1. Intro

Cookie Cats is a mobile puzzle game by Tactile Entertainmen. It's a "connect three" puzzle game where the player must connect tiles of the same color to clear the board and win the level. Check out this short demo: https://youtu.be/GaP5f0jVTWE

As players progress through the levels of the game, they will occasionally encounter gates that force them to wait a non-trivial amount of time or make an in-app purchase to progress. In addition to driving in-app purchases, these gates serve the important purpose of giving players an enforced break from playing the game, hopefully resulting in that the player's enjoyment of the game being increased and prolonged.

But where should the gates be placed? Initially the first gate was placed at level 30, but in this notebook we're going to analyze an AB-test where we moved the first gate in Cookie Cats from level 30 to level 40. In particular, we will look at the impact on player retention. But before we get to that, a key step before undertaking any analysis is understanding the data.

In [1]:
```
# Importing pandas
import pandas as pd

# Reading in the data
df = pd.read_csv('datasets/cookie_cats.csv')

# Showing the first few rows
df.head()
```

Out[1]:

	userid	version	sum_gamerounds	retention_1	retention_7
0	116	gate_30	3	False	False
1	337	gate_30	38	True	False
2	377	gate_40	165	True	False
3	483	gate_40	1	False	False
4	488	gate_40	179	True	True

2. The AB-test data

The data we have is from over 90,000 players that installed the game while the AB-test was running. The variables are:

- `userid` - a unique number that identifies each player.
- `version` - whether the player was put in the control group (`gate_30` - a gate at level 30) or the group with the moved gate (`gate_40` - a gate at level 40).
- `sum_gamerounds` - the number of game rounds played by the player during the first 14 days after install.
- `retention_1` - did the player come back and play **1 day** after installing?
- `retention_7` - did the player come back and play **7 days** after installing?

When a player installed the game, they were randomly assigned to either `gate_30` or `gate_40` . As a sanity check, check the number of players in each AB group.

In [2]:
```
# Counting the number of players in each AB group.
df.groupby('version')['userid'].count()
```

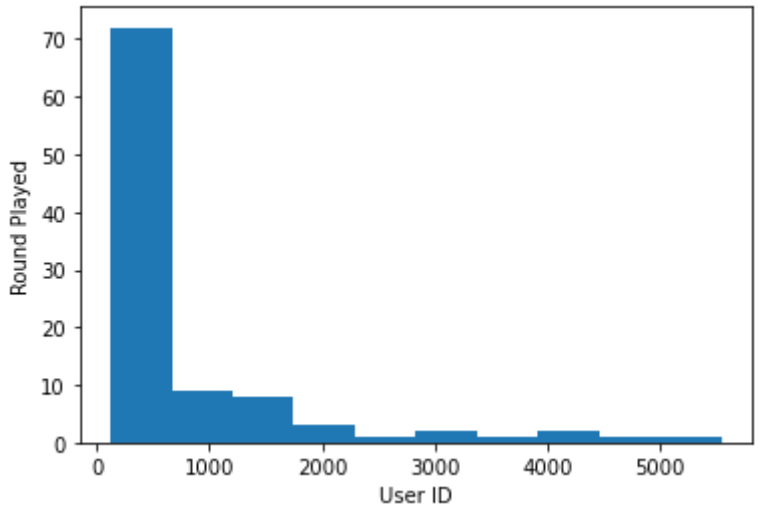
Out[2]:
```
version
gate_30    44700
gate_40    45489
Name: userid, dtype: int64
```

3. The distribution of game rounds

There is roughly the same number of players in each group. The focus of this analysis will be on how the gate placement affects player retention, just to be sure, Plot the distribution of the number of game rounds players played during their first week playing the game.

In [3]:
```
# This command makes plots appear in the notebook
%matplotlib inline

# Counting the number of players for each number of gamerounds
plot_df = df.groupby('sum_gamerounds')['userid'].count()

# Plotting the distribution of players that played 0 to 100 game rounds
ax = plot_df.head(100).plot(x='sum_gamerounds', y='userid', kind='hist')
ax.set_xlabel("User ID")
ax.set_ylabel("Round Played")
```

Out[3]:
```
Text(0, 0.5, 'Round Played')
```



4. Overall 1-day retention

In the plot above we can see that some players install the game but then never play it (0 game rounds), some players just play a couple of game rounds in their first week, and some get really hooked!

What we want is for players to like the game and to get hooked. A common metric in the games industry for how fun and engaging a game is 1-day retention: The percentage of players that

comes back and plays the game one day after they have installed it. The higher 1-day retention is, the easier it is to retain players and build a large player base.

As a first step, let's look at what 1-day retention is overall. Step 1 take a look at overall day-1 retention.

