Technical Appendix
Catch the Pink Flamingo Analysis

Produced by: Luis A. Romero Gamarra

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	This file contains all data from user when clicks on an advertisement.	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	This file contains all data from each purchase of the player in the game.	timestamp: when the purchase was made txId: a unique id for purchase userSessionid: the id of the user session for the user who made the purchase

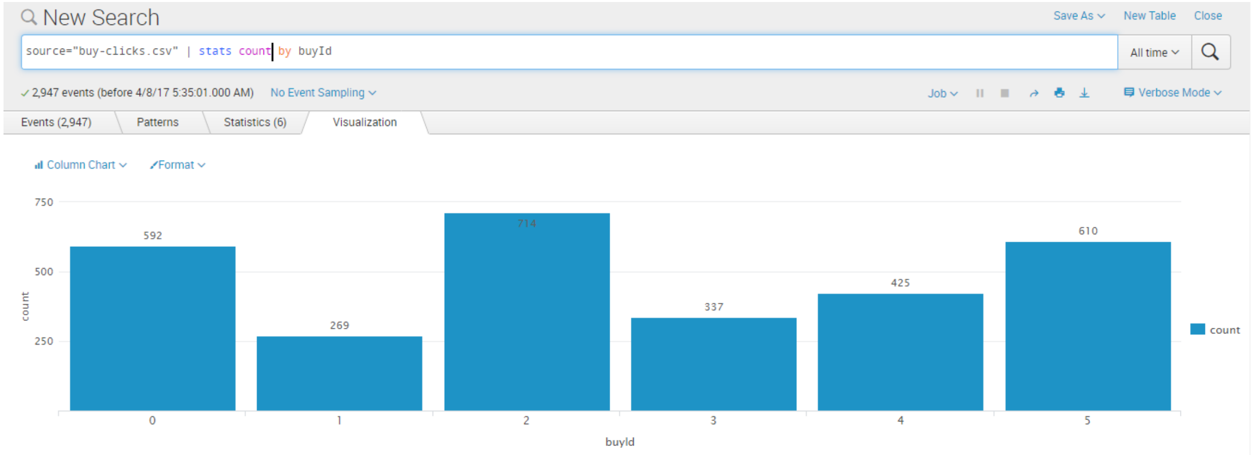
		team: the current team id of the user who made the purchase userid: the user id of the user who made the purchase buyID: the id of the item purchased price: the price of the item purchased
users.csv	This file contains all data of each player.	timestamp: when user first played the game id: the user id 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 all data for each team created or finished in the game	teamid: unique id assigned to each team name: name of the team teamCreationTime: when team record was created teamEndTime: timestamp when the “last” member left the team currentLevel: current level reached by the team
team-assignments.csv	This file contains all data about one player joins a team.	time: when the user joined the team team: team id userid: user id Assignmentid: a unique id assigned to each user-team mapping

level-events.csv	This file contains all data about the levels in the game.	time: when a level start or end occurred eventid: a uniqueid assigned to each record teamid: team id level: level started or completed eventType: start or end
user-session.csv	This file contains all data about user session, their team levels and platform that user uses.	userSessionid: a unique id for the session. assignmentid: the team assignment id for the user to the team. startTimeStamp: a timestamp denoting when the session started. endTimeStamp: a timestamp denoting when the session ended. team_level: the level of the team during this session. platformType: the type of platform of the user during this session
game-clicks.csv	This file contains data about each time a user performs a click when he/she is playing.	time: 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

Aggregation

Amount spent buying items	**21407** (code: source="buy-clicks.csv" |stats sum(price))
# Unique items available to be purchased	**6 items** (code: source="buy-clicks.csv" |stats distinct_count(buyId)) Supose that item available are each buyId which is the identifier each item

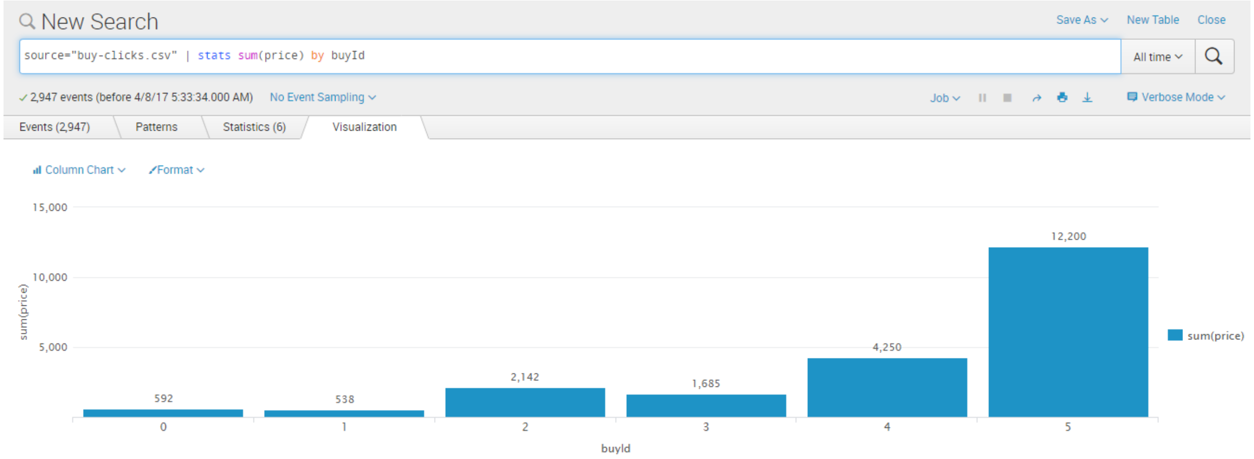
A histogram showing how many times each item is purchased:



```
source="buy-clicks.csv" | stats count by buyId
```

