In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
```

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

In [3]:
```
df=pd.read_csv(r"E:\154\C9_Data - C9_Data.csv").dropna()
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.head()
```

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

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.describe()
```

Out[6]:

	row_id	user_id	gate_id
count	37518.000000	37518.000000	37518.000000
mean	18758.500000	28.219015	6.819607
std	10830.658036	17.854464	3.197746
min	0.000000	0.000000	-1.000000
25%	9379.250000	12.000000	4.000000
50%	18758.500000	29.000000	6.000000
75%	28137.750000	47.000000	10.000000
max	37517.000000	57.000000	16.000000

In [7]:
```
df.columns
```

Out[7]:
```
Index(['row_id', 'user_id', 'timestamp', 'gate_id'], dtype='object')
```

In [8]:
```
feature_matrix = df[['row_id','user_id']]
target_vector = df[['gate_id']]
```

In [9]:
```
fs=StandardScaler().fit_transform(feature_matrix)
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 expected. Please change the shape of y to (n_samples, ), for example using ravel().
  return f(*args, **kwargs)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: 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 (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 (https://scikit-learn.org/stable/modules/line
ar_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

Out[9]:
```
LogisticRegression()
```

In [10]:
```
observation=[[1,2]]
```

In [11]:
```
prediction=logr.predict(observation)
print(prediction)
```

```
[3]
```

In [12]:
```
logr.classes_
```

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

In [13]:
```
logr.predict_proba(observation)[0][0]
```

Out[13]:
```
0.005365176788164149
```

In [14]:
```
logr.predict_proba(observation)[0][1]
```

Out[14]:
```
2.4322107532317633e-05
```

In [15]:
```
x=df[['row_id','user_id']]
y=df['gate_id']
```

In [16]:
```
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 [ ]:
```
```

In [17]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[17]:
```
LinearRegression()
```

In [18]:
```
lr.intercept_
```

Out[18]:
```
7.30281829089302
```

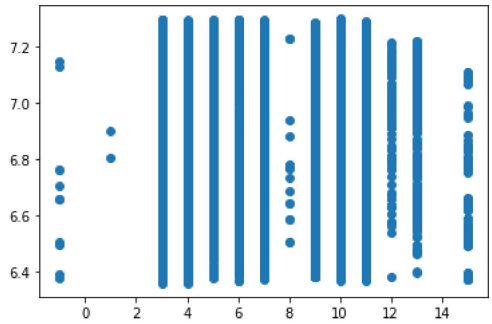
In [19]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[19]:

	Co-efficient
row_id	-0.000006
user_id	-0.012976

In [20]:
```
prediction =lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[20]:
```
<matplotlib.collections.PathCollection at 0x1ef6d2b0730>
```



Random Forest

In [21]:
```
df['gate_id'].value_counts()
```

Out[21]:
```
 4     8170
 3     5351
 10    4767
 5     4619
 11    4090
 9     3390
 7     3026
 6     1800
 13    1201
 12     698
 15     298
-1       48
 8       48
 1        5
 16       4
 0        2
 14       1
Name: gate_id, dtype: int64
```

In [22]:
```
x=df[['row_id','user_id']]
y=df['gate_id']
```

In [ ]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.70)
```

In [24]:
```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[24]:
```
RandomForestClassifier()
```

In [25]:
```
parameters = {'max_depth':[1,2,3,4,5],'min_samples_leaf':[5,10,15,20,25],
              'n_estimators': [10,20,30,40,50]
              }
```

In [26]:
```
from sklearn.model_selection import GridSearchCV
grid_search = GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring="accuracy")
grid_search.fit(x_train,y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\model_selection\_split.py:666: UserWarning: The least populated class in y h
as only 1 members, which is less than n_splits=2.
  warnings.warn(("The least populated class in y has only %d"
```

Out[26]:
```
GridSearchCV(cv=2, estimator=RandomForestClassifier(),
             param_grid={'max_depth': [1, 2, 3, 4, 5],
                         'min_samples_leaf': [5, 10, 15, 20, 25],
                         'n_estimators': [10, 20, 30, 40, 50]},
             scoring='accuracy')
```

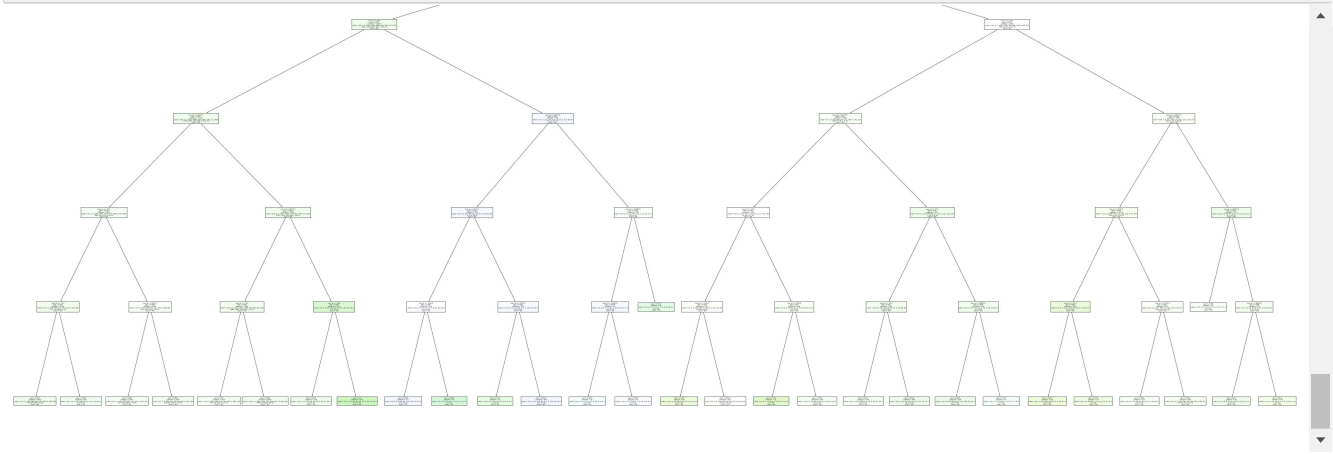
In [27]:
```
grid_search.best_score_
```

Out[27]:
```
0.22469728124286042
```

In [28]:
```
rfc_best = grid_search.best_estimator_
```

In [30]:

```
', 'No','Yes', 'No','Yes', 'No','Yes', 'No','Yes', 'No','Yes', 'No','Yes', 'No','Yes', 'No','Yes', 'No','Yes', 'No','Yes', 'No',]
```



In [ ]: