Technical Appendix
Catch the Pink Flamingo Analysis

Produced by: Tihomir Nikolov

I. Acquiring, Exploring and Preparing the Data

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	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
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.
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
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
user-session.csv	Each line in this file describes a user session, which denotes when a user starts and stops playing the game.	timestamp: a timestamp denoting when the event occurred. userSessionId: a unique id for the session.

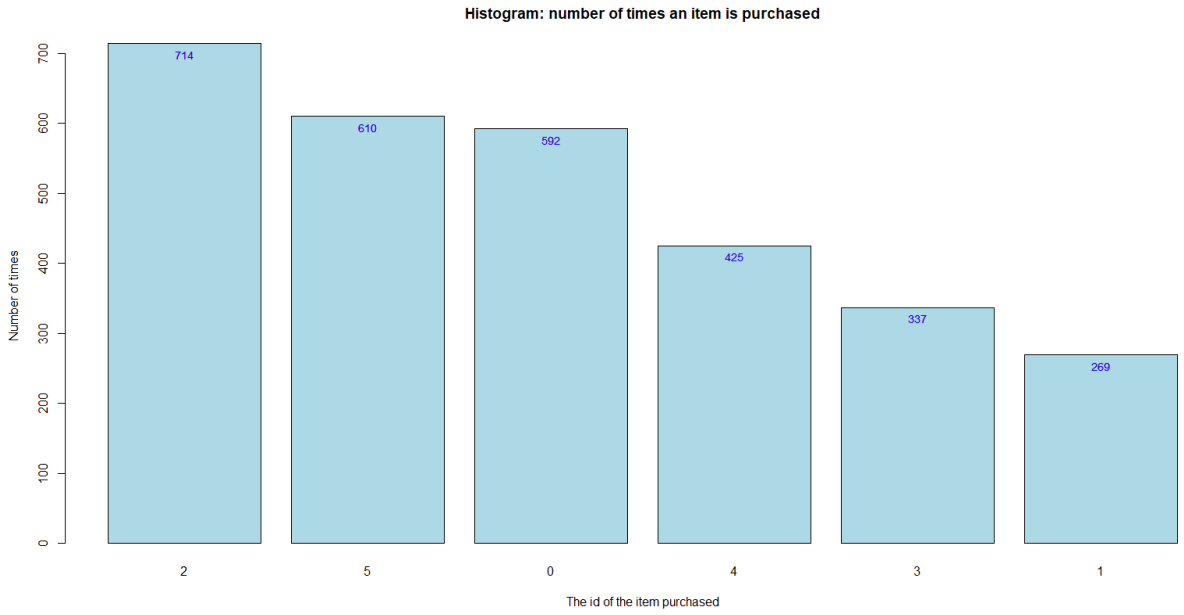
	Additionally, when a team goes to the next level in the game, the session is ended for each user in the team and a new one started.	userId: the current user's ID. teamId: the current user's team. assignmentId: the team assignment id for the user to the team. sessionType: whether the event is the start 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.
combined_data.csv	The combined_data.csv combines data from 3 of the log files: user-session.csv, buy-clicks.csv, and game-clicks.csv.	userId: the user id of the user who made the purchase userSessionId: the id of the user session for the user who made the purchase teamLevel: the level started or completed platformType: the type of platform of the user during this session. count_gameclicks: Total number of game clicks for user session count_hits: total number of game hits for user session count_buyid: total number of purchases for user session avg_price: average purchase price for user session

2. Aggregation

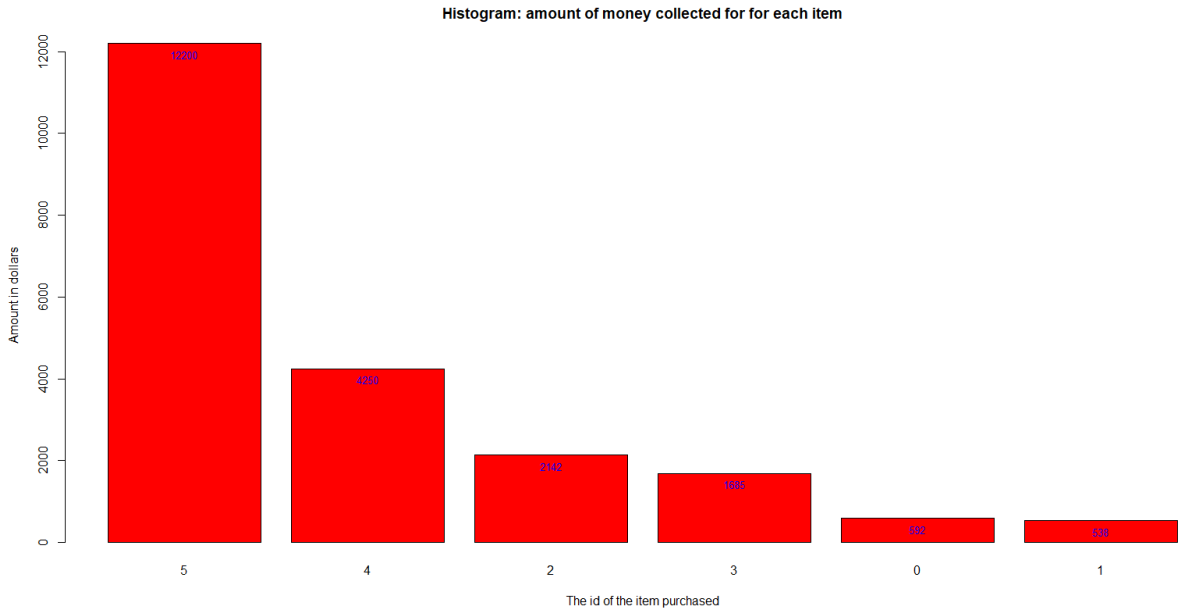
- Some general information

Amount spent buying items(in dollars)	21407
Number of unique items available to be purchased	6