✓ 2,947 events (before 4/8/17 5:35:01.000 AM)

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

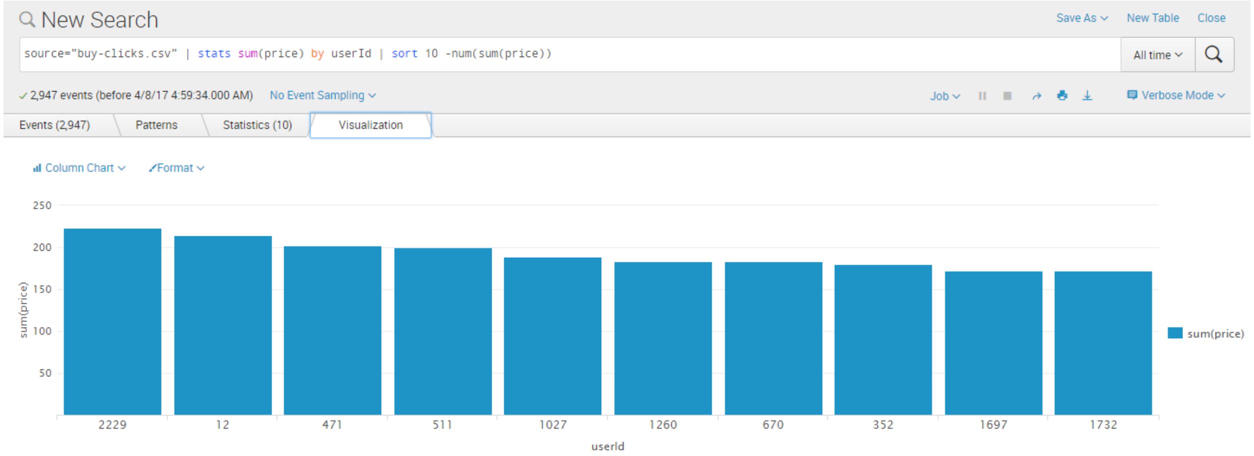


```
source="buy-clicks.csv" | stats sum(price) by buyId
```

✓ 2,947 events (before 4/8/17 5:33:34.000 AM)

Filtering

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



```
source="buy-clicks.csv" | stats sum(price) by userId | sort 10 -num(sum(price))
```

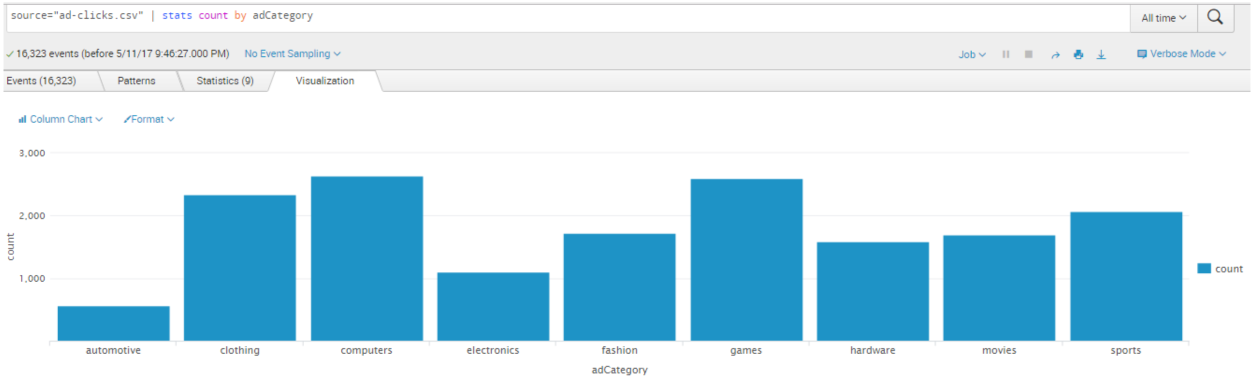
✓ 2,947 events (before 4/8/17 4:59:34.000 AM)

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 %**

3	**471**	**IPhone**	**14.5 %**

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



```
source="ad-clicks.csv" | stats count by adCategory
```

✓ 16,323 events (before 5/11/17 9:46:27.000 PM)

The following table shows the total amount of ad-click revenue for a set of specific values based on the advertisement category. All non-listed categories generate .25 revenue.

Scenario #	**Electronics**	**Fashion**	**Automotive**	**Total Revenue**
1 - even	**0.50**	**0.50**	**0.50**	**$4928,25**
2 - uneven	**0.55**	**0.60**	**0.55**	**$5184,10**

