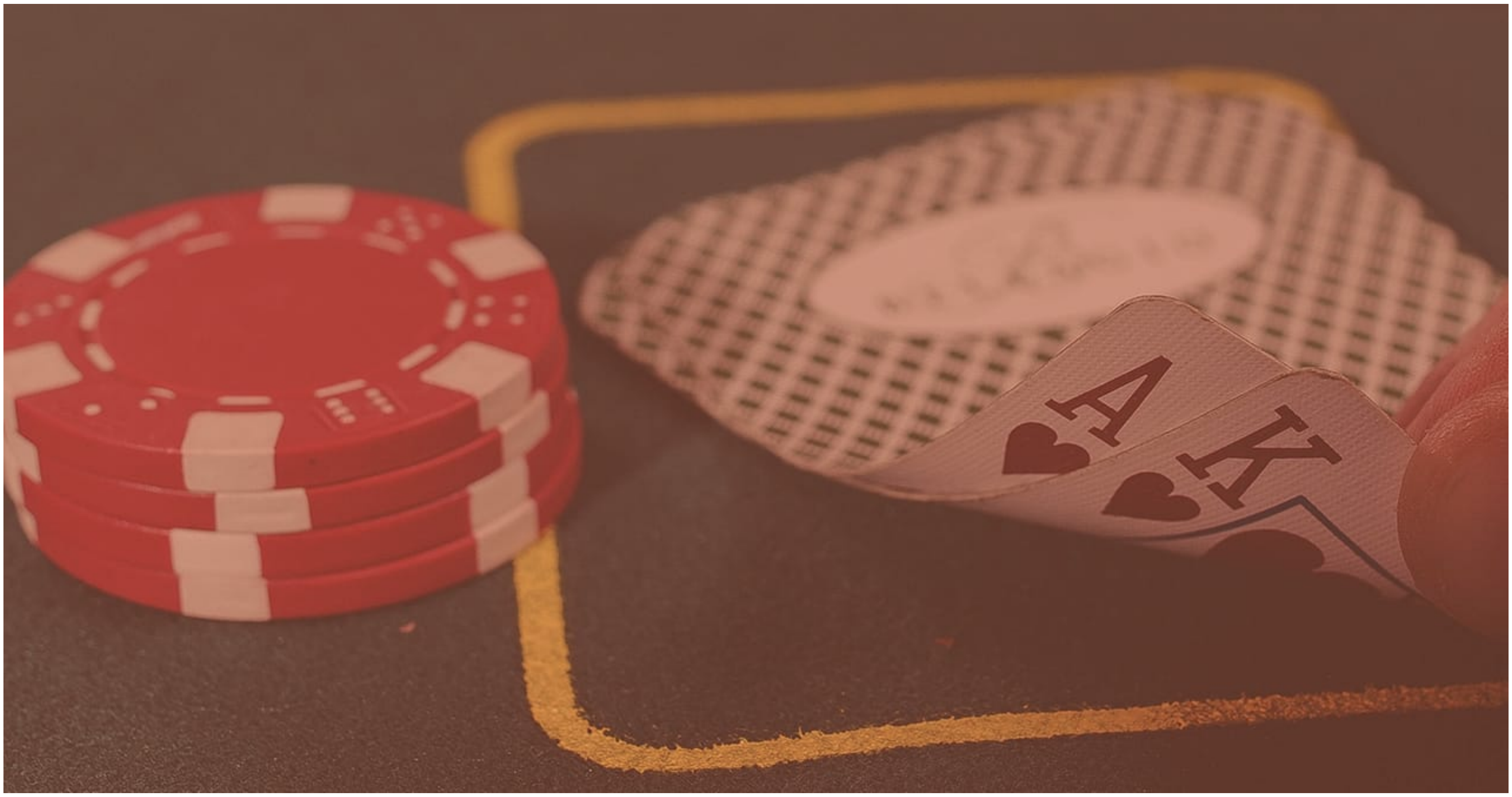
CA4015 Assignment 1

Introduction

Iowa Gambling Task¶

The Iowa Gambling Task(IGT) is a pyscological card game thought to portray real world decision making and help understand how and why people make decisions. Typically, the particpiants start with a loan of $2000 and are typically given a deck of four cards to pick from. The deck either rewards or punishes the particpiant each time and the end goal of the game is to make more money than what they started with. Some of the cards are more favourable than others, providing a steady winning return over the long term whereas others can be typically destructive but sporadically produce high winnings.



Dataset

For the purpose of this assignment there is varying trial lengths for each task. Typically, the task is run in 100 trials but in this instance we have a 95 trial, 100 trial and 150 trial experiments. We were given a variety of data for the trials. There were 617 particpiants across all the studies and all were reported as "healthy". We had each subjects choice on their respective trial, their winnings on the respective rounds combined with their losses for each round. We also had an "index" file which told us which study the subject was part of. An overview of our data and other information such as the particpiant demographics can be seen here.

Methodology

Initially, we start by breaking down the data and seeing any interesting trends. We try to see which studies produced the highest winnings/losses and how subjects decision making flowed over time (i.e. did they consistently win small or try change strategy by going for broke and another set of cards). We try to combine this with the background demographic information provided to see can we tell anything from the studies. We then cluster the data accordingly by winnings and studys to detect any trends and try to analyse card selection on age demography, gender balance in a study or more.

1. Data Analysis and Experiments

Read in data

```
In [1]: import pandas as pd
        import matplotlib.pyplot as plt
        import numpy as np
        import seaborn as sns
```

```
In [2]: df = pd.read_csv('data/choice_100.csv')
        df.head()
```

Out[2]:

	Choice_1	Choice_2	Choice_3	Choice_4	Choice_5	Choice_6	Choice_7	Choice_8	Choice_9	Choice_10	...	Choice_91	Choice
Subj_1	1	1	2	4	3	2	1	2	4	2	...	1	
Subj_2	2	1	4	4	3	2	3	2	1	2	...	4	
Subj_3	4	2	3	1	4	2	4	4	4	3	...	3	
Subj_4	4	3	4	2	1	4	3	2	2	2	...	4	
Subj_5	1	2	2	2	2	3	4	1	4	1	...	2	

5 rows × 100 columns

```
In [3]: index95 = pd.read_csv('data/index_95.csv')
        index100 = pd.read_csv('data/index_100.csv')
        index150 = pd.read_csv('data/index_150.csv')
```

Let's just confirm our python version we are using on our scripts

```
In [55]: from platform import python_version
         print(python_version())

3.8.8
```

Initial data exploration - How many per study made profit?

Here we check the study that used 95 trials of the experiment and see how many of the 15 subjects made profit.

```
In [4]: win95 = pd.read_csv('data/wi_95.csv')
        loss95 = pd.read_csv('data/lo_95.csv')
        totalloss95 = loss95.sum(axis=1)
        totalloss95.head()

Out[4]: Subj_1   -4650
        Subj_2   -7925
        Subj_3   -7850
        Subj_4   -7525
        Subj_5   -6350
        dtype: int64
```

```
In [5]: totalwin95 = win95.sum(axis=1)
        totalwin95.head()

Out[5]: Subj_1    5800
        Subj_2    7250
        Subj_3    7100
        Subj_4    7000
        Subj_5    6450
        dtype: int64
```

```
In [6]: margin95 = totalwin95 + totalloss95
        margin95.head()

Out[6]: Subj_1    1150
        Subj_2    -675
        Subj_3    -750
        Subj_4    -525
        Subj_5     100
        dtype: int64
```

```
In [7]: columns = ['Margin']

        margin95df = pd.DataFrame(margin95, columns=columns)
        margin95df
        ax1 = margin95df.plot.kde(title='Density plot of Profit/Loss margin for 95 people experiments', color=
        'turquoise')
        ax1.axvline(x=0, linestyle='--', color='red')

Out[7]: <matplotlib.lines.Line2D at 0x1c611ab4eb0>
```


```
In [8]: sum(margin95df.select_dtypes(np.number).gt(0).sum(axis=1))

Out[8]: 7
```

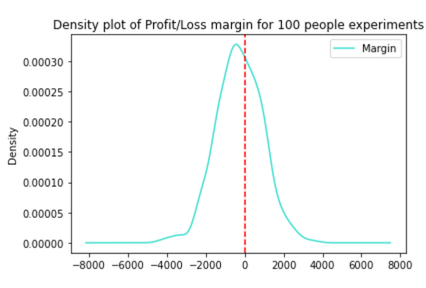
Under half (7/15) of the participants made profit in the 95 trial experiment. We now do the same for both the 100 trial and 150 trial experiments

```
In [9]: win100 = pd.read_csv('data/wi_100.csv')
        loss100 = pd.read_csv('data/lo_100.csv')
        totalloss100 = loss100.sum(axis=1)
        totalloss100.head()

Out[9]: Subj_1   -9950
        Subj_2   -8250
        Subj_3   -8600
        Subj_4   -5650
        Subj_5   -8600
        dtype: int64
```

```
In [10]: totalwin100 = win100.sum(axis=1)
         totalwin100.head()

Out[10]: Subj_1    8150
         Subj_2    7450
         Subj_3    8150
         Subj_4    6850
         Subj_5    7300
         dtype: int64
```

```
In [11]: margin100 = totalwin100 + totalloss100
         margin100.head()

Out[11]: Subj_1   -1800
         Subj_2    -800
         Subj_3    -450
         Subj_4    1200
         Subj_5   -1300
         dtype: int64
```

```
In [12]: columns = ['Margin']

         margin100df = pd.DataFrame(margin100, columns=columns)
         margin100df
         ax2 = margin100df.plot.kde(title='Density plot of Profit/Loss margin for 100 people experiments', color
         ='turquoise')
         ax2.axvline(x=0, linestyle='--', color='red')

Out[12]: <matplotlib.lines.Line2D at 0x1c61330ea60>
```




```
In [13]: sum(margin100df.select_dtypes(np.number).gt(0).sum(axis=1))

Out[13]: 208
```

