Week 1 Technical Appendix: Data Models

Data Exploration

Data Set Overview

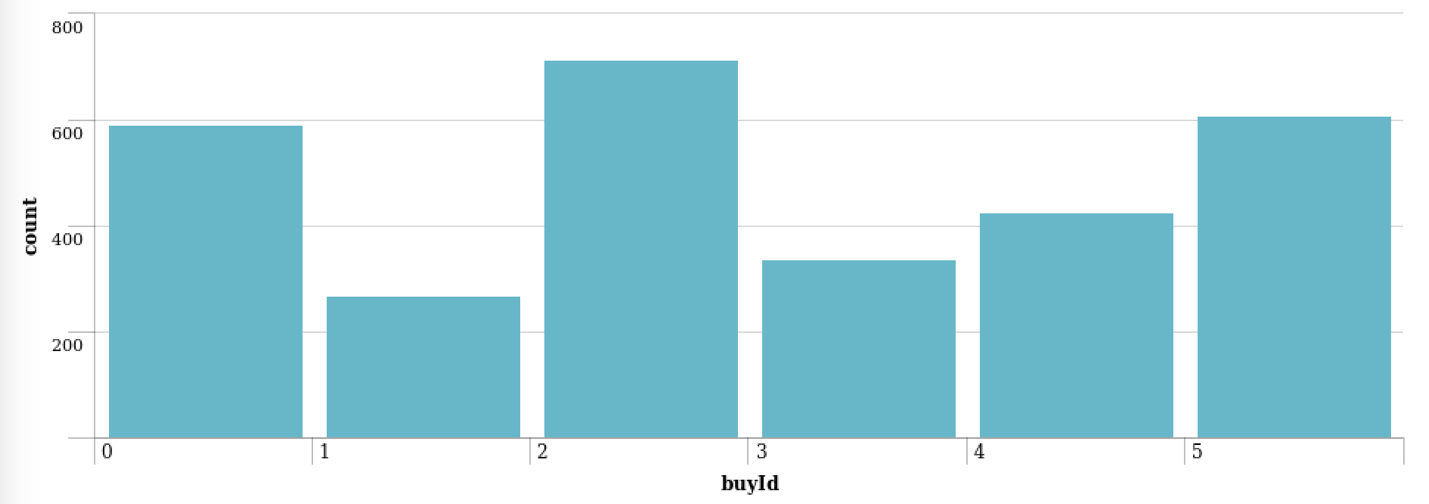
The table below lists each of the files available for analysis with a short description of what is found in each one.

|  |  |  |
| --- | --- | --- |
| **File Name** | **Description** | **Fields** |
| ad-clicks.csv | A record of each ad that was clicked on. | timestamp: date / time the click occurred.  txId: a unique id 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 of ad clicked on |
| buy-clicks.csv | A list of purchases. | 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 record of each click a user performed during the game. | timestamp: when the click occurred.  clickId: a unique id for the click.  userId: the id of the user performing the click.  userSessionId: the id of the session of the user when the click is performed.  isHit: denotes if the click was on a flamingo (value is 1) or missed the flamingo (value is 0)  teamId: the id of the team of the user  teamLevel: the current level of the team of the user |
| level-events.csv | A record of each level event for a team. Level events are recorded when a team ends or begins a new level | timestamp: when the event occurred.  eventId: a unique id for the event  teamId: the id of the team  teamLevel: the level started or completed  eventType: the type of event, either start or end |
| team-assignments.csv | A record of each time a user joins a team. | 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 |
| team.csv | A record of each team 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 |
| user-session.csv | A record of each session a user plays.  When a team levels up, each current user session ends and a new session begins with the new level. | 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 |
| users.csv | A record 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. |