Data Classification Analysis

Data Preparation

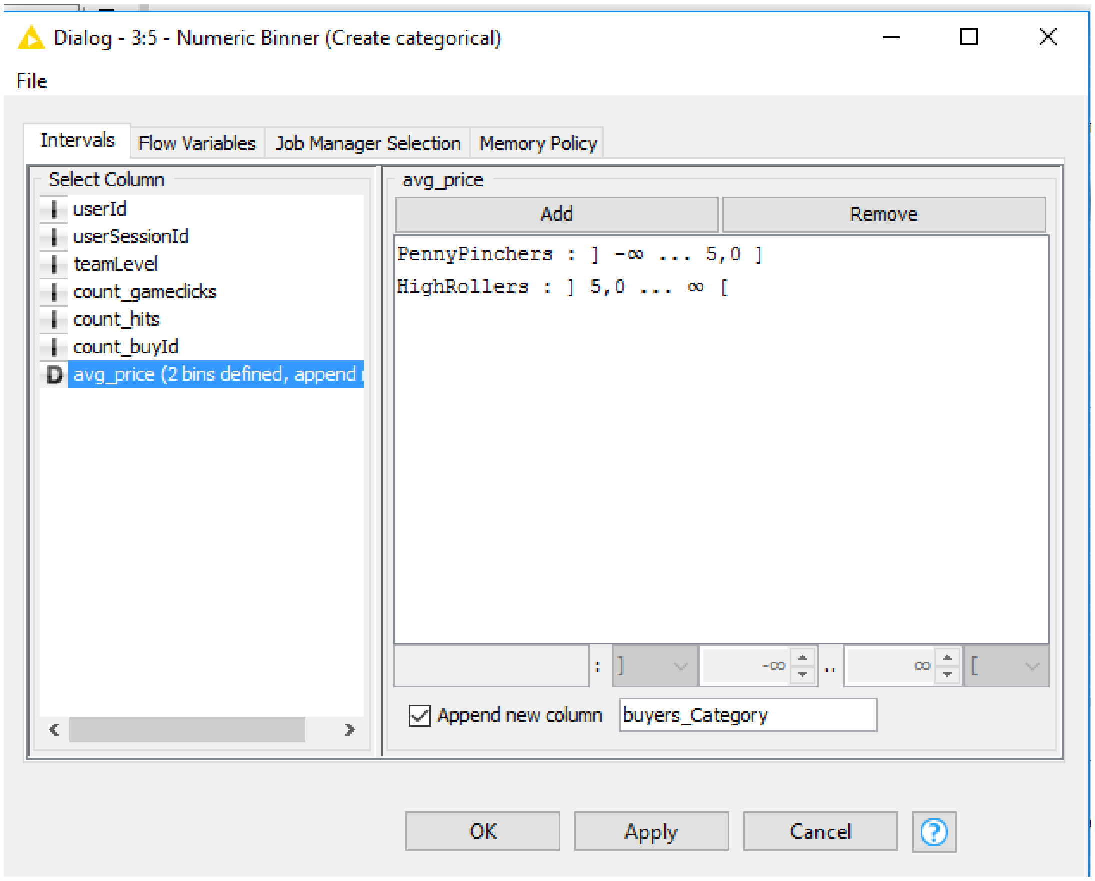
Analysis of combined_data.csv

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:



```
PennyPinchers : ] -∞ ... 5,0 ]
HighRollers : ] 5,0 ... ∞ [
```

☑ Append new column: buyers_Category

HighRollers are buyers of items that cost more than $5.00 and PennyPinchers are buyers of items that cost $5.00 or less. This new category is named buyers_Catgory.
This new categorical attribute will be derived by binning an existing numeric attribute, in this case is avg_price and convert this a categorical attribute.

The creation of this new categorical attribute was necessary because ***we need convert numeric attribute to categorical attribute which it will be the target for the prediction.***

Attribute Selection

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

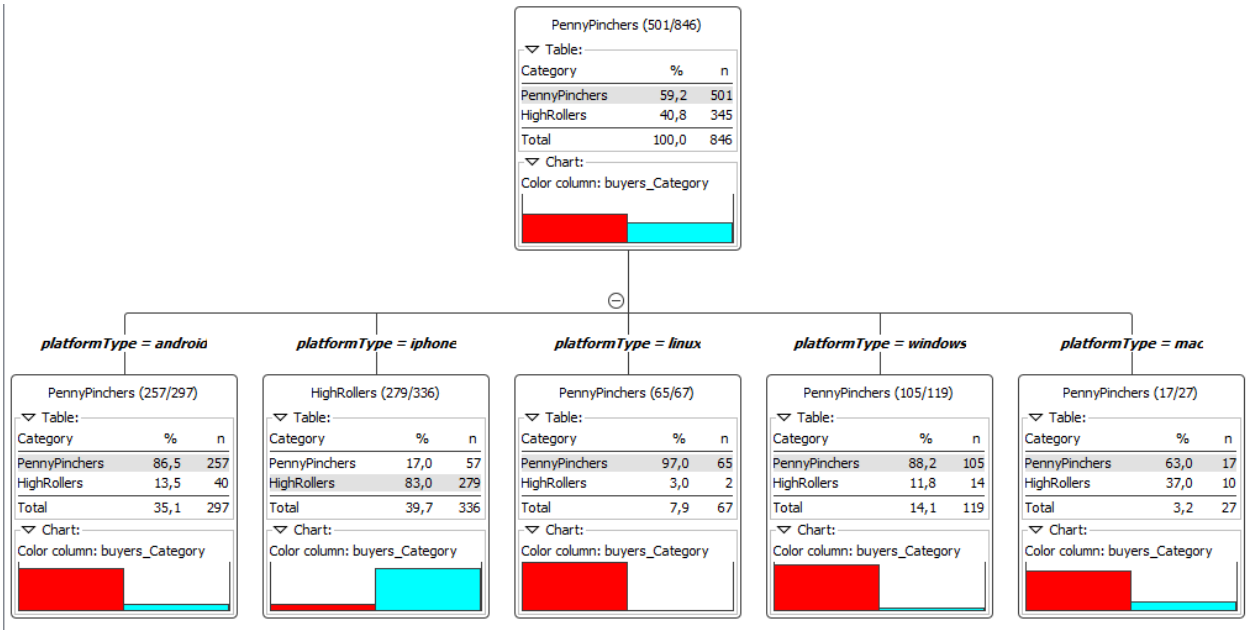
Attribute	**Rationale for Filtering**
Avg_price	**This is the source attribute to realize the new categorical attribute buyers_Categorical, so this column is excluded.**
UserId	**This attribute is no correlationed with the target attribute, not is necessary to have this column, so it is excluded.**
UsesSessionId	**This column has the Id values for user session, then this attribute don't use for make the prediction because It have not correlation with new category, It is excluded.**

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…
This is important because this two partitions allow define the different parts to achieve our prediction. In this case, train data allow to learn of input data and create a model that, in next step, it will be apply to test data through decision tree predictor. For last, the result is compared for perform the accuracy through error measure.

