This project you find

1.Improt data set with python 2.Check and Cleaning data set 3.Explor data 4.AB test 5.summary article

import data

This dataset is from Kaggle, and it goes by the name of 'Cookie Cats' mobile game. You can find the dataset here. Thank you for providing the dataset, and now we can proceed to the next step, which is importing the data.

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

In [ ]:
```
path = 'E:\python\AB test mobile puzzle game\cookie_cats.csv'
df = pd.read_csv(path)
df.head()
```

Out[ ]:

	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

Nice, we have imported the necessary libraries and the dataset is now ready.Go next

Check and Cleaning data set

If you'd like to check the sample data in the dataset, you can use the `head()` function in Python.

In [ ]:
```
#check type of data
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 90189 entries, 0 to 90188
Data columns (total 5 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   userid          90189 non-null  int64 
 1   version         90189 non-null  object
 2   sum_gamerounds  90189 non-null  int64 
 3   retention_1     90189 non-null  bool  
 4   retention_7     90189 non-null  bool  
dtypes: bool(2), int64(2), object(1)
memory usage: 2.2+ MB
```

In [ ]:

```
#check null value in data set
df.isna().sum()
```

Out[ ]:

```
userid            0
version           0
sum_gamerounds    0
retention_1       0
retention_7       0
dtype: int64
```

`info()` function will display the names of all columns along with their data types.

`.isna().sum()` function will reveal the number of null values in each column.

In [ ]:

```
df['version'] = df['version'].replace({'gate_30':'A','gate_40':'B'})
```

We will replace values from 'gate_30' with 'A' and 'gate_40' with 'B' for easier use

Explor data

In [ ]:

```
df.groupby('version')['sum_gamerounds'].sum()
```

Out[ ]:

```
version
A    2344795
B    2333530
Name: sum_gamerounds, dtype: int64
```

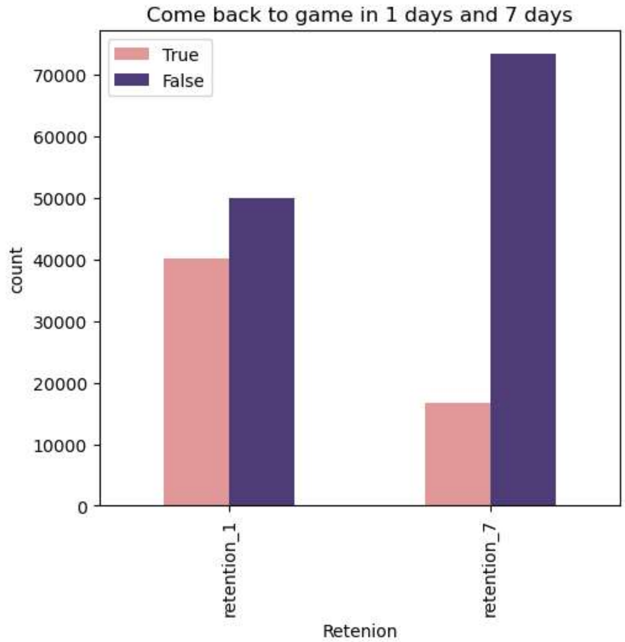
We know that each version has a total number of played game rounds.

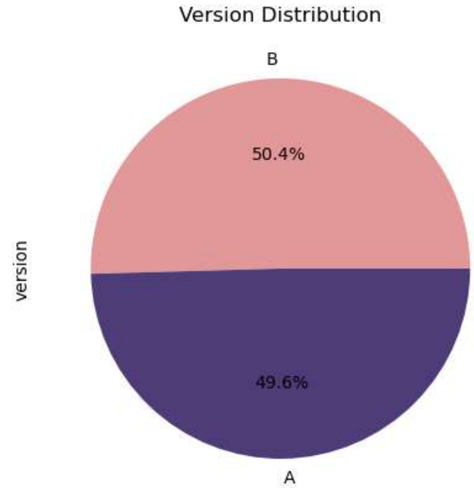
In [ ]:

```
#Export data
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 5))

reten_df = pd.DataFrame(np.zeros(4).reshape(2,2),index=["retention_1","retention_7"],c
reten_df['True'] = [df['retention_1'].sum(), df['retention_7'].sum()]
reten_df['False'] = [len(df) - df['retention_1'].sum(), len(df) - df['retention_7'].su
plot1 = reten_df.plot(kind='bar',title='Come back to game in 1 days and 7 days',color=
plot1.set_xlabel('Retenion')
plot1.set_ylabel('count')


df['version'].value_counts().plot(kind='pie',ax=ax2,autopct='%1.1f%%',colors=['#E1989
ax2.set_title('Version Distribution')
```