- A histogram showing how many times each item is purchased:



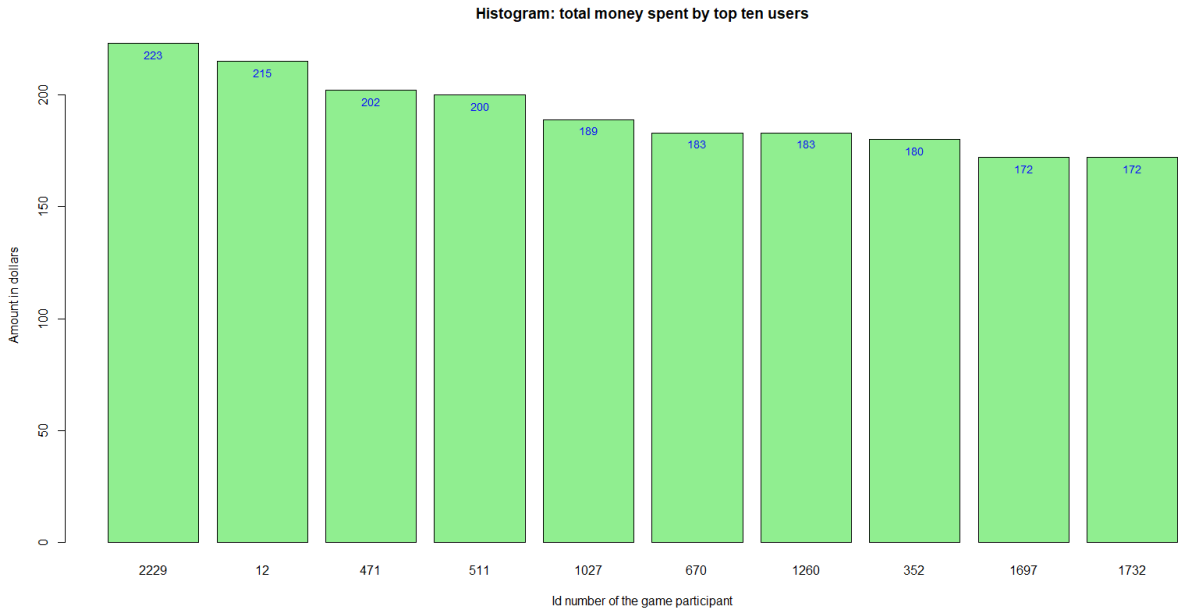
- A histogram showing how much money was made from each item:



We can see that the most frequently purchased item is not the most lucrative. That is the case since the prices vary, e.g. item number 2 cost only 2 dollars, whereas item number 5 earns 20 dollars. That explains why item number 5 is the cash cow, whiles item number 2 is the most frequently purchased.

2. 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	Hit-Ratio (%)	Platform
1	2229	11.59696	iphone
2	12	13.06818	iphone
3	471	14.50382	iphone

We can see that the hit ratio of the top three users is somewhat higher than the average hit ratio of the all participants, which is 11.03233, suggesting that how well an user performs in the game probably does influence his/her purchasing behavior. Better player probably spend more, while playing the game. On the other hand we might reasonably expect that the iphone users are more lavish spenders.