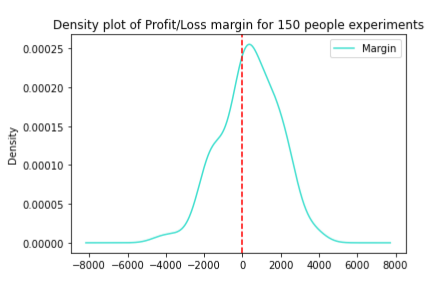
Only 41% of participants in the 100 trial experiment made money!

```
In [14]: win150 = pd.read_csv('data/wi_150.csv')
         loss150 = pd.read_csv('data/lo_150.csv')
         totalloss150 = loss150.sum(axis=1)
         totalloss150.head()

Out[14]: Subj_1   -12200
         Subj_2   -13950
         Subj_3    -9300
         Subj_4    -6750
         Subj_5    -6300
         dtype: int64
```

```
In [15]: totalwin150 = win150.sum(axis=1)
         totalwin150.head()

Out[15]: Subj_1    11650
         Subj_2    12350
         Subj_3    10200
         Subj_4     8950
         Subj_5     8200
         dtype: int64
```

```
In [16]: margin150 = totalwin150 + totalloss150
         margin150.head()

Out[16]: Subj_1    -550
         Subj_2   -1600
         Subj_3     900
         Subj_4    2200
         Subj_5    1900
         dtype: int64
```

```
In [17]: columns = ['Margin']

         margin150df = pd.DataFrame(margin150, columns=columns)
         margin150df
         ax = margin150df.plot.kde(title='Density plot of Profit/Loss margin for 150 people experiments', color=
         'turquoise')
         ax.axvline(x=0, linestyle='--', color='red')

Out[17]: <matplotlib.lines.Line2D at 0x1c613363d60>
```




```
In [18]: sum(margin150df.select_dtypes(np.number).gt(0).sum(axis=1))

Out[18]: 62
```

There is a much higher % of people making profit in the 150 trial experiment (63%). There is far less people taking part in this experiment than the 100 trial experiment but it may be worth drilling down further into the data to check the differences between the different groups undertaking the experiments here. It might be interesting to see if there is any trends surrounding the age profiles surveyed. We will now try to combine the study groups with our "margin" results.

Adding in Study undertakers to participant data

```
In [21]: margin95df['Study'] = index95['Study'].values
```

```
In [24]: margin150df['Study'] = index150['Study'].values
```

```
In [26]: margin100df['Study'] = index100['Study'].values
```

Here we will investigate the Study groups in the 150 person experiment. We will try to see does one group achieve better results in the task than the other.

```
In [28]: margin150df['Study'].value_counts()

Out[28]: Steingroever2011    57
         Wetzels             41
         Name: Study, dtype: int64
```

```
In [29]: print("Profit for the 150 trial Wetzels study")
         margin150df.loc[margin150df['Study'] == 'Wetzels', 'Margin'].sum()

         Profit for the 150 trial Wetzels study

Out[29]: 24000
```

```
In [30]: print("Profit for the 150 trial Steingroever2011 study")
         margin150df.loc[margin150df['Study'] == 'Steingroever2011', 'Margin'].sum()

         Profit for the 150 trial Steingroever2011 study

Out[30]: 12550
```

Assessment of 150 margin results for each study

This is interesting to note these values. Both studies make a large cumulative profit over the course of the 150 trials undertaken. There is a net profit between the two studies of 36,550 dollars, almost an average profit of 375 dollars per participant. It is interesting to note that the 41 students in "Wetzels" study were exclusively students, while in Steingroever2011's study there was a young average age (19.9) but no specific mention of if the participants were students or not. Although they made money it was almost half of the other group studied. We will measure this against the other datasets to see if age is a factor between the decision making of the groups.

```
In [31]: margin100df['Study'].value_counts()

Out[31]: Horstmann             162
         Wood                  153
         SteingroverInPrep      70
         Maia                   40
         Worthy                 35
         Premkumar              25
         Kjome                  19
         Name: Study, dtype: int64
```

```
In [32]: print("Loss for the 100 trial Horstmann study")
         margin100df.loc[margin100df['Study'] == 'Horstmann', 'Margin'].sum()

         Loss for the 100 trial Horstmann study

Out[32]: -6200
```

```
In [33]: print("Loss for the 100 trial Wood study")
         margin100df.loc[margin100df['Study'] == 'Wood', 'Margin'].sum()

         Loss for the 100 trial Wood study

Out[33]: -119410
```

```
In [34]: print("Loss for the 100 trial SteingroverInPrep study")
         margin100df.loc[margin100df['Study'] == 'SteingroverInPrep', 'Margin'].sum()

         Loss for the 100 trial SteingroverInPrep study

Out[34]: -4700
```

```
In [35]: print("Profit for the 100 trial Maia study")
         margin100df.loc[margin100df['Study'] == 'Maia', 'Margin'].sum()

         Profit for the 100 trial Maia study

Out[35]: 13600
```

```
In [36]: print("Loss for the 100 trial Worthy study")
         margin100df.loc[margin100df['Study'] == 'Worthy', 'Margin'].sum()

         Loss for the 100 trial Worthy study

Out[36]: -15100
```

```
In [37]: print("Profit for the 100 trial Premkumar study")
         margin100df.loc[margin100df['Study'] == 'Premkumar', 'Margin'].sum()

         Profit for the 100 trial Premkumar study

Out[37]: 5995
```

```
In [38]: print("Loss for the 100 trial Kjome study")
         margin100df.loc[margin100df['Study'] == 'Kjome', 'Margin'].sum()

         Loss for the 100 trial Kjome study

Out[38]: -8750
```

Assessment of margin for 100 trial experiments

The results here are in stark contrast to the 150 trial experiments. Despite less trials there is some significant losses accumalated by participants. Although the study conducted by Wood has a large number of participants in 153, the loss of 119,410 is certainly a major outlier. This equates to an average loss of roughly 780 dollars per person. It is particularly interesting to note here in table 1, we see this group has the oldest average age of any group in the study by some distance. The next oldest average age specified actually makes a profit (Premkumar). Again we see students with strong results in the Maia study as undergraduate students here make a strong profit, similar to the groups in the 150 trial experiments. We now check the 95 trial study as our last part of our margin analysis.

```
In [39]: margin95df.head()
```

Out[39]:

	Margin	Study
Subj_1	1150	Fridberg
Subj_2	-675	Fridberg
Subj_3	-750	Fridberg
Subj_4	-525	Fridberg
Subj_5	100	Fridberg

```
In [40]: print("Profit for the 95 trial Fridberg study")
         margin95df.loc[margin95df['Study'] == 'Fridberg', 'Margin'].sum()

         Profit for the 95 trial Fridberg study

Out[40]: 1250
```

Assessment of margin for the 95 trial study

In this trial as mentioned previously less than half of the participants made profit, but the 15 participant group made 1,250 over the course of the task. This obviously points to some bigger wins mitigating a collection of smaller losses in the group. The group who took part in this study were slightly older than some of the groups who made bigger profits (mean age of 29.6 years old). One thing consistent through this analysis of profit/loss margins has been student groups making more money than older groups. Age appears to be a factor but it may be interesting to look at the flow of each study too. By this, we mean looking at how participants profit/loss fluctuated on each turn and see did a series of wins lead them to change strategy and go for broke for example? Or did a series of losses at the start of the game potentially set the tone for some participants? To do this we will need to combine win and loss dataframes together and graph our results accordingly.

Analysis of participants selection flow

We will start by looking at the participants in the 95 trial experiment. To merge our dataframes together we will need to change the column names so that they share common names.

```
In [41]: columnnames95 = [f'Trial{num}' for num in range(1,96)]
         wins95test = win95
         wins95test = wins95test.set_axis(columnnames95, axis=1)
```

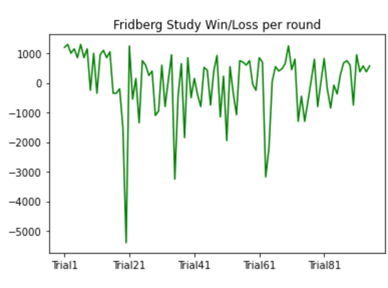
```
In [42]: loss95test = loss95
         loss95test = loss95test.set_axis(columnnames95, axis=1)
```

```
In [43]: df95_added = wins95test.add(loss95test, fill_value=0)
```

This study was all part of the Fridberg study so we don't need to worry about comparing other studies here.

```
In [44]: per_trial_95 = df95_added.sum(axis=0)
```

```
In [45]: per_trial_95.plot(title='Fridberg Study Win/Loss per round', color='green')

Out[45]: <AxesSubplot:title={'center':'Fridberg Study Win/Loss per round'}>
```



Next we move onto the 150 trial experiments and see how these studies flowed over the course of their trials

```
In [46]: columnnames150 = [f'Trial{num}' for num in range(1,151)]
         #columnnames95
         wins150test = win150
         #wins95test.head()
         wins150test = wins150test.set_axis(columnnames150, axis=1)
```

```
In [47]: loss150test = loss150
         loss150test = loss150test.set_axis(columnnames150, axis=1)
```

```
In [48]: loss150test.head()
```

Out[48]:

	Trial1	Trial2	Trial3	Trial4	Trial5	Trial6	Trial7	Trial8	Trial9	Trial10	...	Trial141	Trial142	Trial143	Trial144	Trial145	Trial146
Subj_1	-250	0	-350	0	0	-200	0	0	0	0	...	0	-250	0	-1250	0	0
Subj_2	-250	-350	0	0	0	0	0	0	0	0	...	0	0	0	-1250	0	-250
Subj_3	0	0	0	0	-150	-1250	0	-50	-350	0	...	0	0	0	0	0	0
Subj_4	0	0	0	0	-150	0	0	-50	-250	0	...	0	0	0	0	-250	0
Subj_5	0	0	0	0	0	0	-250	0	0	0	...	0	0	0	0	0	0

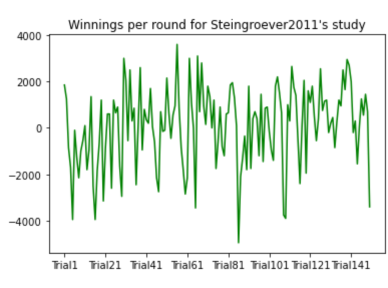
5 rows × 150 columns

```
In [49]: df150_added = wins150test.add(loss150test, fill_value=0)
```

```
In [50]: df150_added['Study'] = index150['Study'].values
```

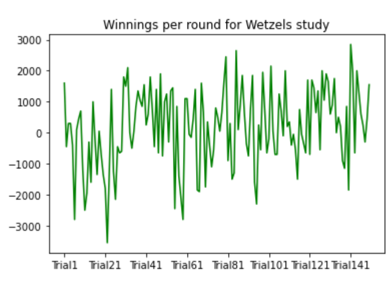
```
In [51]: teststein = df150_added.loc[df150_added['Study'] == 'Steingroever2011']
         del teststein['Study']
         per_trial_150_stein = teststein.sum(axis=0)
```

```
In [52]: per_trial_150_stein.plot(title="Winnings per round for Steingroever2011's study", color="green")

Out[52]: <AxesSubplot:title={'center':"Winnings per round for Steingroever2011's study"}>
```




```
In [53]: testWetzels = df150_added.loc[df150_added['Study'] == 'Wetzels']
         del testWetzels['Study']
         per_trial_150_wetzels = testWetzels.sum(axis=0)
```

```
In [54]: per_trial_150_wetzels.plot(title="Winnings per round for Wetzels study", color="green")

Out[54]: <AxesSubplot:title={'center':'Winnings per round for Wetzels study'}>
```



After assessing the participants winnings in each study and how their respective win / losses flowed per trial I decided to delve deeper into why these studies appeared more profitable than others. It is easy to point to the varying amounts of cards that pay out on each study and the number of respective trials each study undertook or the number of participants each study had. I felt it would be interesting to build on the win / loss flow per trials mentioned just above and try to pick out patterns accordingly. This would be something like if studies lost a lot of money is this down to a lower average choice of deck or constant changing of choosen cards and see if we could link this to research and studies on the IGT. This study and research could be related to gender based decision making or age demographies and see if these related studies findings hold true to our sample data.

2. Data preparation for Clustering

For our clustering analysis we need to prepare the data accordingly. Following on from our data analysis we want to try to cluster on the profit margin of participants against the number of times the subjects picked their most common deck choice or their average choice. We will then combine this with a scatter plot showing the study each subject was a part of and see what information we can gather from this. We will be looking at age demographies more so but also look to combine this with the amount of cards that pay out in each study and gender breakdowns also. To do this we need to create appropriate CSV files that we can then use for clustering.

In [73]:
```
import pandas as pd
import seaborn as sn
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans, AgglomerativeClustering
from sklearn.metrics import silhouette_score
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import LabelEncoder
from sklearn import preprocessing
```

In [74]:
```
index95 = pd.read_csv('data/index_95.csv')
index100 = pd.read_csv('data/index_100.csv')
index150 = pd.read_csv('data/index_150.csv')
win95 = pd.read_csv('data/wi_95.csv')
win100 = pd.read_csv('data/wi_100.csv')
win150 = pd.read_csv('data/wi_150.csv')
loss95 = pd.read_csv('data/lo_95.csv')
loss100 = pd.read_csv('data/lo_100.csv')
loss150 = pd.read_csv('data/lo_150.csv')
choice95 = pd.read_csv('data/choice_95.csv')
choice100 = pd.read_csv('data/choice_100.csv')
choice150 = pd.read_csv('data/choice_150.csv')
```

Creating margin csv files

In [75]:
```
columnnames95 = [f'Trial{num}' for num in range(1,96)]
wins95 = win95
wins95 = wins95.set_axis(columnnames95, axis=1)
wins95.head()
```

Out[75]:

	Trial1	Trial2	Trial3	Trial4	Trial5	Trial6	Trial7	Trial8	Trial9	Trial10	...	Trial86	Trial87	Trial88	Trial89	Trial90	Trial91
Subj_1	100	100	100	100	100	100	100	100	100	100	...	50	50	50	50	50	50
Subj_2	100	100	50	100	100	100	100	100	100	100	...	50	100	100	100	100	100
Subj_3	50	50	50	100	100	100	100	100	100	100	...	100	100	100	50	50	50
Subj_4	50	50	100	100	100	100	100	50	100	100	...	100	50	50	50	50	50
Subj_5	100	100	50	50	50	100	100	100	100	100	...	50	50	50	50	50	50

5 rows × 95 columns

In [76]:
```
losses95 = loss95
losses95 = losses95.set_axis(columnnames95, axis=1)
losses95.head()
```

Out[76]:

	Trial1	Trial2	Trial3	Trial4	Trial5	Trial6	Trial7	Trial8	Trial9	Trial10	...	Trial86	Trial87	Trial88	Trial89	Trial90	Trial91
Subj_1	0	0	0	0	0	0	0	0	-1250	0	...	0	0	0	0	0	0
Subj_2	0	0	0	0	0	0	0	0	0	0	...	-50	-300	0	-350	0	0
Subj_3	0	0	0	0	0	0	0	-150	0	0	...	0	0	0	0	0	0
Subj_4	0	0	0	0	-150	0	0	0	0	0	...	0	-50	0	-50	-50	0
Subj_5	0	0	0	0	0	0	-150	0	0	0	...	-75	0	0	0	0	0

5 rows × 95 columns

In [77]:
```
df95_sum = wins95.add(losses95, fill_value=0)
df95_sum.head()
```

Out[77]:

	Trial1	Trial2	Trial3	Trial4	Trial5	Trial6	Trial7	Trial8	Trial9	Trial10	...	Trial86	Trial87	Trial88	Trial89	Trial90	Trial91
Subj_1	100	100	100	100	100	100	100	100	-1150	100	...	50	50	50	50	50	50
Subj_2	100	100	50	100	100	100	100	100	100	100	...	0	-200	100	-250	100	100
Subj_3	50	50	50	100	100	100	100	-50	100	100	...	100	100	100	50	50	50
Subj_4	50	50	100	100	-50	100	100	50	100	100	...	100	0	50	0	0	50
Subj_5	100	100	50	50	50	100	-50	100	100	100	...	-25	50	50	50	50	50

5 rows × 95 columns

In [78]:
```
profit95 = df95_sum.sum(axis=1)
profit95df = pd.DataFrame(data=profit95)
profit95df.rename(columns={0: 'Margin'}, inplace=True)
profit95df.head()
```

Out[78]:

	Margin
Subj_1	1150
Subj_2	-675
Subj_3	-750
Subj_4	-525
Subj_5	100

In [79]:
```
choice95.head()
```

Out[79]:

	Choice_1	Choice_2	Choice_3	Choice_4	Choice_5	Choice_6	Choice_7	Choice_8	Choice_9	Choice_10	...	Choice_86
Subj_1	2	2	2	2	2	2	2	2	2	1	...	4
Subj_2	1	2	3	2	2	2	2	2	2	2	...	3
Subj_3	3	4	3	2	2	1	1	1	1	2	...	2
Subj_4	4	3	1	1	1	2	2	3	2	2	...	2
Subj_5	1	2	3	4	3	1	1	2	2	2	...	3

5 rows × 95 columns

In [80]:
```
most_common_count = choice95.apply(pd.Series.value_counts, axis=1)
most_common_count = most_common_count.max(axis=1)
profit95df['Most Common Choice Picked'] = most_common_count
profit95df.head()
```

Out[80]:

	Margin	Most Common Choice Picked
Subj_1	1150	71
Subj_2	-675	33
Subj_3	-750	38
Subj_4	-525	38
Subj_5	100	46

In [81]:
```
mode95 = choice95.mode(axis=1)
mode95.rename(columns={0: 'Most Common Choice'}, inplace=True)
```

In [82]:
```
profit95df['Most Common Choice'] = mode95['Most Common Choice'].values
```

In [83]:
```
profit95df['Study'] = index95['Study'].values
profit95df.head()
```

Out[83]:

	Margin	Most Common Choice Picked	Most Common Choice	Study
Subj_1	1150	71	4	Fridberg
Subj_2	-675	33	4	Fridberg
Subj_3	-750	38	4	Fridberg
Subj_4	-525	38	4	Fridberg
Subj_5	100	46	4	Fridberg

In [84]:
```
mean95 = choice95.mean(axis=1)
mean95df = pd.DataFrame(data=mean95)
mean95df.rename(columns={0: 'Average Choice'}, inplace=True)
profit95df['Average Choice'] = mean95df['Average Choice'].values
profit95df.head()
```

Out[84]:

	Margin	Most Common Choice Picked	Most Common Choice	Study	Average Choice
Subj_1	1150	71	4	Fridberg	3.400000
Subj_2	-675	33	4	Fridberg	2.568421
Subj_3	-750	38	4	Fridberg	2.778947
Subj_4	-525	38	4	Fridberg	2.810526
Subj_5	100	46	4	Fridberg	3.021053

In [85]:
```
profit95df.to_csv('Data/cleaned95.csv')
```

We now do this for the 100 trial and 150 trial experiments

In [86]:
```
columnnames100 = [f'Trial{num}' for num in range(1,101)]
wins100 = win100
wins100 = wins100.set_axis(columnnames100, axis=1)
```

In [87]:
```
losses100 = loss100
losses100 = losses100.set_axis(columnnames100, axis=1)
```

In [88]:
```
df100_sum = wins100.add(losses100, fill_value=0)
```

In [89]:
```
profit100 = df100_sum.sum(axis=1)
profit100df = pd.DataFrame(data=profit100)
profit100df.rename(columns={0: 'Margin'}, inplace=True)
```

In [90]:
```
profit100df['Study'] = index100['Study'].values
```

In [91]:
```
mode100 = choice100.mode(axis=1)
mode100.rename(columns={0: 'Most Common Choice'}, inplace=True)
profit100df['Most Common Choice'] = mode100['Most Common Choice'].values
```

In [92]:
```
profit100df['Most Common Choice'].value_counts()
```

Out[92]:
```
2.0    221
4.0    171
3.0     98
1.0     14
Name: Most Common Choice, dtype: int64
```

In [93]:
```
profit100df['Most Common Choice'] = profit100df['Most Common Choice'].astype('int64')
```

In [94]:
```
most_common_count100 = choice100.apply(pd.Series.value_counts, axis=1)
most_common_count100 = most_common_count100.max(axis=1)
profit100df['Most Common Choice Picked'] = most_common_count100
profit100df.head()
```

Out[94]:

	Margin	Study	Most Common Choice	Most Common Choice Picked
Subj_1	-1800	Horstmann	2	42
Subj_2	-800	Horstmann	2	35
Subj_3	-450	Horstmann	2	42
Subj_4	1200	Horstmann	4	35
Subj_5	-1300	Horstmann	2	31

In [95]:
```
mean100 = choice100.mean(axis=1)
mean100df = pd.DataFrame(data=mean100)
mean100df.rename(columns={0: 'Average Choice'}, inplace=True)
profit100df['Average Choice'] = mean100df['Average Choice'].values
```

In [96]:
```
profit100df.to_csv('Data/cleaned100.csv')
```

Lastly, we take the 150 trial data.

In [97]:
```
columnnames150 = [f'Trial{num}' for num in range(1,151)]
wins150 = win150
wins150 = wins150.set_axis(columnnames150, axis=1)
```

In [98]:
```
losses150 = loss150
losses150 = losses150.set_axis(columnnames150, axis=1)
```

In [99]:
```
df150_sum = wins150.add(losses150, fill_value=0)
```

In [100]:
```
profit150 = df150_sum.sum(axis=1)
profit150df = pd.DataFrame(data=profit150)
profit150df.rename(columns={0: 'Margin'}, inplace=True)
```

In [101]:
```
profit150df['Study'] = index150['Study'].values
```

In [102]:
```
mode150 = choice150.mode(axis=1)
mode150.rename(columns={0: 'Most Common Choice'}, inplace=True)
profit150df['Most Common Choice'] = mode150['Most Common Choice'].values
```

In [103]:
```
most_common_count150 = choice150.apply(pd.Series.value_counts, axis=1)
most_common_count150 = most_common_count150.max(axis=1)
most_common_count150 = most_common_count150.astype('int64')
profit150df['Most Common Choice Picked'] = most_common_count150
profit150df.head()
```

Out[103]:

	Margin	Study	Most Common Choice	Most Common Choice Picked
Subj_1	-550	Steingroever2011	1.0	46
Subj_2	-1600	Steingroever2011	2.0	57
Subj_3	900	Steingroever2011	4.0	88
Subj_4	2200	Steingroever2011	4.0	111
Subj_5	1900	Steingroever2011	4.0	135

In [104]:
```
mean150 = choice150.mean(axis=1)
mean150df = pd.DataFrame(data=mean150)
mean150df.rename(columns={0: 'Average Choice'}, inplace=True)
profit150df['Average Choice'] = mean150df['Average Choice'].values
```

In [105]:
```
profit150df.to_csv('Data/cleaned150.csv')
```

In [106]:
```
merged95_150 = pd.concat([profit95df, profit150df])
```

In [114]:
```
mergedall = pd.concat([merged95_150, profit100df])
mergedall['Most Common Choice'] = mergedall['Most Common Choice'].astype('int64')
mergedall
```

Out[114]:

	Margin	Most Common Choice Picked	Most Common Choice	Study	Average Choice
Subj_1	1150	71	4	Fridberg	3.400000
Subj_2	-675	33	4	Fridberg	2.568421
Subj_3	-750	38	4	Fridberg	2.778947
Subj_4	-525	38	4	Fridberg	2.810526
Subj_5	100	46	4	Fridberg	3.021053
...	...	...	...	...	...
Subj_500	75	29	2	Worthy	2.630000
Subj_501	600	44	3	Worthy	2.840000
Subj_502	-1525	32	2	Worthy	2.380000
Subj_503	-750	27	1	Worthy	2.460000
Subj_504	175	54	4	Worthy	3.100000

617 rows × 5 columns

For some of our comparisons we may want to draw on in our k-means clusters we need to change our study values from strings to integers. After plotting our k-means algorithm this will allow us to plot a scatter plot comprising of the different studies which will be colour coded based on their numbers here.

In [117]:
```
replacements_study = {
  r'Fridberg': 0,
  r'Horstmann': 1,
  r'Kjome': 2,
  r'Maia': 3,
  r'SteingroverInPrep': 4,
  r'Premkumar': 5,
  r'Wood': 6,
  r'Worthy': 7,
  r'Steingroever2011': 8,
  r'Wetzels': 9,
}

mergedall['StudyNumber'] = mergedall.Study.replace(replacements_study, regex=True)
mergedall = mergedall.drop(columns=['Study'])
mergedall
```

Out[117]:

	Margin	Most Common Choice Picked	Most Common Choice	Average Choice	StudyNumber
Subj_1	1150	71	4	3.400000	0
Subj_2	-675	33	4	2.568421	0
Subj_3	-750	38	4	2.778947	0
Subj_4	-525	38	4	2.810526	0
Subj_5	100	46	4	3.021053	0
...	...	...	...	...	...
Subj_500	75	29	2	2.630000	7
Subj_501	600	44	3	2.840000	7
Subj_502	-1525	32	2	2.380000	7
Subj_503	-750	27	1	2.460000	7
Subj_504	175	54	4	3.100000	7

617 rows × 5 columns

In [118]:
```
mergedall.to_csv('Data/cleaned_all.csv')
```

Standardize our Data

To work best with our k-means algorithm we choose to standardize our values in our joined dataset. This is because the k-means algorithm is a distance based algorithm, calculating the similarity between points based on distance. This gives the data a mean of 0 and standard deviation of 1 and gives common ground between features which would use different values such as our margin and average choice columns.

In [129]:
```
scaler = preprocessing.StandardScaler().fit(mergedall)
X_scaled = scaler.transform(mergedall)
X_scaled.std(axis=0)
```

Out[129]:
```
array([1., 1., 1., 1., 1.])
```

In [130]:
```
standard_all = pd.DataFrame(X_scaled)
```

In [131]:
```
standard_all = standard_all.rename(columns={0:'Margin', 1: 'Most Common Choice Picked', 2: 'Most Common Choice', 3: 'Average Choice',
                                            4: 'StudyNumber'})
```

In [132]:
```
standard_all
```

Out[132]:

	Margin	Most Common Choice Picked	Most Common Choice	Average Choice	StudyNumber
0	1.044988	1.062672	1.234405	2.271920	-1.609380
1	-0.414346	-0.804770	1.234405	-0.410317	-1.609380
2	-0.474318	-0.559054	1.234405	0.268730	-1.609380
3	-0.294400	-0.559054	1.234405	0.370587	-1.609380
4	0.205371	-0.165908	1.234405	1.049635	-1.609380
...	...	...	...	...	...
612	0.185381	-1.001343	-0.923181	-0.211696	0.952696
613	0.605189	-0.264195	0.155612	0.465654	0.952696
614	-1.094035	-0.853913	-0.923181	-1.018065	0.952696
615	-0.474318	-1.099629	-2.001975	-0.760027	0.952696
616	0.265344	0.227238	1.234405	1.304277	0.952696

617 rows × 5 columns

In [134]:
```
standard_all.to_csv('data/standardized_all.csv')
```

3. Clustering

After our analysis of the data we must now cluster the data accordingly

In [1]:
```
import pandas as pd
import seaborn as sn
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans, AgglomerativeClustering
from sklearn.metrics import silhouette_score
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

First, we read in our data

In [2]:
```
index95 = pd.read_csv('data/index_95.csv')
index100 = pd.read_csv('data/index_100.csv')
index150 = pd.read_csv('data/index_150.csv')
win95 = pd.read_csv('data/wi_95.csv')
win100 = pd.read_csv('data/wi_100.csv')
win150 = pd.read_csv('data/wi_150.csv')
loss95 = pd.read_csv('data/lo_95.csv')
loss100 = pd.read_csv('data/lo_100.csv')
loss150 = pd.read_csv('data/lo_150.csv')
choice95 = pd.read_csv('data/choice_95.csv')
choice100 = pd.read_csv('data/choice_100.csv')
choice150 = pd.read_csv('data/choice_150.csv')
```

Cleaned data from processing

In [3]:
```
cleaned95 = pd.read_csv('data/cleaned95.csv', index_col='Unnamed: 0')
cleaned100 = pd.read_csv('data/cleaned100.csv', index_col='Unnamed: 0')
cleaned150 = pd.read_csv('data/cleaned150.csv', index_col='Unnamed: 0')
```

Initially, I decided to cluster based on the profit/loss margin for each subject and their most common choice. However, the most common choice would limit the clusters greatly I felt. There would only be 4 possible values (1,2,3 or 4) and this would limit what we could learn from the approriate cluster analysis. I looked into clustering on the number of times each deck was selected but this would involve multiple clusters, one for each choice against the profit margin but I decided against it. This lead me to going back to my data processing and creating the average choice column to add to my data. This was the sum of all the subjects selection divided by the number of trials and I felt this would provide better cluster analysis as a result as there would be far more variety in the range of values. I also decided to look into how many times subjects picked their most common choice. I felt this would give us a good overview of the respective studies and how they played the game, did they play safe and stick to what they know would win or would they attempt riskier decks in the hope of winning more? We are going to use our standardized data from our data processing as standardized data tends to work better with the k-means algorithm.

K-Means analysis

Finding our value for k

Silhouette Scores for our dataset

We now use another metric to test for the optimal number of clusters in our datasets. We try the silhouette coefficient which calculates the robustness of a clustering technique. This metric measures the degree of seperation between clusters. The score scales from -1 to 1. 1 means the clusters are very distinguished and perfectly easy to identify, 0 means the clusters are indifferent or hard to identify and -1 means the clusters are assigned in the wrong way. We will try to use this with our earlier elbow coefficient to confirm the optimal number of clusters for our datasets. The formula for the silhouette score is defined as follows:

$$silhouette - score = \frac{b_i - a_i}{max(bi, a_i)}$$

In the formula above from here bi represents the shortest mean distance between a point to all points in any other cluster of which i is not a part whereas ai is the mean distance of i and all data points from the same cluster.

In [4]:
```
standard = pd.read_csv('data/standardized_all.csv', index_col='Unnamed: 0')
standard.head()
```

Out[4]:

	Margin	Most Common Choice Picked	Most Common Choice	Average Choice	StudyNumber
0	1.044988	1.062672	1.234405	2.271920	-1.60938
1	-0.414346	-0.804770	1.234405	-0.410317	-1.60938
2	-0.474318	-0.559054	1.234405	0.268730	-1.60938
3	-0.294400	-0.559054	1.234405	0.370587	-1.60938
4	0.205371	-0.165908	1.234405	1.049635	-1.60938

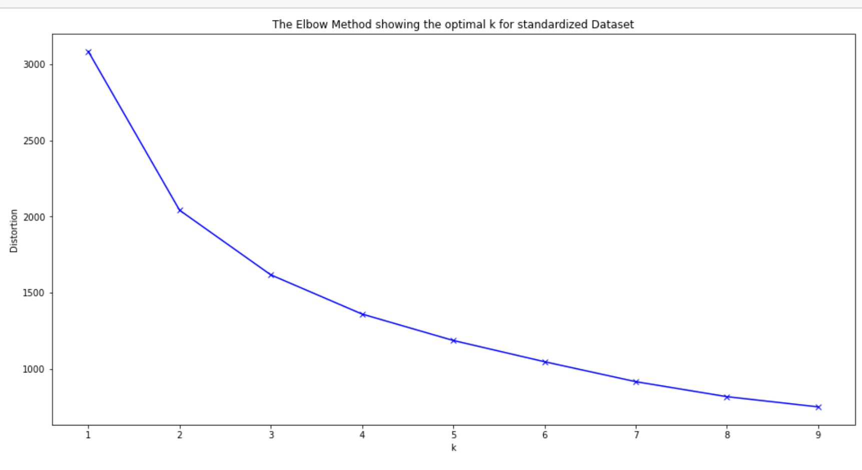
In [5]:
```
for n in range(2, 11):
    km = KMeans(n_clusters=n)
#
# Fit the KMeans model
# Have to pick subset of columns as Study column is in string format
    km.fit_predict(standard)
#
# Calculate Silhoutte Score
#
    score = silhouette_score(standard, km.labels_, metric='euclidean')
#
# Print the score
#
    print('N = ' + str(n) + ' Silhouette Score: %.3f' % score)
```

```
N = 2 Silhouette Score: 0.313
N = 3 Silhouette Score: 0.289
N = 4 Silhouette Score: 0.267
N = 5 Silhouette Score: 0.273
N = 6 Silhouette Score: 0.277
N = 7 Silhouette Score: 0.290
N = 8 Silhouette Score: 0.299
N = 9 Silhouette Score: 0.302
N = 10 Silhouette Score: 0.294
```

Elbow Method

In [6]:
```
distortions_joined_st = []
K = range(1, 10)
for k in K:
    kmeanModel = KMeans(n_clusters=k)
    kmeanModel.fit(standard)
    distortions_joined_st.append(kmeanModel.inertia_)
```

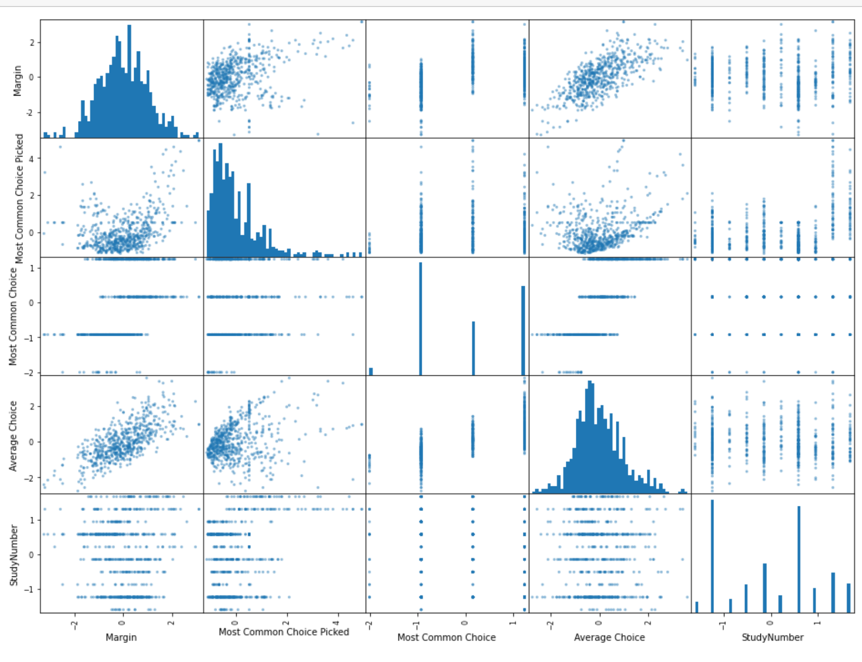
In [7]:
```
plt.figure(figsize=(16,8))
plt.plot(K, distortions_joined_st, 'bx-')
plt.xlabel('k')
plt.ylabel('Distortion')
plt.title('The Elbow Method showing the optimal k for standardized Dataset')
plt.show()
```



Looking at our elbow method and silhouette scores for the dataset as whole we can conclude using k=2 or k=3 is a safe value to use for the amount of clusters in our data.

Now we look at what we will cluster on in our data

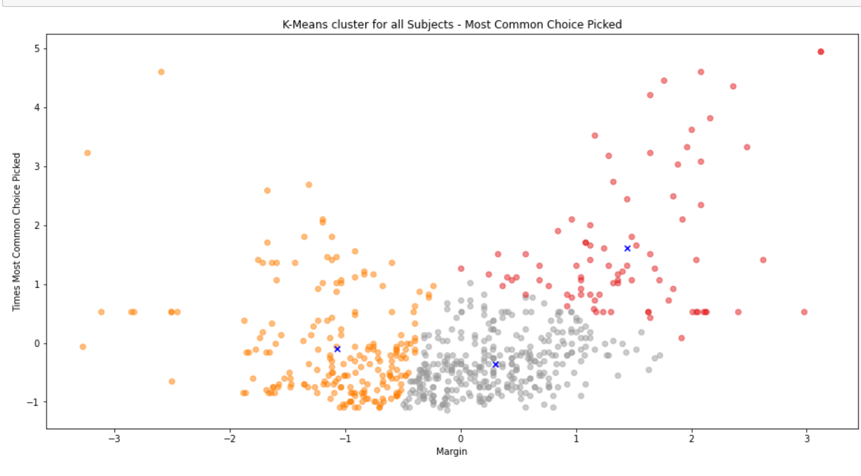
In [9]:
```
pd.plotting.scatter_matrix(standard, figsize=(16,12), hist_kwds=dict(bins=50), cmap="Set1")
plt.show()
```



After looking at this scatterplot comparing the various columns against each other in a scatter plot I felt it might be worth looking into how regularly a participant picked their most common choice and the profit margins. We could further look into this by looking at what was their most common choice and the respective study they belonged to.

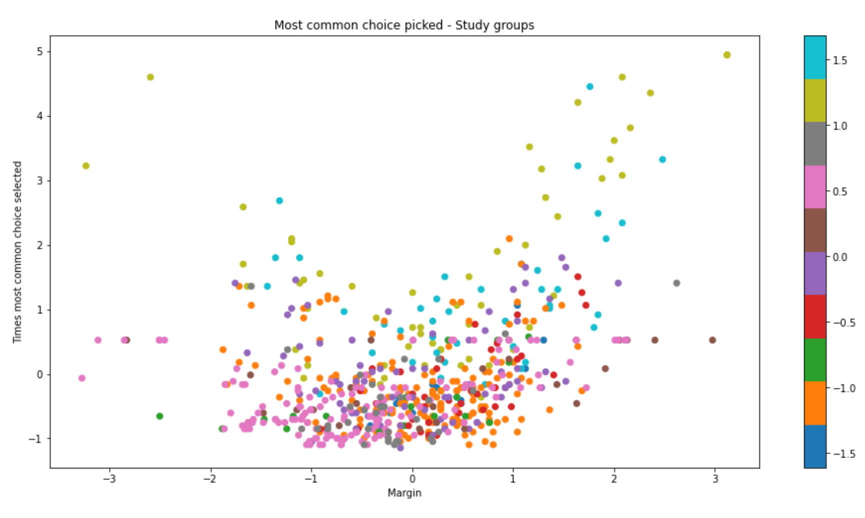
In [11]:
```
kmeans_margin_standard = KMeans(n_clusters=3).fit(standard[["Margin", "Most Common Choice Picked"]])
centroids_betas_standard = kmeans_margin_standard.cluster_centers_
```

In [12]:
```
plt.figure(figsize=(16,8))
plt.scatter(standard['Margin'], standard['Most Common Choice Picked'], c= kmeans_margin_standard.labels
_, cmap = "Set1", alpha=0.5)
plt.scatter(centroids_betas_standard[:, 0], centroids_betas_standard[:, 1], c='blue', marker='x')
plt.title('K-Means cluster for all Subjects - Most Common Choice Picked')
plt.xlabel('Margin')
plt.ylabel('Times Most Common Choice Picked')
plt.show()
```



In [13]:
```
plt.figure(figsize=(16,8))
plt.scatter(standard['Margin'], standard['Most Common Choice Picked'], c=standard['StudyNumber'], cmap=
'tab10')
plt.title("Most common choice picked - Study groups")
plt.xlabel('Margin')
plt.ylabel('Times most common choice selected')
plt.colorbar()
```

Out[13]: <matplotlib.colorbar.Colorbar at 0x20fd1c423a0>



From earlier our studies are as follows: Fridberg: 0, Horstmann: 1, Kjome: 2, Maia: 3, SteingroverInPrep: 4, Premkumar: 5, Wood: 6, Worthy: 7, Steingroever2011: 8 and Wetzels: 9. Immediately we notice our neon scatters here on both the left and right side of the plot. There is a large amount of subjects from Steingroever2011's study that regularly pick their favoured outcome, a lot of these can be seen to the right of the plot in the more profitable outcome while picking this selection 100 trials or more in the 150 trial experiment. The majority of this study picks their most common choice at least 80 times or more from our plot also. We can see a couple of outliers in this study also, picking their favoured outcome well over a hundred times with poor results. Again, this study is based on the youngest specifically mentioned mean of subjects (19.9 years old) and it certainly seems to play a part in their decision making. We can see from our scatterplot something similar with the Wetzels study, a student based study with a noticeable group of these subjects picking their favoured choice 80 times or more and winning money over the course of their trials. It is also interesting to note the clusters to the left and centre from our K-means plot containing a sizable proportion of subjects from the Wood study (in pink). This is a particularly interesting study as the first 90 participants were between the ages of 18-40 and the rest had a mean age of 76.98 years old. Despite it being a 100 trial study the vast majority of this study pick their preferred choice 40-50 times, less than or equal to half of their trials. A lot of these participants also lose money over the course of their trials which raises interesting questions about how age can impair decision making. It is interesting to note when researching the Wood et al study that it was regarded as an outlier in that age over time impairs performance in the IGT {cite:p} `beitziowa`. However, in general it is seen as something that can negatively impact decision making over time. It also suggests in {cite:p} `beitziowa` in later adulthood that a low loss strategy is seen more predominately than the end profit. The Horstmann study follows a similar dispersion to the Woods study with a large amount of subjects centred around the middle cluster and also picking their preffered choice 40-50 out of 100 trials which could be seen as a relatively low numbers and maybe hints at an adaptive approach to the game, where after a period of maybe settling for preferred decks they adapt as their wins/losses dictate. With a similar number of participants in this trial to the Woods trial the one key difference is far more of these subjects lean to the right cluster (breaking even or making a marginal gain). Again, looking at the age demography this is a young adult group with a mean age of 25.6 years old. Something suggested in {cite:p} `beitziowa` could hold true here in that adults past adolescence prefer experimental decision making instead of just frequency preference. This could also explain the cohort of the Horstmann study who followed a similar decision making process (40-50 most common choice) but lost money in the end. Another study which also has an interesting cluster is the Worthy study, which has 35 subjects the majority of which (22) is female. This study contains a lot of subjects who picked their most common choice less than 40 times and very few subjects are above 50 or so. It is something that ties in with some of the potential behavioural findings of {cite:p} `denbosgender` in that women are more sensitive to losses in the long term profitable decks. This would explain the constant tinkering between the decks and the lower most common choice value among many subjects in this study. The aforementioned Horstmann study also has a large number of female participants in it (82/162) and as mentioned previously has similarly lower count of the times the most common choice was picked. It is something mentioned in {cite:p} `denbosgender2` that women pick more disadvantageous decks as they mix a policy of exploration and exploitation, whereas as men will after initial exploration focus on exploitation then. This could definitely explain why these studies with a lower common choice pick and specified amount of females taking part in the study have a more mixed approach to the task and why they float around breaking even. They obviously accept some losses in exploration but exploit the winning cards regularly enough to be around even. We see further information from {cite:p} `healthiowa` in gender decision making from the task. Similar to our data there is 40 healthy males and females in this article and it suggests females are more sensitive to losses than males. This could explain the more varied approach females seem to take in our k-means plot.

In [16]:
```
joined = pd.read_csv('data/cleaned_all.csv', index_col='Unnamed: 0')
joined['cluster'] = kmeans_margin_standard.labels_.tolist()
```

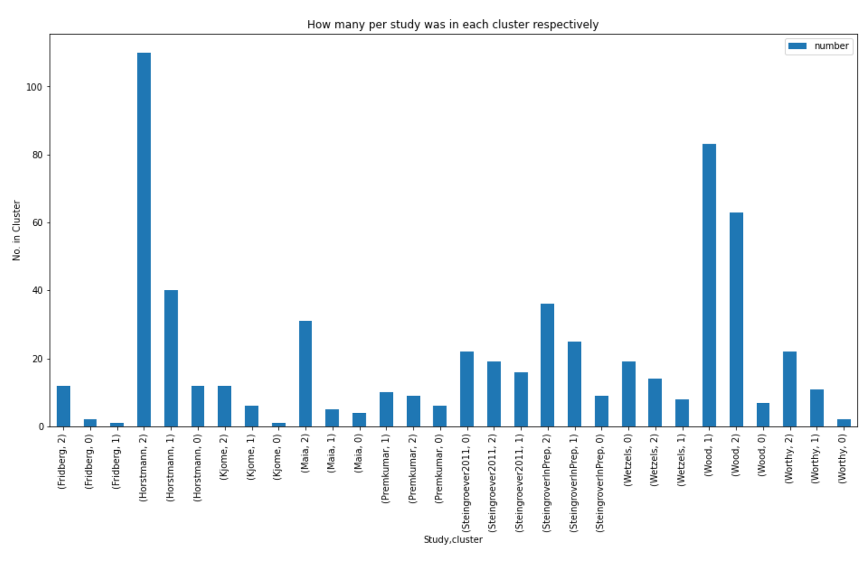
In [18]:
```
heatmap = joined[['StudyNumber', 'cluster']]
replacements = {
  0: r'Fridberg',
  1: r'Horstmann',
  2: r'Kjome',
  3: r'Maia',
  4: r'SteingroverInPrep',
  5: r'Premkumar',
  6: r'Wood',
  7: r'Worthy',
  8: r'Steingroever2011',
  9: r'Wetzels',
}

heatmap['Study'] = heatmap.StudyNumber.replace(replacements, regex=True)
heatmap = heatmap.drop(columns=['StudyNumber'])
```

```
<ipython-input-18-35f83923f008>:15: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexin
g.html#returning-a-view-versus-a-copy
  heatmap['Study'] = heatmap.StudyNumber.replace(replacements, regex=True)
```

In [19]:
```
counts = heatmap.groupby('Study')['cluster'].value_counts()
```

In [72]:
```
histdf = pd.DataFrame(counts)
histdf = histdf.rename(columns={'cluster': 'number'})
plt = histdf.plot.bar(figsize=(16, 8), ylabel='No. in Cluster', title='How many per study was in each c
luster respectively')
plt
```

Out[72]: <AxesSubplot:title={'center':'How many per study was in each cluster respectively'}, xlabel='Study,cl
uster', ylabel='No. in Cluster'>



In [21]:
```
centroids_betas_standard
```

Out[21]:
```
array([[ 1.443018  ,  1.61018515],
       [-1.07289372, -0.09542955],
       [ 0.29751279, -0.35872379]])
```

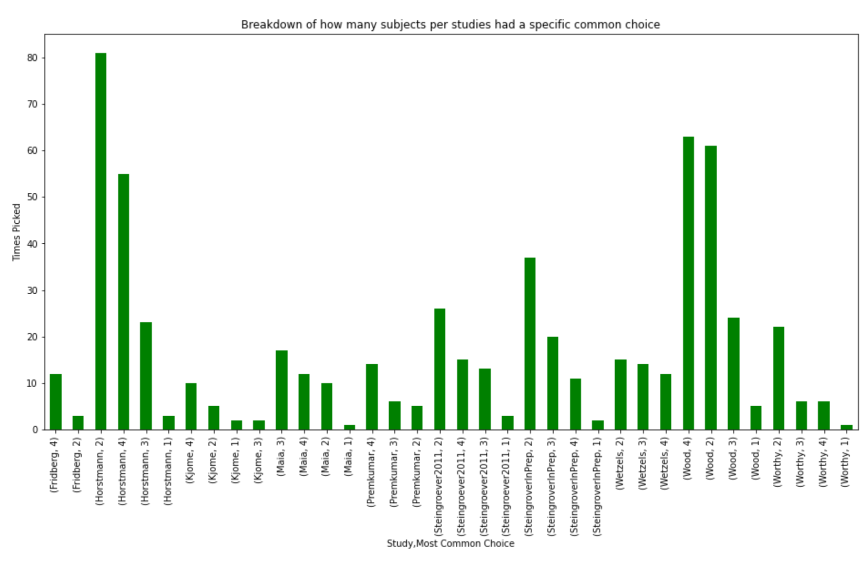
We see our cluster centres above, the left most cluster is denoted as 1, the right most cluster denoted as 0 and the central cluster as 2.

Using our histogram above it confirms some of our previous statements from earlier. We do indeed see a large majority of Wood study subjects in the less profitable and more varied choice selection cluster to the left of our k-means scatter plot. We also see a large number of subjects from the Horstmann study in the central cluster which from our earlier analysis of profitable studies adds up also. We also see can confirm how unprofitable the Wood study was with very few subjects in the right most cluster.

In [71]:
```
commonchoice = joined[['Most Common Choice', 'StudyNumber']]
replacements = {
  0: r'Fridberg',
  1: r'Horstmann',
  2: r'Kjome',
  3: r'Maia',
  4: r'SteingroverInPrep',
  5: r'Premkumar',
  6: r'Wood',
  7: r'Worthy',
  8: r'Steingroever2011',
  9: r'Wetzels',
}

commonchoice['Study'] = commonchoice.StudyNumber.replace(replacements, regex=True)
commonchoice = commonchoice.drop(columns=['StudyNumber'])
common_counts = commonchoice.groupby('Study')['Most Common Choice'].value_counts()
plt2 = common_counts.plot.bar(figsize=(16, 8), color='green', ylabel='Times Picked', title='Breakdown o
f how many subjects per studies had a specific common choice')
plt2
```

```
<ipython-input-71-4fbfd07259c0>:15: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexin
g.html#returning-a-view-versus-a-copy
  commonchoice['Study'] = commonchoice.StudyNumber.replace(replacements, regex=True)
```

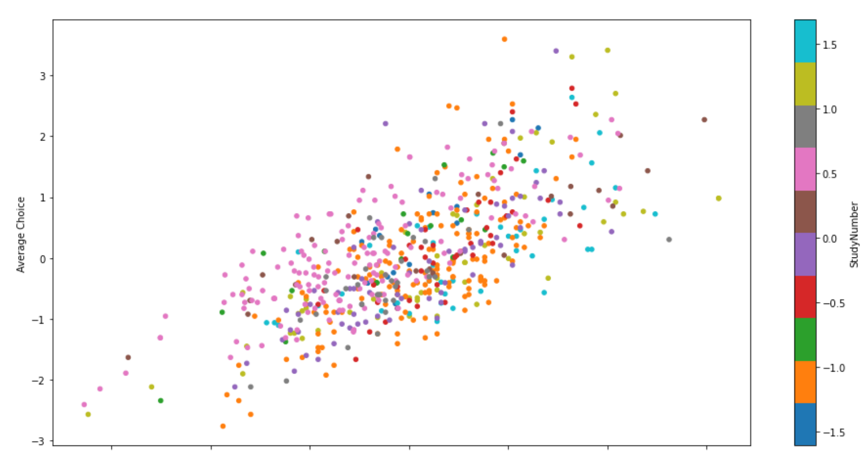
Out[71]: <AxesSubplot:title={'center':'Breakdown of how many subjects per studies had a specific common choic
e'}, xlabel='Study,Most Common Choice', ylabel='Times Picked'>



To further add on to our observations from our k-means analysis for most common choice picked and margin we look at the breakdown of each study and how many subjects favoured a specific deck. This graph here adds to the theory of what we discussed earlier with regards how subjects played the game with exploitation versus exploration. We mentioned earlier how females appeared to prefer a policy of exploration over end results. The studies we mentioned then which were the Horstmann and Wood studies and have large numbers which preffered deck 2 which was one of the least favourable decks. They also had large numbers picking deck 4 which could suggest this is the more exploitative section of subjects in the studies. We also see looking at our earlier age studies that two of the seemingly younger studies SteingroeverInPrep and Wetzels follow very consistent decision making patterns in each of the most common choices. This is something we alluded to earlier in that younger adults tend to follow what they know. It is also interesting to note in the SteingroeverInPrep and the Worthy study that have large specified numbers of female participants the variety of spread of subjects across the most common choices. They both have large numbers picking deck 2 and reasonably equal spread across decks 3 and 4 too. This definitely follows on from our previous findings of potential differences in task approaches along gender lines.

In [64]:
```
standard.plot.scatter(x='Margin', y='Average Choice', c='StudyNumber', cmap='tab10', figsize=(16, 8))
```

Out[64]: <AxesSubplot:xlabel='Margin', ylabel='Average Choice'>



We will look at the margin vs average choice plot and try to combine this with our earlier graphs too to further our knowledge on subjects decison making. Using our colour bar we can deduce the clusters from what study they are a part of. If we look at the cluster to the left in our k-means scatter plot we can see that a substantial amount of this cluster contains subjects from the Wood et al study. It is interesting to note this had the highest mean average age of any study in the datasets. It also had a large number of participants but looking at the scatter plot very few participants made money over the course of the trials. The majority had an average choice of below 3 and certainly fell into the category of average lower choice and lower financial gain. This study also features heavily in the second cluster (the one most central) and this cluster also contains subjects who struggled to break even. The Horstmann study also features heavily in this cluster as does the Worthy study in yellow. The Worthy study leans more towards the first cluster again in the lower choice average, lower money made category. It is interesting to note that this study does not explicitly state the age demography of the group studied but tells us it was a solely female, undergraduate student study, which hints at it being a younger age group. In the third cluster to the right, which is the higher average choice, higher profit group we can see a large mix of groups with comparatively less subjects in this cluster compared to the other two. We can see a significant amount of subjects from the Maia study and also quite a few from the previously mentioned Horstmann study. We also see even with a small sample size from the study there is a significant number of Premkumar participants in this profitable cluster. Two of these studies contain a very young mean age again. The Maia study is another that focuses on undergraduate students again, but with better results than previous.

4. K-Means Variation

In [2]:
```
import pandas as pd
import seaborn as sn
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans, AgglomerativeClustering
from sklearn.metrics import silhouette_score
```

Import our data

In [3]:
```
cleaned95 = pd.read_csv('data/cleaned95.csv', index_col='Unnamed: 0')
cleaned100 = pd.read_csv('data/cleaned100.csv', index_col='Unnamed: 0')
cleaned150 = pd.read_csv('data/cleaned150.csv', index_col='Unnamed: 0')
joined = pd.read_csv('data/cleaned_all.csv', index_col='Unnamed: 0')
standard = pd.read_csv('data/standardized_all.csv', index_col='Unnamed: 0')
```

In [4]:
```
standard.head()
```

Out[4]:

	Margin	Most Common Choice Picked	Most Common Choice	Average Choice	StudyNumber
0	1.044988	1.062672	1.234405	2.271920	-1.60938
1	-0.414346	-0.804770	1.234405	-0.410317	-1.60938
2	-0.474318	-0.559054	1.234405	0.268730	-1.60938
3	-0.294400	-0.559054	1.234405	0.370587	-1.60938
4	0.205371	-0.165908	1.234405	1.049635	-1.60938

Methodology

We are going to attempt to follow the methods stated in {cite:p}`LinPP` in his attempt at constructing a privacy preserving clustering technique based on the k-means algorithm. This involves a 2 step process which is as follows:

1. Data Protection Phase and
2. Data Recovery Phase

The first phase involving the data protection phase involves 4 key steps. Firstly, we apply the K-means algorithm on our dataset and then we select one of the clusters from the result. In our cluster let's say A, we select the furthest data away from the centroid of A. We generate the set of noises by the using the following equation:

$$n_i^u = d^u + \alpha \times (distance(c, d)) \tag{1}$$

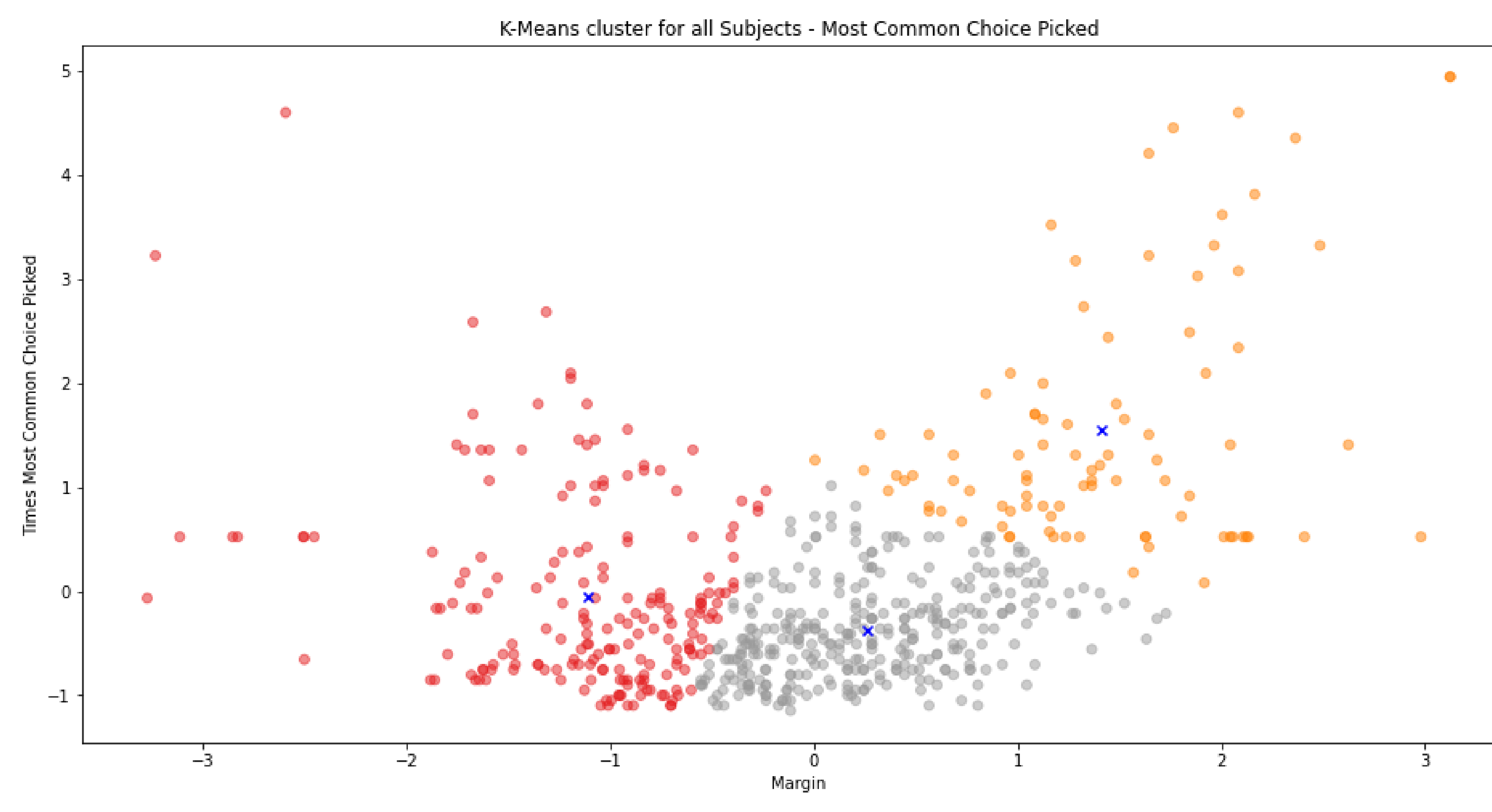
We then use the following equation:

$$p_i = |D| \times Rand(s) \tag{2}$$

This is to obtain the position of the noise from eq(1) in dataset D. This leads us on to our data recovery phase. Our first step in the phase is to use eq(2) and obtain p_i for position of noise in D' and commence removals. Then we delete all the noises and the original dataset D can be recovered immediately. The end result should be a dataset that shares cluster information but protects the privacy of the individuals at hand.

In [5]:
```
kmeans_margin_joined = KMeans(n_clusters=3).fit(standard[["Margin", "Most Common Choice Picked"]])
centroids_betas_joined = kmeans_margin_joined.cluster_centers_
```

In [6]:
```
plt.figure(figsize=(16,8))
plt.scatter(standard['Margin'], standard['Most Common Choice Picked'], c= kmeans_margin_joined.labels_,
 cmap = "Set1", alpha=0.5)
plt.scatter(centroids_betas_joined[:, 0], centroids_betas_joined[:, 1], c='blue', marker='x')
plt.title('K-Means cluster for all Subjects - Most Common Choice Picked')
plt.xlabel('Margin')
plt.ylabel('Times Most Common Choice Picked')
plt.show()
```



In [7]:
```
centroids_betas_joined
```

Out[7]:
```
array([[-1.10872288, -0.06074673],
       [ 1.41597414,  1.55028183],
       [ 0.25911929, -0.38263846]])
```

We can tell from our above cluster centres that cluster 0 is in red, cluster 1 is the right most cluster and cluster 2 is in grey.

In [8]:
```
standard['cluster'] = kmeans_margin_joined.labels_.tolist()
standard.head()
```

Out[8]:

	Margin	Most Common Choice Picked	Most Common Choice	Average Choice	StudyNumber	cluster
0	1.044988	1.062672	1.234405	2.271920	-1.60938	1
1	-0.414346	-0.804770	1.234405	-0.410317	-1.60938	2
2	-0.474318	-0.559054	1.234405	0.268730	-1.60938	2
3	-0.294400	-0.559054	1.234405	0.370587	-1.60938	2
4	0.205371	-0.165908	1.234405	1.049635	-1.60938	2

In [9]:
```
cluster0 = standard[standard.cluster==0]
cluster0.head()
```

Out[9]:

	Margin	Most Common Choice Picked	Most Common Choice	Average Choice	StudyNumber	cluster
16	-1.154008	0.374667	-0.923181	-1.233097	1.318707	0
20	-1.074044	-0.067622	-2.001975	-1.319109	1.318707	0
26	-0.594263	1.357531	-0.923181	-0.889046	1.318707	0
28	-1.673771	2.586111	-0.923181	-1.899695	1.318707	0
37	-1.193990	2.094679	-0.923181	-0.631008	1.318707	0

From looking at our graph and cluster 0, I feel data points with margin values less than -2 would be classified as noise.

In [10]:
```
noises = cluster0[cluster0.Margin <= -2]
noises
```

Out[10]:

	Margin	Most Common Choice Picked	Most Common Choice	Average Choice	StudyNumber	cluster
43	-2.593351	4.600983	-0.923181	-2.114727	1.318707	0
49	-3.233059	3.224973	-0.923181	-2.566293	1.318707	0
287	-2.501393	-0.657340	-2.001975	-2.340510	-0.877358	0
404	-2.829243	0.522097	-0.923181	-1.630905	0.220674	0
450	-3.273040	-0.067622	-0.923181	-2.405019	0.586685	0
501	-2.853232	0.522097	-0.923181	-1.888943	0.586685	0
531	-2.453415	0.522097	-0.923181	-0.953555	0.586685	0
547	-3.113113	0.522097	-0.923181	-2.146981	0.586685	0
572	-2.505391	0.522097	-0.923181	-1.308358	0.586685	0
573	-2.505391	0.522097	-0.923181	-1.308358	0.586685	0

In [12]:
```
marginnoise = pd.DataFrame(np.random.uniform(-3.4,-2.5,15))
```

In [13]:
```
marginnoise = marginnoise.rename(columns={0: 'Margin'})
marginnoise.head()
```

Out[13]:

	Margin
0	-2.542259
1	-3.255683
2	-3.016242
3	-2.921898
4	-3.195019

In [14]:
```
choicenoise = pd.DataFrame(np.random.uniform(-0.1,4,15))
```

Out[14]:

	0
0	2.220171
1	2.045314
2	3.831233
3	3.094479
4	2.395261

In [15]:
```
choicenoise = choicenoise.rename(columns={0: 'Most Common Choice Picked'})
choicenoise.head()
```

Out[15]:

	Most Common Choice Picked
0	2.220171
1	2.045314
2	3.831233
3	3.094479
4	2.395261

In [16]:
```
noise = pd.concat([marginnoise, choicenoise], axis=1)
```

Our noise data is now generated we add this back to the original dataset now.

In [39]:
```
noisesf = noises[['Margin', 'Most Common Choice Picked']]
noisesf = noisesf.to_numpy()
noisesf
```

Out[39]:
```
array([[-2.59335066,  4.6009828 ],
       [-3.23305855,  3.22497306],
       [-2.50139265, -0.65734015],
       [-2.82924294,  0.52209677],
       [-3.2730403 , -0.06762169],
       [-2.85323199,  0.52209677],
       [-2.45341456,  0.52209677],
       [-3.11311332,  0.52209677],
       [-2.50539082,  0.52209677],
       [-2.50539082,  0.52209677]])
```

In [43]:
```
centroids_betas_joined[0]
```

Out[43]:
```
array([-1.10872288, -0.06074673])
```

In [46]:
```
from scipy.spatial.distance import cdist
from scipy.spatial import distance

