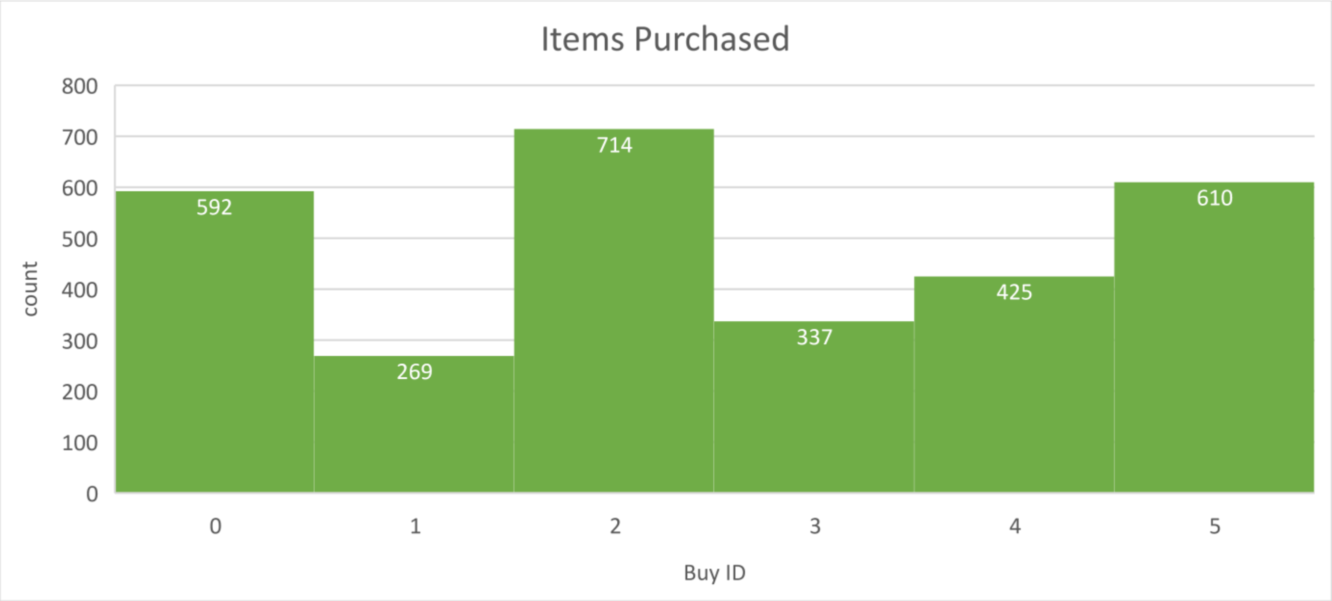
		➢ userId: the id of the user performing the click. ➢ userSessionId: the id of the session of the user when the click is performed. ➢ isHit: denotes if the click was on a flamingo (value is 1) or missed the flamingo (value is 0) ➢ teamId: the id of the team of the user ➢ teamLevel: the current level of the team of the user
level-events.csv	A line is added to this file each time a team starts or finishes a level in the game.	➢ timestamp: when the event occurred. ➢ eventId: a unique id for the event ➢ teamId: the id of the team ➢ teamLevel: the level started or completed ➢ eventType: the type of event, either start or end
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.	➢ timestamp: 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
users.csv	This file contains a line for each user playing the game.	➢ timestamp: when user first played the game.

		➢ userId: 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.
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 level in the game, the session is ended for each user in the team and a new one started.	➢ timestamp: a timestamp denoting when the event occurred. ➢ userSessionId: a unique id for the session. ➢ userId: the current user's ID. ➢ teamId: the current user's team. ➢ assignmentId: the team assignment id for the user to the team. ➢ sessionType: whether the event is the start or end of a session. ➢ teamLevel: the level of the team during this session. ➢ platformType: the type of platform of the user during this session.

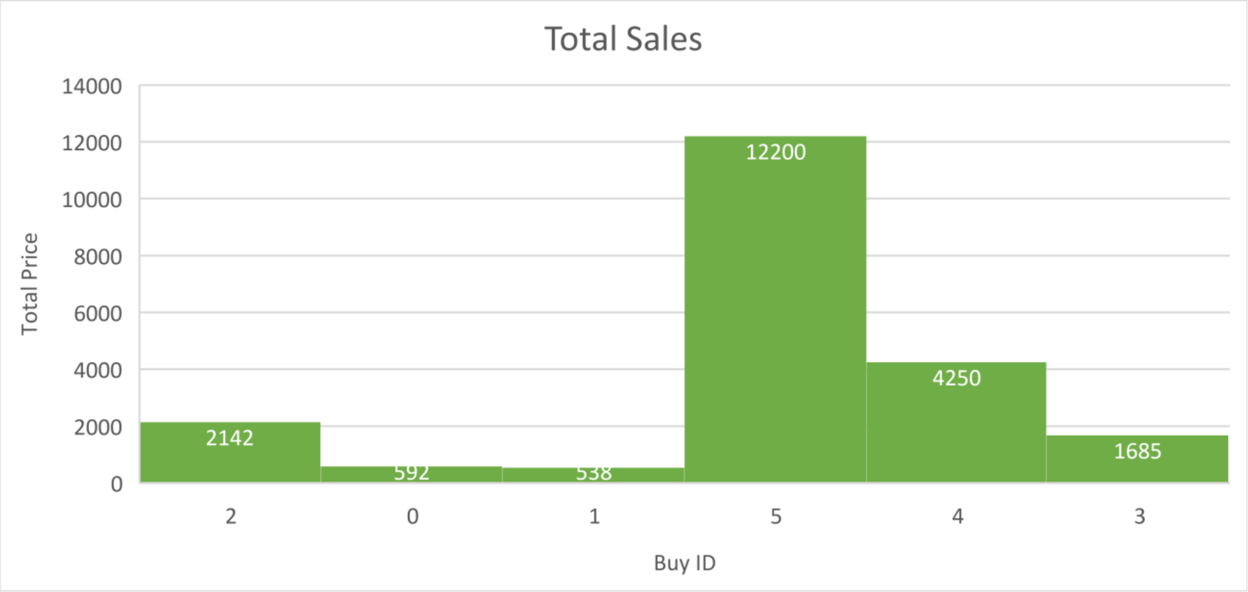
Aggregation

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

A histogram showing how many times each item is purchased:

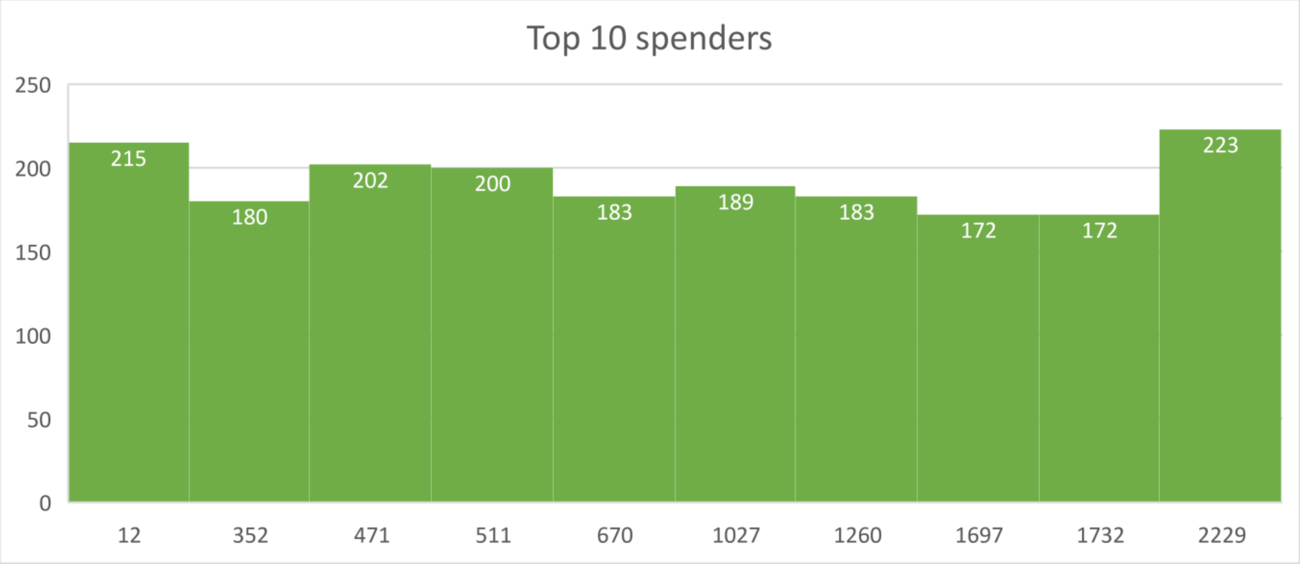


A histogram showing how much money was made from each item:



Filtering

A histogram showing total amount of money spent by the top ten users (ranked by how much money they spent).



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	229	iphone	11.60
2	12	iphone	13.07
3	471	iphone	14.50

Data Preparation

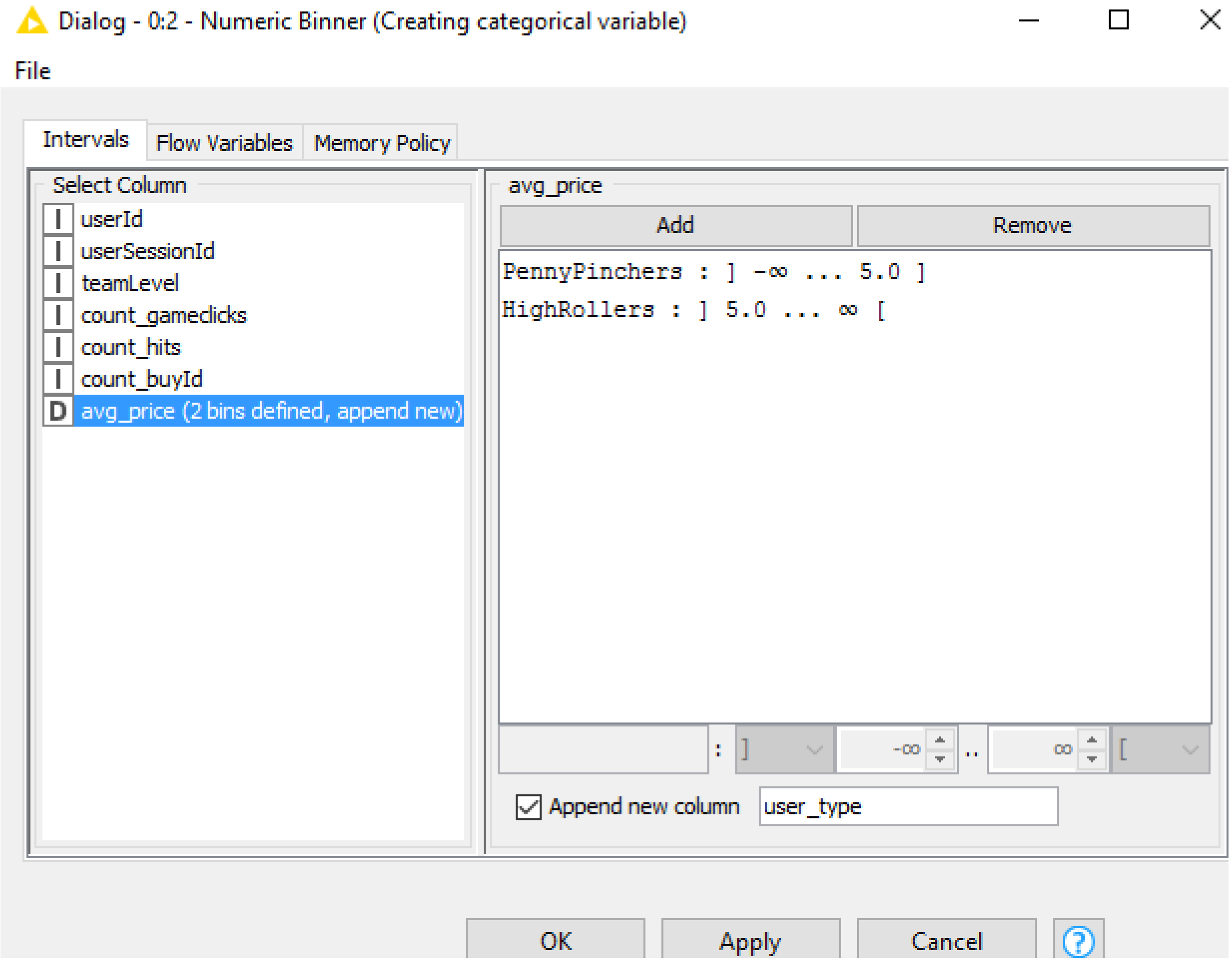
Analysis of combined_data.csv

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:



Dialog - 0:2 - Numeric Binner (Creating categorical variable)

Select Column

- I userId
- I userSessionId
- I teamLevel
- I count_gameclicks
- I count_hits
- I count_buyId
- D avg_price (2 bins defined, append new)

avg_price

```
PennyPinchers : ] -∞ ... 5.0 ]
HighRollers : ] 5.0 ... ∞ [
```

:] -∞ .. ∞ [

☑ Append new column user_type

The creation of this new categorical attribute was necessary because it helps to categorize HighRollers(more than $5.00) and PennyPinchers(less than or equal to $5.00)

Attribute Selection

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

Attribute	**Rationale for Filtering**
userId	This attribute can be removed because our target is predicting which user is likely to purchase big-ticket items and this attribute is unnecessary
userSessionId	The userSession id attribute has no use in our analysis so it can be removed because our target is predicting which user is likely to purchase big-ticket items
avg_price	The creation of new attribute 'user_type' makes the attribute 'avg_price' unnecessary so we can remove it.

Data Partitioning and Modeling

The data was partitioned into train and test datasets.

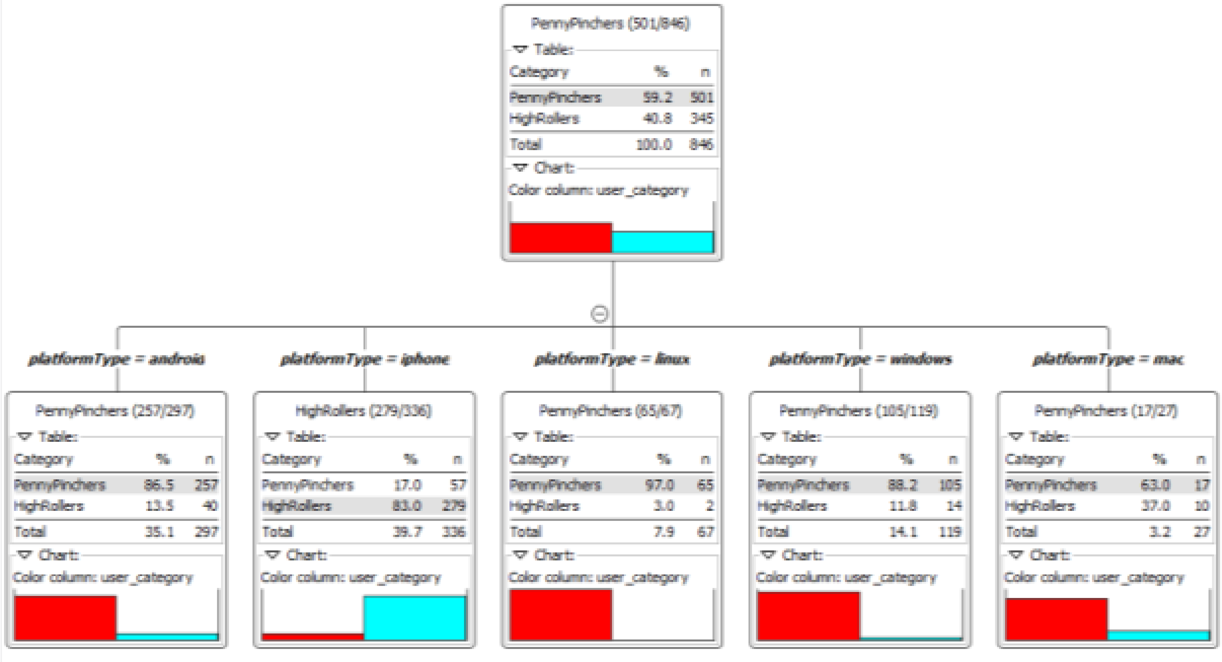
The train data set was used to create the decision tree model.

The trained model was then applied to the test dataset.

This is important because the train datasets helps in building the decision tree model and when the model is evaluated using test dataset we can obtain accuracy result.

When partitioning the data using sampling, it is important to set the random seed because to avoid getting same data partition everytime.

Decision Tree



PennyPinchers (501/846)

▽ Table:

Category	%	n
PennyPinchers	59.2	501
HighRollers	40.8	345
Total	100.0	846

▽ Chart:

Color column: user_category

platformType = androia

PennyPinchers (257/297)

▽ Table:

Category	%	n
PennyPinchers	86.5	257
HighRollers	13.5	40
Total	35.1	297

▽ Chart:

Color column: user_category

platformType = iphone

HighRollers (279/336)

▽ Table:

Category	%	n
PennyPinchers	17.0	57
HighRollers	83.0	279
Total	39.7	336

▽ Chart:

Color column: user_category

platformType = linux

PennyPinchers (65/67)

▽ Table:

Category	%	n
PennyPinchers	97.0	65
HighRollers	3.0	2
Total	7.9	67

▽ Chart:

Color column: user_category

platformType = windows

PennyPinchers (105/119)

▽ Table:

Category	%	n
PennyPinchers	88.2	105
HighRollers	11.8	14
Total	14.1	119

▽ Chart:

Color column: user_category

platformType = mac

PennyPinchers (17/27)