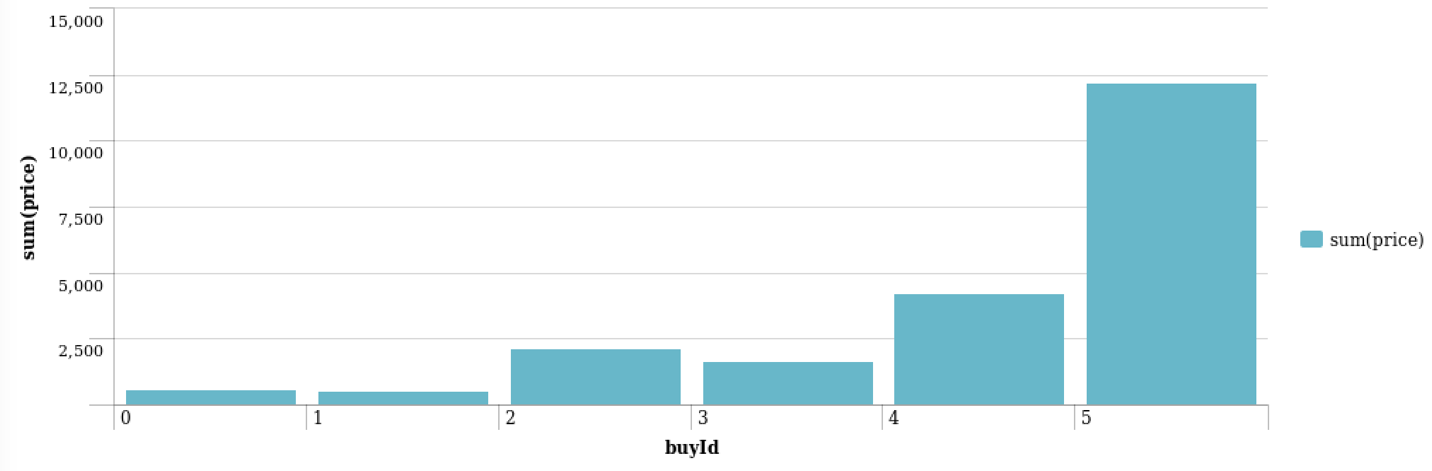
Aggregation

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

A histogram showing how many times each item is purchased:

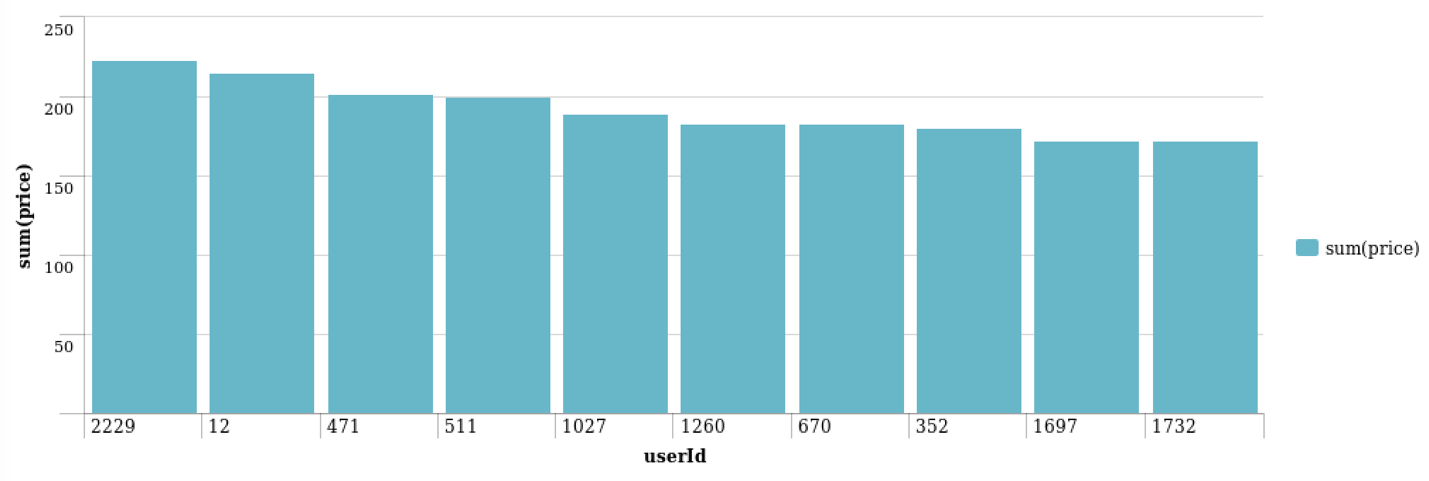


A histogram showing how much money was made from each item:



Filtering

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



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

|  |  |  |  |
| --- | --- | --- | --- |
| **Rank** | **User Id** | **Platform** | **Hit-Ratio (%)** |
| 1 | 2229 | iPhone | 11.5% |
| 2 | 12 | iPhone | 13% |
| 3 | 471 | iPhone | 14.5% |

