Data Exploration

1. 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 ERD table: AdClicks	A line is added to this file when a player clicks on an advertisement in the Flamingo app.	timestamp, txID, userSessionid, teamid, userid, adID, adCategory **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 ERD table: InAppPurchases	A line is added to this file when a player makes an in-app purchase in the Flamingo app.	timestamp, txID, userSessionid, team, userid, buyid, price 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
users.csv ERD table: User	This file contains a line for each user playing the game.	timestamp, id, nick, twitter, dob, country 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 **ERD table: Team**	This file contains a line for each team terminated in the game.	teamid, name, teamCreationTime, teamEndTime, strength, currentLevel 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

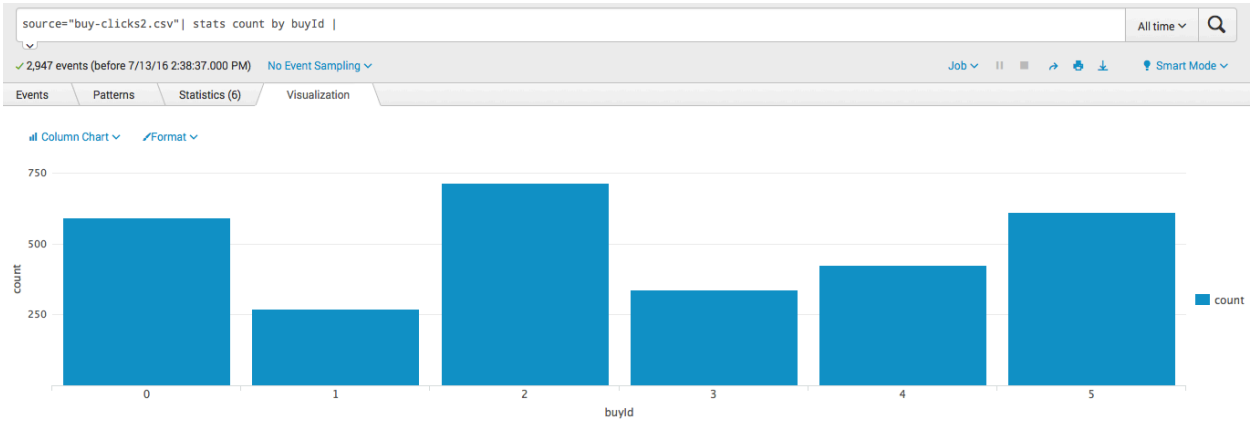
team-assignments.csv **ERD table: TeamAssignment**	A line is added to this file each time a user joins a team. A user can be in at most a single team at a time.	time, team, userid, assignmentid time: when the user joined the team. team: the id of the team userid: the id of the user assignmentid: a unique id for this assignment
level-events.csv **ERD table: LevelEvent**	A line is added to this file each time a team starts or finishes a level in the game	time, eventid, teamid, level, eventType 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 **ERD table: User_Sessions**	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,userSessionId,userId,teamId,assignmentId,sessionType,teamLevel,platformType 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.
game-clicks.csv **ERD table: GameClicks**	A line is added to this file each time a user performs a click in the game.	time, clickid, userid, usersessionid, isHit, teamId, teamLevel 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

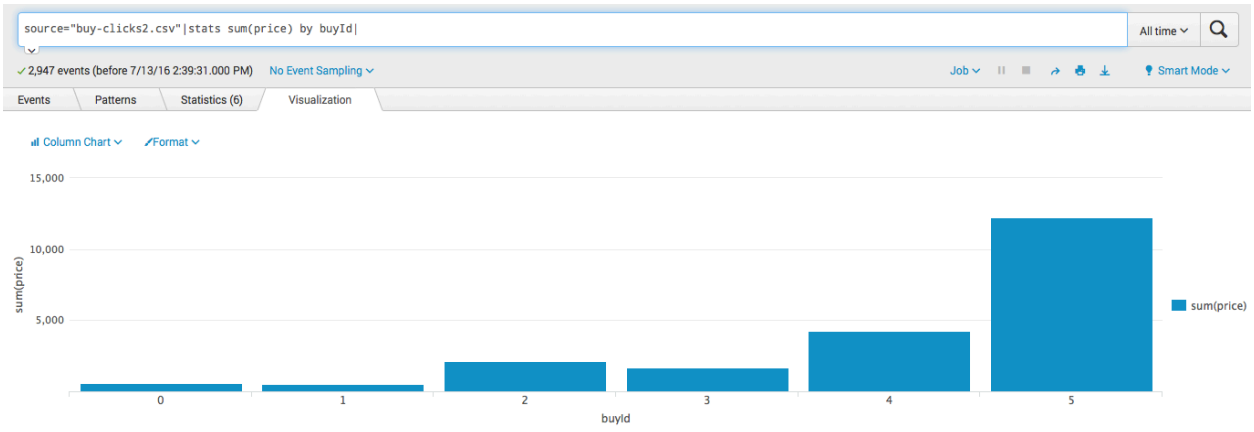
Aggregation

Amount spent buying items	21407.0
# Unique items available to be purchased	6

A histogram showing how many times each item is purchased:
(source="buy-clicks.csv"| stats count by buyId |)