distances = distance.cdist(centroids_betas_joined, noisesf, 'euclidean')
distances[0]
```

Out[46]:
```
array([4.89242699, 3.91264062, 1.51507518, 1.81656155, 2.16432834,
       1.8392984 , 1.46557233, 2.0874117 , 1.51340276, 1.51340276])
```

Our furtherest point away from the centroid of our choosen cluster 0 is the first point we see in the array here. This will be denoted as our data "d" the furtherest point from our centroid, C of cluster 0.

We now need to use this distance value obtained and combine it with our "noise offset ratio" denoted as α. From our previousy cited paper this value is typically in the range 0 - 0.05. We will use 0.05 for α here.

In [54]:
```
d = distances[0][0]
alpha = 0.05
```

In [55]:
```
d * alpha
```

Out[55]:
```
0.2446213497147065
```

We will now add our noises to dataset we generated above. This is the last step as part of our data protection phase.

In [58]:
```
clusterframes = standard[['Margin', 'Most Common Choice Picked']]
```

In [59]:
```
clusterframes = clusterframes.append(noise)
clusterframes
```

Out[59]:

	Margin	Most Common Choice Picked
0	1.044988	1.062672
1	-0.414346	-0.804770
2	-0.474318	-0.559054
3	-0.294400	-0.559054
4	0.205371	-0.165908
...	...	...
10	-3.122292	3.384886
11	-3.313970	0.733010
12	-2.732256	1.588493
13	-2.889044	3.483566
14	-2.705425	2.669901

632 rows × 2 columns

We now need to randomly shuffle our dataframe with noises added so it is harder to identify our noise.

In [60]:
```
sample = clusterframes.sample(frac=1).reset_index(drop=True)
sample
```

Out[60]:

	Margin	Most Common Choice Picked
0	0.613185	0.522097
1	-0.462324	-0.018478
2	-0.354373	-0.706483
3	-0.914117	-0.313338
4	-1.321931	-0.755627
...	...	...
627	0.245353	-0.952199
628	-0.194446	0.079808
629	0.045444	-0.411624
630	-3.313970	0.733010
631	-0.698216	-0.313338

632 rows × 2 columns

With our dataframe randomly assigned we now begin our data recovery phase.

In [ ]:
```
```