Week 2 Technical Appendix: Classification

**Data Preparation**

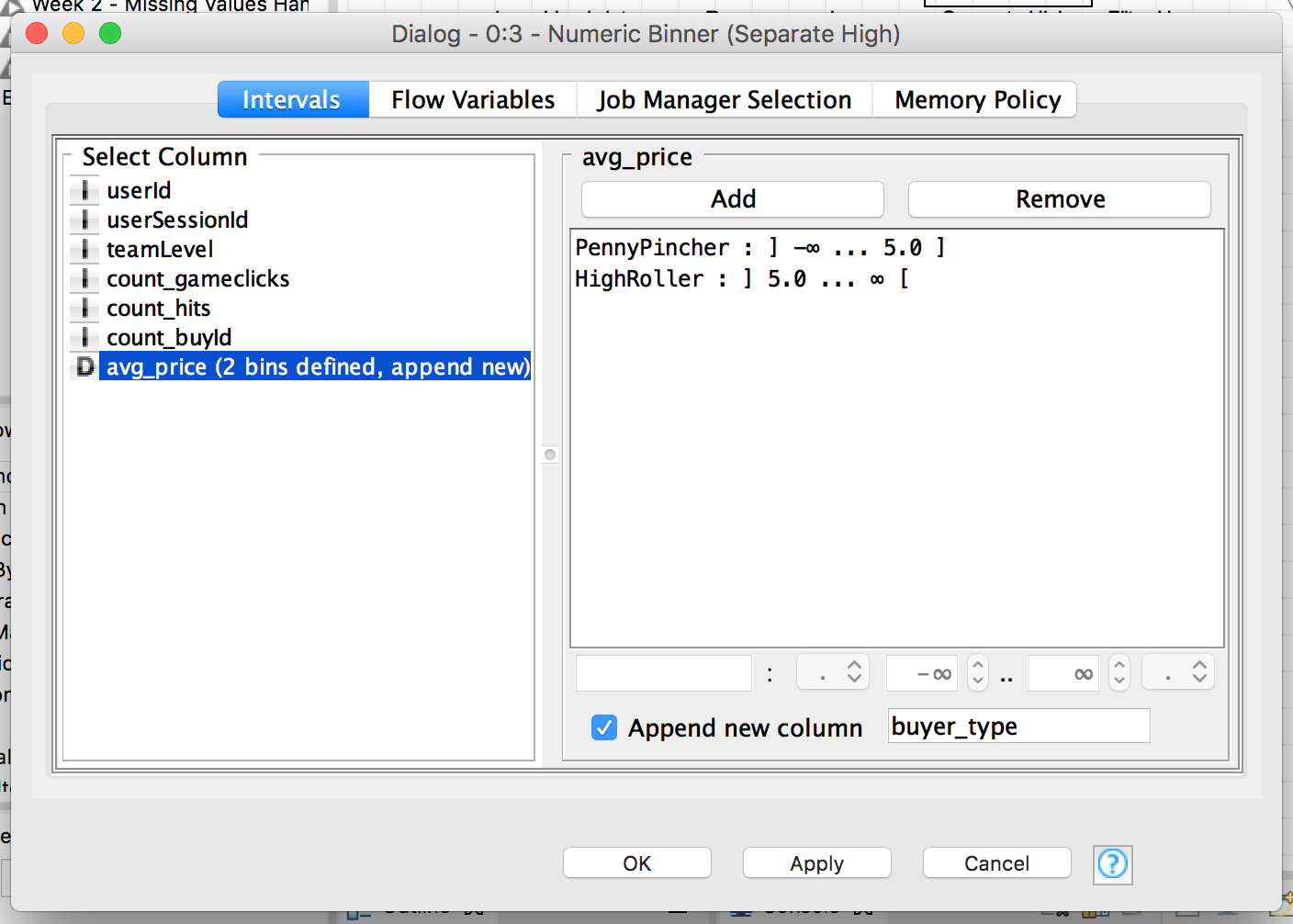
Analysis of combined\_data.csv

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:



High rollers are defined as those who purchase items over $5.00. Defining a new column based on the avg\_price allows us to classify users accordingly.

The creation of this new categorical attribute was necessary because:

Our goal is to understand the attributes of who makes large purchases. This categorical variable is what we are going to base our decision tree upon.

Attribute Selection

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

|  |  |
| --- | --- |
| **Attribute** | **Rationale for Filtering** |
| userId | Not relevant in determining the user’s behavior. |
| userSessionId | Not relevant in determining the user’s behavior. |
| avg\_price | This variable was used to create the categorical variable “buyer\_type”, the variable we are trying to predict based on other data elements. Therefore, we do not want to include this as a criteria in our decision tree model. |

**Data Partitioning and Modeling**

The data was partitioned into train and test datasets.