When partitioning the data using sampling, it is important to set the random seed was set because **when we modify the random seed, we ensure that the results are the same by a given seed. Otherwise, each execution will be with different seeds.**

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



PennyPinchers (501/846)

Category	%	n
PennyPinchers	59,2	501
HighRollers	40,8	345
Total	100,0	846

PennyPinchers (257/297)

Category	%	n
PennyPinchers	86,5	257
HighRollers	13,5	40
Total	35,1	297

HighRollers (279/336)

Category	%	n
PennyPinchers	17,0	57
HighRollers	83,0	279
Total	39,7	336

PennyPinchers (65/67)

Category	%	n
PennyPinchers	97,0	65
HighRollers	3,0	2
Total	7,9	67

PennyPinchers (105/119)

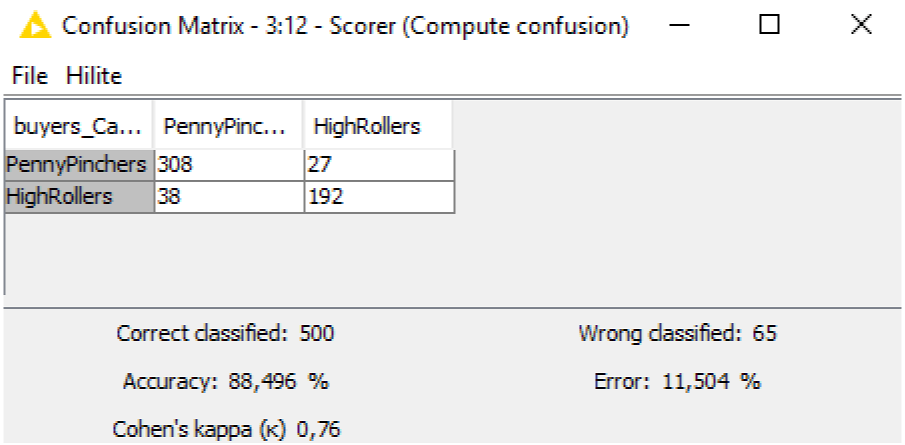
Category	%	n
PennyPinchers	88,2	105
HighRollers	11,8	14
Total	14,1	119

PennyPinchers (17/27)

Category	%	n
PennyPinchers	63,0	17
HighRollers	37,0	10
Total	3,2	27

Evaluation

A screenshot of the confusion matrix can be seen below:



buyers_Ca...	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

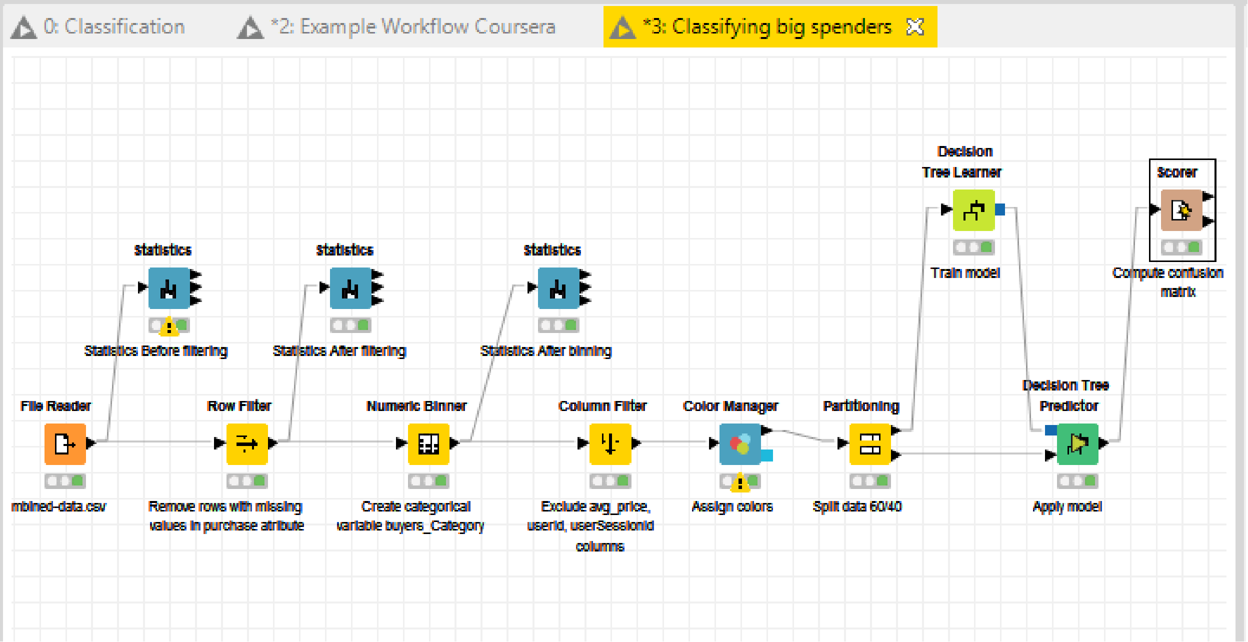
As seen in the screenshot above, the overall accuracy of the model is **88,496%**

- **There are 308 PennyPinchers correctly classified.**

- **There are 27 PennyPinchers have been wrongly classified as HighRollers.**
- **There are 192 HighRollers have been correctly classified**
- **There are 38 have been classified as PennyPinchers, so this is wrong.**

Analysis Conclusions

The final KNIME workflow is shown below:



What makes a HighRoller?

Users that use IPhone platform are most likely to be a HighRoller, IPhone users spend more money than others. The user that use linux, android, windows and mac have most probability to be a PennyPincher because spend less in purchase.

