# Technical Appendix Template

## 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** | Each line in the file represents when a player clicks on an ad in a Flamingo game. | -timestamp: when the click occurred.<br>-txId: a unique id for the click<br>-userSessionid: the id of the user session for the user who made the click<br>-teamid: the current team id of the user who made the click<br>-userid: the user id of the user who made the click<br>-adId: the id of the ad clicked on<br>-adCategory: the category/type of ad clicked on |
| **buy-clicks.csv** | Each line in this file represents when a player makes an in-app purchase in a Flamingo game. | -timestamp: when the purchase was made.<br>-txId: a unique id for the purchase<br>-userSessionId: the id of the user session for the user who made the purchase<br>-team: the current team id of the user who made the purchase<br>-userId: the user id of the user who made the purchase<br>-buyId: the id of the item purchased<br>-price: the price of the item purchased |

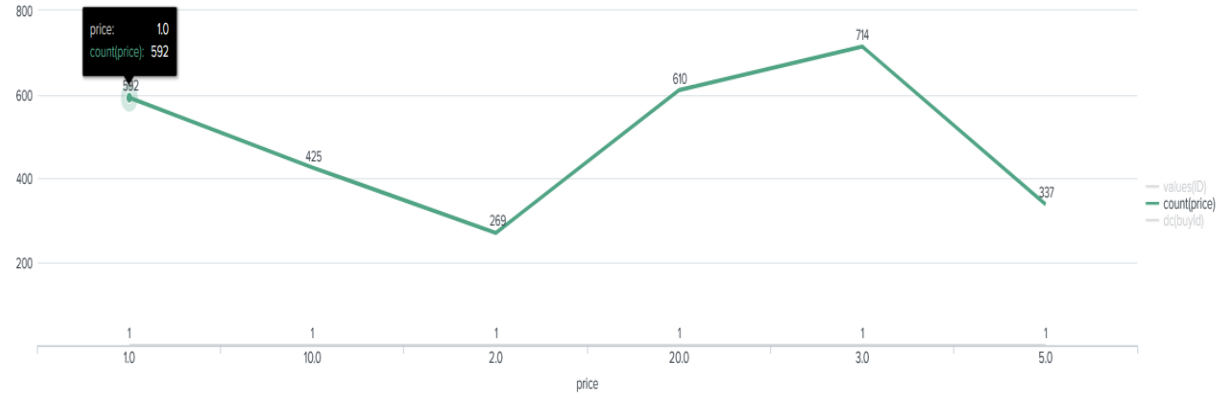
| | | |
|---|---|---|
| **users.csv** | This file contains a line for each user who plays the game. | -timestamp: when user first played the game.<br>-userId: the user id assigned to the user.<br>-nick: the nickname chosen by the user.<br>-twitter: the twitter handles of the user.<br>-dob: the date of birth of the user.<br>-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<br>-name: the name of the team<br>-teamCreationTime: the timestamp when the team was created<br>-teamEndTime: the timestamp when the last member left the team<br>-strength: a measure of team strength, roughly corresponding to the success of a team<br>-currentLevel: the current level of the team |
| **team-assignments.csv** | Each line in this file represents each time a user joins a team. The user can be in at most one team at a time. | -timestamp: when the user joined the team.<br>-team: the id of the team<br>-userId: the id of the user<br>-assignmentId: a unique id for this assignment |
| **level-events.csv** | Each line in this file represents each time a team starts or ends with a level in the game | -timestamp: when the event occurred.<br>-eventId: a unique id for the event<br>-teamId: the id of the team<br>-teamLevel: the level started or completed<br>-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 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.<br>-userSessionId: a unique id for the session.<br>-userId: the current user's ID.<br>-teamId: the current user's team.<br>-assignmentId: the team assignment id for the user to the team.<br>-sessionType: whether the event is the start or end of a session.<br>-teamLevel: the level of the team during this session.<br>-platformType: the type of platform of the user during this session. |
| **game-clicks.csv** | Each line in this file represents each time the user makes a click in the game. | -timestamp: when the click occurred.<br>-clickId: a unique id for the click.<br>-userId: the id of the user performing the click.<br>-userSessionId: the id of the session of the user when the click is performed.<br>-isHit: denotes if the click was on a flamingo (value is 1) or missed the flamingo (value is 0).<br>-teamId: the id of the team of the user.<br>-teamLevel: the current level of the team of the user. |