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

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

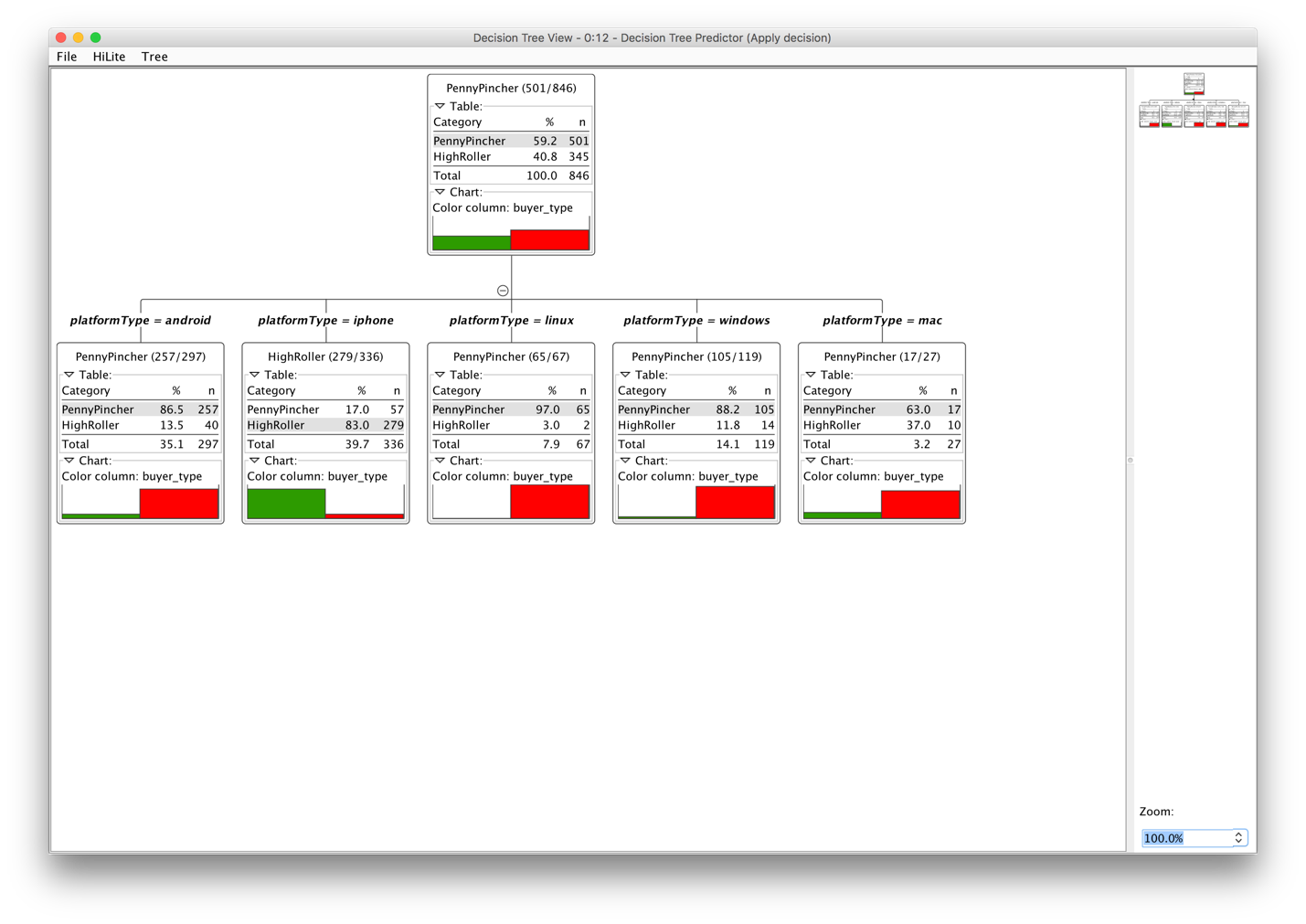
This is important because…

**Partitioning the data set into training and test data allows us to verify the accuracy of the trained model. If we used all the data for training the model, we would have no idea how the model performs.**

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

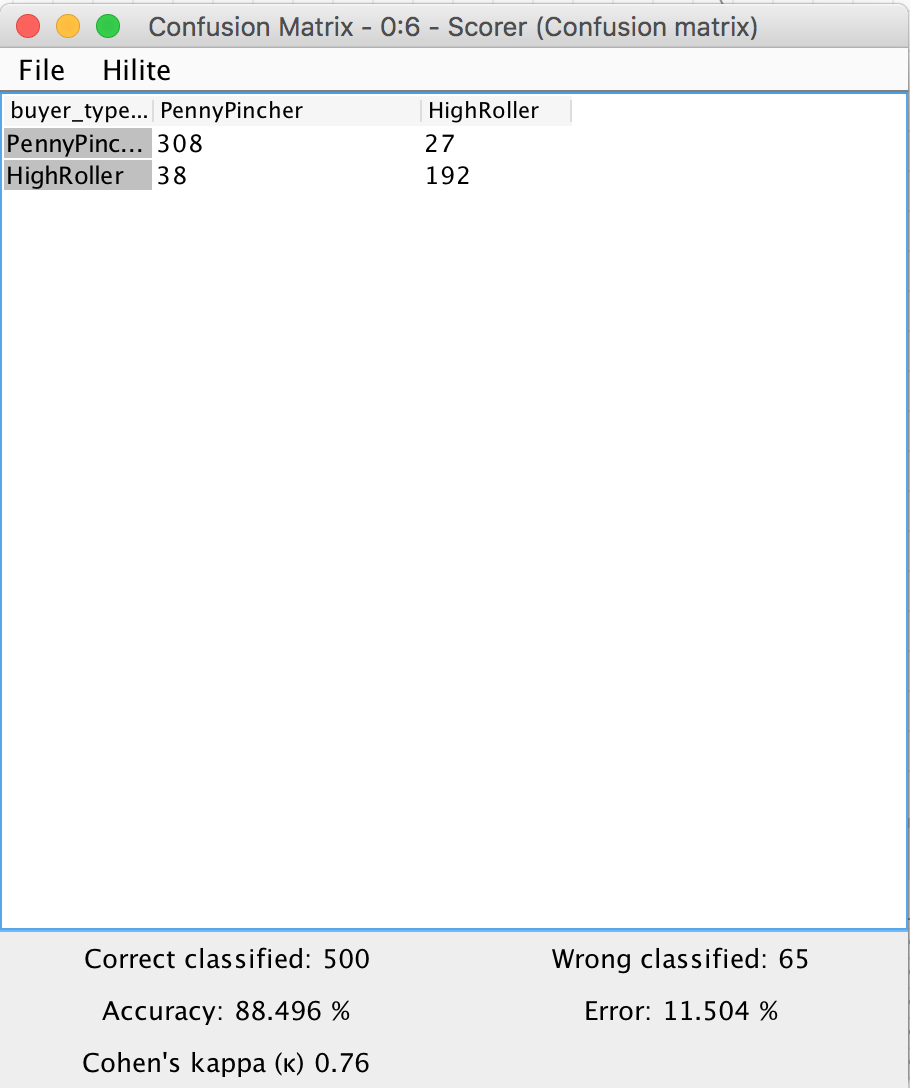
**A random seed will partition the data set consistently. This allows you to obtain reproducible results each time you run the partition. When validating the accuracy of different models, you need to keep the data used to test the models consistent. If the test data is inconsistent, you can’t compare the accuracy between models.**