Out[ ]:

```
Text(0.5, 1.0, 'Version Distribution')
```





In this plot, we can observe the return of players to the game after 1 day and 7 days. In the second plot, each color represents a player in this dataset.

AB test

In this method, I create a function to calculate the variance and the mean. The reason for calculating the variance first is to determine if each group has an equal distribution or not. We start with an F-test to assess the equality of variances; this step is crucial because the subsequent T-test assumes equal variances when comparing means. If the F-test suggests that the variances are statistically equal, we can confidently proceed with the T-test to compare the means of the two groups. The T-test helps us determine whether the means are statistically equal or different. However, if the F-test indicates unequal variances, we may need to adjust our analysis approach or consider using a Welch's T-test, which doesn't assume equal variances. So, the sequence of using the F-test first is essential to ensure the validity of the subsequent T-test, allowing us to draw reliable conclusions about the means of the groups in our AB test.

In [ ]:
```
def A_B_function(dataframe, group, target):
    import scipy.stats as stats

    # Split data by group
    groupA = dataframe[dataframe[group]=='A'][target]
    groupB = dataframe[dataframe[group]=='B'][target]

    n = len(groupB)-(len(groupB)-len(groupA))

    groupA = groupA.sample(n=n,random_state=42)
    groupB = groupB.sample(n=n,random_state=42)
    print("""hypothesis test
    HO: Variances is equal
    H1:Variances is not equal""")
    statistic_Ftest, p_value_Ftest = stats.levene(groupA,groupB)

    alpha = 0.05
```

```
    if p_value_Ftest < alpha:
        print("----------------------------------------------------")
        print("Reject H0 hypothesis:Variances is not equal")
        print("----------------------------------------------------")
    else:
        print("----------------------------------------------------")
        print("Fail to reject HO: Variances is equal")
        print("----------------------------------------------------")

    print("""hypothesis test
    HO: mean is not different
    H1:mean is  different""")

    statistic_ttest, p_value_ttest = stats.ttest_rel(groupA,groupB)
    if p_value_ttest < alpha:
        print("----------------------------------------------------")
        print("Reject H0 hypothesis:Variances is not equal")
        print("----------------------------------------------------")
    else:
        print("----------------------------------------------------")
        print("Fail to reject HO: Variances is equal")
        print("----------------------------------------------------")

    meanA = np.mean(groupA)
    meanB = np.mean(groupB)

    print("meandiff:",meanA-meanB)
```

In [ ]:

```
A_B_function(dataframe=df, group='version', target='sum_gamerounds')
```

```
hypothesis test
    HO: Variances is equal
    H1:Variances is not equal
----------------------------------------------------
Fail to reject HO: Variances is equal
----------------------------------------------------
hypothesis test
    HO: mean is not different
    H1:mean is  different
----------------------------------------------------
Fail to reject HO: Variances is equal
----------------------------------------------------
meandiff: 1.1505145413870252
```