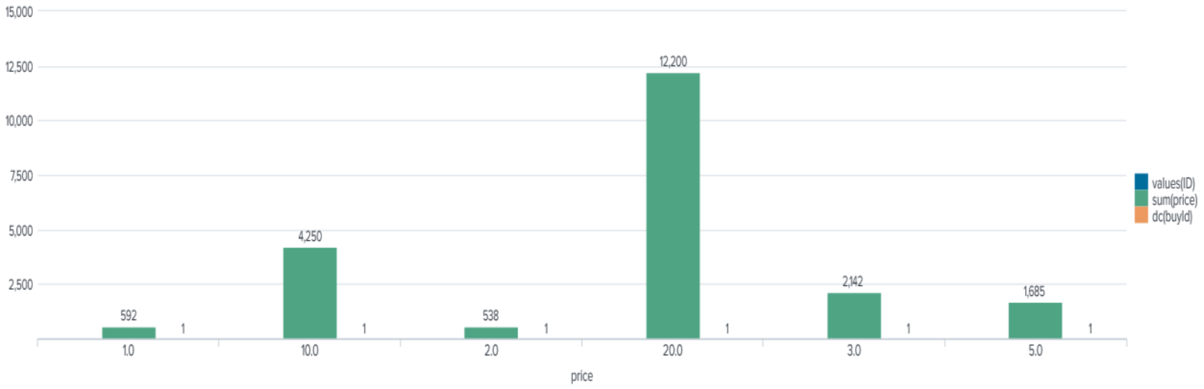
## Aggregation

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

A histogram showing how many times each item is purchased:

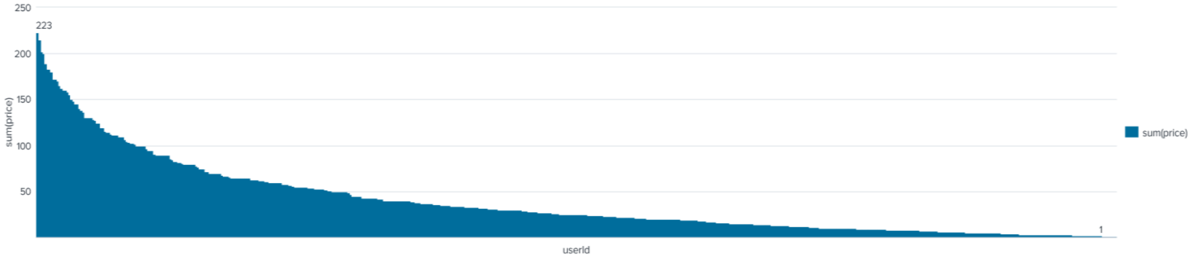


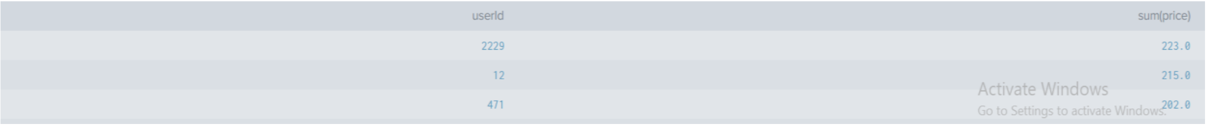
A histogram showing how much money was made from each item:



## Filtering

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





| userId | sum(price) |
|---|---|
| 2229 | 223.0 |
| 12 | 215.0 |
| 471 | 202.0 |

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 | 13.07% |
| 3 | 471 | iPhone | 14.5% |