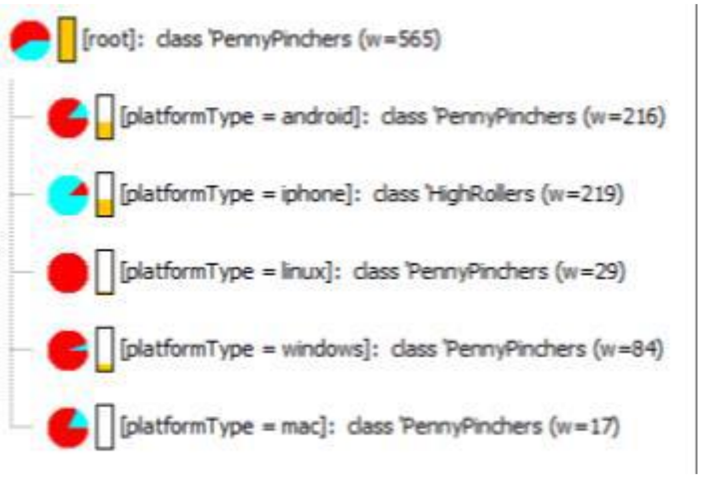
▽ Table:

Category	%	n
PennyPinchers	63.0	17
HighRollers	37.0	10
Total	3.2	27

▽ Chart:

Color column: user_category

Decision Tree(Simple)



[root]: class 'PennyPinchers (w=565)

[platformType = android]: class 'PennyPinchers (w=216)

[platformType = iphone]: class 'HighRollers (w=219)

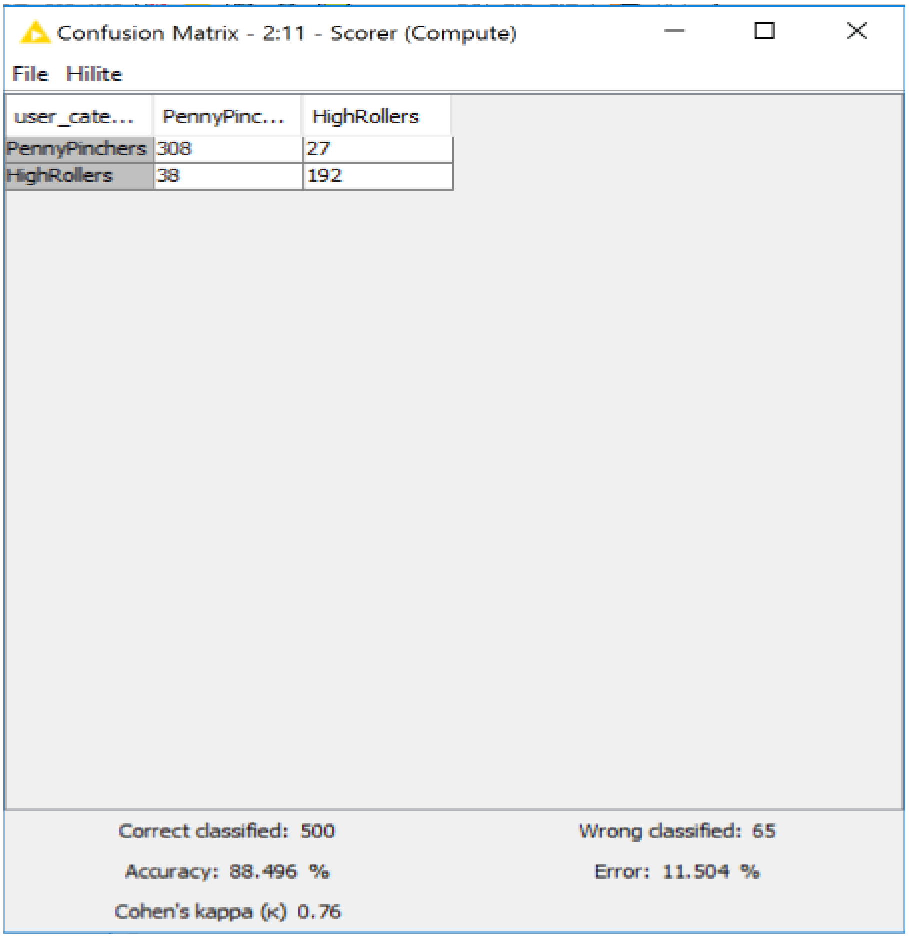
[platformType = linux]: class 'PennyPinchers (w=29)

[platformType = windows]: class 'PennyPinchers (w=84)

[platformType = mac]: class 'PennyPinchers (w=17)

Evaluation

A screenshot of the confusion matrix can be seen below:



user_cate...	PennyPinc...	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%

PennyPinchers:

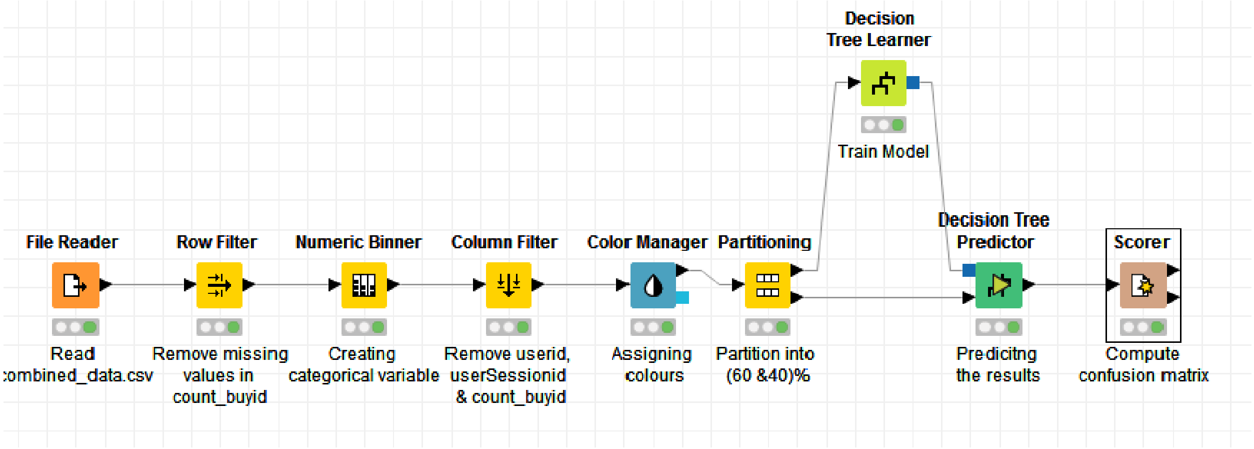
The model predicted that 346 users belong to the user_type PennyPinchers, out of which 308 users were correctly predicted which means that out of 346 , only 308 users are PennyPinchers and 38 HighRollers are wrongly predicted as PennyPinchers.

