In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:

```
df2=pd.read_csv(r'C:\Users\user\Downloads\C9_Data.csv')
df2
```

Out[3]:

	row_id	user_id	timestamp	gate_id
0	0	18	2022-07-29 09:08:54	7
1	1	18	2022-07-29 09:09:54	9
2	2	18	2022-07-29 09:09:54	9
3	3	18	2022-07-29 09:10:06	5
4	4	18	2022-07-29 09:10:08	5
...	...	...	...	...
37513	37513	6	2022-12-31 20:38:56	11
37514	37514	6	2022-12-31 20:39:22	6
37515	37515	6	2022-12-31 20:39:23	6
37516	37516	6	2022-12-31 20:39:31	9
37517	37517	6	2022-12-31 20:39:31	9

37518 rows × 4 columns

In [4]:

```
df0=df2.head(10)
df=df0.fillna("30.0")
df
```

Out[4]:

	row_id	user_id	timestamp	gate_id
0	0	18	2022-07-29 09:08:54	7
1	1	18	2022-07-29 09:09:54	9
2	2	18	2022-07-29 09:09:54	9
3	3	18	2022-07-29 09:10:06	5
4	4	18	2022-07-29 09:10:08	5
5	5	18	2022-07-29 09:10:34	10
6	6	18	2022-07-29 09:32:47	11
7	7	18	2022-07-29 09:33:12	4
8	8	18	2022-07-29 09:33:13	4
9	9	1	2022-07-29 09:33:16	7

In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   row_id     10 non-null     int64 
 1   user_id    10 non-null     int64 
 2   timestamp  10 non-null     object
 3   gate_id    10 non-null     int64 
dtypes: int64(3), object(1)
memory usage: 448.0+ bytes
```

In [6]:

```
df.describe()
```

Out[6]:

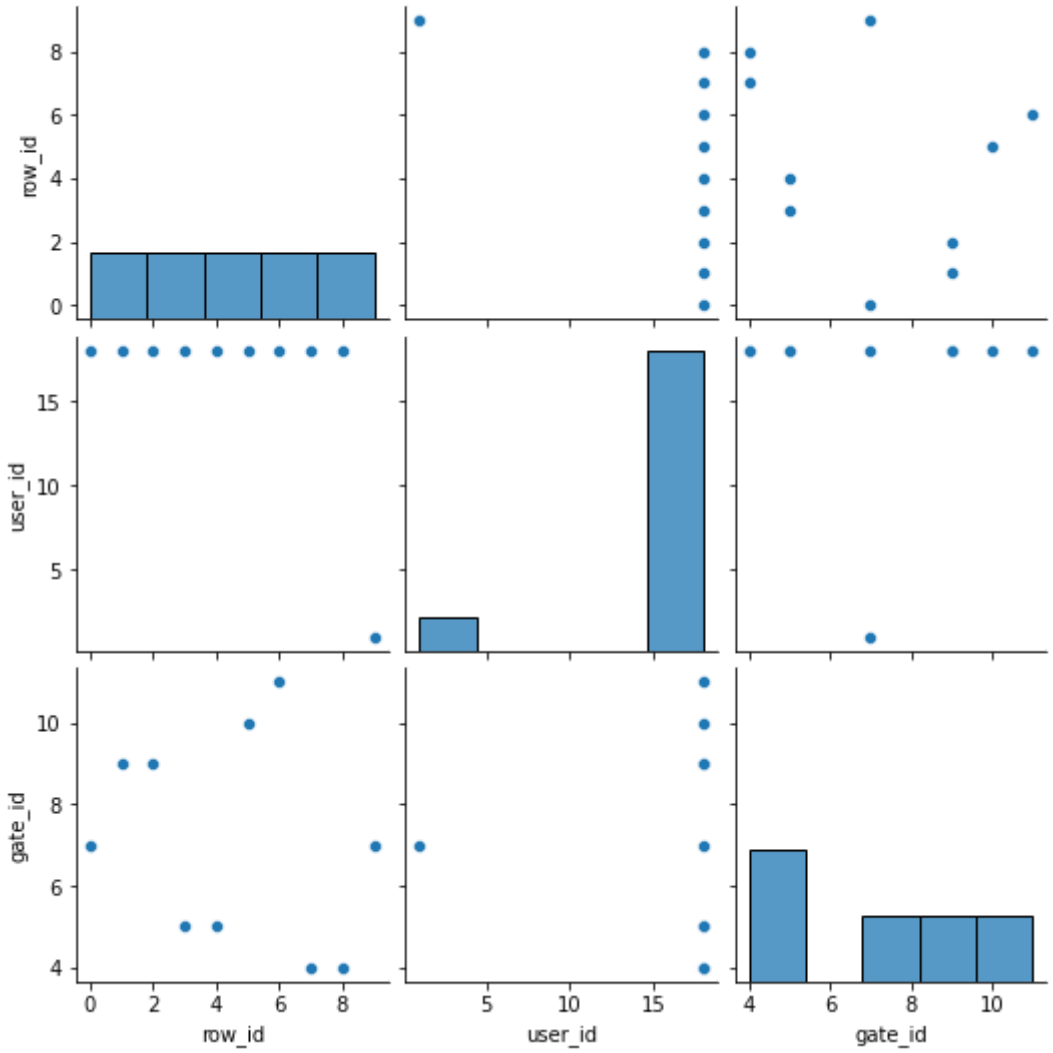
	row_id	user_id	gate_id
count	10.00000	10.000000	10.000000
mean	4.50000	16.300000	7.100000
std	3.02765	5.375872	2.558211
min	0.00000	1.000000	4.000000
25%	2.25000	18.000000	5.000000
50%	4.50000	18.000000	7.000000
75%	6.75000	18.000000	9.000000
max	9.00000	18.000000	11.000000

In [7]:

```
sns.pairplot(df)
```

Out[7]:

```
<seaborn.axisgrid.PairGrid at 0x1d8a8c247f0>
```



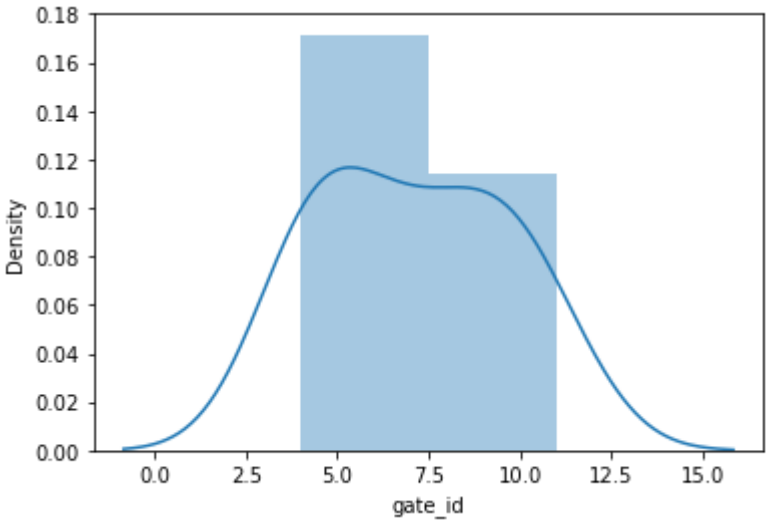
In [8]:

```
sns.distplot(df['gate_id'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557:
FutureWarning: `distplot` is a deprecated function and will be removed in
a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `histplot` (an axes-level fun
ction for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[8]:

```
<AxesSubplot:xlabel='gate_id', ylabel='Density'>
```



In [9]:

```
df.columns
```

Out[9]:

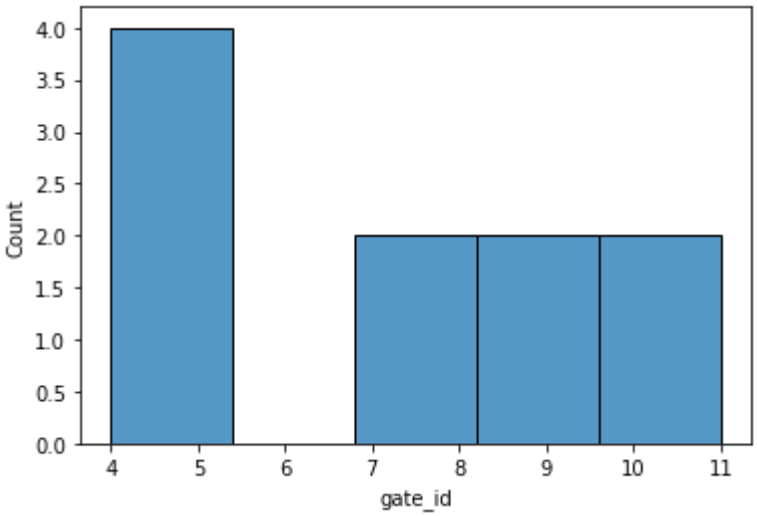
```
Index(['row_id', 'user_id', 'timestamp', 'gate_id'], dtype='object')
```

In [10]:

```
sns.histplot(df["gate_id"])
```

Out[10]:

```
<AxesSubplot:xlabel='gate_id', ylabel='Count'>
```



In [11]:

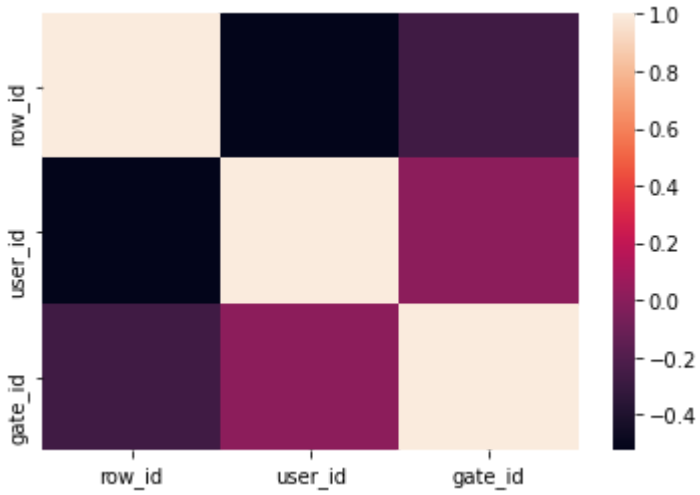
```
df1=df[['row_id', 'user_id', 'timestamp', 'gate_id']]
```

In [12]:

```
sns.heatmap(df1.corr())
```

Out[12]:

```
<AxesSubplot:>
```



In [14]:

```
x=df1[['row_id', 'user_id', 'gate_id']]
y=df1[[ 'gate_id']]
```

In [15]:

```
from sklearn.model_selection import train_test_split
```

In [16]:

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3)
```

Linear Regression

In [17]:

```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x_train,y_train)#ValueError: Input contains NaN, infinity or a value too large for
```

Out[17]:

```
LinearRegression()
```

In [18]:

```
print(lr.intercept_)
```

```
[2.66453526e-15]
```

In [19]:

```
coef= pd.DataFrame(lr.coef_)
coef
```

Out[19]:

	0	1	2
0	-3.029670e-16	0.0	1.0

In [20]:

```
print(lr.score(x_test,y_test))
```

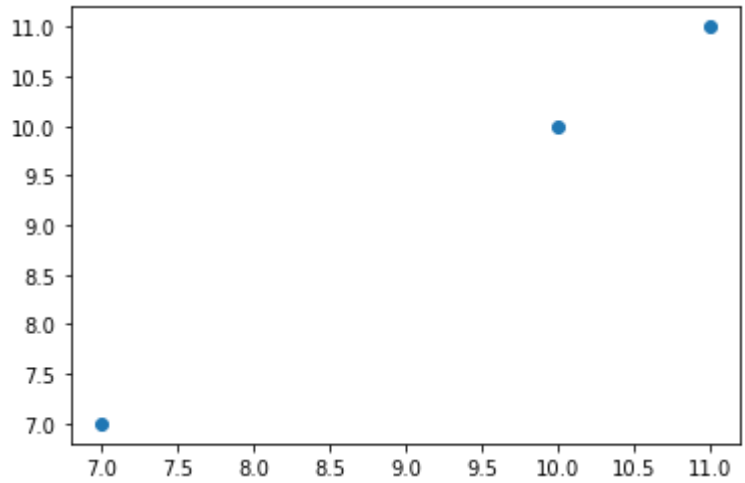
```
1.0
```

In [21]:

```
prediction = lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[21]:

```
<matplotlib.collections.PathCollection at 0x1d8a9516610>
```



In [22]:

```
lr.score(x_test,y_test)
```

Out[22]:

```
1.0
```

In [23]:

```
lr.score(x_train,y_train)
```

Out[23]:

```
1.0
```

In [24]:

```
from sklearn.linear_model import Ridge,Lasso
```

In [25]:

```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[25]:

```
Ridge(alpha=10)
```

In [26]:

```
rr.score(x_test,y_test)
```

Out[26]:

```
-0.5021777484431988
```

In [27]:

```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[27]:

```
Lasso(alpha=10)
```

In [28]:

```
la.score(x_test,y_test)
```

Out[28]:

```
-3.523547880690736
```

Elastic Net

In [29]:

```
from sklearn.linear_model import ElasticNet
en = ElasticNet()
en.fit(x_train,y_train)
```

Out[29]:

```
ElasticNet()
```

In [30]:

```
print(en.coef_)
```

```
[-0.12615457  0.          0.66057324]
```

In [31]:

```
print(en.intercept_)
```

```
[2.53560214]
```

In [32]:

```
prediction=en.predict(x_test)
print(prediction)
```

```
[9.04498035 8.51056168 6.02422366]
```

In [33]:

```
print(en.score(x_test,y_test))
```

```
0.19315371642564338
```

Logistic Regression

In [34]:

```
from sklearn.linear_model import LogisticRegression
```

In [36]:

```
feature_matrix=df1[['row_id', 'user_id', 'gate_id']]
target_vector = df1[['gate_id']]
```

In [37]:

```
feature_matrix.shape
```

Out[37]:

```
(10, 3)
```

In [38]:

```
target_vector.shape
```

Out[38]:

```
(10, 1)
```

In [39]:

```
from sklearn.preprocessing import StandardScaler
```

In [40]:

```
fs = StandardScaler().fit_transform(feature_matrix)
```

In [41]:

```
logr = LogisticRegression()
logr.fit(fs,target_vector)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:63:
DataConversionWarning: A column-vector y was passed when a 1d array was ex
pected. Please change the shape of y to (n_samples, ), for example using r
avel().
  return f(*args, **kwargs)
```

Out[41]:

```
LogisticRegression()
```

In [43]:

```
observation=df1[['row_id', 'user_id', 'gate_id']]
```

In [44]:

```
prediction = logr.predict(observation)
print(prediction)
```

```
[11 11 11 11 11 11 11  4  4 11]
```

