Contents

CA4015 Clustering Assignment Introduction

Introduction to the Iowa Gambling Task

This Jupyter Book will hold an analysis of data from 617 Healthy Participants Performing the Iowa Gambling Task (IGT). The data, which originates from 10 individual studies, is pooled together by {cite}'Steingroever2015'. All participants are healthy (have no known neurological impairments). Participants were assessed on a computerised version of the IGT. The playing conditions vary between each study, with participant's attempts ranging between *95 - 150 tries*. The *payoff scheme* varies also, with some servers hosting **harsher penalties** and **more lucrative rewards**.

Introduction to K-Means Clustering Algorithm

K-Means Clustering is a classical machine learning algorithm developed well over 50 years ago. K-Means is an *unsupervised* machine learning technique meaning, unlike classification it does not need to be trained on annotated training data. Although K-Means is a rhobust algorithm, it does suffer from the 'curse of dimensionality'. This means that techniques such as *Principal Component Analysis* are commonly used in conjunction with K-Means to perform dimensionality reduction.

Introduction to Federated Learning

Federated Learning (FL) is a deep learning approach which involves training a model over disconnected or siloed data centres such as mobile phones. Rather than both the data and model being centralised on one system, data is preserved in its local environment. A machine learning model is sent to the system hosting the data, rather than the reverse. This approach makes a step forward in protecting the privacy of user-generated data. In FL, the user data is not transmitted across a network. However, there are challenges associated with FL, including: *the expensive nature*, *system heterogeneity*, *statistical heterogeneity*, and *privacy* {cite}'Li2020'.

Dataset Description

As stated previously, the data originates from 10 individual studies. These studies can be found listed below:

Study	Amount of Participants	Number of Trials
{cite}'FRIDBERG201028'	15	95
{cite}'hortsmanm2012'	162	100
{cite}'KJOME2010299'	19	100
{cite}'Maia16075'	40	100
{cite}'PREMKUMAR20082002'	25	100
{cite}'stein_sub_study_1'	70	100
{cite}'stein_sub_study_2'	57	100
{cite}'WETZELS201014'	41	150
{cite}'Wood2005'	153	100
{cite}'Worthy2013'	35	100

Quality Control

All studies were administered through a computerized version of the IGT to ensure quality {cite}'Steingroever2015'.

Clustering of Total Dataset

In the followig Jupyter Notebook, we will be clustering and performing some analysis on the total dataset of 617 healthy participants performing the ***Iowa Gambling Task*** (IGT). The clustering algorithm we will be using is the **K-Means Algorithm**. In the following analysis, we will not be taking varying testing conditions between studies (such as *Reward Scheme* and *number of attempts*) into consideration. We will be analysing users based on their **Net Win \(\times\) Net Loss**.

We will perform the following tasks in this Notebook:

1. Analysis Setup
2. Understand Data Distribution
    - Histogram
    - QQPlot
3. Clustering
    - Finding the Elbow Point (Inertia Reduction)
    - K-Means Clustering
4. Comparison Between Contrasting Clusters
    - Time-Series Analysis
5. Comparison of All Clusters
    - Boxplot

In our next Notebook, we will attempt to employ a **Federated Learning** (FL) approach to clustering. FL is a deep-learning technique not usually associated with clustering. However, we will attempt to use some of the basic principles of FL when clustering.

It's time to focus on this Notebook however, so let's begin with the setup:

Initial Setup

Importing Packages

```
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
from scipy.stats import pearsonr
from statsmodels.graphics.gofplots import qqplot
from matplotlib import pyplot
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_8040/935022693.py in <module>
      4 import matplotlib.pyplot as plt
      5 from scipy.stats import pearsonr
----> 6 from statsmodels.graphics.gofplots import qqplot
      7 from matplotlib import pyplot

ModuleNotFoundError: No module named 'statsmodels'
```

Importing Data

```
#Importing challenge involving 95 attempts
index_95 = pd.read_csv("../data/index_95.csv")
choice_95 = pd.read_csv("../data/choice_95.csv")
win_95 = pd.read_csv("../data/wi_95.csv")
loss_95 = pd.read_csv("../data/lo_95.csv")

#Importing challenge involving 100 attempts
index_100 = pd.read_csv("../data/index_100.csv")
choice_100 = pd.read_csv("../data/choice_100.csv")
win_100 = pd.read_csv("../data/wi_100.csv")
loss_100 = pd.read_csv("../data/lo_100.csv")

#Importing challenge involving 150 attempts
index_150 = pd.read_csv("../data/index_150.csv")
choice_150 = pd.read_csv("../data/choice_150.csv")
win_150 = pd.read_csv("../data/wi_150.csv")
loss_150= pd.read_csv("../data/lo_150.csv")
```

Merging Data

We will firstly merge all of our data together into large csv files representing the index, choice, win, and loss. For now we will ignore the different payoff schemes introduced for different studies.