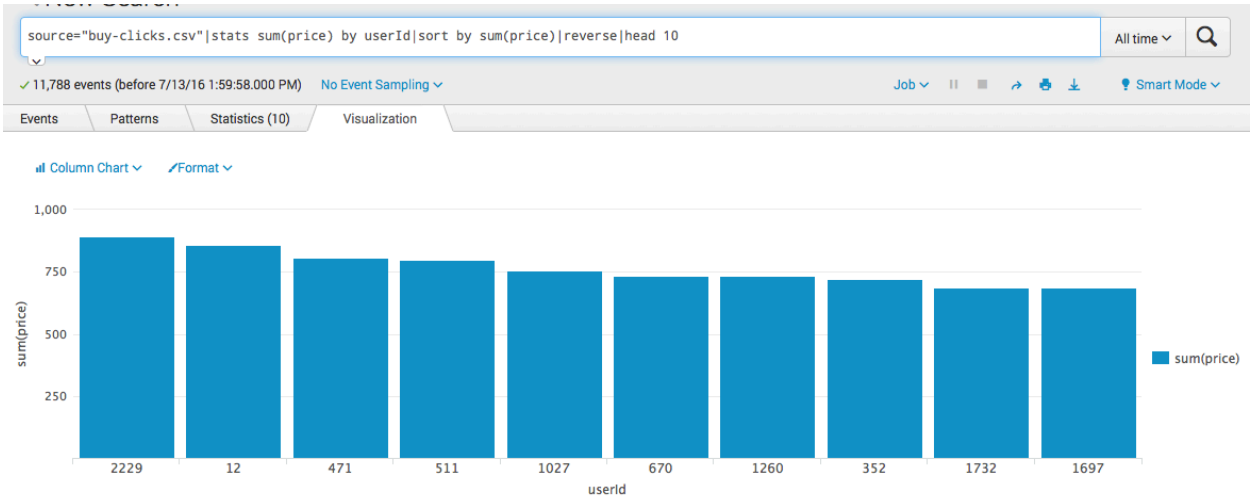


A histogram showing how much money was made from each item:
(source="buy-clicks.csv"|stats sum(price) by buyId|)



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 by sum(price)|reverse|head 10)



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

(source="user-session.csv" userId=2229|stats count by platformType
source="user-session.csv" userId=12|stats count by platformType
source="user-session.csv" userId=471|stats count by platformType
source="game-clicks.csv" userId=2229|stats avg(isHit)|
source="game-clicks.csv" userId=12|stats avg(isHit)|
source="game-clicks.csv" userId=471|stats avg(isHit)|)

Rank	User Id	Platform	Hit-Ratio (%)
1	2229	iphone	11.5
2	12	iphone	13.0
3	471	iphone	14.5

