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

In [2]:
```
from sklearn.linear_model import LogisticRegression
```

In [3]:
```
df=pd.read_csv("C9_Data.csv")
df
```

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]:
```
df=df.dropna()
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
...	...	...	...	...
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

	row_id	user_id	timestamp	gate_id
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 [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 37518 entries, 0 to 37517
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   row_id     37518 non-null  int64 
 1   user_id    37518 non-null  int64 
 2   timestamp  37518 non-null  object
 3   gate_id    37518 non-null  int64 
dtypes: int64(3), object(1)
memory usage: 1.4+ MB
```

In [6]:
```
df.columns
```

Out[6]:
```
Index(['row_id', 'user_id', 'timestamp', 'gate_id'], dtype='object')
```

In [7]:
```
feature_matrix=df[['row_id', 'user_id']]
target_vector=df[ 'gate_id']
```

In [8]:
```
feature_matrix.shape
```

Out[8]:
```
(37518, 2)
```

In [9]:
```
target_vector.shape
```

Out[9]:
```
(37518,)
```

In [10]:
```
from sklearn.preprocessing import StandardScaler
```

In [11]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [12]:
```
logr=LogisticRegression()
logr.fit(fs,target_vector)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:763: Conver
genceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
```

```
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[12]:
```
LogisticRegression()
```

In [13]:
```
observation=[[1,2]]
```

In [14]:
```
prediction=logr.predict(observation)
print(prediction)
```

```
[3]
```

In [15]:
```
logr.classes_
```

Out[15]:
```
array([-1,  0,  1,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16],
      dtype=int64)
```

In [16]:
```
logr.predict_proba(observation)[0][0]
```

Out[16]:
```
0.005365176788164149
```

In [17]:
```
logr.predict_proba(observation)
```

Out[17]:
```
array([[5.36517679e-03, 2.43221075e-05, 9.36568351e-05, 2.22025633e-01,
        2.19695882e-01, 7.52352405e-02, 5.84513730e-02, 7.17956781e-02,
        2.68284044e-03, 7.98655513e-02, 1.24425419e-01, 1.07054385e-01,
        2.51118120e-03, 7.57336969e-03, 2.68214159e-05, 2.29125763e-02,
        2.60893089e-04]])
```

In [18]:
```
x=df[['row_id', 'user_id']]
y=df[ 'gate_id']
```

In [19]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

In [20]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[20]:
```
LinearRegression()
```

In [21]:
```
lr.intercept_
```

Out[21]:
```
7.278299996623737
```

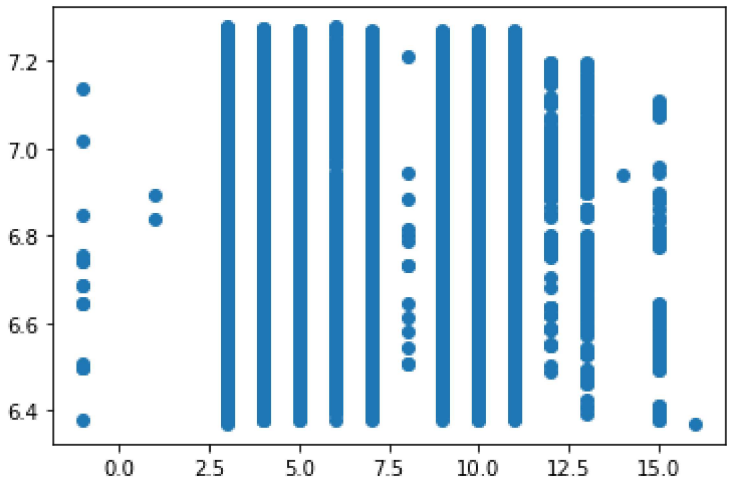
In [22]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[22]:

	Co-efficient
row_id	-0.000005
user_id	-0.012829

In [23]:
```
prediction =lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[23]: <matplotlib.collections.PathCollection at 0x2099ed6c4c0>



In [24]:
```
lr.score(x_test,y_test)
```

Out[24]: 0.005925010775539419

In [25]:
```
lr.score(x_train,y_train)
```

Out[25]: 0.005346973822092593