HighRollers:

The model predicted that 219 users belong to the user_type HighRollers, out of which 192 users were correctly predicted which means that out of 219 , only 192 users are HighRollers and 27 PennyPinchers are wrongly predicted as HighRollers.

Analysis Conclusions

The final KNIME workflow is shown below:



What makes a HighRoller vs. a PennyPincher?

The user_type who buy more items like the iphone users are HighRollers. The user_type who tend to not purchase or purchase small items like users on android, linux, windows, mac are called PennyPinchers.

Specific Recommendations to Increase Revenue
1. Iphone users are constant buyers, increase in money can be done , combo offers can be provided so that the users can purchase more items.
2. Provide offers to PennyPinchers to make them consider buying items, give away additional bonus items when they purchase an item.

Attribute Selection

Attribute	Rationale for Selection
Total number of ad-clicks per user	This attribute helps in determining user's behavior their interest based on the types of ads.
Game-clicks per hour by	This attribute helps in determining user's behavior based on clicking in

each user	the game.
Sum of money spent on ads by each user	This attribute helps in determining each user’s spending capacity which in turn helps us categorize the users.

Training Data Set Creation

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

- Read of the following files game-clicks.csv and buy-clicks.csv
- Feature Selection of the selected columns
  - user_purchases = buyclicks_df[['userId','price']]
  - user_hits = gameclicks_df[['userId','isHit’']]
- Sum the target columns per User Id
  - hits_per_user = user_hits.groupby('userId').sum()
  - revenue_per_user = user_purchases.groupby('userId').sum()
- Merge the datasets into one for the analysis
  - combined_df = hits_per_user.merge(revenue_per_user, on='userId')

Dimensions of the final data set:

training_df.shape

(543,3)

# of clusters created:3

Cluster Centers

Cluster #	Center
1	array([25.12037037, 362.50308642, 35.35802469])
2	array([32.05, 2393.95, 41.2])
3	array([36.47486034, 953.82122905, 46.16201117])

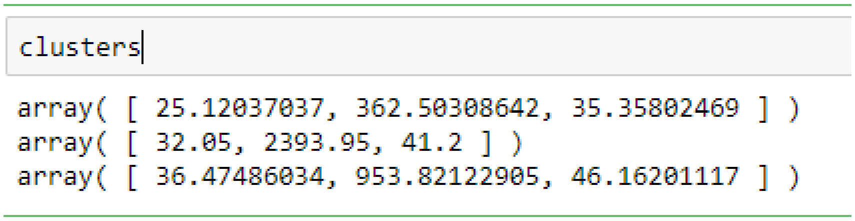
These clusters can be differentiated from each other as follows:

Cluster 1 is different from the others in that the gamers(users) in ad-clicks, game-clicks and their total spending are all less than others this kind of users can be called "**Less spending gamers**".

Cluster 2 is different from the others in that the gamers(users) ad-clicks is not the least, game-clicks is the most but here their sum of total cost spent is not that high so this kind of users can be called "**Neutral gamers**".

Cluster 3 is different from the others in that the gamers(users) ad-clicks, game-clicks and total spending are all more than other users hence this kind of users can be called "**High spending gamers**".

Below you can see the summary of the train data set:



```
clusters
```

```
array( [ 25.12037037, 362.50308642, 35.35802469 ] )
array( [ 32.05, 2393.95, 41.2 ] )
array( [ 36.47486034, 953.82122905, 46.16201117 ] )
```

Recommended Actions

Action Recommended	**Rationale for the action**
Providing additional items to "**High spending gamers**".	These **High spending gamers** may click less than others but they tend to buy more than others, we can increase the revenue by providing additional items or bonus items.
Providing some extra items, packages or promotion to gamers, especially to "**Less spending gamers**".	To improve the revenue additional packages or promotional offers can be given frequently to these gamers in order to make them buy more items

Graph Analytics

Modeling Chat Data using a Graph Data Model

A Graph Data Model is used here to portray the interactions among the users in the game. A user can create a chat session and include other team members and have discussion threads between them. A user can also join a chat session, leave a chat session if he wishes to. A user can mention other users in the chat session. This data model helps us to analyze which team interacts more and helps us to improvise the game.

Creation of the Graph Database for Chats

Steps:

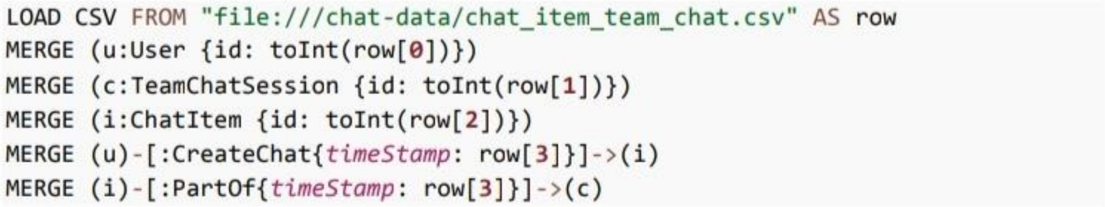
1. **Schema of 6 CSV files**

Name of the File	Attributes	Description
chat_create_team_chat.csv	UserID	It represents the ID of the user.
	TeamID	It represents the ID of the team.
	TeamChatSessionID	It represents the ID of the team's chat session.
	Timestamp	It represents the creation time of chat session.
chat_item_team_chat.csv	UserID	It represents the ID of the user.
	TeamChatSessionID	It represents the ID of the team's chat session.
	ChatItemID	It represents the ID of a chat item.
	Timestamp	It represents the creation time of chat session.
chat_join_team_chat.csv	UserID	It represents the ID of the user.
	TeamChatSessionID	It represents the ID of the team's chat session.
	Timestamp	It represents the creation time of chat session.
chat_leave_team_chat.csv	UserID	It represents the ID of the user.
	TeamChatSessionID	It represents the ID of the team's chat session.