In [4]:
```
# The % of users that came back the day after they installed
df.retention_1.mean()
```

Out[4]:
```
0.4452095044850259
```

5. 1-day retention by AB-group

A little less than half (44.5%) of the players are playing again one day after installing the game. Now that we have a benchmark, let's look at how 1-day retention differs between the two AB-groups.

In [5]:
```
# Calculating 1-day retention for each AB-group
df.groupby('version')['retention_1'].mean()
```

Out[5]:
```
version
gate_30    0.448188
gate_40    0.442283
Name: retention_1, dtype: float64
```

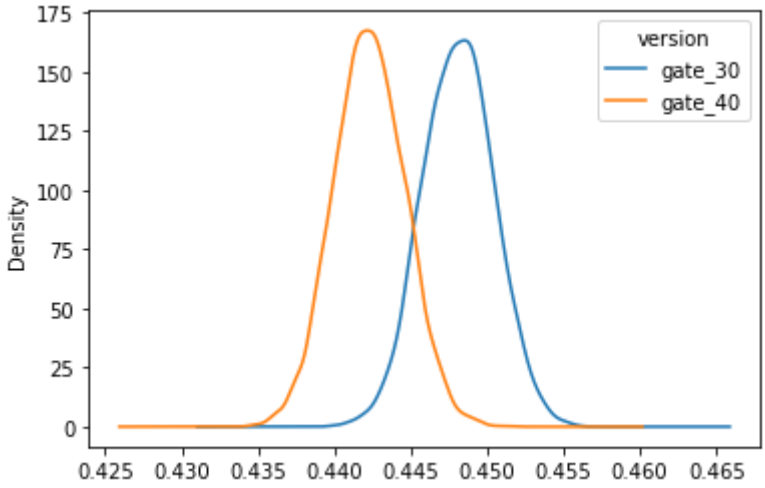
6. Should we be confident in the difference?

It appears that there was a slight decrease in 1-day retention when the gate was moved to level 40 (44.2%) compared to the control when it was at level 30 (44.8%). Small change, but even small changes in retention can have a substantial impact. But while we are certain of the difference in the data, how certain should we be that a gate at level 40 will be worse in the future?

There are a couple of ways we can get at the certainty of these retention numbers. Here, we will use bootstrapping: We will repeatedly re-sample our dataset (with replacement) and calculate 1-day retention for those samples. The variation in 1-day retention will give us an indication of how uncertain the retention numbers are.

In [6]:
```
# Creating an list with bootstrapped means for each AB-group
boot_1d = []
for i in range(10000):
    boot_mean = df.sample(frac=1, replace=True).groupby('version')['retention_1'].me
    boot_1d.append(boot_mean)

# Transforming the list to a DataFrame
boot_1d = pd.DataFrame(boot_1d)

# A Kernel Density Estimate plot of the bootstrap distributions
boot_1d.plot(kind='kde')
```

Out[6]:
```
<AxesSubplot:ylabel='Density'>
```

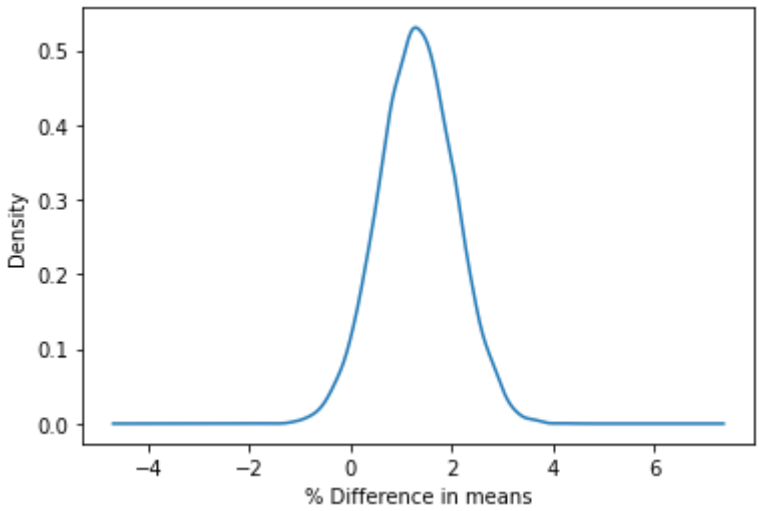


7. Zooming in on the difference

Two distributions above represent the bootstrap uncertainty over what the underlying 1-day retention could be for the two AB-groups. Just by looking at the plot, we can see that there seems to be some evidence of a difference, albeit small. Let's zoom on the difference in 1-day retention

In [7]:
```
# Adding a column with the % difference between the two AB-groups
boot_1d['diff'] = ((boot_1d.gate_30-boot_1d.gate_40)/boot_1d.gate_40)*100

# Ploting the bootstrap % difference
ax = boot_1d['diff'].plot.kde()
ax.set_xlabel('% Difference in means')
```

Out[7]:
```
Text(0.5, 0, '% Difference in means')
```



8. The probability of a difference

From this chart, we can see that the most likely % difference is around 1% - 2% give or take, and that most of the distribution is above 0%, in favor of a gate at level 30. But what is the probability that the difference is above 0%? Let's calculate that as well.