2. Data Preparation

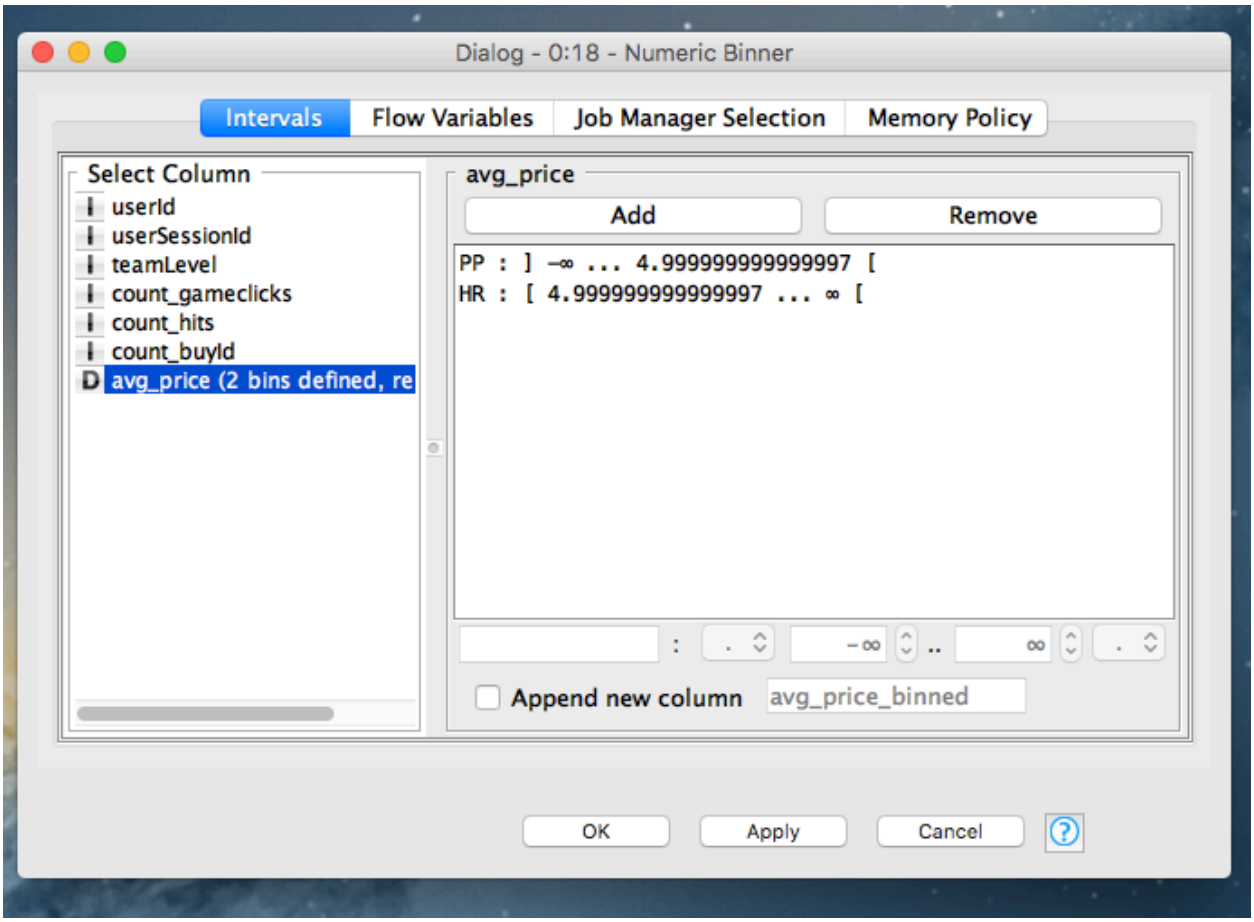
Analysis of combined_data.csv

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:



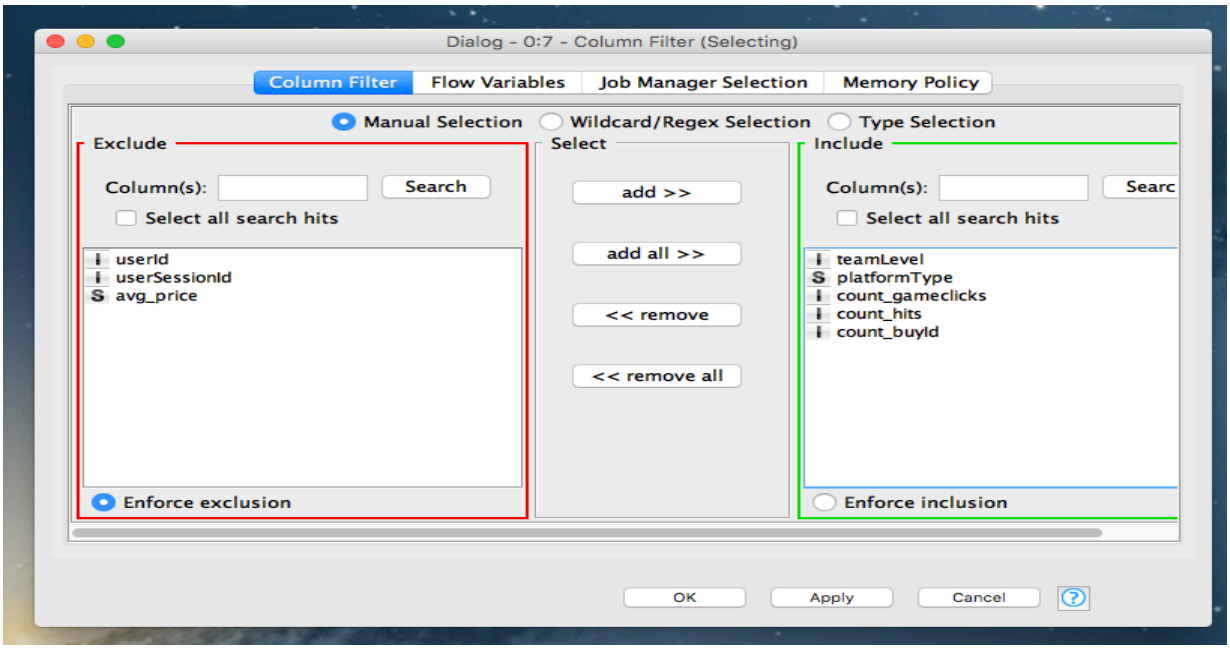
Description for how attribute was designed:
I uploaded the data and filtered those rows where count_buyId is nonzero. Then I applied Math formula (Avg_price >=5) to create binary attribute variable called HighRollers. The attribute value =1 represents HighRolles and value 0 represents PennyPinchers

The creation of this new categorical attribute was necessary because <Fill in 1-2 sentences>.

