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

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

In [3]:
```
df=pd.read_csv("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.dropna(inplace=True)
```

In [5]:
```
df['gate_id'].value_counts()
```

Out[5]:
```
 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 [6]:
```
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 [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: 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/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

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

In [13]:
```
feature_matrix.shape
```

Out[13]:
```
(37518, 2)
```

In [14]:
```
target_vector.shape
```

Out[14]:
```
(37518,)
```

In [15]:
```
from sklearn.preprocessing import StandardScaler
```

In [16]:
```
fs = StandardScaler().fit_transform(feature_matrix)
```

Regression

In [17]:
```
logr = LogisticRegression()
logr.fit(fs,target_vector)
```

```
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/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

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

In [18]:
```
observation=df[['row_id','user_id']]
```

In [19]:
```
prediction = logr.predict(observation)
prediction
```

Out[19]:
```
array([-1, -1, -1, ..., 16, 16, 16], dtype=int64)
```

In [20]:
```
logr.classes_
```

Out[20]:
```
array([-1,  0,  1,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16],
      dtype=int64)
```

In [21]:
```
logr.predict_proba(observation)[0][1]
```

Out[21]:
```
1.7263815682078809e-09
```

In [22]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [23]:
```
x = df[['row_id','user_id']]
y = df['gate_id']
```

In [24]:
```
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 [25]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
rr.score(x_test,y_test)
rr.score(x_train,y_train)
```

Out[25]:
```
0.005122582651229557
```

In [26]:
```
from sklearn.linear_model import LinearRegression
lr= LinearRegression()
lr.fit(x_train,y_train)
```

Out[26]:
```
LinearRegression()
```

In [27]:
```
lr.intercept_
```

Out[27]:
```
7.248714067408693
```

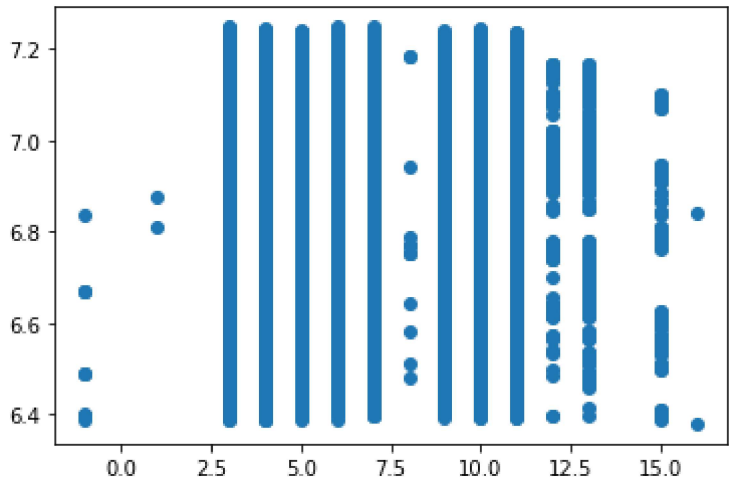
In [28]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[28]:

	Co-efficient
row_id	-0.000004
user_id	-0.012597

In [29]:
```
prediction = lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[29]:
```
<matplotlib.collections.PathCollection at 0x210d36eebe0>
```



Random Forest

In [30]:
```
df['gate_id'].value_counts()
```

Out[30]:
```
 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 [31]:
```
x=df[['row_id','user_id']]
y=df['gate_id']
```

In [32]:
```
g1={'gate_id':{"4":1, "3":2, "10":3, "5":4, "11":5, "9":6, "7":7, "6":8, "13":
df=df.replace(g1)
df
```

Out[32]:

	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 [33]:
```
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 [34]:
```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[34]:
```
RandomForestClassifier()
```

In [35]:
```
parameters = {'max_depth':[1,2,3,4,5],'min_samples_leaf':[5,10,15,20,25],
              'n_estimators': [10,20,30,40,50]
              }
```

In [36]:
```
from sklearn.model_selection import GridSearchCV
grid_search = GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring="a
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 has only 1 members, which is
less than n_splits=2.
  warnings.warn(("The least populated class in y has only %d"
```

Out[36]:
```
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 [37]:
```
grid_search.best_score_
```

Out[37]:
```
0.22005178585027796
```

In [38]:
```
rfc_best = grid_search.best_estimator_
```

In [39]:
```
from sklearn.tree import plot_tree
plt.figure(figsize = (80,40,))
plot_tree(rfc_best.estimators_[5],feature_names=x.columns,class_names=['Yes','
```

Out[39]:
```
[Text(2185.5, 1993.2, 'user_id <= 49.5\ngini = 0.872\nsamples = 16632\nvalu
e = [37, 1, 2, 3787, 5622, 3257, 1212, 2241, 42, 2410\n3324, 2779, 485, 80
4, 2, 254, 3]\nclass = Yes'),
 Text(1097.4, 1630.8000000000002, 'row_id <= 14076.5\ngini = 0.873\nsamples
= 13840\nvalue = [24, 1, 2, 2717, 4588, 2997, 904, 1888, 42, 2166\n2788, 23
03, 457, 755, 2, 208, 0]\nclass = Yes'),
 Text(595.2, 1268.4, 'row_id <= 2032.0\ngini = 0.878\nsamples = 5156\nvalue
= [11, 1, 1, 948, 1666, 1186, 359, 685, 9, 631, 962\n759, 328, 457, 0, 44,
0]\nclass = Yes'),
 Text(297.6, 906.0, 'row_id <= 1858.0\ngini = 0.864\nsamples = 732\nvalue =
[4, 0, 0, 132, 290, 151, 29, 93, 0, 73, 147, 139\n61, 44, 0, 7, 0]\nclass =
Yes'),
 Text(148.8, 543.5999999999999, 'user_id <= 19.5\ngini = 0.873\nsamples = 6
51\nvalue = [4, 0, 0, 123, 221, 142, 22, 89, 0, 67, 142, 113\n61, 43, 0, 7,
0]\nclass = Yes'),
 Text(74.4, 181.19999999999982, 'gini = 0.873\nsamples = 348\nvalue = [0,
0, 0, 51, 117, 83, 7, 42, 0, 33, 66, 59, 57\n36, 0, 0, 0]\nclass = Yes'),
 Text(223.20000000000002, 181.19999999999982, 'gini = 0.863\nsamples = 303
\nvalue = [4, 0, 0, 72, 104, 59, 15, 47, 0, 34, 76, 54, 4\n7, 0, 7, 0]\ncla
```