II . Data Classification Analysis

1. Data Preparation

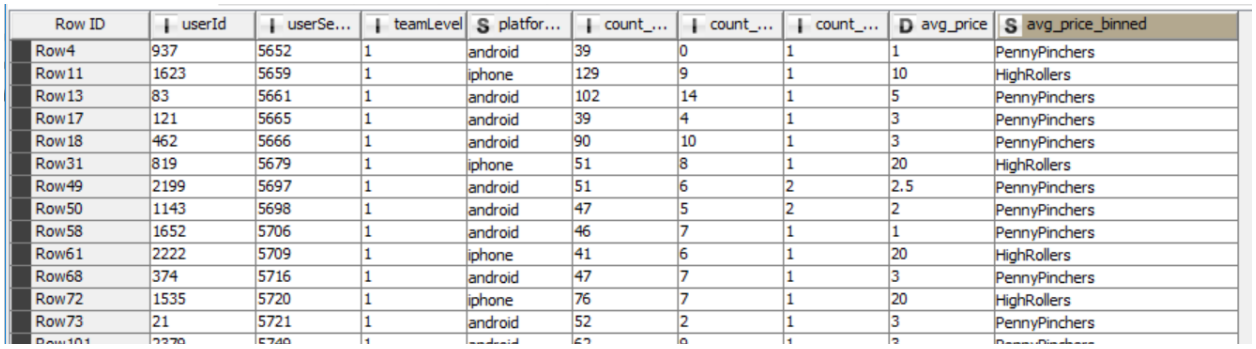
Analysis of combined_data.csv

Sample Selection

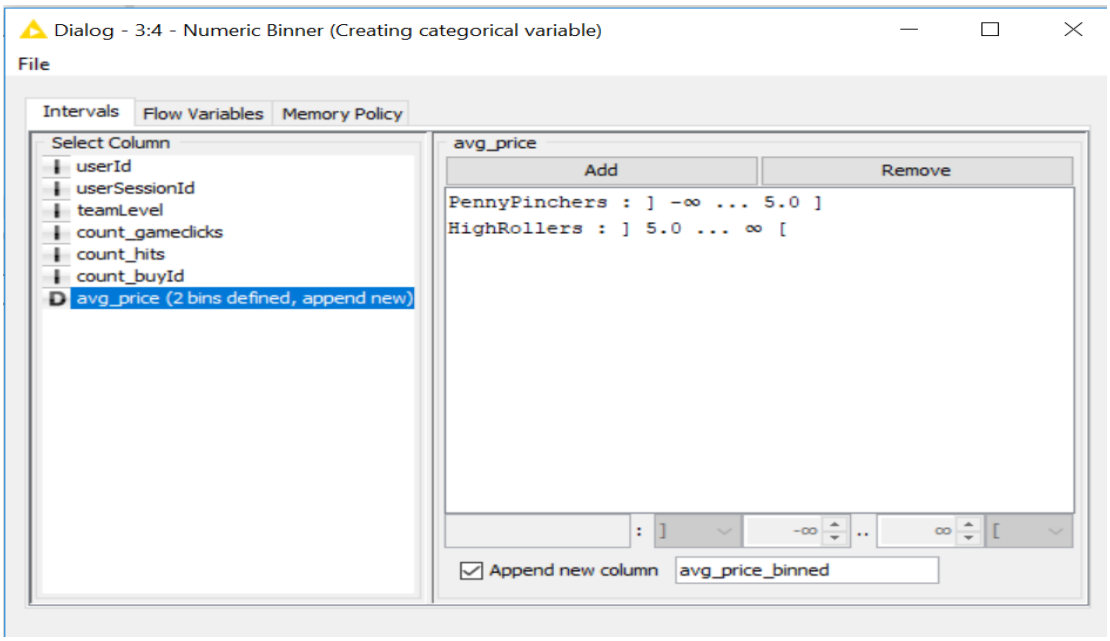
Item	Amount
# of Samples	4619
# of Samples with Purchases	1411

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:



Row ID	userId	userSe...	teamLevel	platfor...	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



The categorical variable is created by dividing the data into two parts. Less than or equal to average price of 5 dollars is coded as ‘PennyPinchers’, and strictly more than 5 dollars of average price is coded as ‘HighRollers’. It must be said here that the 5 dollar threshold is arbitrary and if we chose another one the results would be different.

The creation of this new categorical attribute was necessary because the decision tree algorithm works better with categorical variables as a root node, and also we need to define our categories HighRollers and PennyPinchers to have better insight to our analysis. Alternatively, for our division(our particular choice), we are asking the model to find how the spending is influenced by the other attributes of our data.

Attribute Selection

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

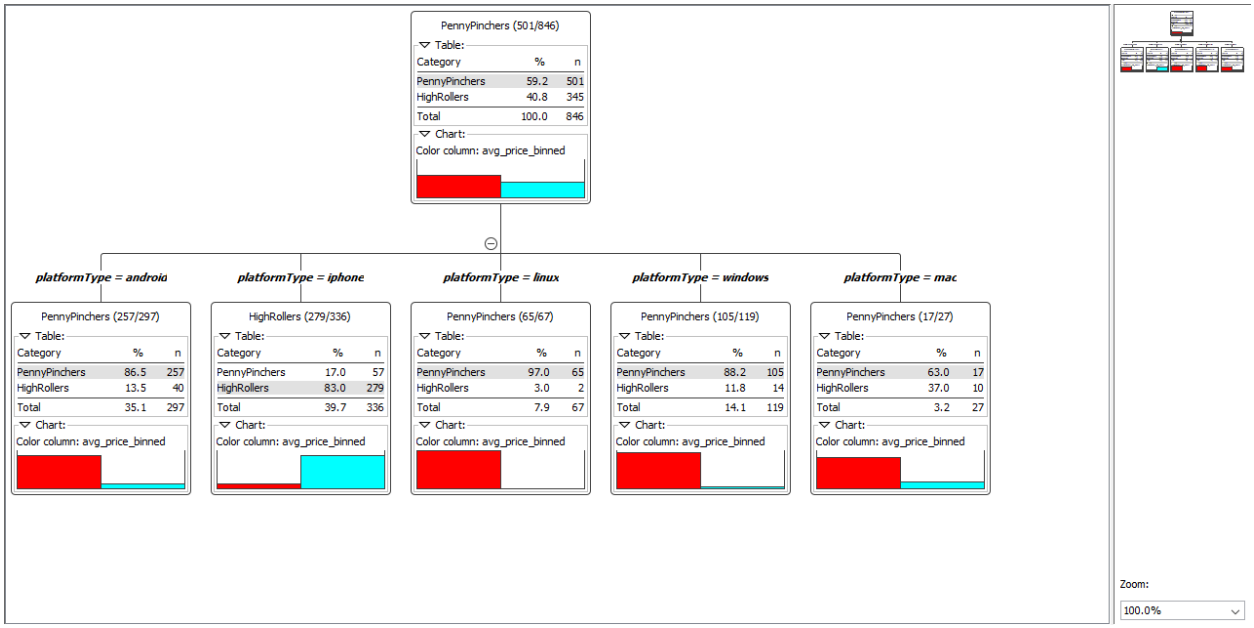
Attribute	**Rationale for Filtering**
userId	User id is just an identifier number(primary or foreign key), and this number has no meaning for the analysis performed by the decision tree algorithm.
userSessionId	Use session id is also an identifier, and for similar reasons has no meaning for this analysis.
avg_price	The average price has to be removed from the decision tree algorithm, since we created categorical variable based on it. If we include it, the algorithm would split the data only based on the price which is redundant and not what we need to do.