	Timestamp	It represents the creation time of chat session.
chat_mention_team_chat.csv	ChatItemID	It represents the ID of a chat item.
	UserID	It represents the ID of the user.
	Timestamp	It represents the creation time of chat session.
chat_respond_team_chat.csv	ChatID1	It represents the ID of chat session 1
	ChatID2	It represents the ID of chat session 2
	Timestamp	It represents the creation time of chat session.

2. Loading Process

LOAD Command



```
LOAD CSV FROM "file:///chat-data/chat_item_team_chat.csv" AS row
MERGE (u:User {id: toInt(row[0])})
MERGE (c:TeamChatSession {id: toInt(row[1])})
MERGE (i:ChatItem {id: toInt(row[2])})
MERGE (u)-[:CreateChat{timeStamp: row[3]}]->(i)
MERGE (i)-[:PartOf{timeStamp: row[3]}]->(c)
```

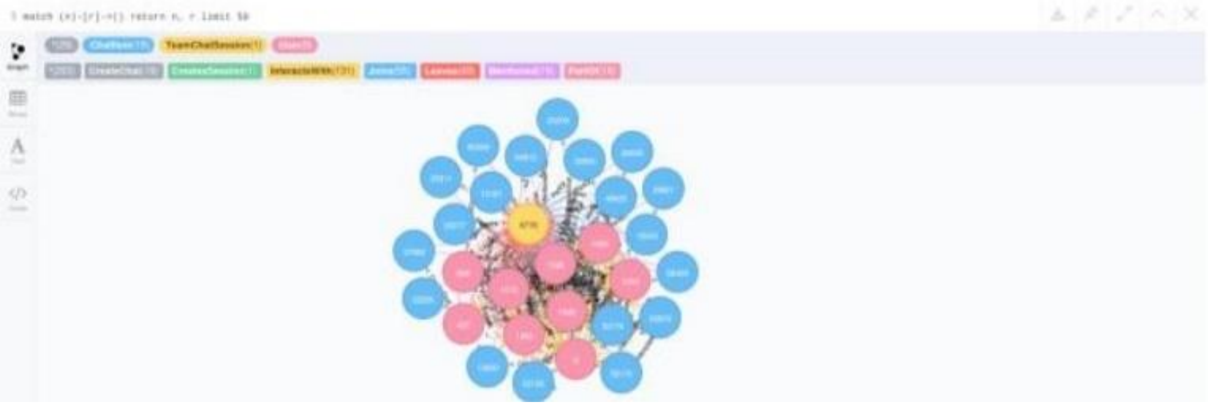
The LOAD CSV command loads the csv data present in that location as row at a time and creates user nodes.

The MERGE (u)-[:CreateChat{timeStamp: row[3]}]->(i) command is used to create an edge labeled “CreateChat” between the User ‘u’ and the ChatItem ‘I’. The edge contains a property called timeStamp. This property is filled by the content of column 3 of the same row.

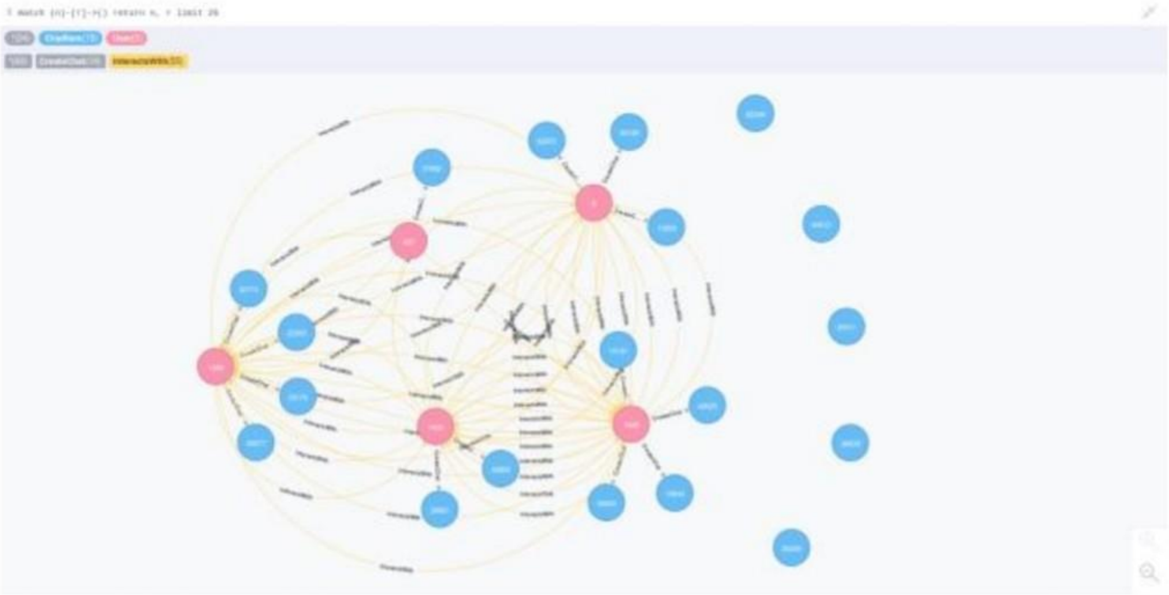
The MERGE (i)-[:PartOf{timeStamp: row[3]}]->(c) Command is used to create an edge labeled “PartOf” between the ChatItem ‘I’ and the TeamChatSession ‘c’. The edge contains a property called timeStamp. This property is filled by the content of column 3 of the same row.