Specific Recommendations to Increase Revenue
1. The company should focus in IPhone users, this user spend more money in purchase, in this case, It will be important create new products for increase the revenues in this platform.
2. Analysis the cost of products in platform with user least spenders, improve the engagement i.e. for each purchase in-app this user receipt gift or unlock different achievements.

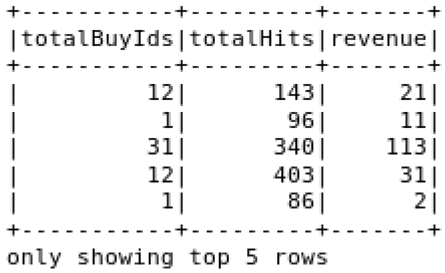
Clustering Analysis

Attribute Selection

Attribute	Rationale for Selection
totalBuyIds	Attribute that count total item buying for each user. This attribute is from buy-clicks.csv columns and it is important for analysis in-app purchase.
totalHits	Attribute that count total hits clicked for each user, if user was click their value is 1, otherwise is 0. This new attribute is from game-clicks.csv file.
revenue	Revenue is the aggregation of the total spending by user. The analysis tried to find out if this number is related to total number of hits and buyIds by users.

Training Data Set Creation

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



```
+-----------+---------+-------+
|totalBuyIds|totalHits|revenue|
+-----------+---------+-------+
|         12|      143|     21|
|          1|       96|     11|
|         31|      340|    113|
|         12|      403|     31|
|          1|       86|      2|
+-----------+---------+-------+
only showing top 5 rows
```

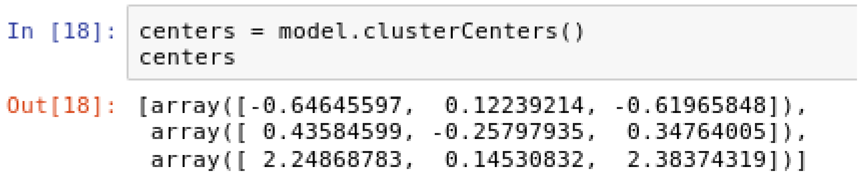
Dimensions of the training data set (rows x columns) : **(546x3)**

# of clusters created: **3 clusters**

Cluster Centers

Cluster #	Cluster Center
Cluster 1	**-0.64645597, 0.12239214, -0.61965848**

Cluster 2	**0.43584599, -0.25797935, 0.34764005**
Cluster 3	**2.24868783, 0.14530832, 2.38374319**



```
In [18]: centers = model.clusterCenters()
         centers

Out[18]: [array([-0.64645597,  0.12239214, -0.61965848]),
          array([ 0.43584599, -0.25797935,  0.34764005]),
          array([ 2.24868783,  0.14530832,  2.38374319])]
```

First number (field1) in each array refers to scaled version of the number of buy-Id and the second number (field2) is the scaled version of the totalHits, and the third number (field3) is the scaled version of the revenue per user.

- Compare the 1st number of each cluster to see how differently users in each cluster behave when it comes to buying items.
- Compare the 2nd number of each cluster to see how differently users in each cluster behave when it comes to hit in the game.
- Compare the 3rd number of each cluster to see how differently users in each cluster behave when it comes to buying stuff

These clusters can be differentiated from each other as follows:

Cluster 1 is different from the others in that it has lower values than clusters 3. All fields have lowest values, so this show that few total buy items means user is new or has no interest in the items.

Cluster 2 is different from the others in that it has high values than clusters 1. All fields have highest values, so this show that more total hits means user tend to purchase more in the app, except in the field 2 (center) since the values is lowest, this mean that currently exist

Cluster 3 is different from the others in that this cluster have high values than other cluster except in the field 2 that mean the user can be click hits more times but not help to increment revenue. Otherwise, the highest values in the fields 1 and 3 show that user with buy more item tending to increase the revenue.

Recommended Actions

Action Recommended	Rationale for the action
Make new promotion and focused in the items	We can make new item according level of user, for each historical of item, we can down their price or we can sell create determinate item for a determinate kind of user for increase their value.
Make offers for each hit > 1 (hits corrects)	We can make that a user win new item for a determinate number of hits corrects. Its increase the purchase and given value to the game.
Create new prices for users engaged	We can raise prices determined for the users who have more clicks since they are what they buy more items

Graph Analytics Analysis

Modeling Chat Data using a Graph Data Model

This graph contains data about players and their information in chat sessions, this data is keep in six files with format csv. The information of this files are:

Nodes		
Name	**Properties**	**Descriptions**
User	Id	User that interacts in the chat
Team	Id	Teams of users in each chat room
TeamChatSession	Id	Each session create for a team of players
ChatItem	Id	Each text in the chat, represents by Id

Relationships		
Name	**Properties**	**Descriptions**
CreateSession	Timestamp	Edge labeled "CreatesSession" between the user node and the TeamChatSession
OwnerBy	Timestamp	Edge labeled "OwnedBy" between the TeamChatSession node and the Team node

Joins	Timestamp	Edge labeled "Joins" from User to TeamChatSession
Leaves	Timestamp	Edge labeled "Leaves" from User to TeamChatSession
Mentioned	Timestamp	Edge labeled "Mentioned", the edge going from the chatItem to the User
PartOf	Timestamp	Edge labeled "PartOf" from the ChatItem node to the TeamChatSession node.
ResponseTo	Timestamp	Edge labeled "ResponseTo" from a ChatItem node to another ChatItem node.
InteractsWith	Timestamp	Edge labeled “InteractsWith " composed only by users.