# Data Preparation

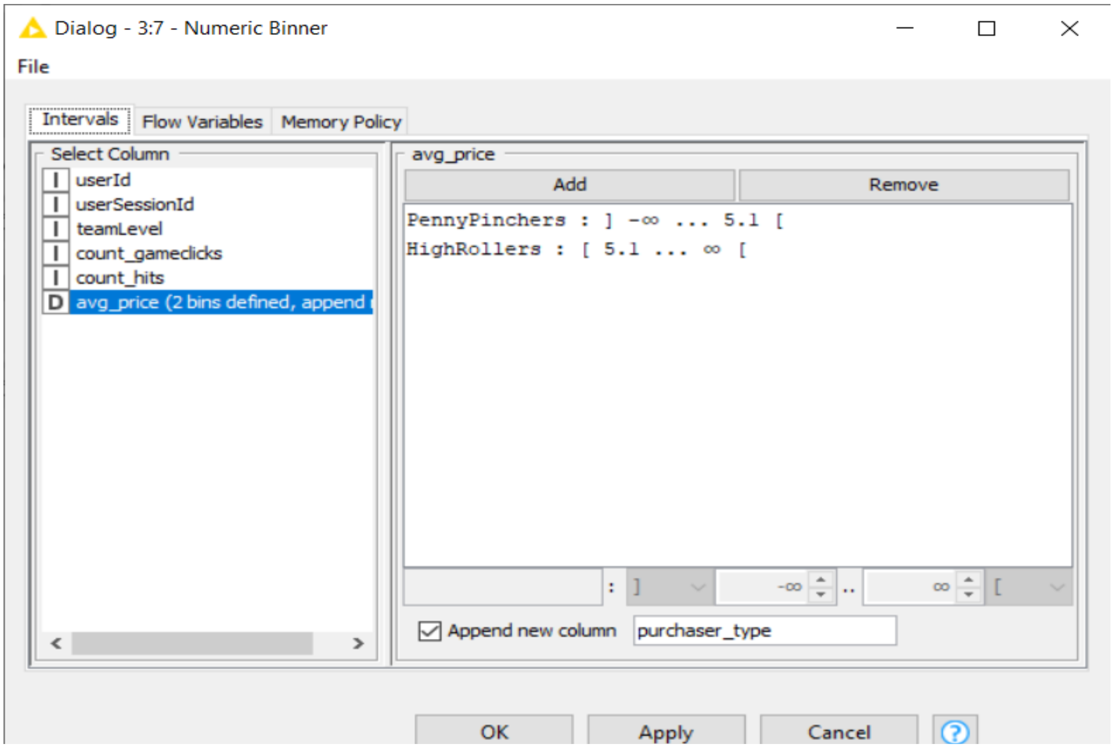
Analysis of combined_data.csv

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:



```
I userId
I userSessionId
I teamLevel
I count_gameclicks
I count_hits
D avg_price (2 bins defined, append
```

avg_price

```
PennyPinchers : ] -∞ ... 5.1 [
HighRollers : [ 5.1 ... ∞ [
```

: ] -∞ .. ∞ [

☑ Append new column purchaser_type

The design description:

- **High roller those who are spending more than 5.00$.**
- **Penny pinchers those who spend 5.00$ or less.**
- **The new column builds based on the average price that make us use it for classifying the users.**

The creation of this new categorical attribute was necessary because:

**We want to understand the attribute that we have and use it to know who make the large purchases in the game in order to make decision and this categorical variable are the foundation in our decision tree.**

Attribute Selection

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

| Attribute | Rationale for Filtering |
|---|---|
| **userId** | **does not have relevant to the workflow.** |
| **userSessionId** | **does not have relevant to the workflow.** |
| **avg_price** | **This column was used to create the categorical column "purchaser_type" we will try to predict based in the new split data so that why we dont need this futaur in our workflow.** |

# Data Partitioning and Modeling

The data was partitioned into train and test datasets.

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

The trained model was then applied to the **test** dataset.