2. 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 we need to test how our model created on the train dataset(60% of the data) would perform on the 'unseen' data of the test dataset(40% of the data). That way we could have a model that would supposedly perform well on new data that is not part of our analysis. By adjusting our model on the train data set, so that it would perform better on the test set we could have a more robust decision tree model.

When partitioning the data using sampling, it is important to set the random seed because we could have reproducible results, i.e. we could compare for example the results from different students. If the random seed was not set, then each time a partition is made it would have been different and the results would have varied. That is not good when it comes to designing the model, and checking for possible mistakes.

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



3. Evaluation

A screenshot of the confusion matrix can be seen below:

avg_price_binned \ Prediction (avg_price_binned)	PennyPinc...	HighRollers
PennyPinchers	308	27
HighRollers	38	192

Correct classified: 500 Wrong classified: 65

Accuracy: 88.496 % Error: 11.504 %

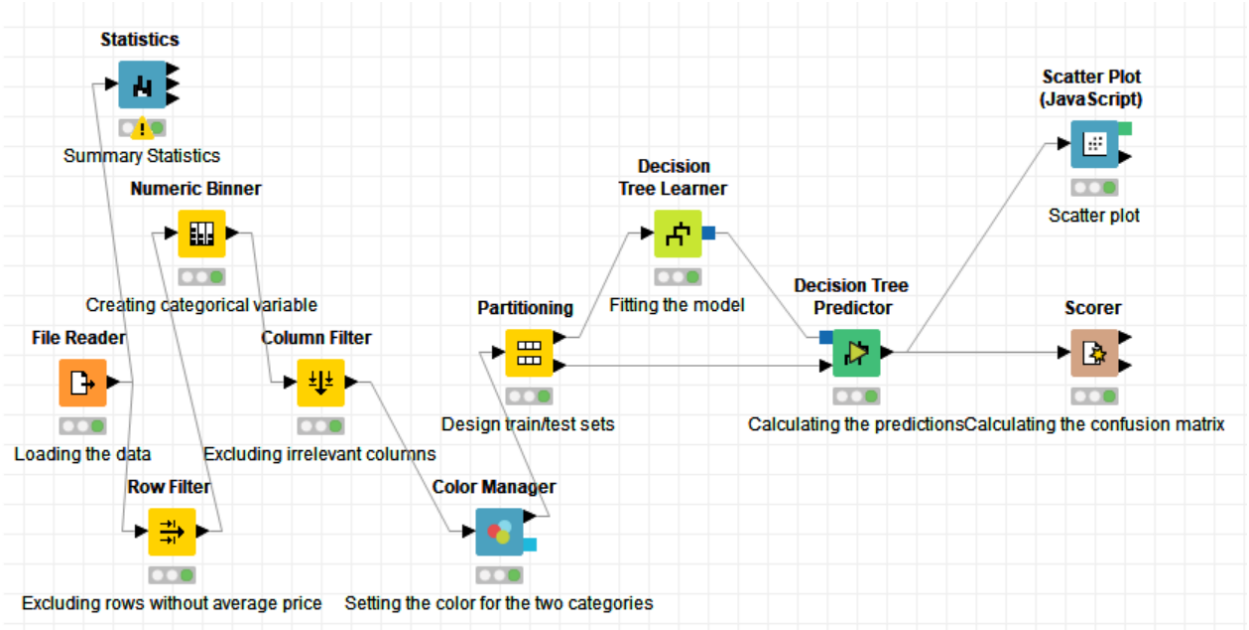
Cohen's kappa (κ) 0.76

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

308 of all PennyPinchers were correctly predicted from the model, while 27 of the actual PennyPinchers were wrongly classified as HighRollers. 192 of the HighRollers were correctly predicted from the model, whereas 38 of them were wrongly classified as PennyPinchers. Overall 308 + 192 = 500 of all samples were correctly classified, whereas, 38 + 27 = 65 were wrongly predicted. Accuracy = 500/(500 + 65) = .88496 , error = 65/(500 + 65) = .11504 .

4. Analysis Conclusions

The final KNIME workflow is shown below:

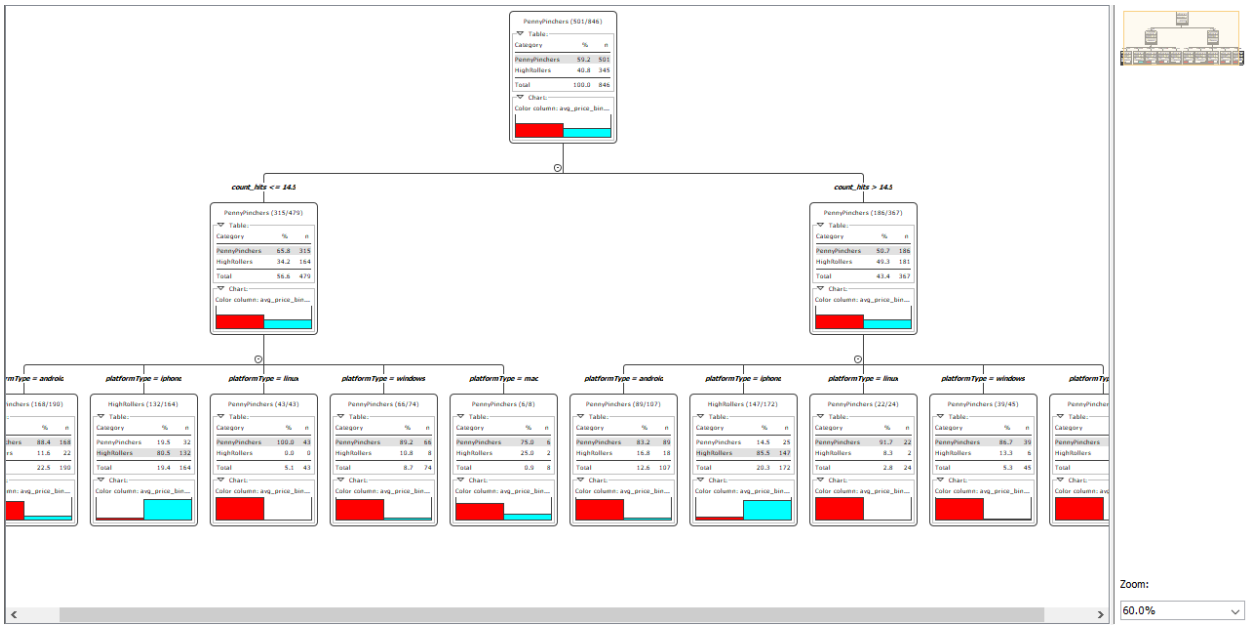


What makes a HighRoller vs. a PennyPincher?

As can be seen from the resulting decision tree clearly the vast majority of HighRollers are the players coming from iphone platform. 83% of iphone users are classified as HighRollers. That is perhaps unsurprising given the price policy of Apple the maker of iphones. The second highest percentage of HighRollers could be found in Mac users, but only 37%. Only 13% of Android users are HighRollers. Windows and Linux users are almost exclusively PennyPinchers.
The other attributes as number of hits, number of clicks or game level are not included in the decision tree, suggesting that they do not influence of variable of interest average price binned. To put it in another way they do not influence how much a player spends in the game.

Specific Recommendations to Increase Revenue
1. Craft specific advertising towards iphone users, knowing that they are most susceptible to spending during the game. Also show more ads to iphone users, compared to Linux/Windows users.
2. Proportionally increase the advertising to Mac users, knowing that 37% of them are HighRollers. Similarly to Android users 13% of them are also prone to spend more.

Forcefully adding number of hits to the decision tree(picture below) shows that there are not big differences between people being a better players and spending more money. That suggests that we should not target users based on anything else than the platform which they use.



III. Clustering Analysis

1. Attribute Selection

Attribute	Rationale for Selection
teamLevel	The team level is a measure of player's sophistication. We would like to see whether better(and long participating) players spend more in the game.
count_gameclicks	The total game clicks per player is a measure of how active he/she is. Again we would like to see if more clicking players actually spend more money.
isHitRate	The hit rate ratio is a composite variable computed as the ratio of number of hits divided by the total number of clicks. This is a measure of how well a player is faring during the game. We would like to find a connection between this measure and his/her spending habits.
revenue	The total revenue per user is a composite measure computed by multiplying the average price times the number of items purchased. This is a feature of the main concern. We want to see how the spending is related to the other characteristics of a player.

2. Training Data Set Creation

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

	teamLevel	count_gameclicks	isHitRate	revenue
0	1	69	0.115942	0.0
1	1	31	0.161290	0.0
2	1	26	0.076923	0.0
3	1	35	0.114286	0.0
4	1	39	0.000000	1.0
5	1	36	0.138889	0.0

- Dimensions of the training data set (rows x columns) : (4619, 4)
- # of clusters created: 3

```
          teamLevel         count_gameclicks     isHitRate          revenue
[array([ 0.02315771, -0.31515387, -0.02822087, -0.28628674]),
 array([ 0.40678906, -0.11040416,  0.35604185,  2.66535918]),
 array([-0.47914333,  2.17228851, -0.09875528, -0.24323711])]
```

Notes:

- The users with NaN for average price are imputed with 0 dollars as a revenue. Hence the 4619 samples in the dataset
- The features are scaled with mean 0 and standard deviation 1, hence the relative numbers in the table.
- The same calculations are made excluding the user who did not spend anything on the game. The results are very similar in nature.
- The number of clusters is arbitrarily chosen to be 3. The elbow method actually shows that 4 is a better choice, however the interpretation of 4 clusters is not straightforward in for our dataset.
- The full calculation could be seen here:

kmeansExample/Week 3 Assignment K Means-final.ipynb at master · Tycho-1/kmeansExample · GitHub

3. Cluster Centers

Cluster #	Cluster Center

1	0.02315771, -0.31515387, -0.02822087, -0.28628674
2	0.40678906, -0.11040416, 0.35604185, 2.66535918
3	-0.47914333, 2.17228851, -0.09875528, -0.24323711

- These clusters can be differentiated from each other as follows:

Cluster 2 is different from the others in that the players who belong to it are playing at a higher level, have high accuracy of playing the game measured through the hit ratio, click not very frequently, and at the same time are high spenders. Those are the most valued player and should be encourage to play more and receive more targeted ads, given that they are most likely to actually buy items

Cluster 1 is different from the others in that the players in it are in a relatively high level, their clicking rate is the lowest of all clusters, while the accuracy is much lower than cluster 1. What is important is that they spend much less money than all other groups, in relative terms. Those people even if a relatively good player are penny pinchers, so maybe they should receive not as many ads.