3. Screenshot

The graphs must include clearly visible examples of most node and edge types. Below are two acceptable examples. The first example is a rendered in the default Neo4j distribution, the second has had some nodes moved to expose the edges more clearly. Both include examples of most node and edge types.



```
$ match (n)-[r]->() return n, r limit 50
```



```
$ match (n)-[r]->() return n, r limit 25
```

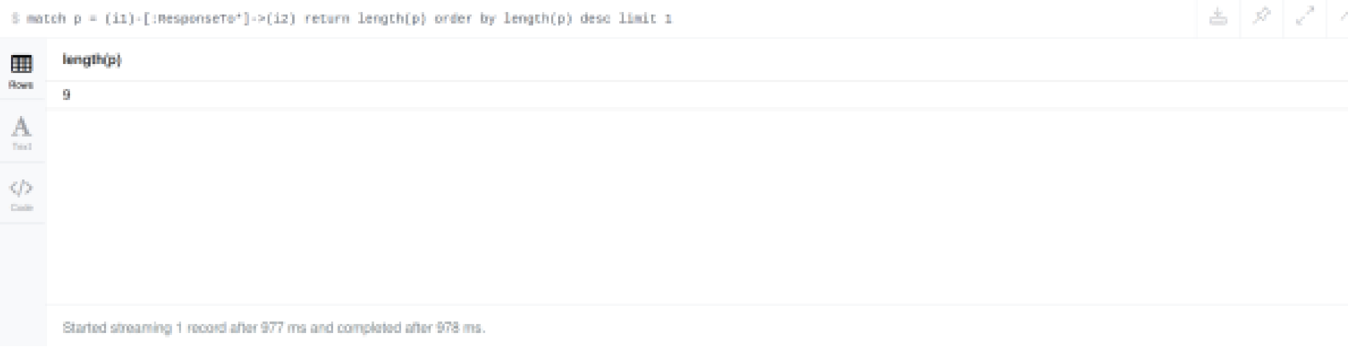
Finding the longest conversation chain and its participants

1) **The longest conversation chain** Is traced via `ChatItem` nodes which are connected by `ResponseTo` edges, to find the longest one order it's length. The following query when executed gives the longest conversation chain that has path length of **9**.

   a) Query

```
match p = (i1)-[:ResponseTo*]->(i2)
return length(p)
order by length(p) desc limit 1
```

   b) Result



```
$ match p = (i1)-[:ResponseTo*]->(i2) return length(p) order by length(p) desc limit 1
```

length(p)
9

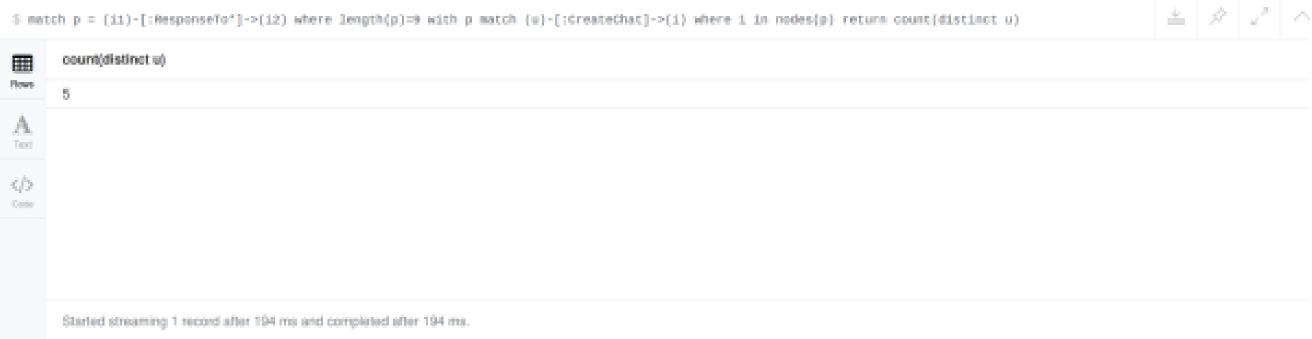
Started streaming 1 record after 977 ms and completed after 978 ms.

2) **Number of unique users** can be identified using the below query, we already know that the path length of the longest conversation chain is 9, hence to find all the distinct users which are part of this path, the query is executed, **5** unique users are extracted.

   a) Query

```
match p = (i1)-[:ResponseTo*]->(i2)
where length(p) = 9
with p
match (u)-[:CreateChat]->(i)
where i in nodes(p)
return count(distinct u)
```

   b) Result



```
$ match p = (i1)-[:ResponseTo*]->(i2) where length(p)=9 with p match (u)-[:CreateChat]->(i) where i in nodes(p) return count(distinct u)
```

```
count(distinct u)
5
```

Started streaming 1 record after 194 ms and completed after 194 ms.

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

1) Chattiest Users

The CreateChat edge from User node is matched with the ChatItem node and then return the ChatItem amount for every user , grouping is done in descending order.

a) Query

```
match (u)-[:CreateChat*]->(i)
return u.id, count(i)
order by count(i) desc limit 10
```

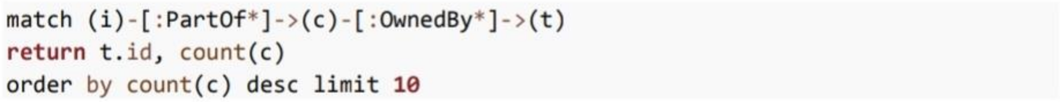
b) Result

Users	Number of Chats
394	115
2067	111
1087	109

2) Chattiest Teams

The part of edge from ChatItem node is matched with the TeamChatSession node , next we match the OwnedBy edge from TeamChatSession node to Team node, TeamChatSession amount per team is returned in adescending order.

a) Query



```
match (i)-[:PartOf*]->(c)-[:OwnedBy*]->(t)
return t.id, count(c)
order by count(c) desc limit 10
```