Creation of the Graph Database for Chats

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

- Write the schema of the 6 CSV files

Chat_create_team_chat.csv: userid, teamid, TeamChatSessionID, timestamp

Chat_item_team_chat.csv: userid, TeamChatSessionID, ChatItemID, timestamp

Chat_join_team_chat.csv: userid, TeamChatSessionID, timestamp

Chat_leave_team_chat.csv: userid, TeamChatSessionID, timestamp

Chat_mention_team_chat.csv: ChatItem, userid, timestamp

Chat_respond_team_chat.csv: ChatItem, ChatItem, timestamp

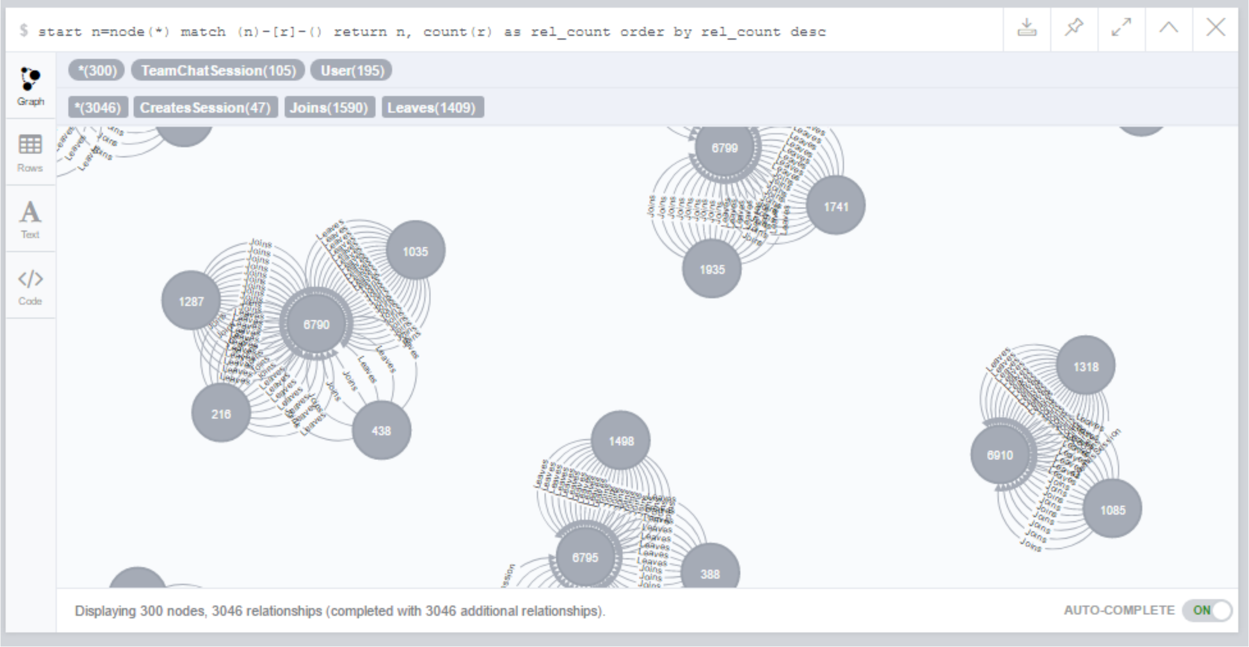
- Explain the loading process and include a sample LOAD command

Create an import directory in
“C:\Users\Alias\OneDrive\Documentos\Neo4j\default.graphdb\import”
Then we save each file csv in this folder. After, we execute each step for each file, so that are 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: toInt(row[0])})
MERGE (c:TeamChatSession {id: toInt(row[1])})
MERGE (u)-[:Joins {timeStamp: row[2]}]->(c)

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

LOAD CSV FROM "file:///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)

LOAD CSV FROM "file:///chat_mention_team_chat.csv" AS row
MERGE (i:ChatItem {id: toInt(row[0])})
MERGE (u:User {id: toInt(row[1])})
MERGE (i)-[:Mentioned {timeStamp: row[2]}]->(u)

LOAD CSV FROM "file:///chat_respond_team_chat.csv" AS row
MERGE (a:ChatItem {id: toInt(row[0])})
MERGE (b:ChatItem {id: toInt(row[1])})
MERGE (a)-[:ResponseTo {timeStamp: row[2]}]->(b)
```

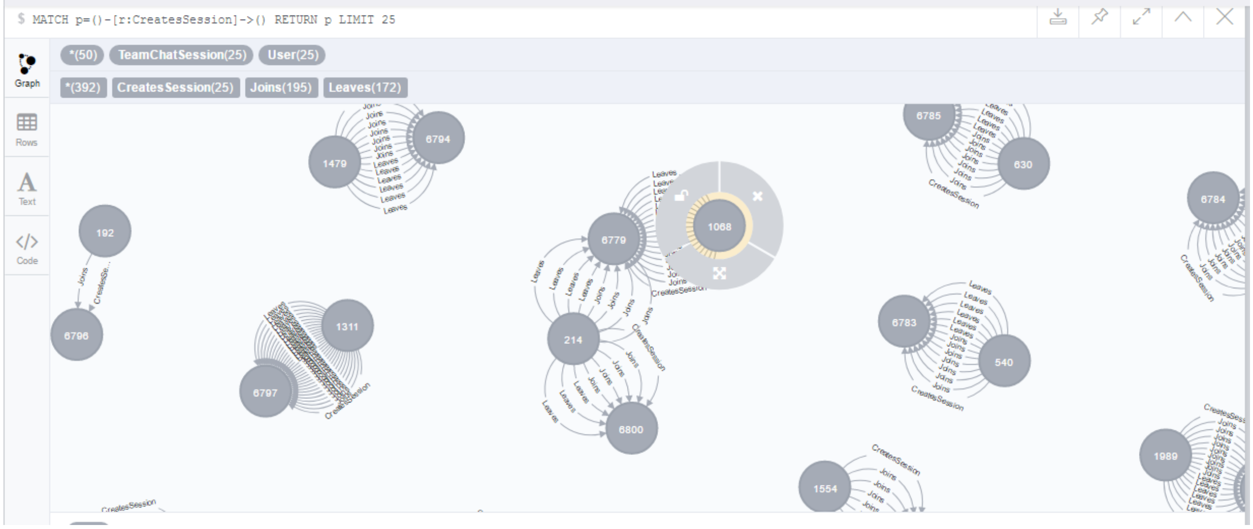
- Present a screenshot of some part of the graph you have generated. The graphs must include clearly visible examples of most node and edge types. Below are two acceptable examples. The first example is a rendered in the default Neo4j distribution, the second has had some nodes moved to expose the edges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