As the index for the csv's will no longer be unique, we will ignore it using `ignore_index=True`. This will also result in some NaN values but they **will not** affect our results at this point.

```
full_index = pd.concat([index_95, index_100, index_150], ignore_index=True)
full_choice = pd.concat([choice_95, choice_100, choice_150], ignore_index=True)
full_win = pd.concat([win_95, win_100, win_150], ignore_index=True)
full_loss = pd.concat([loss_95, loss_100, loss_150], ignore_index=True)
```

KMeans Clustering of Net Win & Net Loss

The first thing we will do is cluster by **Net Win \(\times\) Net Loss** for each participant.

Let's create a new DF with two columns, net_win & net_loss with each row representing a participant. We need:

1. Array representing net win / loss for each participant
2. Suitable dataframe

```
#Creating our arrays
net_win = np.array(full_win.sum(axis=1))
net_loss = np.array(full_loss.sum(axis=1))
```

```
#Creating and displaying our dataframe
net_df = pd.DataFrame(index=full_win.index, columns=["Winnings", "Losses", "Final"],
            data={"Winnings":net_win, "Losses":net_loss, "Final":net_win+net_loss})
net_df.head()
```

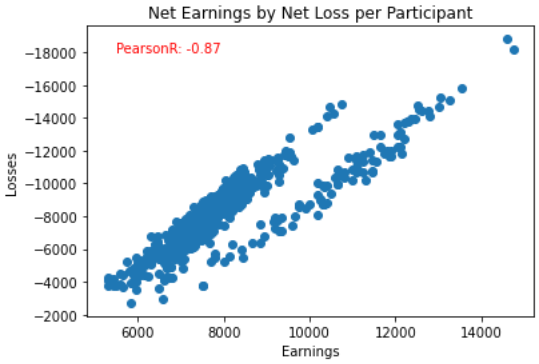
	Winnings	Losses	Final
0	5800.0	-4650.0	1150.0
1	7250.0	-7925.0	-675.0
2	7100.0	-7850.0	-750.0
3	7000.0	-7525.0	-525.0
4	6450.0	-6350.0	100.0

So now that we have a suitable dataframe consisting of winnnings and losses per participant, it's time to do some analysis before we begin clustering.

Let's visualize the data straight away to help us better understand the distribution of our data

Note that we invert the Y axis in the below plot.

```
#Plotting points
plt.scatter(net_df["Winnings"], net_df["Losses"])

#Inverting y-axis
plt.gca().invert_yaxis()

#Axes labels
plt.xlabel('Earnings')
plt.ylabel('Losses')

#Plot Title
plt.title("Net Earnings by Net Loss per Participant")

#Inserting Pearson Correlation
corr, _=pearsonr(net_df["Winnings"], net_df["Losses"])
corr = "PearsonR: " + str((float("{0:.2f}".format(corr))))
plt.text(5500, -18000, corr, color='red')

plt.show()
```



Linear Relationship

From this graph we can see a **strong negative (Y is inverted) linear relationship** indicating that the more "money" a participant earned in the challenge, the more they lost in general.

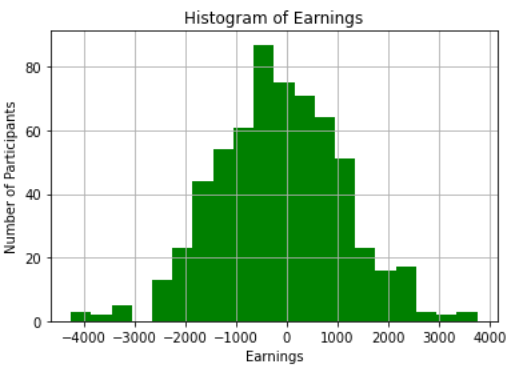
This relationship is confirmed with the inclusion of **Pearson Correlation Coefficient**.

Let's create a histogram representing the total earnings (**Net Win + Net Loss**).