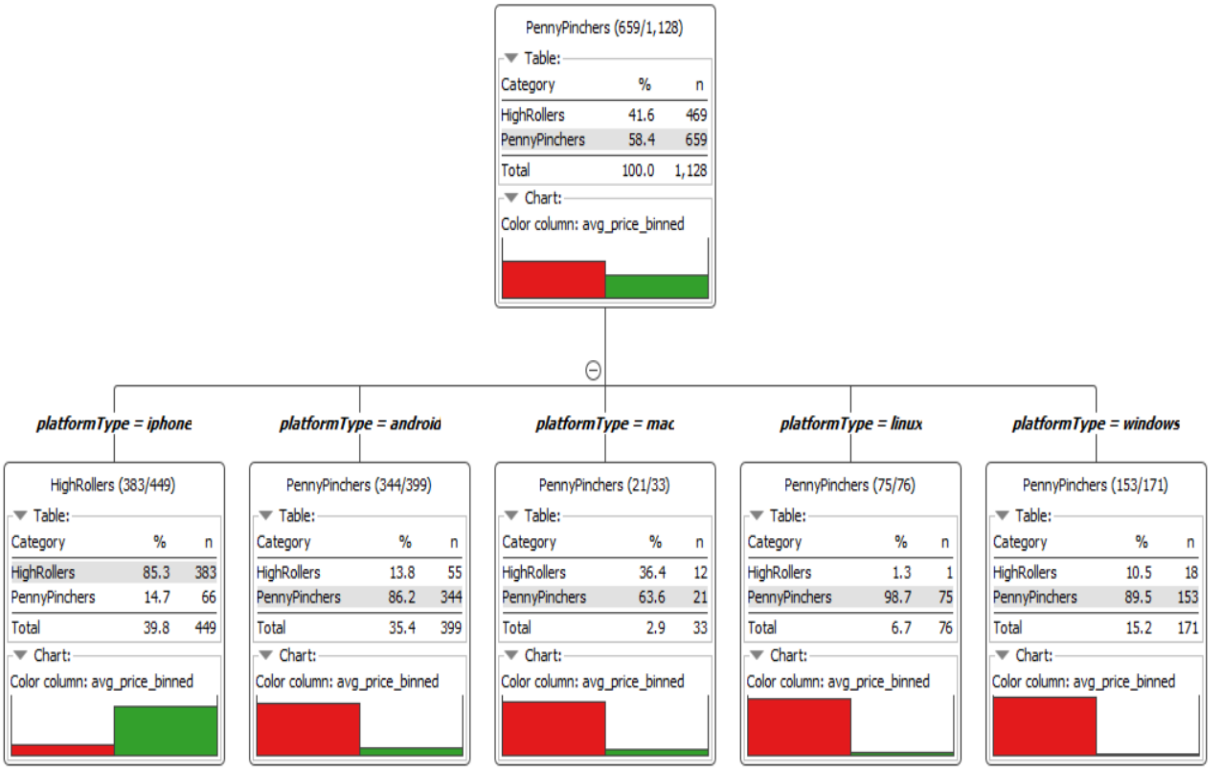
This is important because

- ❖ **Partitioning the data set into a "trained and tested" data sets, ensures that the data trained in the model is accurate.**

When partitioning the data using sampling, it is important to set the random seed because

- ❖ **Allows you to configure the node in a way that each execution of the node with the same configuration or input produces the same results.**

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



PennyPinchers (659/1,128)

Table:

| Category | % | n |
|---|---|---|
| HighRollers | 41.6 | 469 |
| PennyPinchers | 58.4 | 659 |
| Total | 100.0 | 1,128 |

Chart:

Color column: avg_price_binned

HighRollers (383/449)

Table:

| Category | % | n |
|---|---|---|
| HighRollers | 85.3 | 383 |
| PennyPinchers | 14.7 | 66 |
| Total | 39.8 | 449 |

Chart:

Color column: avg_price_binned

PennyPinchers (344/399)

Table:

| Category | % | n |
|---|---|---|
| HighRollers | 13.8 | 55 |
| PennyPinchers | 86.2 | 344 |
| Total | 35.4 | 399 |

Chart:

Color column: avg_price_binned

PennyPinchers (21/33)

Table:

| Category | % | n |
|---|---|---|
| HighRollers | 36.4 | 12 |
| PennyPinchers | 63.6 | 21 |
| Total | 2.9 | 33 |

Chart:

Color column: avg_price_binned

PennyPinchers (75/76)

Table:

| Category | % | n |
|---|---|---|
| HighRollers | 1.3 | 1 |
| PennyPinchers | 98.7 | 75 |
| Total | 6.7 | 76 |

Chart:

Color column: avg_price_binned

PennyPinchers (153/171)

Table:

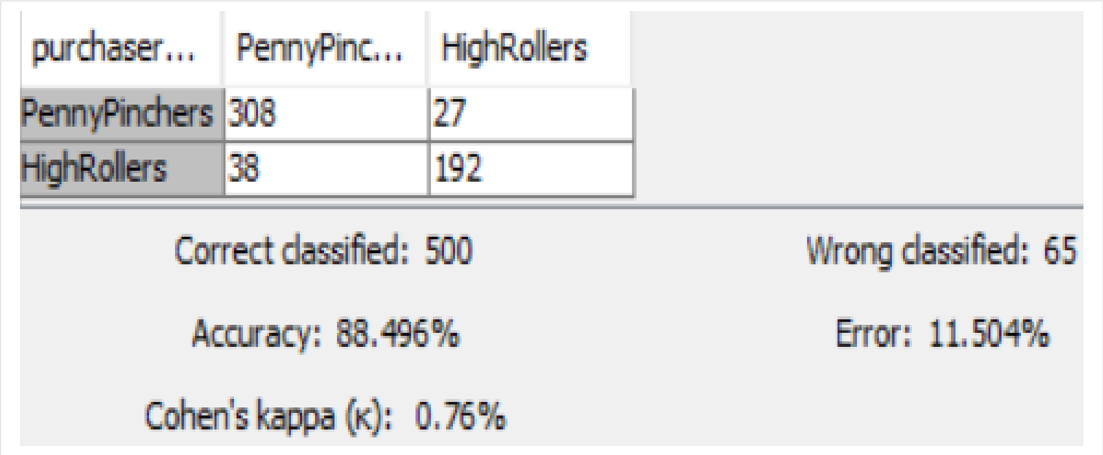
| Category | % | n |
|---|---|---|
| HighRollers | 10.5 | 18 |
| PennyPinchers | 89.5 | 153 |
| Total | 15.2 | 171 |