Attribute Selection

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

Attribute	**Rationale for Filtering**
userId, userSessionId and avg_price	The filters that do not help accurately predict the target should be removed.
userId	Highly-branching attribute. Decision tree bias toward high information gain, so this attribute can prevent generalized pattern.
userSessionId	Highly-branching attribute. Decision tree bias toward high information gain, so this attribute can prevent generalized pattern.
avg_price	The categorical attribute using binning was created based on this attribute. So if this attribute is included than there is nothing to predict, the model will have all the information and will have 100% accuracy.

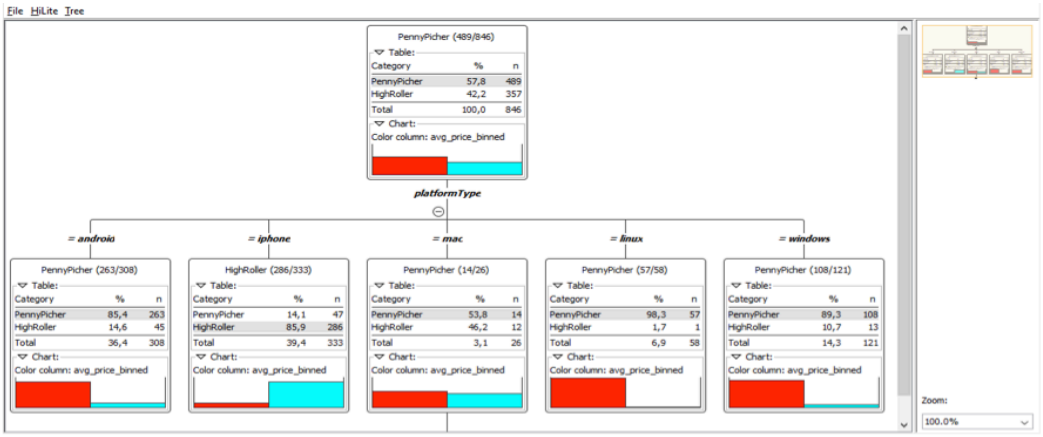


Data Partitioning and Modeling

The data was partitioned into train and test datasets.
The **train data (60%)** data set was used to create the decision tree model.
The trained model was then applied to the **test data (40%)** dataset.
This is important because we need to check accuracy of our model and improve the model performance by testing it on test set of data that was not used to train a model.

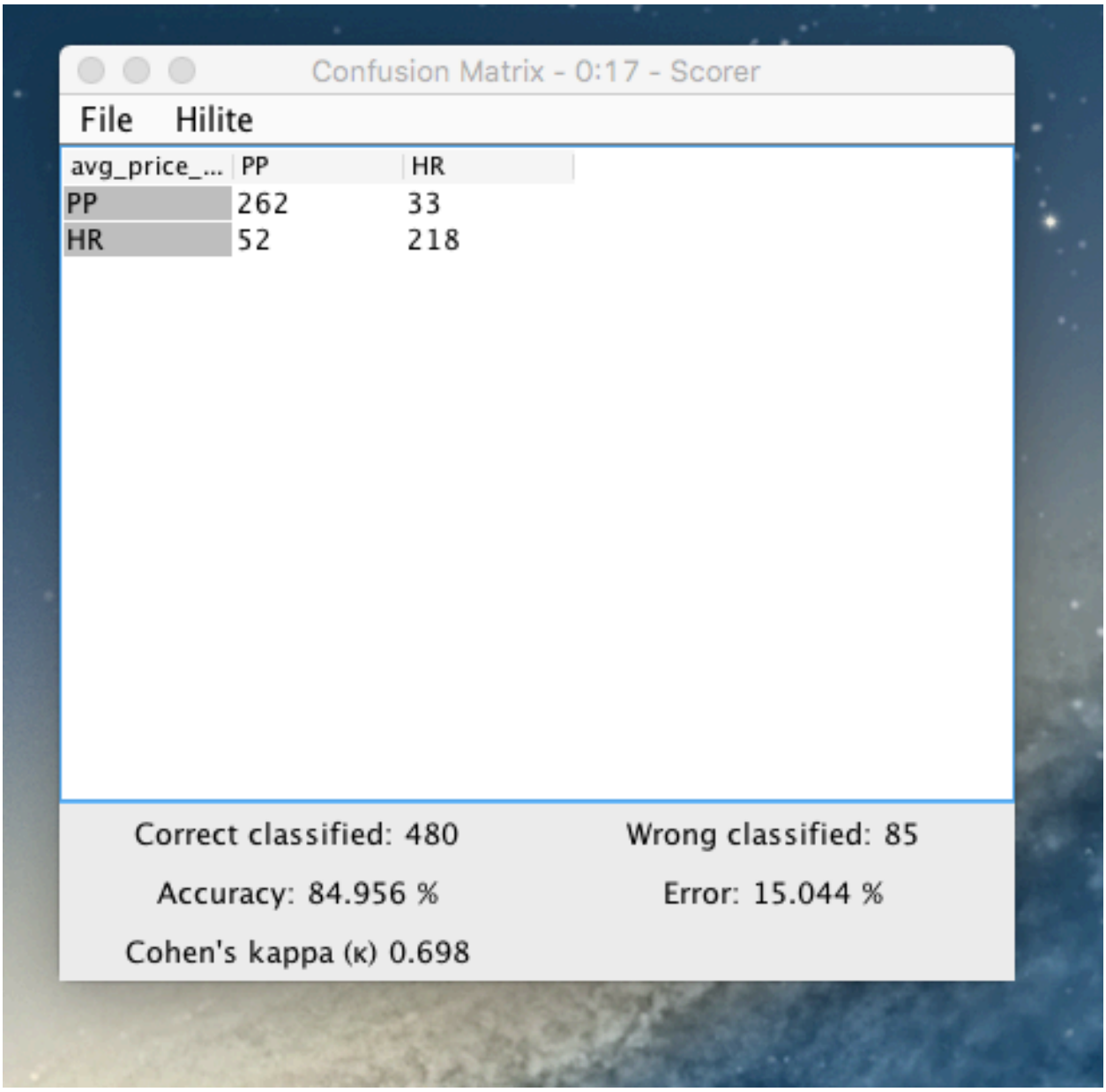
When partitioning the data using sampling, it is important to set the random seed because this ensures that you will get the same partitions every time you execute this node. This is important to get reproducible results.