```
$ start n=node(*) match (n)-[r]-() return n, count(r) as rel_count order by rel_count desc
```

Displaying 300 nodes, 3046 relationships (completed with 3046 additional relationships).

AUTO-COMPLETE ON

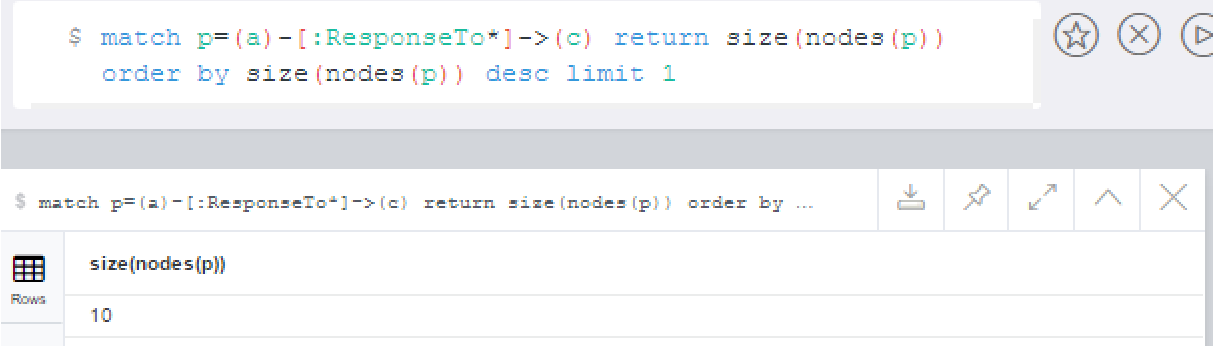


```
$ MATCH p=()-[r:CreatesSession]->() RETURN p LIMIT 25
```

Finding the longest conversation chain and its participants

Report the results including the length of the conversation (path length) and how many unique users were part of the conversation chain. Include an image of the graph with the longest conversation chain.

For this question, we have to do two steps, in the first we find the number of chats in longest conversation chain:



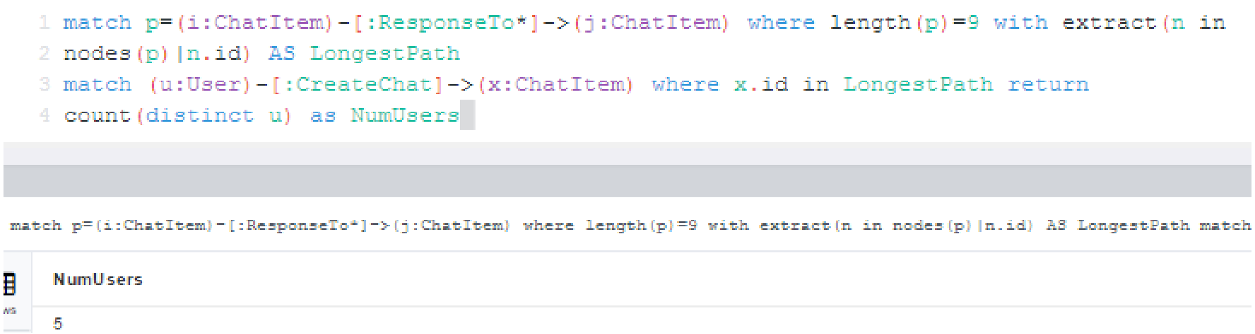
```
$ match p=(a)-[:ResponseTo*]->(c) return size(nodes(p))
  order by size(nodes(p)) desc limit 1
```

```
$ match p=(a)-[:ResponseTo*]->(c) return size(nodes(p)) order by ...
```

size(nodes(p))
10

For the second step we need to find unique users of the conversation chain:

```
match p=(i:ChatItem)-[:ResponseTo*]->(j:ChatItem) where length(p)=9 with extract(n in nodes(p)|n.id)
AS LongestPath  match (u:User)-[:CreateChat]->(x:ChatItem) where x.id in LongestPath return
count(distinct u) as NumUsers
```

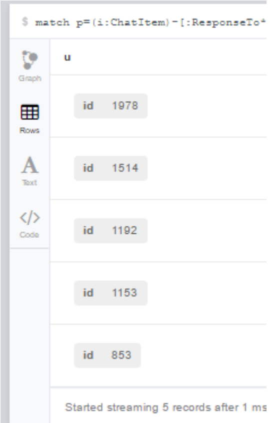


```
match p=(i:ChatItem)-[:ResponseTo*]->(j:ChatItem) where length(p)=9 with extract(n in
nodes(p)|n.id) AS LongestPath
match (u:User)-[:CreateChat]->(x:ChatItem) where x.id in LongestPath return
count(distinct u) as NumUsers
```

```
match p=(i:ChatItem)-[:ResponseTo*]->(j:ChatItem) where length(p)=9 with extract(n in nodes(p)|n.id) AS LongestPath match
```

NumUsers
5

There are 5 unique user in the chain:



```
$ match p=(i:ChatItem)-[:ResponseTo*
```

u
id 1978
id 1514
id 1192
id 1153
id 853

Started streaming 5 records after 1 ms

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

Include your table containing the top 3 chattiest users and teams below, and report whether or not any of the chattiest users are part of any of the chattiest teams.

Chattiest Users