Cluster 3 is different from the others in that the players are the least sophisticated, have very high clicking rate, and the lowest accuracy, i.e. hit ratio. At the same time they spend much less money than the people in cluster 2, but slightly more than those in cluster 1. Those people are relatively inexperienced, who do not spend much on the game. They should be encouraged to play more and eventually to spend more when they become more addicted to the game.

4. Recommended Actions

Action Recommended	**Rationale for the action**
Cluster 2 Higher price for displaying ads to this cluster.	Carefully craft the ads to those users, since they are the most sophisticated and the most lavish spenders. Maybe charge a higher price to advertiser in order to reach those particular customers, knowing that they would most likely react positively and buy.
Cluster 1 Precisely targeted ads.	Those users are relatively sophisticated, but low spenders. A recommended action is to show very precisely targeted ads based on some additional information about the players. The rationale is that they probably would react to goods of services that are particularly attractive to them.
Cluster 3	Those are inexperienced users, who do not spend much. We should encourage those people to play more. Also we could

Promote the game among those users, eventually hoping that they would spend more.	make promotional offers to bring users to buy items useful to upgrade their rank in the game. Send some emails to encourage them to play more.

IV. Graph Analytics

1. Modeling Chat Data using a Graph Data Model

(Describe the graph model for chats in a few sentences. Try to be clear and complete.)
The graph model broadly could be characterized with 4 main node types. ‘User’ and ‘Team’ are self explanatory, and ‘ChatItem’ and ‘TeamChatSession’, which refer to each text a user writes, and respectively to which session it belongs. The edges of the graph are more numerous, ‘CreateChat’ is obvious, ‘CreateSession’ represents an edge which user creates to link with user chat session.. There can be only one 'CreateSession' edge to each Session node. Important edges related to the session are ‘Joins’ and ‘Leaves’ chracterising what an user is doing. ‘Mentioned’ is an edge which is created when a user is mentioned in a chat text. ‘RespondTo’ connects two chat items which are related. ‘PartOf’ edge reflects that each chat item is part of a specific chat session. Finaly the edge ‘OwnedBy’ designates each chat session to be owned by a specific team.

2. 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

File	Fields
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, teamstamp
chat_leave_team_chat.csv	userid, teamchatsessionid, timestamp
chat_mention_team_chat.csv	ChatItem, userid, timeStamp
chat_respond_team_chat.csv	chatid1, chatid2,timestamp

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

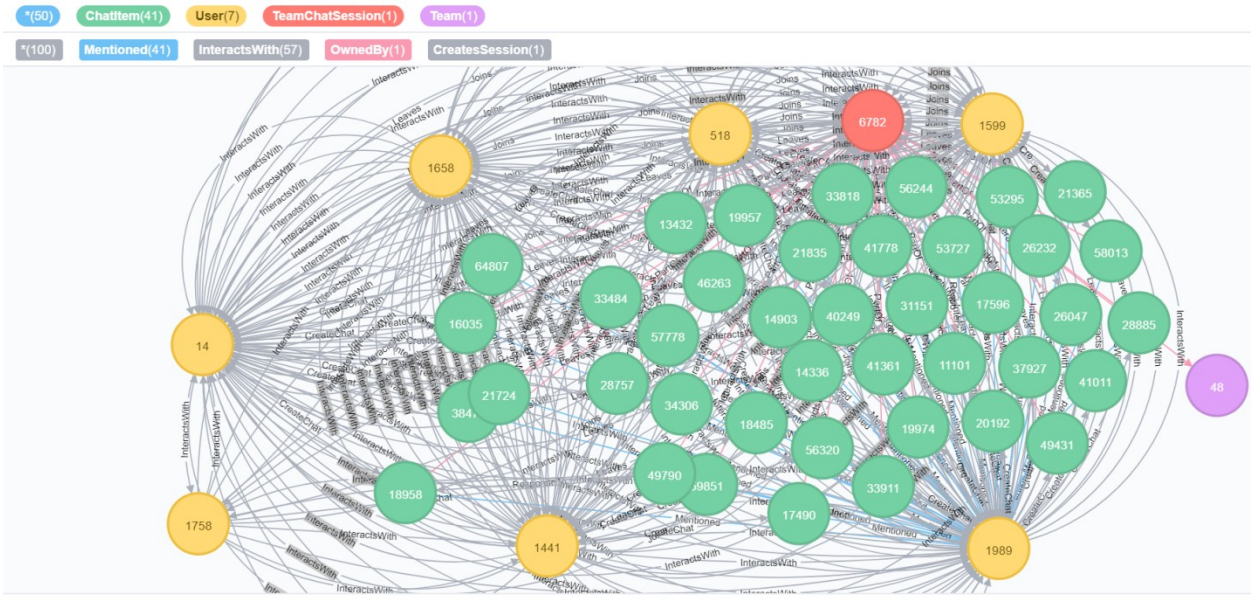
The original:

```
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;

//original
LOAD CSV FROM "file:///C:/Users/121b2/Documents/chat-data/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)
```