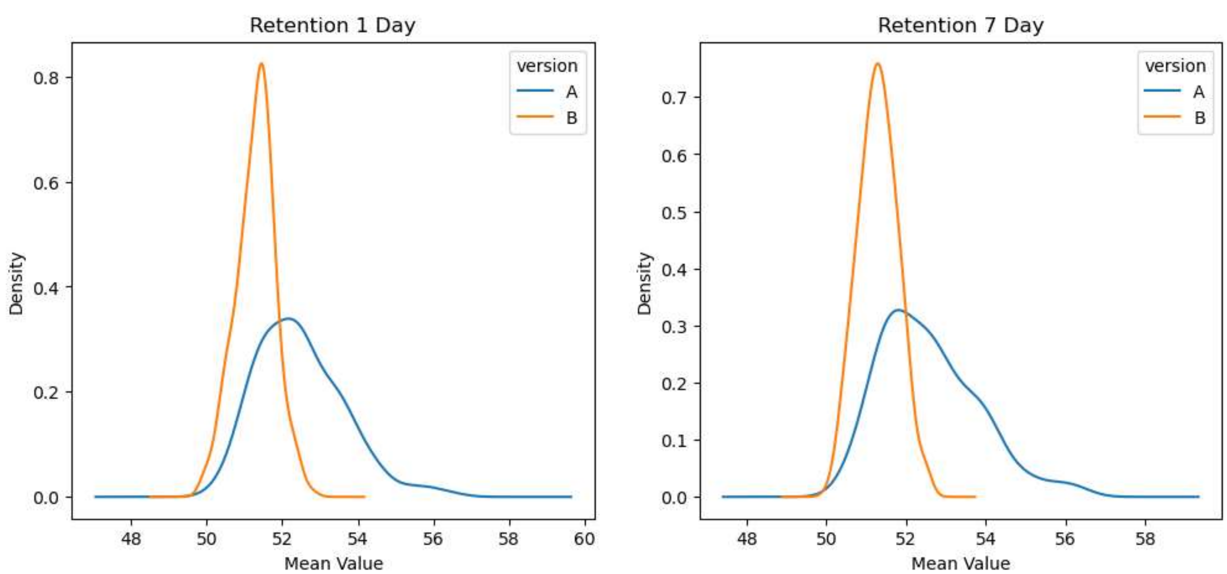
In the result of the F-test: When we fail to reject the null hypothesis (HO) that the variances are equal, it means that the variances are indeed equal, allowing us to proceed to the next step.

In the T-test, when the hypothesis test results in a 'Fail to reject HO: Variances are equal,' it suggests that the means may be equal or have only a slight difference.

After learning this information, I decided to perform further testing using graphs to understand what each mean gate is after conducting 500 rounds of testing. To achieve this, I applied bootstrap sampling as a testing method.

In [ ]:
```
fig, (ax3, ax4) = plt.subplots(1, 2, figsize=(12, 5))
mean_retention_1 = []
for i in range(200):
    bootstrap_sample = df.sample(frac = 1,replace=True)
    mean_1 = bootstrap_sample.groupby('version')['sum_gamerounds'].mean()
    mean_retention_1.append(mean_1)
mean_retention_1 = pd.DataFrame(mean_retention_1)

mean_retention_7 = []
for i in range(200):
    bootstrap_sample = df.sample(frac = 1,replace=True)
    mean_7 = bootstrap_sample.groupby('version')['sum_gamerounds'].mean()
    mean_retention_7.append(mean_7)
mean_retention_7 = pd.DataFrame(mean_retention_7)

mean_retention_1.plot(kind='density',ax=ax3)
ax3.set_title('Retention 1 Day')
ax3.set_xlabel('Mean Value')

mean_retention_7.plot(kind='density',ax=ax4)
plt.title('Retention 7 Day')
plt.xlabel('Mean Value')
```

Out[ ]:
```
Text(0.5, 0, 'Mean Value')
```



The graph provides a clear answer to our choice between gate A and B. Without a doubt, we should choose gate A because its mean average is higher than that of B

summary article

Summary of All Methods:

That the data is free of null values and ready for calculation. Many players do not return to play after installing, while some do. If the door is at stage 30, more players return to play the game compared to when the door is at stage 40. Confirm that the distribution of players between gate 30 and gate 40 is equal.

Proposed Solutions:

Consider retaining the door at gate 30 rather than moving it to gate 40, as this may impact player retention. Explore additional strategies to improve player retention, as it is concerning that many players do not return to the game after installing."