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



Evaluation

A screenshot of the confusion matrix can be seen below:



As seen in the screenshot above, the overall accuracy of the model is 100**%**

Confusion Matrix

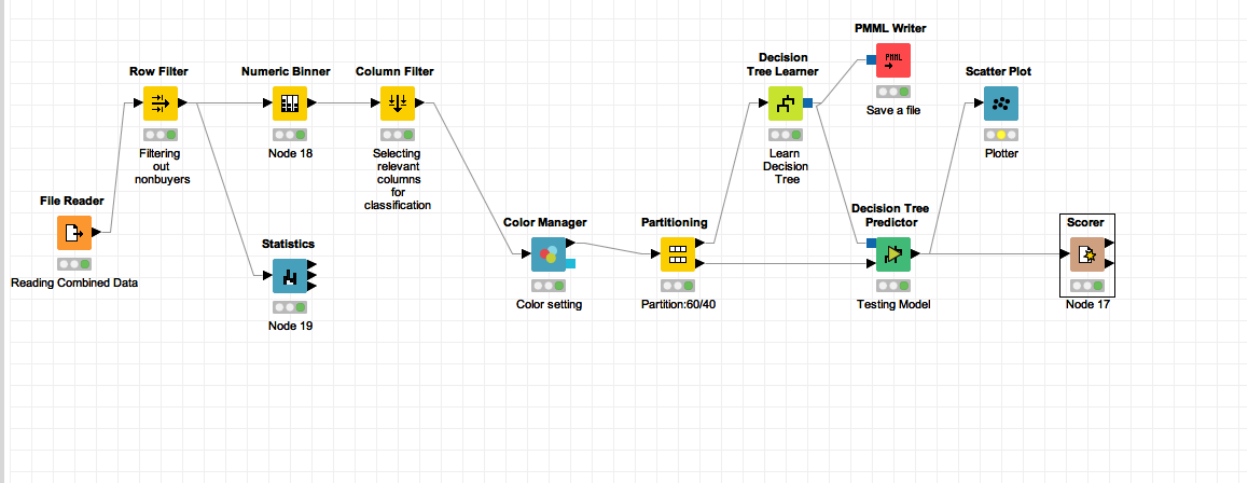
	PennyPitchers	HighRollers
PennyPitchers(PP)	True Positive TP (262)	False Negative (33)
HighRollers(HR)	False Positive (52)	True Negative (218)

So in this case, the
Correctly predicted, PennyPitchers = 262 and Correctly predicted HighRollers = 218
Incorrectly predicted, PennyPitchers = 52 and incorrectly predicted HighRollers = 33

Accuracy = (TP + TN) / (TP +TN + FP + FN)
= 84.956

Analysis Conclusions

The final KNIME workflow is shown below:



What makes a HighRoller vs. a PennyPincher?

The PlatformType is the main decision factor on deciding who wiis going to be HighRoller Vs. PennyPincher. From the prediction we see that users who uses iphone spends more amount and are HighRoller vs the users who play game using devices on other operating system (i.e. android, window, mac or linux etc).

Specific Recommendations to Increase Revenue
1. Since iphone users are HighRollers, the game offers should be high values if platformType is iphone, since that will be more appealing to them and are more likely to buy them.
2. Since other operating system device (i.e. android, mac, windows etc) and linux are generally PennyPinchers, they should be given low values. It is possible that they will purchase low value items but in more volume, so there should be some other strategy.

References:

1. https://www.youtube.com/watch?v=RHsO10q7e2Y

Attribute Selection

Attribute	Rationale for Selection
Price	Characteristics of a user's purchase behavior. It is directly associated to profit made from user.
adCount	Ad clicks represents user's purchase frequency as well as add frequency. This represents dynamic relationship between users, add providers and host.