My solution:

```
//(i)
LOAD CSV FROM "file:///C:/Users/121b2/Documents/chat-data/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)

//(ii)
LOAD CSV FROM "file:///C:/Users/121b2/Documents/chat-data/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)

//(iii) ??? maybe something wrong # now correct!!
LOAD CSV FROM "file:///C:/Users/121b2/Documents/chat-data/chat_item_team_chat.csv" AS row
MERGE (u:User {id: toInt(row[0])})
MERGE (c:TeamChatSession {id: toInt(row[1])})
MERGE (c_i:ChatItem {id: toInt(row[2])})
MERGE (u)-[:CreateChat{timeStamp: row[3]}]->(c_i)
MERGE (c_i)-[:PartOf{timeStamp: row[3]}]->(c)

//(iv)
LOAD CSV FROM "file:///C:/Users/121b2/Documents/chat-data/chat_mention_team_chat.csv" AS row
MERGE (c_i:ChatItem {id: toInt(row[0])})
MERGE (u:User {id: toInt(row[1])})
MERGE (c_i)-[:Mentioned{timeStamp: row[2]}]->(u)

//(v)
LOAD CSV FROM "file:///C:/Users/121b2/Documents/chat-data/chat_respond_team_chat.csv" AS row
MERGE (c_i:ChatItem {id: toInt(row[0])})
MERGE (c_i2:ChatItem {id: toInt(row[1])})
MERGE (c_i)-[:ResponseTo{timeStamp: row[2]}]->(c_i2)
```

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.



In the pictures could be seen most nodes and edges in the graph. The yellow circles are the nodes of user type, the green circles are the chat item nodes. As can be seen they are the bulk of the information. The red circle represents the team chat session node The edges is more difficult to distinguish here, but we can see ‘interact with’, ‘create chat’, ‘owned by’ and so on.

3. Finding the longest conversation chain and its participants

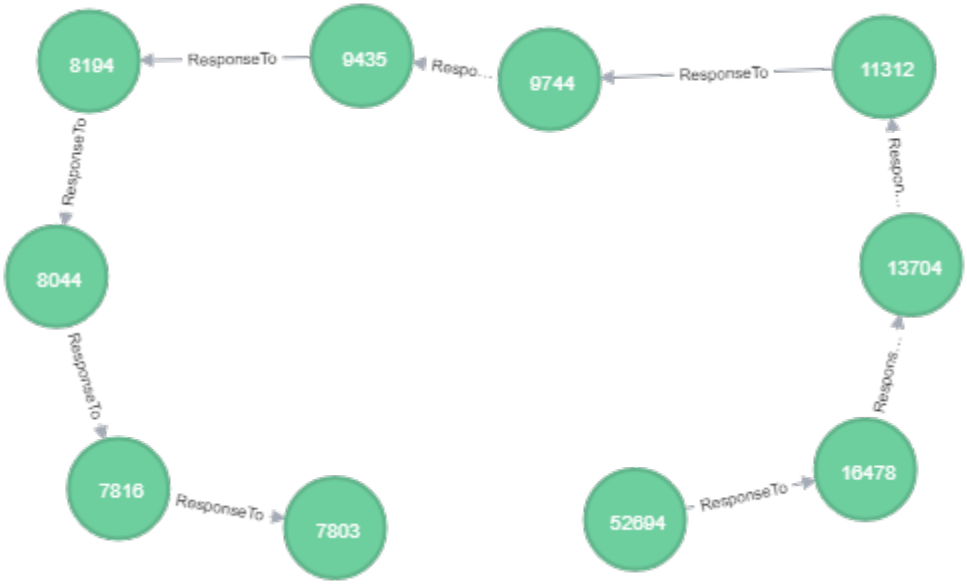
Report the results including the length of the conversation (path length) and how many unique users were part of the conversation chain. Describe your steps. Write the query that produces the correct answer.

- The longest chat consists of 9 chat items.

  MATCH p=()-[r:ResponseTo*]->()
  RETURN length(p)
  ORDER BY length(p) DESC
  LIMIT 1
  Result 9.

- A graph of the longest chat produced by the query

  MATCH p=()-[r:ResponseTo*]->()
  WHERE length(p)=9
  RETURN p



- Five users participate in this longest chat. This can be seen with the following query:

  MATCH p=(a:ChatItem)-[r:ResponseTo*]->(b:ChatItem)
  WHERE length(p)=9
  WITH p AS path
  MATCH (u:User)-[:CreateChat*]->(c_i:ChatItem)
  WHERE c_i IN NODES(path)
  RETURN COUNT(DISTINCT u) AS uniqueUsers
  Result 5.

Relevance for the company's business plan: it would be good for the company to see the longest chats and see what they are about. Similarly, what is the share of the unique users in these chats. That could help the company when analyzing the marketing strategy, and deciding on how to target the users.

4. 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.

- The query for producing the top 10 chattiest users is as follows:

  MATCH p=(u:User)-[r:CreateChat]->()
  RETURN u,COUNT(u) AS cnt
  ORDER BY cnt DESC
  LIMIT 10
  Top 3 of them are shown in the following table(there is a tie):

Chattiest Users

Users	Number of Chats
{"id":394}	115
{"id":2067}	111
{"id":209}	109
{"id":1087}	109

- The query for producing the top 10 chattiest teams is as follows:

  MATCH (c:ChatItem)-[r:PartOf]->(t:TeamChatSession)-[r2:OwnedBy]->(n)
  RETURN n,COUNT(n) AS cnt
  ORDER BY cnt DESC
  LIMIT 10
  The top 3 of the are shown in the following table:

Chattiest Teams

Teams	Number of Chats
{"id":82}	1324
{"id":185}	1036
{"id":112}	957

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

- None of the top three chattiest users belong to any of the top three chattiest teams. This can be seen from the following query:

```
MATCH p=(u:User)-[r:Joins]->(c:TeamChatSession)
where u.id IN [2067,394,209]
WITH p AS path
MATCH (c:TeamChatSession)-[r:OwnedBy]->(t:Team)
WHERE c IN NODES(path)
RETURN DISTINCT t.id AS team
```

The result is just teams with id 7 and 63, i.e users 2067 an 209 belong to team 7, whereas user 394 belongs to team 63.