Chart:

Color column: avg_price_binned

## Evaluation

A screenshot of the confusion matrix can be seen below:



| purchaser... | PennyPinc... | HighRollers |
|---|---|---|
| PennyPinchers | 308 | 27 |
| HighRollers | 38 | 192 |

Correct classified: 500 Wrong classified: 65

Accuracy: 88.496% Error: 11.504%

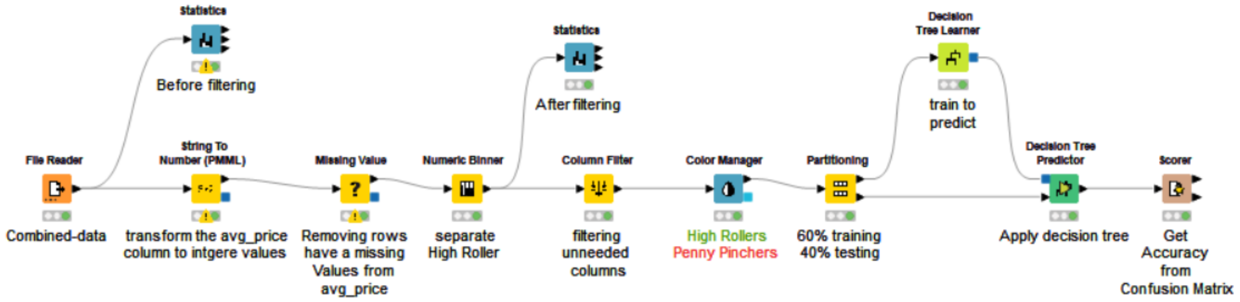
Cohen's kappa (κ): 0.76%

As seen in the screenshot above, the overall accuracy of the model is **88.5%**

- ✓ **When the purchaser type is penny pinchers the model classified correctly 308 times and incorrectly 27 times and when it is high rollers the model classified correctly 192 times and incorrectly 38 times.**

# Analysis Conclusions

The final KNIME workflow is shown below:



What makes a HighRoller vs. a PennyPincher?

- **The OS "platform_type" that the High Rollers use iOS and the Penny Pinchers use Mac, Win, Android and Linux.**

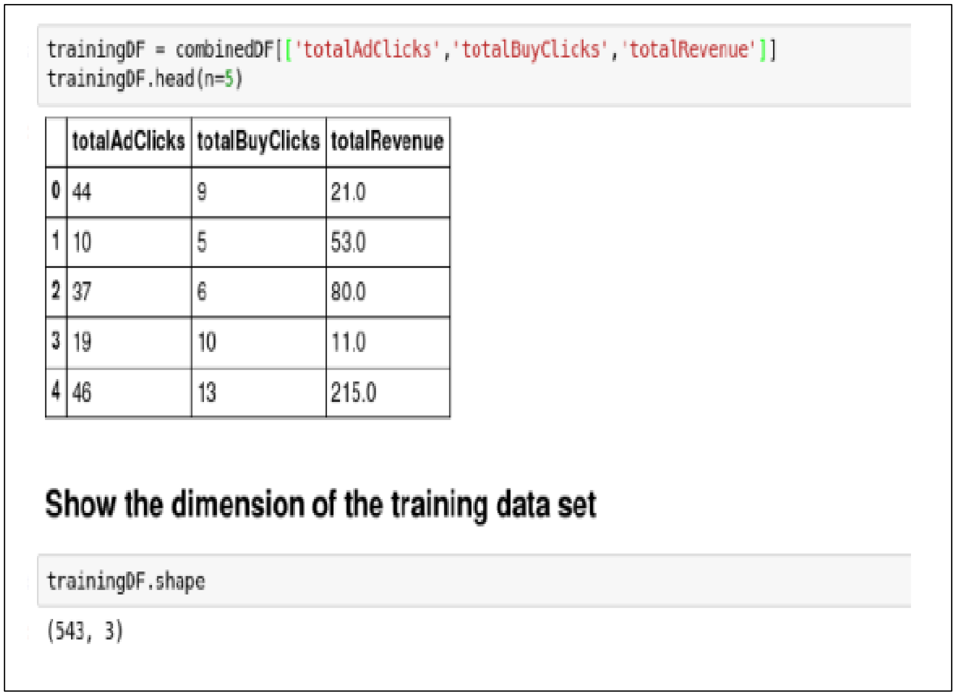
| Specific Recommendations to Increase Revenue |
|---|
| 1. **Targeting players who use (IOS) to advertise a specific purchase package customized for them.** |
| 2. **Making Game develop toward ios with special features.** |
| 3. **Designing an advertisement for playing the game, where the game used is filmed on an iOS device.** |