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



**Evaluation**

A screenshot of the confusion matrix can be seen below:

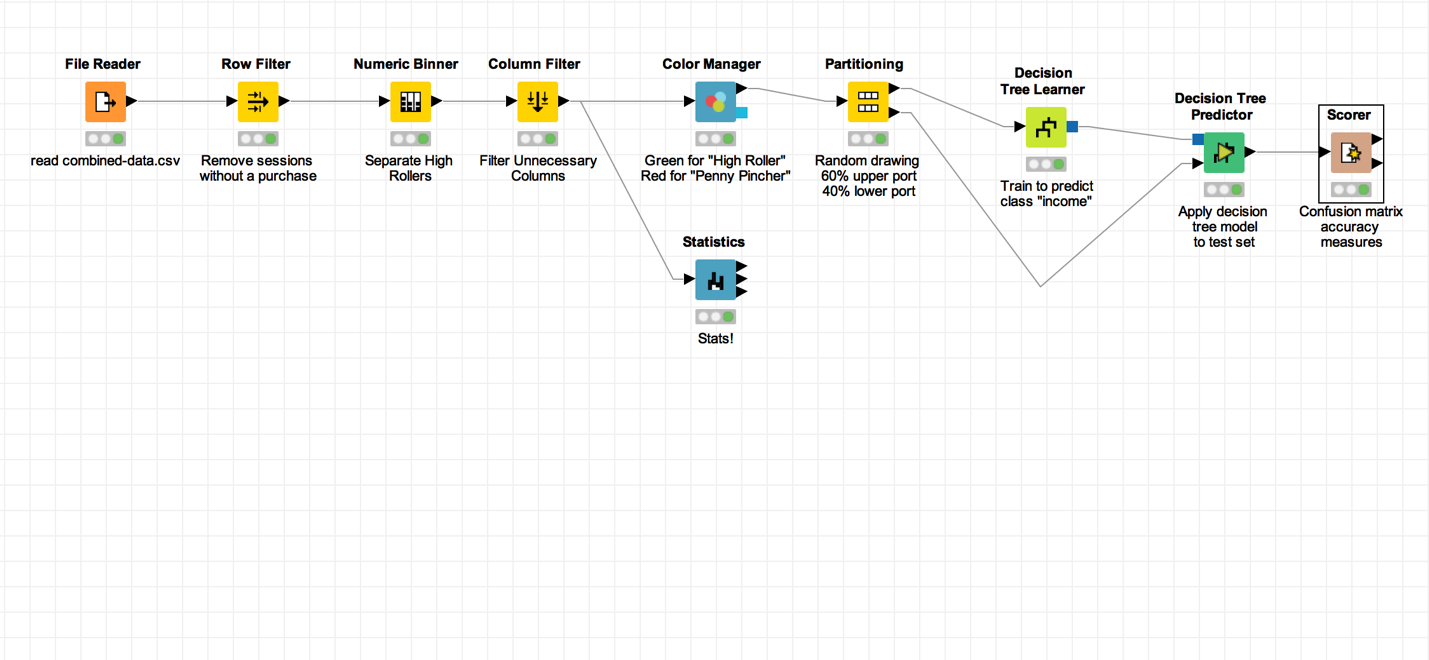


As seen in the screenshot above, the overall accuracy of the model is **88.496%**

**When the buyer type is PennyPincher, the model classified correctly 308 times and incorrectly 27 times. When the byer type is HighRoller, 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 used. Users who are HighRoller use iOS. PennyPincher use Android, Mac, Win, Linux.**

|  |
| --- |
| **Specific Recommendations to Increase Revenue** |
| 1. Target promotions to iOS users. |
| 2. Target future product develop towards iOS. |