Training Data Set Creation

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

```
In [25]: combinedDF.head(n=5) #display how the merged table looks
Out[25]:
   userId  totalAdClicks  revenue
0       1             44     21.0
1       8             10     53.0
2       9             37     80.0
3      10             19     11.0
4      12             46    215.0
```

Dimensions of the training data set (rows x columns) : (543,2)
# of clusters created: 2

Cluster Centers

These clusters can be differentiated from each other as follows:

# of clusters	Cluster
2	(27.39, 23.86), (39.08, 115.26)
3	(34.62, 59.29), (40.87, 138.24), (25.33, 15.29)
4	(40.63, 23.53), (41.09, 146.39), (33.45, 69.41), (14.85, 12.65)

3 cluster analysis is good one because there is over fitting problem in 4-cluster analysis.

Cluster 1 is different from the others in that, in general, players click on ads and spend more. The click: spending ratio is roughly 34:59.

Cluster 2 is different from the others in that, in general, players click on ads and spend much more. The click: spending ratio is roughly 40:138.

Cluster 3 is different from the others in that, in general, players click on ads more and spend less. The click: spending ratio is roughly 25:15.

Recommended Actions

Action Recommended	**Rationale for the action**
Priority to high 'click: revenue' ratio	Priority must be give to those users who are included in the cluster having high 'click: revenue' ratio.
Priority to low 'click: revenue' ratio by selective age	Even though the ration of click: revenue is less, there is important information in age of the users, platform used and geographical regions. Further clustering is required in this case to improve the performance.

Graph Analytics

Modeling Chat Data using a Graph Data Model

(Describe the graph model for chats in a few sentences. Try to be clear and complete.)
This graph model based on chat data consists of nodes

1. User
2. Team
3. ChatItem

4. TeamChatSession

And different edges for different relationship connecting two nodes like:

1. CreatChat,
2. CreatSession
3. Joins
4. Leaves
5. OwnedBy
6. PartOf
7. Mentioned
8. RespondTo

Creation of the Graph Database for Chats

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

i) Write the schema of the 6 CSV files

1. chat_create_team_chat.csv:

Entity	Type	Column
userId	int	0
TeamId	int	1
TeamChatSessionId	int	2
timestamp	time	3

2. chat_item_team_chat.csv:

Entity	Type	Column
userId	int	0
TeamChatSessionId	int	1
chatId	int	2
timestamp	time	3

3. chat_join_team_chat.csv:

Entity	Type	Column
userId	int	0
TeamChatSessionId	int	1
timestamp	time	2

4. chat_leave_team_chat.csv

Entity	Type	Column
userId	int	0
TeamChatSessionId	int	1
timestamp	time	2

5. chat_mention_team_chat.csv

Entity	Type	Column
chatId	int	0
userId	int	1
timestamp	time	2

(ii). Explain the loading process and include a sample LOAD command

LOAD CSV is used to load the neo4j local server database.
Loading process is as follows:

1. These first 5 lines create the nodes and the rest load the files and create the edges.

```
CREATE CONSTRAINT ON (u:User) ASSERT u.id IS UNIQUE;
CREATE CONSTRAINT ON (t:Team) ASSERT t.id IS UNIQUE;
CREATE CONSTRAINT ON (c:TeamChatSession) ASSERT c.id IS UNIQUE;
CREATE CONSTRAINT ON (i:ChatItem) ASSERT i.id IS UNIQUE;
CREATE CONSTRAINT ON (j:ChatItem) ASSERT j.id IS UNIQUE;
```

2. Merge command fill up nodes and creates edges between nodes previously created. We are going to upload 6 files, which are kept in 'Neo4j/default.graphdb/import'

```
LOAD CSV FROM "file:/chat_create_team_chat.csv" AS row
MERGE (u:User {id: toInt(row[0])})
```

```
MERGE (t:Team {id: toInt(row[1])})
MERGE (c:TeamChatSession {id: toInt(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)
```