- Considering the top 10 users the query is:

```
MATCH p=(u:User)-[r:Joins]->(c:TeamChatSession)
where u.id IN [394,2067,209,1087,554,1627,999,516,668,461]
WITH p AS path,u
MATCH (c:TeamChatSession)-[r:OwnedBy]->(t:Team)
WHERE c IN NODES(path)
RETURN distinct t.id,u
```

And the results:

"team_id"	"user"
89	668
77	1087
63	394
52	999
7	2067
104	461
7	516
7	1627
181	554
7	209

"team"	"cnt"
{"id":82}	1324
{"id":185}	1036
{"id":112}	957
{"id":18}	844
{"id":194}	836
{"id":129}	814
{"id":52}	788
{"id":136}	783
{"id":146}	746
{"id":81}	736

The first picture shows to which team each of the top 10 user belongs, the second picture shows the top 10 chattiest teams. As can be seen only user '999' from the top 10 chattiest is the only one that belongs to one of the top 10 teams, i.e. team 52.

Relevance for the company's business plan: the most chattiest users are important to be identified since they probably are influential. The same goes for the most chattiest teams. The fact that the most chattiest users do not belong to the most chattiest teams, probably means that the chat items are more evenly or uniformly distributed. The few chattiest persons could not sway the tally of the whole team. The company should pay special attention therefore for the team as a whole, when designing advertising strategies.

5. 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.

- Creating the relation "InteractWith"

  First:

```
MATCH (u1:User)-[r:CreateChat]->(c_i:ChatItem)-[:Mentioned]->(u2:User)
CREATE (u1)-[:InteractsWith]->(u2)
//Created 11084 relationships
```

  Second:

```
MATCH (u1:User)-[r:CreateChat]->(c_i:ChatItem)-[:ResponseTo]->(c_i_2:ChatItem)-
[:CreateChat]-(u2:User)
create (u1)-[:InteractsWith]->(u2)
//Created 11073 relationships
MATCH (u1)-[r:InteractsWith]->(u2)
RETURN count(r)
//Created  22157 relations
```

  Third deleting the self loops:

```
MATCH (u1)-[r:InteractsWith]->(u1) DELETE r
//Deleted 4377 relationships
MATCH (u1)-[r:InteractsWith]->(u2)
RETURN count(r)
//17780 relations
```

- The coefficients are calculated with the following query(computes the coefficient for only one user):

```
MATCH (a{id:2067})-[:InteractsWith]-(b)
WITH COLLECT(DISTINCT b.id) as neighbours
MATCH (n)-[r:InteractsWith]-(m) WHERE (n.id in neighbours) and (m.id in neighbours)
WITH neighbours,n,m,COUNT(DISTINCT r) AS relation
WITH relation,n,m,neighbours
WITH count(relation) AS cnt_r,size(neighbours)AS size_n,
(size(neighbours)*(size(neighbours)-1)) AS comb, (
toFloat(count(relation))/(size(neighbours)*(size(neighbours)-1))) AS coef
RETURN cnt_r,size_n,comb, coef
```

Explanation: first I find the neighbors of the of the particulat node with respect to a particular relationship, i.e. interact with. Then I compute the distinct relations between the node that belong to this particular neighborhood. Then I form the ratio of the unique relations divided bu all possible combinations i.e.k*(k-1)

Most Active Users (based on Cluster Coefficients)

User ID	Coefficient
{"id":394}	1
{"id":2067}	0.7857142857142857
{"id":209}	0.9523809523809523

Implications for the business plan of the company: identifying highly interacting clusters is beneficial, since the company could target these users with advertising. We can see for example that the most chattiest used id:394 has a coefficient of 1 meaning that he/she interact with everyone in the group. We can also see that the other two users in the list have also very high coefficients of density within their respective groups(though less than 1). This partly reflects that the neighborhood of user 394 is smaller, i.e. 4, while user 2067 interacts with 8 other users and 209 interacts with 7. All in all these are highly dense groups which are beneficial for analysis and advertising considerations from the management of the company.

PS: The computed coefficients for the 10 top users as can be seen below. (ordered by the cluster coefficient) The interpretation is similar.

"id"	"cluster_coefficient"
394	1.0
461	1.0
516	0.952380952380952
209	0.952380952380952
554	0.904761904761904
999	0.866666666666667
1087	0.8
2067	0.785714285714285
1627	0.785714285714285
668	0.7

V. Recommended Actions

Finally, make recommendations to Eglence, Inc. and include examples of how your findings support them. Include this information in Slide 6 of your final presentation.

Data analysis could yield important insights:
1 Pay attention on the platform users use, especially the
Iphone users. Example could be found in the in the classification analysis in Knime.

2 Charge higher price for advertising to users belonging to
cluster #2 Example from the clustering analysis part III of the document.

3 Send precisely targeted ads to users belonging to cluster #1:
most susceptible

4 Promote the game among the users from cluster 3:they may
become more like users from #2 and spend more

5 Send special advertising to chattiest users, take into
consideration their cluster coefficients. Example could be found in part IV of the document.

6 Craft ads towards chattiest teams, they hold the most of the
conversation and hence could be good communicators