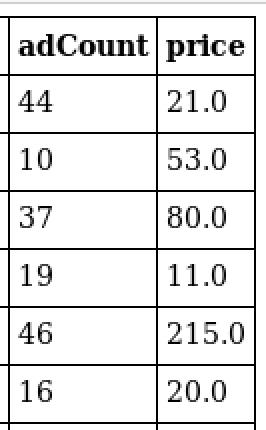
Week 3 Technical Appendix: Clustering

**Attribute Selection**

|  |  |
| --- | --- |
| **Attribute** | **Rationale for Selection** |
| Team level | Used to determine if and/or how a user’s behavior changes by team level.  For example, perhaps users in higher teams make more purchases because the levels are increasingly hard or they have more money invested. |
| Ad clicks | The number of ad clicks per user.  I want to determine how related the number of ad clicks is in relation to the amount of revenue and team level. My hypothesis is that as the team number increases, ad clicks and revenue both increase. |
| Revenue | The total number of revenue per user will tell us how revenue changes when team and ad clicks change. |

**Training Data Set Creation**

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



Dimensions of the training data set (rows x columns) : 543 x 2

# of clusters created: 2

**Cluster Centers**

|  |  |
| --- | --- |
| **Cluster #** | **Cluster Center** |
| 1 | 27.39467849, 23.86474501 |
| 2 | 39.07608696, 115.26086957 |

These clusters can be differentiated from each other as follows:

Cluster 1 is different from the others in that…

**This cluster has a lower ad click rate per user and lower revenue per user.**

Cluster 2 is different from the others in that…

**This cluster has a higher ad click rate per user and higher revenue per user.**

**Recommended Actions**

|  |  |
| --- | --- |
| **Action Recommended** | **Rationale for the action** |
| Determine why users click on ads | The data shows who click on more ads produce more revenue. Therefore, we want to determine the root cause as to why users click ads in order to produce more clicks.  Examples of why users may click ads:   * The ad is for a resource they are lacking. For example, time, diamonds, power, etc. * The ad is a deep discount. * The ad is funny or likeable. |
| Show more relevant ads to users when they are more likely to click on them. | Once we understand why users click the ads they do, we can better target advertising to users.  Examples of more relevant ads:   * Ads the users have clicked on in the past, but did not complete the purchase. * Ads that play to an emotion – like the ability to help their team or their morale. |

Week 4 Technical Appendix: Graph Analytics

**Graph Analytics**

**Modeling Chat Data using a Graph Data Model**

The graph model for chat data models the entities and relationships involved in chat activity. The model allows us to analyze chat behavior to determine how users and teams chat.

For example, the model can tell us which users and teams chat the most frequently (or not at all), if the chats are conversational or not, which users are the most mentioned in a chat (the influencers).

This data could be used to improve the chat feature of the game or target users for promotion.

**Creation of the Graph Database for Chats**

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

1. Write the schema of the 6 CSV files

Chat\_create\_team\_chat : A line is added to this file when a player creates a new chat with their team.

|  |  |
| --- | --- |
| userId | The user creating the chat |
| teamId | The teamId of the chat |
| teamChatSessionId | The Id for the chat session |
| timestamp | The time the chatSession was created |

chat\_item\_team\_chat: A line is created for each chat item (a message posted to a chat)

|  |  |
| --- | --- |
| userId | The user posting the chat item |
| teamChatSessionId | The session the item was posted to |
| chatItemId | The id for the chat item |
| timestamp | The time the item was posted |

chat\_join\_team\_chat : A line is created when a user joins a chat

|  |  |
| --- | --- |
| userId | The user joining the chat |
| teamChatSessionId | The session id being joined |
| timestamp | The time the user joined |

chat\_leave\_team\_chat : A line is created when a user leaves a chat

|  |  |
| --- | --- |
| userId | The user leaving the chat |
| teamChatSessionId | The session id being left |
| timestamp | The time the user left |

chat\_mention \_team\_chat : A line is created when a user is mentioned in a chat item

|  |  |
| --- | --- |
| chatItemId | The id of the chat item which mentioned the user |
| userId | The user mentioned |
| timestamp | The time the mention occurred |

chat\_respond \_team\_chat : A line is created when a user responds to a chat item

|  |  |
| --- | --- |
| chatId1 | The chat item when initiated the response |
| chatId2 | The response chat item |
| timestamp | The time the response occurred |

1. Explain the loading process and include a sample LOAD command

The loading process is the process of loading the graph database with data. In our case, data was loaded from comma separated files. When data is loaded, the entities and relationships are created in the graph database.

In this example, we are loading the team chats. We create the user, team, and teamChatSession entities, along with relationships telling us which user created the chat session and which team owns the chat session.