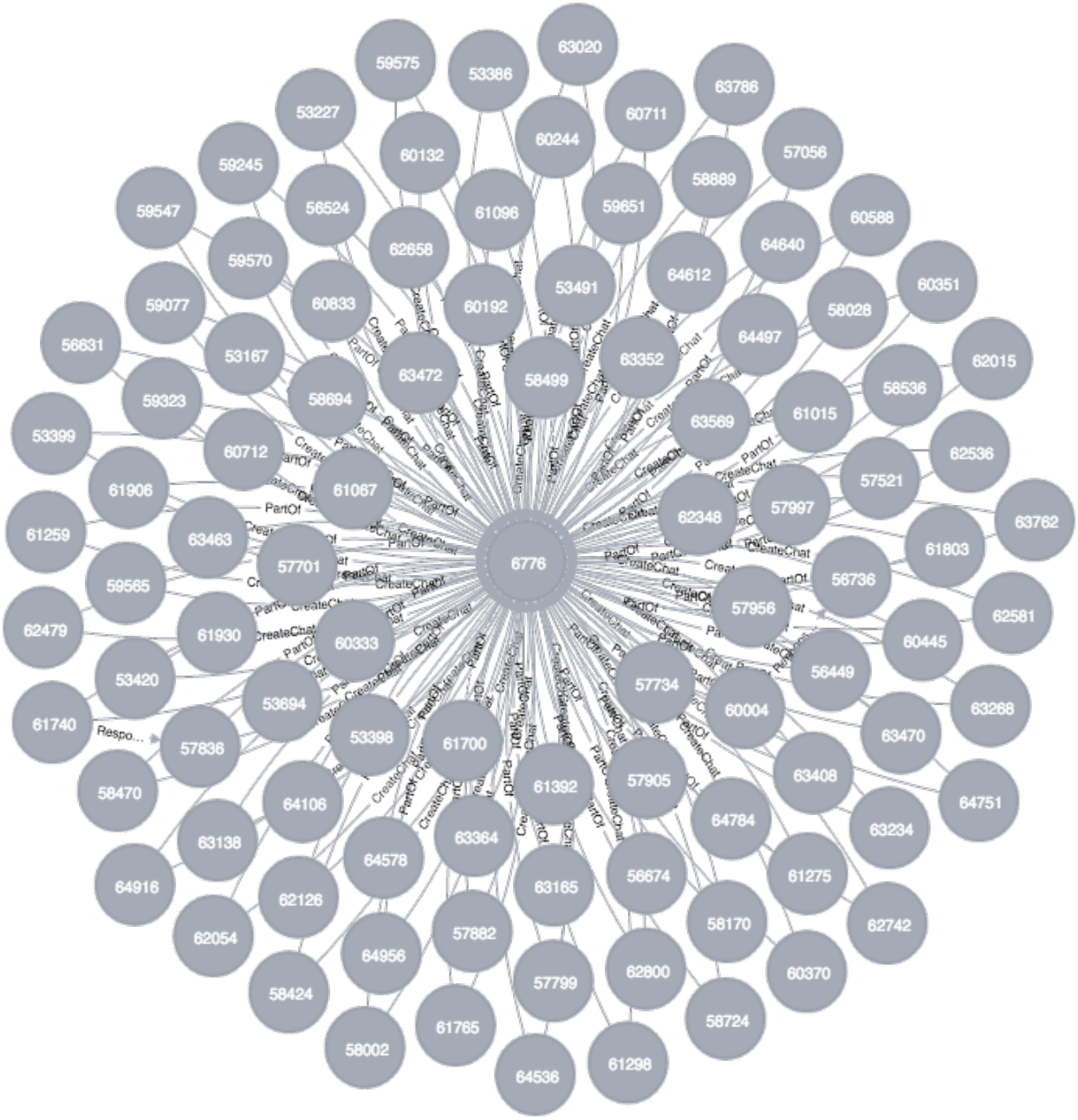
```
LOAD CSV FROM "file:/chat_leave_team_chat.csv" AS row
MERGE (u:User {id: toInt(row[0])})
MERGE (c:TeamChatSession {id: toInt(row[1])})
MERGE (u)-[:Leaves{timeStamp: row[2]}]->(c)
```

```
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 (i)-[:PartOf{timeStamp: row[3]}]->(c)
MERGE (i)-[:CreateChat{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 (i:ChatItem {id: toInt(row[0])})
MERGE (j:ChatItem {id: toInt(row[1])})
MERGE (i)-[:ResponseTo{timeStamp: row[2]}]->(j)
```

(iii) . 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.



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.

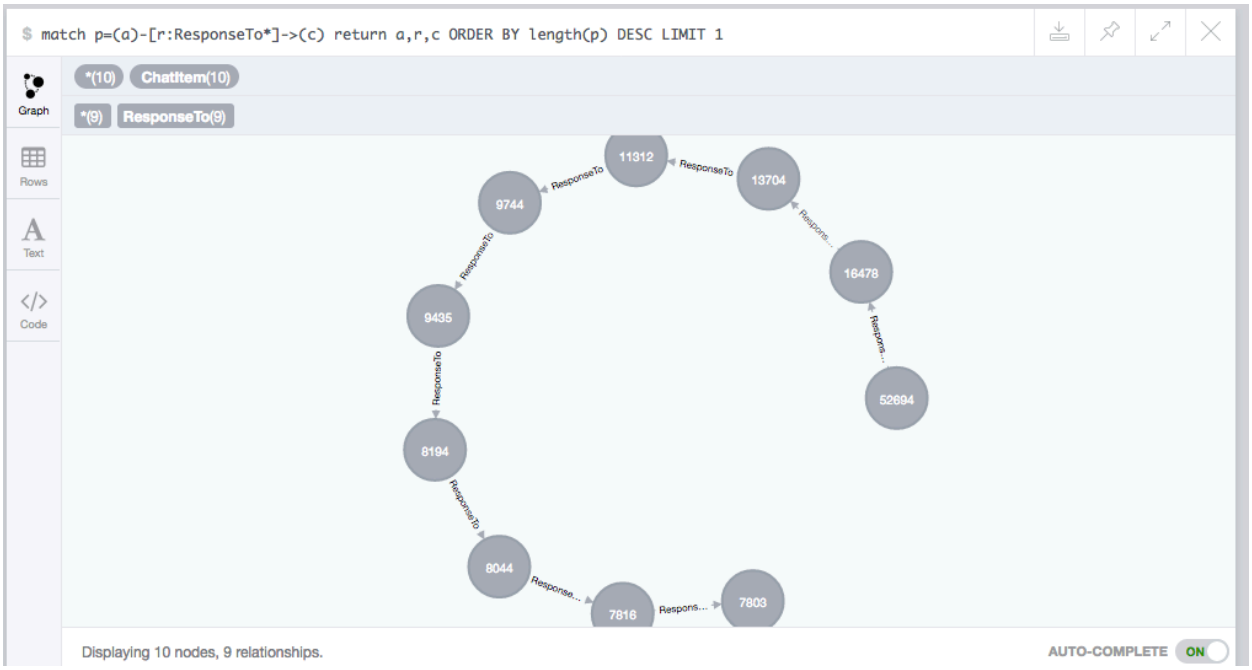
Query to path length

```
MATCH p=(a)-[r:ResponseTo*]->()
RETURN a,r ORDER BY length(p) DESC LIMIT 1
```