```
match p=(a)-[:ResponseTo*]->(c)
return size(nodes(p)) order by size(nodes(p)) desc limit 1
```

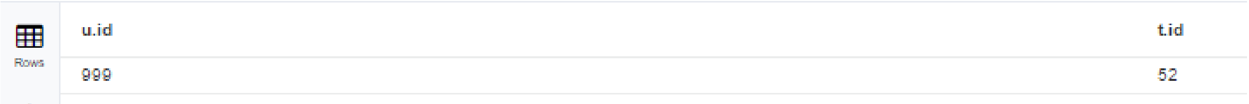
Users	Number of Chats
394	115
2067	111
209	109

Chattiest Teams

```
match p =(i:ChatItem)-[:ResponseTo*]->(j:ChatItem) where size(nodes(p)) = 10
with p match (u:User)-[:CreateChat]->(i:ChatItem) where i in nodes(p)
return distinct u
```

Teams	Number of Chats
82	1324
185	1036
112	957

We conclude that for the top 3 user, none of this user belong to the top 3 teams. But if we analyze the top 10 user and teams, we see that only the user 999 is belong to team 52 of the top 10 teams.



u.id	t.id
999	52

How Active Are Groups of Users?

For the first step, we execute the query that mentioned another user in a chat, this we did in the previous assignment instructions:

· One mentioned another user in a chat:

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

· One created a chatItem in response to another user's chatItem:

```
Match (u1:User)-[:CreateChat]->(i:ChatItem)-[:ResponseTo]->(i2:ChatItem)<-[:CreateChat]-(u2:User)
create (u1)-[:InteractsWith]->(u2)
```

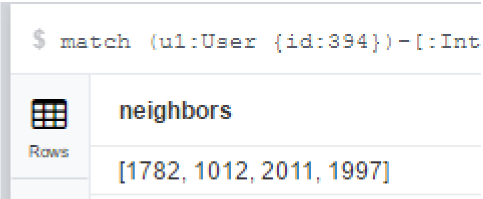
·

Now we need remove all self-loops with the new edge created in the step above:

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

The next step is get list of neighbors for each user Id:

```
match (u1:User {id:394})-[:InteractsWith]-(u2:User)
with collect(distinct u2.id) as neighbors
return neighbors
```

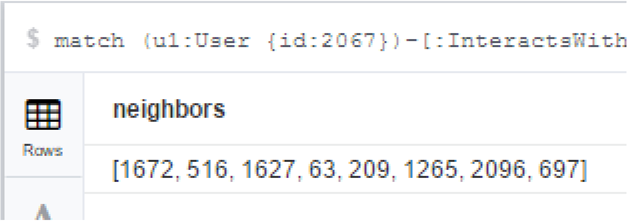


```
$ match (u1:User {id:394})-[:Int
```

neighbors
[1782, 1012, 2011, 1997]

Total neighbors = 4

```
match (u1:User {id:2067})-[:InteractsWith]-(u2:User)
with collect(distinct u2.id) as neighbors
return neighbors
```

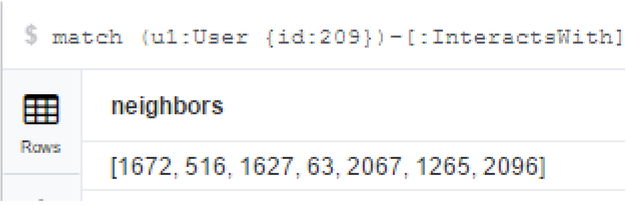


```
$ match (u1:User {id:2067})-[:InteractsWith
```

neighbors
[1672, 516, 1627, 63, 209, 1265, 2096, 697]

Total neighbors = 8

```
match (u1:User {id:209})-[:InteractsWith]-(u2:User)
with collect(distinct u2.id) as neighbors
return neighbors
```



```
$ match (u1:User {id:209})-[:InteractsWith]
```

neighbors
[1672, 516, 1627, 63, 2067, 1265, 2096]

Total neighbors = 7

For the final step, we need to do the formula:

Clustering coefficient = total number of edges on neighbors/[k * (k-1)], where k is the number of neighbors of the node

We know the number of neighbors, so we need to know the edges for each user:

```
Match (u)-[cc:CreateChat]->()
With u,count(u) as chatCount order by chatCount desc limit 10
With collect(u.id) as users match(u:User)-[:InteractsWith]-(u1:User) where u.id in users
With u,count(distinct u1) as k, collect(distinct u1.id) as neighbours
Match (n:User)-[:InteractsWith]-(n1:User),(m:User)-[:InteractsWith]-(m1:User) where n.id=u.id
and n1.id
in neighbours and m1.id in neighbours
With distinct u,n1,m1,k
With u,sum(case when (n1)--(m1) then 1 else 0 end) as edges return u.id, edges
```

Report the top 3 most active users in the table below.

Most Active Users (based on Cluster Coefficients)

User ID	Coefficient
394	1
2067	1.32
209	1.28

Recommended Actions

After the exhaustive analysis carried out in the different phases, we conclude that to increase revenues the company should pay attention to the following points:

Focuses in IPhone user, this user spend more money in purchase than other user in the different platforms. It will be important create new products or promoter the currently offer because through analysis we know that this user spend most money than others.

Make new promotion and focused in the items, we can make new item according level of user, for each historical of item, we can down their price or we can sell it, create determinate item for a determinate kind of user for increase their value. For this last sentence, it is important know the user behavioral, in this case, we know user with more clicks (hits) and many session are good target to show more ads than other users. We recommends create new items to new users for increase engagement and adding more ads for user more actives.

For finished, focuses in the user most active in each chartroom, they are potential consumers for ads and new items. With this recommends we are sure that Eglence Inc. will increase revenues and engaged new users.