LOAD CSV FROM "file:///big-data/datasets/big-data-capstone/chat/chat\_create\_team\_chat.csv" AS row

MERGE (u:User {id: toInteger(row[0])})

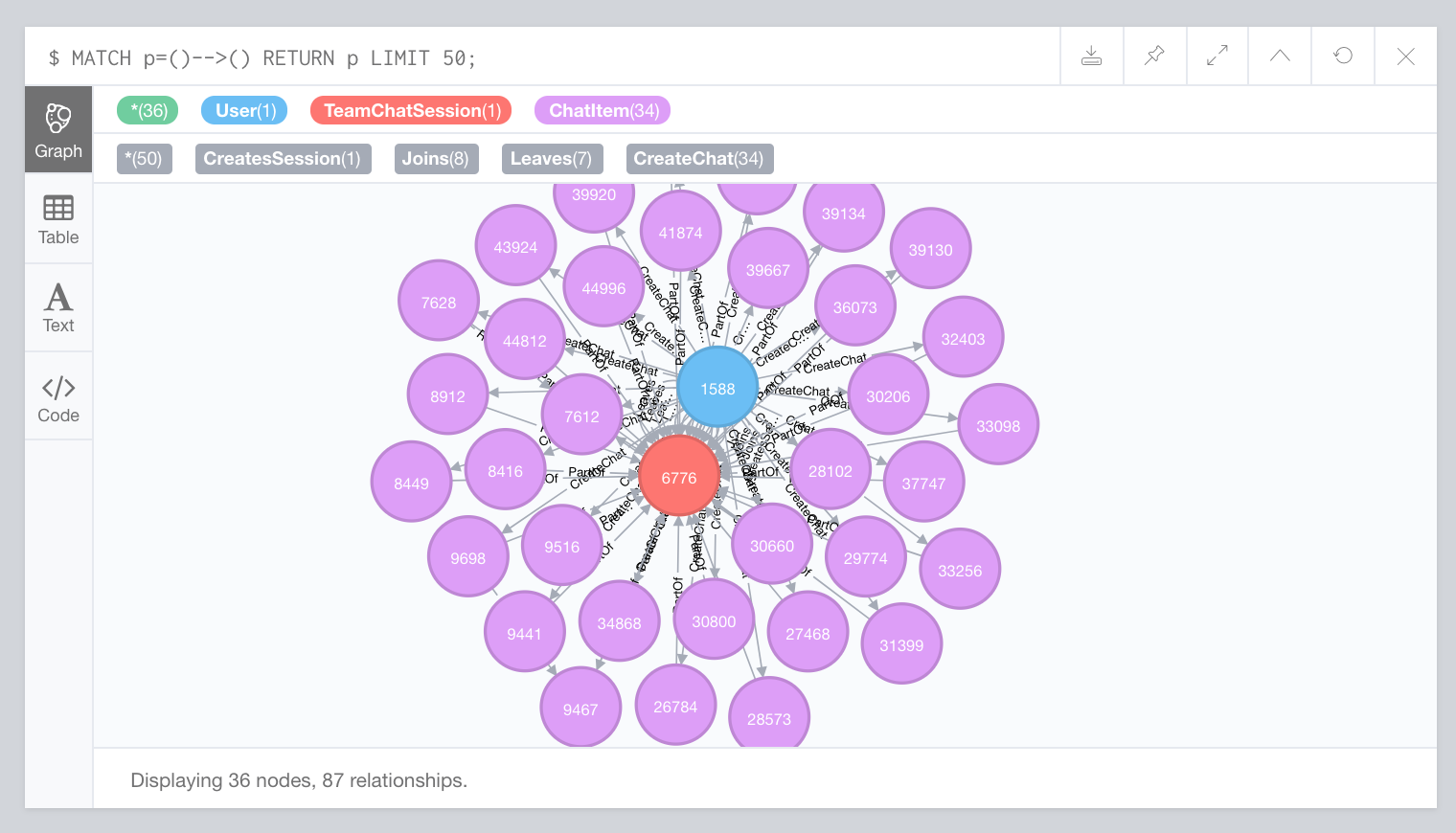
MERGE (t:Team {id: toInteger(row[1])})