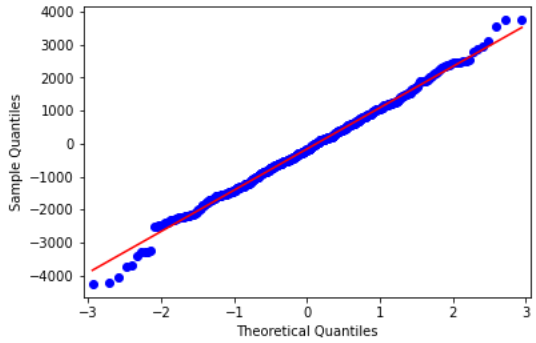
```
#Let's see the range of earnings in our data:
print("Minimum: {} Maximum: {}".format(
    net_df["Final"].describe()["min"], net_df["Final"].describe()["max"]))
```

```
Minimum: -4250.0 Maximum: 3750.0
```

```
#The histogram of the data
n, bins, patches = plt.hist(net_df["Final"], 20, facecolor='g')

plt.xlabel('Earnings')
plt.ylabel('Number of Participants')
plt.title('Histogram of Earnings')
plt.grid(True)
plt.show()
```



```
#Let's learn more about the distribution using a QQPlot
qqplot(net_df["Final"], line='s')
pyplot.show()
```



Interpretation of Histogram and QQPlot

The histogram with a total of 20 bins shows us that the distribution of the data somewhat resembles a normal distribution. This is to be expected when working with human-generated data.

However, it is clear that there is data at the two extremeties of the bell curve indicating groups of **outliers**. These outliers are also clearly represented in the QQplot which has tails straying from the centre line.

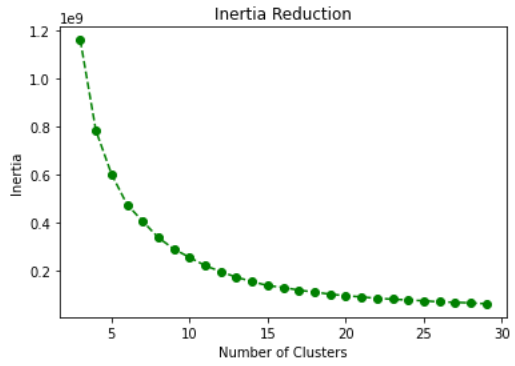
These groups of outliers are representative of participants which had **total earnings considerably higher or lower** than the average. Later on, we will create a boxplot representative of participant's choices with respect to their clusters where we will further discuss the existence of outliers.

Clustering

Visually, it is unclear how many clusters will be most suitable for this graph. We will create a **lineplot** representative of the **total inertia** for each varying quantity of clusters. From this plot, we can interprete the **elbow point** of the line which indicates a suitable number of clusters. Any amount of clusters after this point results in a negligible reduction of inertia and is therefore unnecessary.

This graph will allow us to **visualise the decrease in inertia** between cluster quantities. We can then decide by eye which amount is most suitable. This method is called finding **the elbow-point** of a line curve.

```
#Let X equal our points
X = net_df[["Winnings", "Losses"]]

#List of inertias
inertias = []

#Finding and appending Inertia to list
for i in range(3, 30):
    kmeans = KMeans(n_clusters=i).fit(X)
    inertias.append(kmeans.inertia_)

#Create Inertia Line Plot
plt.plot(range(3,30), inertias, "go--")
plt.title('Inertia Reduction')
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia')
plt.show()
```



Interpreting the Inertia Graph

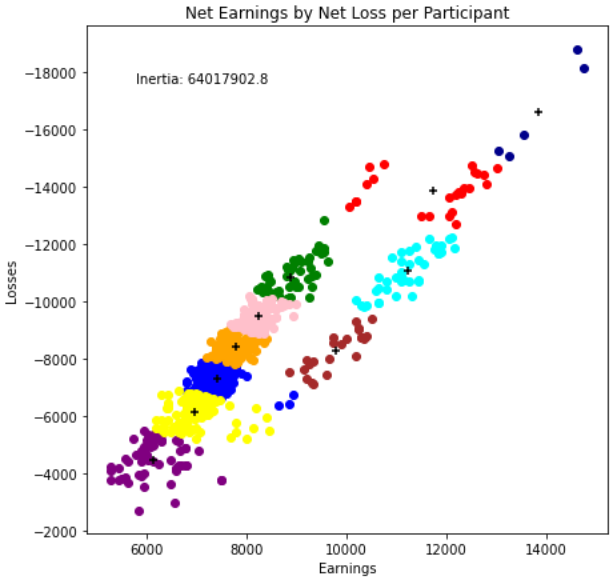
We can see that the rate of change (or the slope) decreases drastically for more than 10 clusters. Therefore, we can assume that **10 clusters** is a suitable number for this particular graph.

Let's now continue the process by employing the KMeans algorithm with a cluster allowance of 10. We will then calculate the **label** associated with each data point. This will allow us to seperate the data points on our graph by colour depending on the cluster they belong to. We will also find the **centroids** of each cluster and plot them with a **+** symbol.

Cluster Graph

```
#Let X equal our points
X = net_df[["Winnings", "Losses"]]

#Perform kmeans and fit X
kmeans = KMeans(n_clusters=10).fit(X)

#Calculating Labels
label = kmeans.labels_

#Calculating Centroids
centroids = kmeans.cluster_centers_
```

```
plt.figure(figsize=(7,7))

#filter rows of original data by label
filtered_label0, filtered_label1, filtered_label2 = net_df[label == 0], net_df[label == 1],
net_df[label == 2]
filtered_label3, filtered_label4, filtered_label5 = net_df[label == 3], net_df[label == 4],
net_df[label == 5]
filtered_label6, filtered_label7, filtered_label8, filtered_label9 = net_df[label == 6],
net_df[label == 7], net_df[label == 8], net_df[label == 9]

filtered_labels =
[filtered_label0,filtered_label1,filtered_label2,filtered_label3,filtered_label4,filtered_label5
                 ,filtered_label6,filtered_label7,filtered_label8,filtered_label9]
colors=["blue", "red", "green", "purple", "orange", "yellow", "aqua", "darkblue", "brown",
"pink"]
#Inverting y-axis
plt.gca().invert_yaxis()

#plotting the clusters
for i in range(0, len(filtered_labels)):
    plt.scatter(filtered_labels[i]["Winnings"], filtered_labels[i]["Losses"], color=colors[i])

#Plotting the centroids
plt.scatter(centroids[:,0], centroids[:,1], marker='+', color="black")

#Calculating & placing inertia
inertia= "Inertia: " + str(float("{0:.2f}".format(kmeans.inertia_)))

#Placing Text
plt.text(5800, -18000, inertia,  verticalalignment='top')

#Axes labels
plt.xlabel('Earnings')
plt.ylabel('Losses')

#Plot Title
plt.title("Net Earnings by Net Loss per Participant")

#Saving image for reference
plt.savefig("all_clusters")
plt.show()
```



Cluster Graph Analysis

In the above graph it is clear that the data does not all come from the same distribution. There are **two very apparent lines**.

We can easily imagine that the more dense line on the left is likely a **combination of data points** from studies allowing participants a total of **95 & 100 tries**. The less dense line further right in the diagram being made up of the remaining participants allowed a total of **150 tries**. This would make sense as more tries would allow users to generally receive more rewards and penalties causing their datapoint to be further to the top right.

As stated previously, the studies employed **various payoff schemes**. This means that harsher penalties and more lucrative rewards were available in some studies but **not in others**. This has an effect on the total amount of profit or loss a participant can make during the game.

We can see that the clusters do a reasonably good job dividing the data. However, some clusters seem less reasonable, particularly the **yellow, blue, and red clusters**. Later on we will be performing a **federated cluster analysis** meaning studies will be clustered independently of each other. This should result in more reasonable clusters.

Comparison Between Opposite Groups

From the above diagram I am interested in viewing the difference between the choices made by users in the **purple cluster** (bottom-left) versus users in the **dark blue** cluster (top-right). We have already segmented our dataset using the `filtered_label3` & `filtered_label7` sets above. We can filter our `full_choice` dataset in a similar fashion.

For simplicity, we will choose **6** participants out of the **57** in the purple group for analysis. We will choose **3 users with the highest profit and 3 users with the lowest profit**.

Clustering participants based on the decks they chose from is a difficult task. Instead, a **time-series chart** may reveal trends in the data more easily. A time-series chart will reveal to us the decision making process of users. We expect to see choices steadily lean towards the **more favourable decks of C and D** in situations where the participant performed well.

As we are viewing the polar extremes of the data, we should expect to see contrasting strategies employed by subjects.

Note: Re-running the cluster algorithm above will cause the clusters to change colour and no longer be associated with the same labels

Pre-Processing

```
#Filter out top and bottom 3 results
purples = pd.concat([filtered_label3.sort_values("Final").iloc[:3],
filtered_label3.sort_values("Final").iloc[-3:]])
```

```
#Here we see the table of top 3 and bottom 3 earners.
purples
```

	Winnings	Losses	Final
176	5950.0	-5500.0	450.0
116	5750.0	-5200.0	550.0
123	6050.0	-5350.0	700.0
323	6570.0	-3000.0	3570.0
526	7500.0	-3750.0	3750.0
575	7500.0	-3750.0	3750.0

```
#Let's extract the choices of these participants by using their index numbers
purple_choices = full_choice[full_choice.index.isin(purples.index)]
```

```
#Let's do the same for the dark blue group. We can use the label 7 already provided to extract
the choices as
#there are only 5 members of the cluster.

blue_choices=full_choice[label == 7]
```

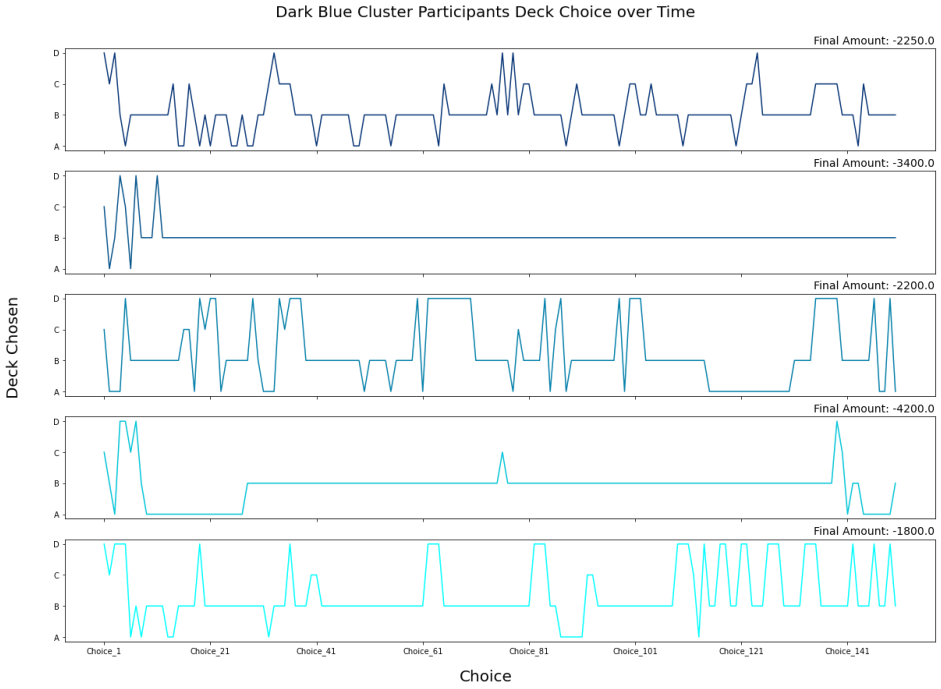
```
#Let's see the blue choices table
blue_choices.head()
```

	Choice_1	Choice_2	Choice_3	Choice_4	Choice_5	Choice_6	Choice_7	Choice_8	Choic
532	4	3	4	2	1	2	2	2	
547	3	1	2	4	3	1	4	2	
549	3	1	1	1	4	2	2	2	
553	3	2	1	4	4	3	4	2	
616	4	3	4	4	4	1	2	1	

5 rows × 150 columns

Dark Blue Cluster Choice Time-Series

```
#Creating figure and subplots
fig, axs = plt.subplots(5,1,figsize=(20,14), sharex=True)
axs[0].plot(blue_choices.columns, blue_choices.iloc[0], color='#083577')
axs[1].plot(blue_choices.columns, blue_choices.iloc[1], color='#07558d')
axs[2].plot(blue_choices.columns, blue_choices.iloc[2], '#0581ab')
axs[3].plot(blue_choices.columns, blue_choices.iloc[3], '#02c6d9')
axs[4].plot(blue_choices.columns, blue_choices.iloc[4], '#00feff')

#Adjusting xticks and yticks
plt.setp(axs, yticks=[1,2,3,4], yticklabels=["A", "B", "C", "D"])
plt.xticks(np.arange(0,151,step=20))

#Include Final Amount as title to subplot
for i in range(0, len(axs)):
    fin="Final Amount: {}".format(net_df.loc[blue_choices.index[i]]["Final"])
    axs[i].set_title(fin, loc='right', fontsize=14)

#Setting Title
fig.text(0.5, 0.92, "Dark Blue Cluster Participants Deck Choice over Time", fontsize=20,
ha="center")

#Setting axes labels
fig.text(0.5, 0.08, 'Choice', ha='center', va='center', fontsize=20)
fig.text(0.08, 0.5, 'Deck Chosen', ha='center', va='center', rotation='vertical', fontsize=20)

plt.show()
```



Interpretation of Dark Blue Participants' Choices

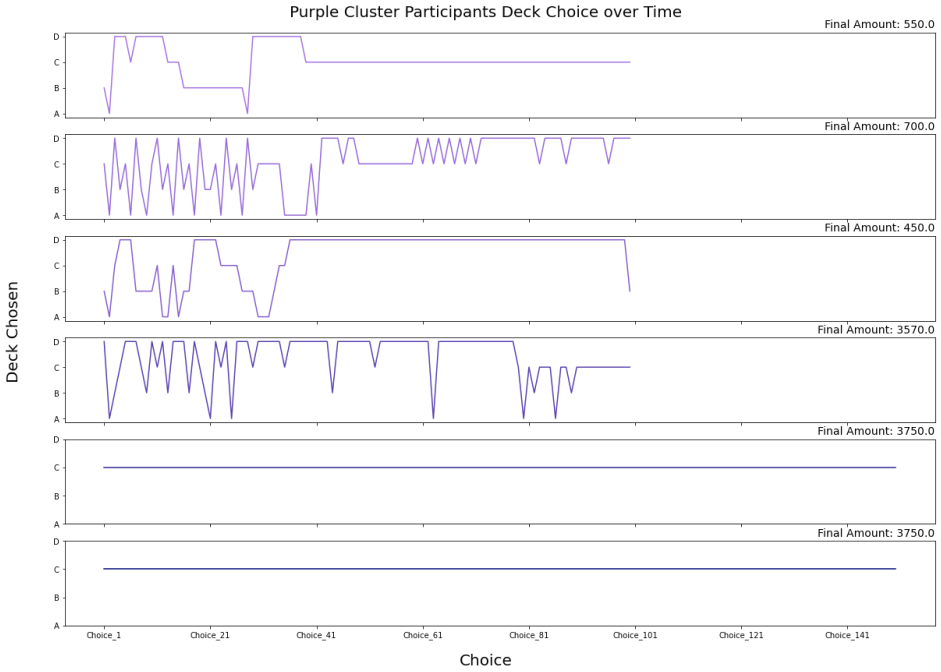
Overall, it is clear that a tendency to choose 'Deck B' is a similarity between participants of this cluster. This is particularly evident in the **second** and **fourth** subplots. Both of these users selected 'Deck B' for the majority of their choices.

From viewing the remaining 3 time-series charts, we can see that participants **regularly returned to choosing 'Deck B'** for long periods of time, only **occassionally choosing from a different deck**. We understand that 'Deck B' was not considered a favourable deck.

The less-favourable nature of 'Deck B' is made more apparent by the fact that the two subplots (2nd, and 4th) choosing 'Deck B' the most ended up with the **Greatest Losses** as their final amount.

Purple Cluster Choice Time-Series

```
#Creating figure and subplots
fig, axs = plt.subplots(6,1,figsize=(20,14), sharex=True)
axs[0].plot(purple_choices.columns, purple_choices.iloc[0], color='#a673e3')
axs[1].plot(purple_choices.columns, purple_choices.iloc[1], color='#9669d9')
axs[2].plot(purple_choices.columns, purple_choices.iloc[2], '#865ece')
axs[3].plot(purple_choices.columns, purple_choices.iloc[3], '#5941b1')
axs[4].plot(purple_choices.columns, purple_choices.iloc[4], '#2d2495')
axs[5].plot(purple_choices.columns, purple_choices.iloc[5], '#040a7c')

#Adjusting xticks and yticks
plt.setp(axs, yticks=[1,2,3,4], yticklabels=["A", "B", "C", "D"])
plt.xticks(np.arange(0,151,step=20))

#Include Final Amount as title to subplot
for i in range(0, len(axs)):
    fin="Final Amount: {}".format(net_df.loc[purple_choices.index[i]]["Final"])
    axs[i].set_title(fin, loc='right', fontsize=14)

#Setting Title
fig.text(0.5, 0.9, "Purple Cluster Participants Deck Choice over Time", fontsize=20,
ha="center")

#Setting axes labels
fig.text(0.5, 0.08, 'Choice', ha='center', va='center', fontsize=20)
fig.text(0.08, 0.5, 'Deck Chosen', ha='center', va='center', rotation='vertical', fontsize=20)

plt.show()
```



Interpretation of the Purple Participants' Choices

The above 6 time-series charts represent those particpants of the purple cluster with **the highest and lowest** final amount scores. We can see that all users ended the experiment with a positive amount of score. Although some scores are low, they are considerably greater than the scores of the dark blue participants we previously examined. As the lowest earners of the cluster ended with a positive score, we know **the entire cluster of participants ended with a positive score**.

It should be noted that we see **varying time-series lengths** as some participants included here took part in a study involving 95 tries while others had 150. Something we alluded to in the Cluster Graph Analysis section is that the **number of attempts a subject is allowed can affect their position in the cluster**. Maybe we should account for the fact that participants in the Dark Blue cluster were allowed more attempts and therefore had more opportunity to lose score. In the next notebook, we will re-cluster participants based on their Net Win and Net Loss normalised to their number of attempts.

Visually, we can see these **users favour decks 'C' and 'D'**. We understand these decks to be **favourable** in this experiment. The first 4 subplots from the top visually describe the thought process of the participants. These subjects began by testing each deck but over time favoured 'C' and 'D'.

The remaining 2 subplots exhibit strange choice patterns. Both users selected **only 'Deck C'** for the duration of the test. This may indicate to us that these particular participants had prior knowledge before beginning the study. However, there is also a possiblity that these are legitimate scores. Maybe this is more indicative of a neurological condition such as OCD.

Comparison of All Clusters

In the following section, we will examine some of the differences between the clusters. We will use a **Box and Whisker Plot** to show the general distribution of choices for each cluster. We should expect the boxplot to confirm our analysis above in that;

1. The Dark Blue cluster should be centralised around 'Deck B'
2. The Purple cluster should be centralised around 'Deck C'

Analysis of the boxplot will also allow us to see the **spread** of choice amongst the clusters. It is possible that some clusters may have large spreads between their 25% amd 75% quartiles, while others may be more condensed. Maybe there will be a correlation between clusters with a large spread and clusters that seem less natural on the cluster diagram above.

```
#Let's create a series object representing the average deck chosen per choice for each cluster
darkblue_avg = full_choice[label==7].T.mean(axis=1)
purple_avg = full_choice[label==3].T.mean(axis=1)
orange_avg = full_choice[label==4].T.mean(axis=1)
red_avg = full_choice[label==1].T.mean(axis=1)
yellow_avg = full_choice[label==5].T.mean(axis=1)
aqua_avg = full_choice[label==6].T.mean(axis=1)
brown_avg = full_choice[label==8].T.mean(axis=1)
pink_avg = full_choice[label==9].T.mean(axis=1)
green_avg = full_choice[label==2].T.mean(axis=1)
blue_avg = full_choice[label==0].T.mean(axis=1)

#Let's combine these objects to create a dataframe of average deck chosen per choice by each
cluster.
avg_choices=pd.DataFrame(data=[blue_avg, red_avg, green_avg, purple_avg, orange_avg, yellow_avg,
aqua_avg, darkblue_avg, brown_avg, pink_avg],
            index=["Blue", "Red","Green","Purple","Orange","Yellow", "Aqua", "Dark Blue",
"Brown", "Pink"])
```

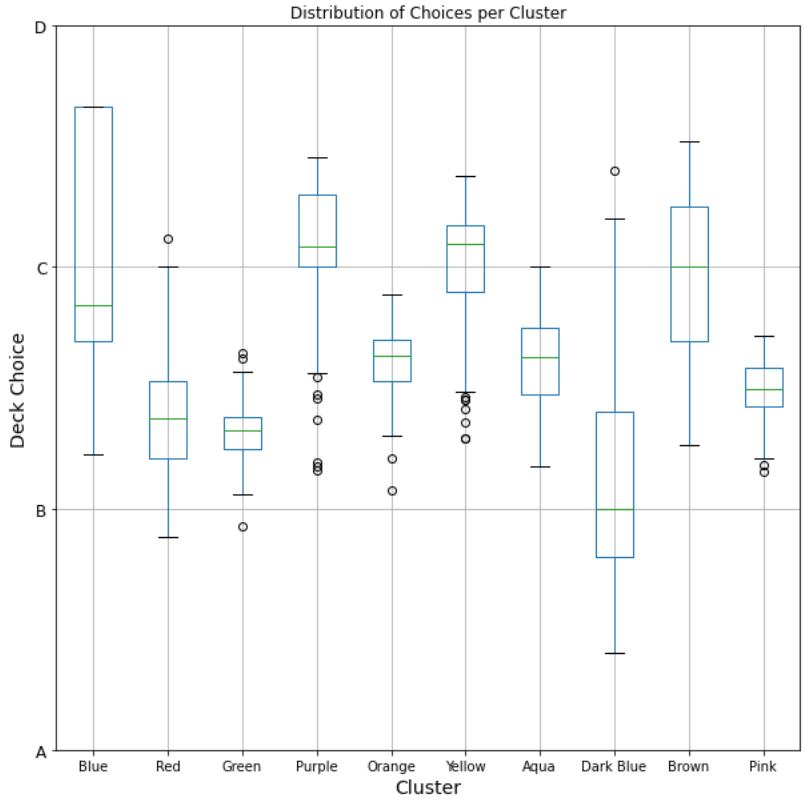
```
#Let's see the dataframe
avg_choices.head()
```