Path length = 9

Query to path plot

```
MATCH p=(a)-[r:ResponseTo*]->()
RETURN a,r ORDER BY length(p) DESC LIMIT 1
```



Unique users involved in the conversation: 5

Query to unique users:

```
MATCH p=(a)-[:ResponseTo*]->()
WHERE length(p) = 9
WITH p  MATCH (u)-[:CreateChat]->(i)
WHERE i in nodes(p)
RETURN count(distinct u)
```

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

Process: We query users that connect to other user nodes with a CreateChat edge. The chattiest user is found by counting the outdegree of the node. We order the results in descending order and limiting to see top results.

```
MATCH (u)-[r:CreateChat*]-()
RETURN u, count(r) as Outdegree
ORDER BY Outdegree desc
LIMIT 10
```

Users	Number of Chats
394	115
2067	111

209,1087	109

Chattiest Teams

Process: To find the chattiest teams, **we** query teams that connect to TeamChatSession(c) nodes that in order connect to ChatItem(i) node. Then we compute the indegree of the team nodes, order the results in descending order and limit the top results

```
MATCH (i)-[r:PartOf*]->(c)-[o:OwnedBy]->(t)
RETURN t.id, COUNT(o) as Indegree
ORDER BY Indegree desc
LIMIT 10
```

Teams	Number of Chats
82	1324
185	1036
112	957

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

Process: To analyze if top chattiest users belong to chattiest teams, we first identified those chattiest user and team and put the restriction in each such graph by WHERE statement to find chattiest user in chattiest team

```
MATCH (u)-[: CreateChat]->(i)-[:PartOf]->(c)-[:OwnedBy]->(t)
WHERE u.id in [394, 2067, 209,1087,554,516,1627,999,668,461] and t.id in [82, 1085,
112,18,194,129,52,136,146,81]
RETURN DISTINCT u.id, t.id
```

Result: u.id = 999, t.id = 52 (i.e. Only one of the chattiest player 999 belongs to one of the Chattiest team 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.

Process and Commands:

1. First we construct the neighborhood relationship:

```
MATCH (u1:User)-[:CreateChat]->(i)-[:Mentioned]->(u2:User)
CREATE(u1)-[:InteractsWith]->(u2)
MATCH (u1)-[:CreateChat]->(i1)-[:RespondTo]->(i2)
WITH u1, i2 MATCH (u2)-[:CreateChat]->(i2)
CREATE (u1)-[:InteractsWith]->(u2)
```

2. Then we delete all the loops

```
MATCH (u)-[r:InteractsWith]->(u) DELETE r
```

3. To compute the Clustering Coefficient we follow the steps:

- Obtain the neighbors of each of the top chattiest users
- Compute all distinct edges linking each neighbor with other neighbors
- Divide the number of distinct edges by K*(K-1) where K is the number of neighbors

```
MATCH (u)-[r:InteractsWith]-(v)
WHERE u.id in [394, 2067,1087,209,554,516,1627,999,668,461]
WITH u.id as user,collect(DISTINCT v.id) as  neighbors,
COUNT(DISTINCT v) as noOfNeighbors
MATCH (n)-[r:InteractsWith]-(m)
WHERE
n.id in neighbors
and
m.id in neighbors
WITH user as user, collect(DISTINCT [n.id,m.id]) as listOfEdges,
(noOfNeighbors *(noOfNeighbors-1)) as den
RETURN user,length(listOfEdges)*1.0 / den as  clusteringCoefficient
ORDER BY clusteringCoefficient DESC
```

Most Active Users (based on Cluster Coefficients)

User ID	Coefficient
209	0.95
554	0.90
1087	0.8