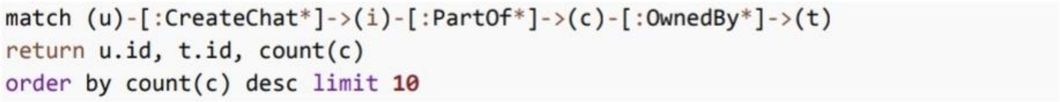
b) Result

Teams	Number of Chats
82	1324
185	1036
112	957

3) Check Whether Chattiest users belong to the Chattiest teams

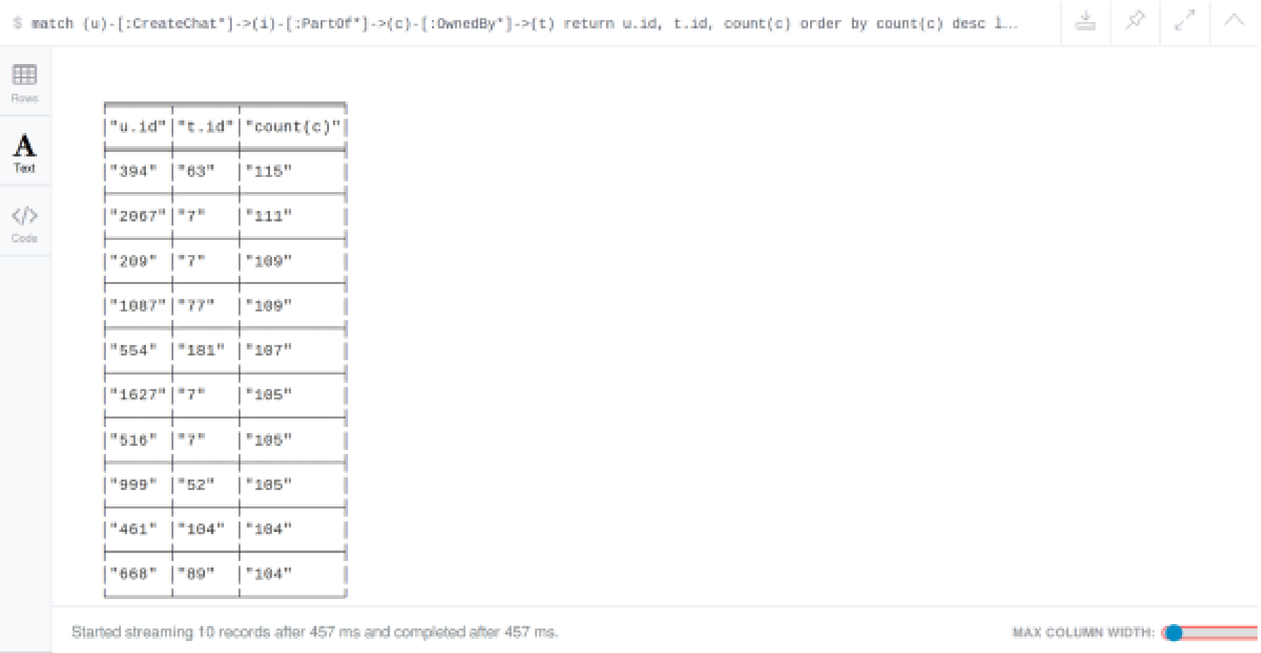
To perform checking we combine the above two queries:

a) Query



```
match (u)-[:CreateChat*]->(i)-[:PartOf*]->(c)-[:OwnedBy*]->(t)
return u.id, t.id, count(c)
order by count(c) desc limit 10
```

b) Result



```
$ match (u)-[:CreateChat*]->(i)-[:PartOf*]->(c)-[:OwnedBy*]->(t) return u.id, t.id, count(c) order by count(c) desc l...
```

```
|"u.id"|"t.id"|"count(c)"|
|"394" |"63"  |"115"     |
|"2067"|"7"   |"111"     |
|"209" |"7"   |"109"     |
|"1087"|"77"  |"109"     |
|"554" |"181" |"107"     |
|"1627"|"7"   |"105"     |
|"516" |"7"   |"105"     |
|"999" |"52"  |"105"     |
|"461" |"104" |"104"     |
|"668" |"89"  |"104"     |
```

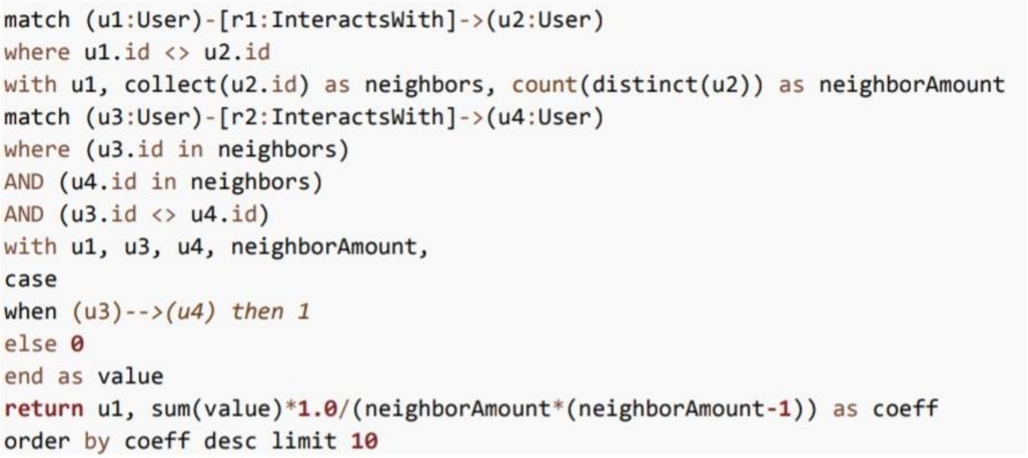
Started streaming 10 records after 457 ms and completed after 457 ms.

From the above result it’s evident that only one user ‘999’ belonging to team ‘52’ is part of the chattiest teams, but other users are not part of the chattiest teams.

Therefore, Most of the chattiest users do not belong to the chattiest teams.

How Active Are Groups of Users?

To know how active are groups of users it is necessary to determine the highly interactive neighborhoods. Activity of a group can be determined based on their chatting. The following query is executed to determine this result.



```
match (u1:User)-[r1:InteractsWith]->(u2:User)
where u1.id <> u2.id
with u1, collect(u2.id) as neighbors, count(distinct(u2)) as neighborAmount
match (u3:User)-[r2:InteractsWith]->(u4:User)
where (u3.id in neighbors)
AND (u4.id in neighbors)
AND (u3.id <> u4.id)
with u1, u3, u4, neighborAmount,
case
when (u3)-->(u4) then 1
else 0
end as value
return u1, sum(value)*1.0/(neighborAmount*(neighborAmount-1)) as coeff
order by coeff desc limit 10
```

Most Active Users (based on Cluster Coefficients)

User ID	Coefficient
209	0.9523809523809523
554	0.9047619047619048
1087	0.8