	Choice_1	Choice_2	Choice_3	Choice_4	Choice_5	Choice_6	Choice_7	Choice_8	Ch
Blue	2.250000	2.629310	2.224138	2.439655	2.284483	2.405172	2.387931	2.310345	2.
Red	2.583333	2.458333	2.625000	2.750000	2.291667	2.041667	2.375000	2.333333	2.
Green	1.924528	2.320755	2.377358	2.094340	2.226415	2.264151	2.056604	2.301887	2.
Purple	2.192982	2.157895	2.456140	2.473684	2.175439	2.368421	2.543860	2.631579	2.
Orange	2.076923	2.461538	2.401709	2.495726	2.205128	2.299145	2.418803	2.384615	2.

5 rows × 150 columns

Boxplot

```
#Using .boxplot() gives an annoying warning so we turn warnings off!
import warnings
warnings.filterwarnings('ignore')

plt.figure(figsize=(10,10))

#We use avg_choices.boxplot() as plt.boxplot(avg_choices) refuses to show clusters containing
NaN values
avg_choices.T.boxplot()

#Adjusting xticks and yticks
plt.xticks(np.arange(1,11),["Blue", "Red","Green","Purple","Orange","Yellow", "Aqua", "Dark
Blue", "Brown", "Pink"])
plt.yticks(np.arange(1,5),["A", "B", "C", "D"], fontsize=12)

#Setting Labels and Title
plt.title("Distribution of Choices per Cluster")
plt.xlabel("Cluster", fontsize=14)
plt.ylabel("Deck Choice", fontsize=14)
plt.show()
```



Interpretation of the Box Plot

Outliers in this plot represent a particular choice where the mean of all participants choices is far greater or less than the mean of all the other choices in the game. The existince of outliers could indicate that there is a common point in which subjects begin to try a different strategy. However, it could also be down to chance.

The **median** and the **spread** tell us more than the outliers here. As expected, the Dark Blue median rests firmly on 'B', while the Purple median lies just above 'C'. The distribution of the Yellow cluster is similar to the Purple cluster. They are also beside each other on the cluster diagram. This could indicate that performance of these two clusters are similar. The 'Blue' cluster has the largest **inter-quartile range**. Its median also rests near 'C'.

If we consider clusters with medians and ranges residing closer to 'Deck C' and 'Deck D' in the boxplot as more performant, it becomes clear that these clusters, namely **Blue**, **Purple**, **Yellow**, and **Brown**, are found closer to the **bottom-left** corner of the cluster diagram. The remaining clusters are found further towards the **top-right**.

Conclusion

This is the end of this Jupyter Notebook. Here we clustered our data by Net Earnings and Net Losses irrespective of study conditions such as **reward scheme** and **number of attempts**. We were able to see a linear relationship in the data which told us that in most cases, as a participant earns more reward, they consequently have more loses. This is related to participants deck choice. Users who favoured Deck 'B' such as the dark blue cluster received greater rewards but endured harsher penalties resulting in a net loss over 10 turns. We found that the data closely resembles a **normal distribution** but does contain some **outliers** with large earnings or losses.

From viewing the cluster diagram, we understand that there are **two appararent lines** visible in the graph. This could indicate two prominent and contrasting testing conditions. This will be investigated further in the next Notebook. We will employ a *federated clustering approach*. In this approach, data is considered private and cannot be merged together. We will cluster each dataset individually, and then attempt to recalculate the k-means based on the original k-means of the individual studies.

By Seán Cummins
© Copyright 2021.