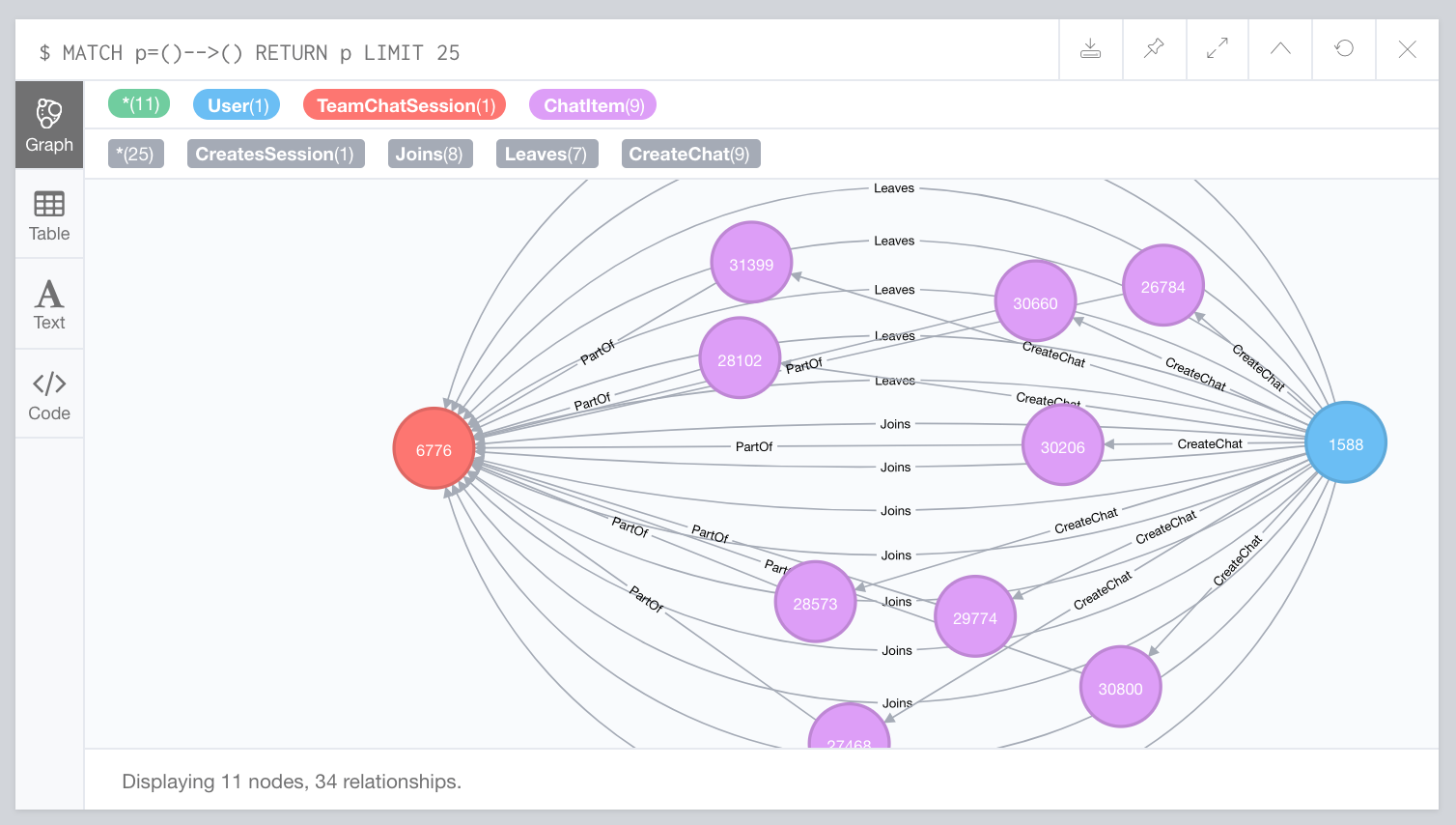
MERGE (c:TeamChatSession {id: toInteger(row[2])})

MERGE (u)-[:CreatesSession{timeStamp: row[3]}]->(c)

MERGE (c)-[:OwnedBy{timeStamp: row[3]}]->(t);

1. 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.

The longest conversation chain (by path length) is the set of chat items which respond to previous chat items. The longest chain in our graph is 9.

The steps to find the longest chain is to find all the paths with ResponseTo relationships between ChatItem nodes. Sort all the paths by their path lengths descending and select the first element in the list (the longest). Once you have that longest path, you can find the users who created the ChatItems in that particular path. The number of users in that particular chat is 5.

MATCH p=(:ChatItem)-[:ResponseTo\*]->(:ChatItem)

WITH p as Path

ORDER BY LENGTH(p) DESC LIMIT 1

MATCH (u:User)-[:CreateChat]->(ci:ChatItem)

WHERE ci IN NODES(Path)

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**

|  |  |
| --- | --- |
| **Users** | **Number of Chats** |
| 394 | 115 |
| 2067 | 111 |
| 209 | 109 |

**Chattiest Teams**

|  |  |
| --- | --- |
| **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.

User 999 is one of the chattiest users is in one of the top 10 chattiest teams (team 52). In general, the chattiest users are **not** part of the chattiest teams.

**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.

Users are active if their neighbors are active with each other. Users are less active if they have no neighbors or their neighbors are not active with eachother.

To find out active a group of users is, we find all the “neighbors” of a user. Given User 1, we first need to find all users who User 1 interacted with. These users are called “neighbors”. We then determine how active those users are between eachother using a “cluster coefficient”. If two neighbor users interact with eachother, they are considered “active”. The more active neighbors are, the higher the cluster coefficient.

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

|  |  |
| --- | --- |
| **User ID** | **Coefficient** |
| 209 | 0.9523809523809523 |
| 554 | 0.9047619047619048 |
| 1087 | 0.8 |