## Attribute Selection

| Attribute | Rationale for Selection |
|---|---|
| **totalAdClicks** | **Total of ad-clicks per user. This attribute is correlated to the profit's company.** |
| **totalBuyClicks** | **Total money of in-app purchase per user. This attributes is correlated to the profit's company.** |
| **totalRevenue** | **Total money spent on in-app purchase items per user.** |

## Training Data Set Creation

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



```
trainingDF = combinedDF[['totalAdClicks','totalBuyClicks','totalRevenue']]
trainingDF.head(n=5)
```

| | totalAdClicks | totalBuyClicks | totalRevenue |
|---|---|---|---|
| 0 | 44 | 9 | 21.0 |
| 1 | 10 | 5 | 53.0 |
| 2 | 37 | 6 | 80.0 |
| 3 | 19 | 10 | 11.0 |
| 4 | 46 | 13 | 215.0 |

### Show the dimension of the training data set

```
trainingDF.shape
```

```
(543, 3)
```

Dimensions of the final data set: **543 rows x 3 columns**

# of clusters created: **3**

# Cluster Centers

Cluster centers formed are given in the table below

| Cluster # | Center[totalAdClics,totalBuyClicks,totalRevenue] |
|---|---|
| 1 | [41.07, 10.29, 145.51] |
| 2 | [34.28, 6.45, 67.22] |
| 3 | [26.30, 4.48, 17.07] |

These clusters can be differentiated from each other as follows:

Cluster 1 is different from the others in that the players in the cluster have the highest 'totalAdClics', 'totalBuyClicks' and 'totalRevenue'.

Cluster 2 is different from the others in that the players in the cluster have the second highest 'totalAdClics', 'totalBuyClicks' and 'totalRevenue'.

Cluster 3 is different from the others in that the players in the cluster have the lowest 'totalAdClics', 'totalBuyClicks' and 'totalRevenue'

# Recommended Actions

| Action Recommended | Rationale for the action |
|---|---|
| Increase the prices for advertisements showed to players into first cluster | Players into the first cluster are frequent ad-clickers and increase the price of their ad, could increase the company's revenue. |
| Charge players into third cluster lower fees for the price of the in-app purchase items | Players into the third cluster only purchase items with lower prices. Lowering the price of the in-app purchase or giving them coupons could encourage them to spend more. |

# Graph Analytics

## Modeling Chat Data using a Graph Data Model

The graph model is a network based on chat interactions between users.

A chat session can be initiated by a user, other users on the same team are able to join and leave the session.

Interactions between users begins when a user create a post.

it's possible for a user, mention another user.

All relationship between entities are logged with timestamp.

## Creation of the Graph Database for Chats

The schema of the 6 CSV files:

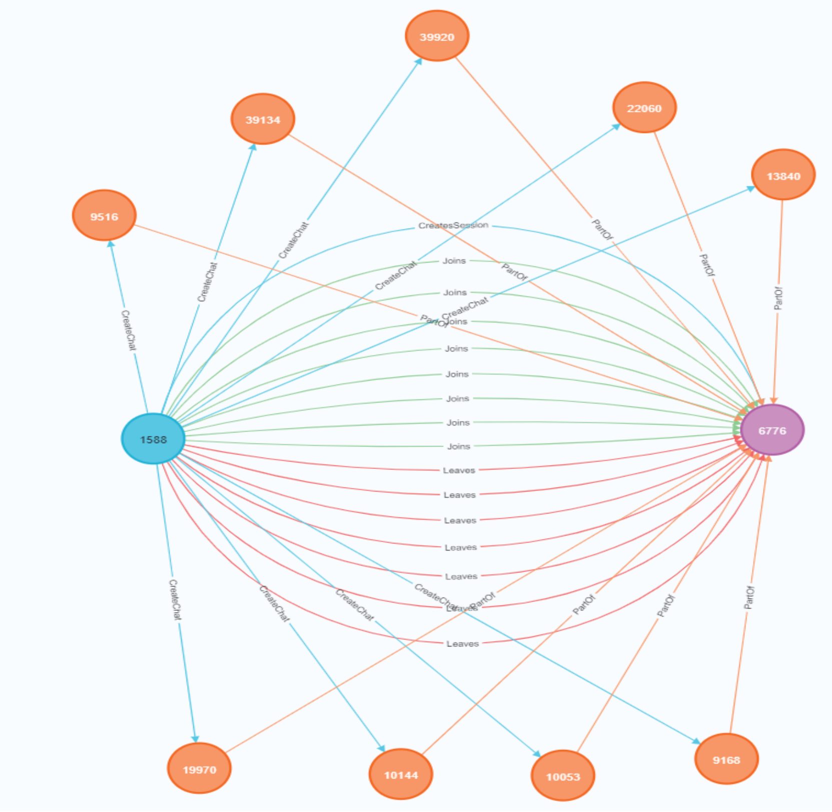
| Files | Fields |
|---|---|
| chat_create_team_chat | -userid<br>-teamid<br>-teamChatSessionID<br>-timestamp |
| chat_item_team_chat | -userid<br>-teamchatsessionid<br>-chatitemid<br>-timestamp |
| chat_join_team_chat | -userid<br>-teamChatSessionID<br>-teamstamp |
| chat_leave_team_chat | -userid<br>-teamchatsessionid<br>-timestamp |
| chat_mention_team_chat | -chatItem<br>-userid<br>-timeStamp |
| chat_respond_team_chat | -chatid1<br>-chatid2<br>-timestamp |

Explain the loading process and include a sample LOAD command:

```
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)
```

**The first line imports the csv file from specific path. second, third and fourth lines creates nodes with different labels (User, Team, TeamChatSession) and have id properties. fifth and sixth lines create a relatiosship under lables specified like this:**
**Between User nodes and TeamChatSessionnodes called CreateSession**
**Between TeamChatSession nodes and Team nodes called OwnedBy.**

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 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:

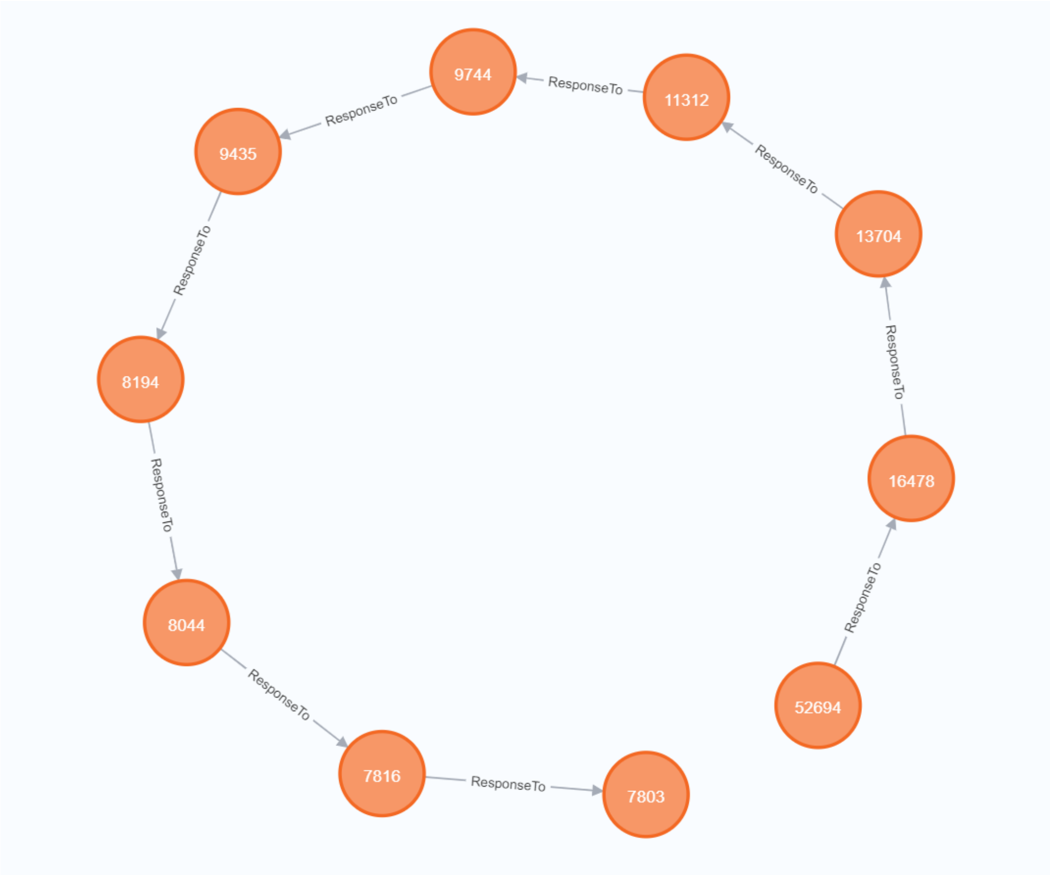


## Finding the longest conversation chain and its participants

Report the results including the length of the conversation (path length) and how many unique users were part of the conversation chain. Describe your steps. Write the query that produces the correct answer:

**How many chats are involved between tow nodes?**

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



**The longest conversation chain in the chat data has path length (9) and there is (10) chats are involved on it.**

**How many Users are participated in this chain?**

```
match p=(c:ChatItem)-[:ResponseTo*]->(j:ChatItem)
where length(p) = 9
with p
match q=(u:User)-[:CreateChat]-(c:ChatItem)
where (c in NODES(p))
return count(DISTINCT u)
```

**With (9) as longest path, count the number of distinct Users who create a ChatItem in the longest path and the query result is (5) Users.**

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

Describe your steps from Question 2. In the process, create the following two tables. You only need to include the top 3 for each table. Identify and report whether any of the chattiest users were part of any of the chattiest teams.

**Chattiest Users:**

```
match (u:User)-[:CreateChat]-(i:ChatItem)
return u.id as Users , count(u.id) as Num_Chats
order by count(u.id) desc limit 10
```

| Users | Number of Chats |
|---|---|
| **394** | **115** |
| **2067** | **111** |
| **1087** | **109** |
| **209** | **109** |
| **554** | **107** |
| **1627** | **105** |
| **999** | **105** |

| | |
|---|---|
| 516 | 105 |
| 668 | 104 |
| 461 | 104 |

## Chattiest Teams:

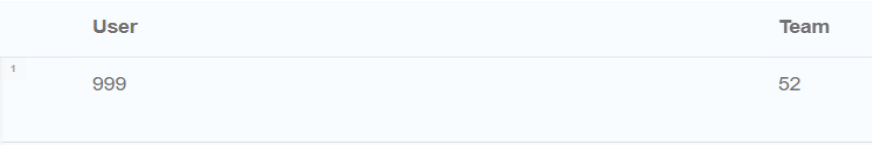
```
match (:ChatItem)-[:PartOf]-(:TeamChatSession)-[:OwnedBy]-(t:Team)
return t.id as Teams , count(t.id) as Num_Chats
order by count(t.id) desc limit 10
```

| Teams | Number of Chats |
|---|---|
| 82 | 1324 |
| 185 | 1036 |
| 112 | 957 |
| 18 | 844 |
| 194 | 836 |
| 129 | 814 |
| 52 | 788 |
| 136 | 783 |
| 146 | 746 |
| 81 | 736 |

Finally, present your answer, i.e. whether or not any of the chattiest users are part of any of the chattiest teams.

```
match (u:User)-[:CreateChat]-(:ChatItem)-[:PartOf]-(:TeamChatSession)-[:OwnedBy]-(t:Team)
where u.id in [394,2067,1087,209,554,1627,999,516,668,461]
and t.id in [82,185,112,18,194,129,52,136,146,81]
return distinct u.id as User , t.id as Team
```

The result be like:



|  | User | Team |
|---|---|---|
| 1 | 999 | 52 |

## 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.

connect mention Users:

```
match (u1:User)-[:CreateChat]->(:ChatItem)-[:Mentioned]->(u2:User)
merge (u1)-[:InteractsWith]->(u2)
```

connect Users response with the chat creator:

```
match (u1:User)-[:CreateChat]->(:ChatItem)-[:ResponseTO]-(:ChatItem)<-[:CreateChat]-(u2:User)
merge (u1)-[:InteractsWith]->(u2)
```

Eliminate all self-interactions:

```
match (u1)-[r:InteractsWith]->(u2) delete r
```

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

```
match (u:User {id=349})-[:InteractsWith]-(u2:User)
with collect(u2.id) as neighbours, count(u2) as k
match (u3:User)-[iw:InteractsWith]-(u4:User)
where (u3.id in (neighbours)) and (u4.id in (neighbours))
return count(iw)/(k * (k - 1) * 1.0) as clusterCoefficient
```

| User ID | Coefficient |
|---|---|
| **394** | **0.9167** |
| **2067** | **0.7679** |
| **209** | **0.9524** |