In [45]:

```
logr.classes_
```

Out[45]:

```
array([ 4,  5,  7,  9, 10, 11], dtype=int64)
```

In [46]:

```
logr.predict_proba(observation)[0][0]
```

Out[46]:

```
0.0004888074773710361
```

In [47]:

```
logr.predict_proba(observation)[0][1]
```

Out[47]:

```
2.5166149601853298e-05
```

Evaluation Metrics

In [48]:

```
from sklearn import metrics
```

In [49]:

```
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
---------------------------------------------------------------------------
-
ValueError                                Traceback (most recent call las
t)
<ipython-input-49-4b3ebc69c71d> in <module>
----> 1 print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,pr
ediction))

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in
inner_f(*args, **kwargs)
     61             extra_args = len(args) - len(all_args)
     62             if extra_args <= 0:
---> 63                 return f(*args, **kwargs)
     64
     65             # extra_args > 0

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_regression.py
in mean_absolute_error(y_true, y_pred, sample_weight, multioutput)
    180     0.85...
    181     """
--> 182     y_type, y_true, y_pred, multioutput = _check_reg_targets(
    183         y_true, y_pred, multioutput)
    184     check_consistent_length(y_true, y_pred, sample_weight)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_regression.py
in _check_reg_targets(y_true, y_pred, multioutput, dtype)
     86         the dtype argument passed to check_array.
     87     """
---> 88     check_consistent_length(y_true, y_pred)
     89     y_true = check_array(y_true, ensure_2d=False, dtype=dtype)
     90     y_pred = check_array(y_pred, ensure_2d=False, dtype=dtype)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in
check_consistent_length(*arrays)
    260     uniques = np.unique(lengths)
    261     if len(uniques) > 1:
--> 262         raise ValueError("Found input variables with inconsistent
numbers of"
    263                          " samples: %r" % [int(l) for l in length
s])
    264

ValueError: Found input variables with inconsistent numbers of samples:
[3, 10]
```

In [50]:

```
print("Mean Squared Error:",metrics.mean_squared_error(y_test,prediction))
```

```
---------------------------------------------------------------------------
-
ValueError                                Traceback (most recent call las
t)
<ipython-input-50-a8cbb0b7f78a> in <module>
----> 1 print("Mean Squared Error:",metrics.mean_squared_error(y_test,pred
iction))

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in
inner_f(*args, **kwargs)
     61             extra_args = len(args) - len(all_args)
     62             if extra_args <= 0:
---> 63                 return f(*args, **kwargs)
     64
     65             # extra_args > 0

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_regression.py
in mean_squared_error(y_true, y_pred, sample_weight, multioutput, squared)
    333     0.825...
    334     """
--> 335     y_type, y_true, y_pred, multioutput = _check_reg_targets(
    336         y_true, y_pred, multioutput)
    337     check_consistent_length(y_true, y_pred, sample_weight)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_regression.py
in _check_reg_targets(y_true, y_pred, multioutput, dtype)
     86         the dtype argument passed to check_array.
     87     """
---> 88     check_consistent_length(y_true, y_pred)
     89     y_true = check_array(y_true, ensure_2d=False, dtype=dtype)
     90     y_pred = check_array(y_pred, ensure_2d=False, dtype=dtype)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in
check_consistent_length(*arrays)
    260     uniques = np.unique(lengths)
    261     if len(uniques) > 1:
--> 262         raise ValueError("Found input variables with inconsistent
numbers of"
    263                          " samples: %r" % [int(l) for l in length
s])
    264

ValueError: Found input variables with inconsistent numbers of samples:
[3, 10]
```

In [51]:

```
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
---------------------------------------------------------------------------
-
ValueError                                Traceback (most recent call las
t)
<ipython-input-51-c32fd676945a> in <module>
----> 1 print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_erro
r(y_test,prediction)))

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in
inner_f(*args, **kwargs)
     61             extra_args = len(args) - len(all_args)
     62             if extra_args <= 0:
---> 63                 return f(*args, **kwargs)
     64
     65             # extra_args > 0

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_regression.py
in mean_squared_error(y_true, y_pred, sample_weight, multioutput, squared)
    333     0.825...
    334     """
--> 335     y_type, y_true, y_pred, multioutput = _check_reg_targets(
    336         y_true, y_pred, multioutput)
    337     check_consistent_length(y_true, y_pred, sample_weight)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_regression.py
in _check_reg_targets(y_true, y_pred, multioutput, dtype)
     86         the dtype argument passed to check_array.
     87     """
---> 88     check_consistent_length(y_true, y_pred)
     89     y_true = check_array(y_true, ensure_2d=False, dtype=dtype)
     90     y_pred = check_array(y_pred, ensure_2d=False, dtype=dtype)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in
check_consistent_length(*arrays)
    260     uniques = np.unique(lengths)
    261     if len(uniques) > 1:
--> 262         raise ValueError("Found input variables with inconsistent
numbers of"
    263                          " samples: %r" % [int(l) for l in length
s])
    264

ValueError: Found input variables with inconsistent numbers of samples:
[3, 10]
```

Model Saving

In [52]:

```
import pickle
```

In [53]:

```
filename="prediction"
pickle.dump(lr,open(filename,'wb'))
```