In [8]:
```
# Calculating the probability that 1-day retention is greater when the gate is at le
prob = (boot_1d['diff'] > 0).sum()/(len(boot_1d['diff']))
```

```
# Pretty printing the probability
'{:.1%}'.format(prob)
```

Out[8]: '96.1%'

9. 7-day retention by AB-group

The bootstrap analysis tells us that there is a high probability that 1-day retention is better when the gate is at level 30. However, since players have only been playing the game for a day, it is likely that most players haven't reached level 30 yet. That is, many players won't have been affected by the gate, even if it's as early as level 30.

But after having played for a week, more players should have reached level 40, and therefore it makes sense to also look at 7-day retention. That is: What percentage of the people that installed the game also showed up a week later to play the game again.

Calculate 7-day retention for the two AB-groups.

In [9]:
```
# Calculating 7-day retention for both AB-groups
df.groupby('version')['retention_7'].mean()
```

Out[9]:
```
version
gate_30    0.190201
gate_40    0.182000
Name: retention_7, dtype: float64
```

10. Bootstrapping the difference again

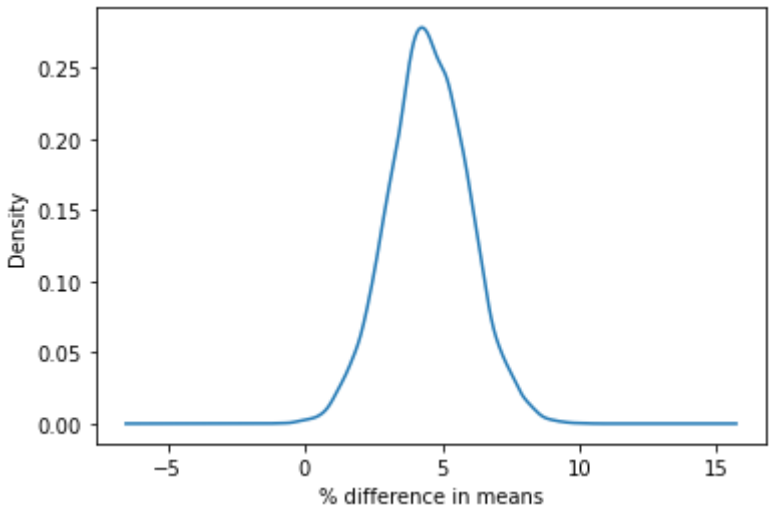
Similar with 1-day retention, we see that 7-day retention is slightly lower (18.2%) when the gate is at level 40 than when the gate is at level 30 (19.0%). This difference is larger than for 1-day retention, presumably because more players have time to reach the first gate. We also see that the overall 7-day retention is lower than the overall 1-day retention; fewer people play a game a week after installing than a day after installing.

let's use bootstrap analysis to figure out how certain we should be of the difference between the AB-groups.

In [10]:
```
# Creating a list with bootstrapped means for each AB-group
boot_7d = []
for i in range(10000):
    boot_mean = df.sample(frac=1, replace=True).groupby('version')['retention_7'].me
    boot_7d.append(boot_mean)

# Transforming the list to a DataFrame
boot_7d = pd.DataFrame(boot_7d)

# Adding a column with the % difference between the two AB-groups
boot_7d['diff'] = ((boot_7d.gate_30-boot_7d.gate_40)/boot_7d.gate_40)*100

# Ploting the bootstrap % difference
ax = boot_7d['diff'].plot.kde()
ax.set_xlabel("% difference in means")

# Calculating the probability that 7-day retention is greater when the gate is at le
prob = (boot_7d['diff'] > 0).sum() / len(boot_7d)
```

```
# Pretty printing the probability
'{:.1%}'.format(prob)
```

Out[10]: '99.9%'



In [11]:
```
# So, given the data and the bootstrap analysis
# Should we move the gate from level 30 to level 40 ?
move_to_level_40 = prob > 1-prob
if move_to_level_40 == True:
    print('We should not move the gate to level 40')
else:
        print('Move the gate to level 40')
```

```
We should not move the gate to level 40
```

11. The conclusion

The bootstrap result tells us that there is strong evidence that 7-day retention is higher when the gate is at level 30 than when it is at level 40. The conclusion is: If we want to keep retention high — both 1-day and especially 7-day retention — we should not move the gate from level 30 to level 40. There are, of course, other metrics we could look at, like the number of game rounds played or how much in-game purchases are made by the two AB-groups. But retention is one of the most important metrics. If we don't retain our player base, it doesn't matter how much money they spend in-game.