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
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: 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://sciki
t-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-regres
sion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regr
ession)
  n_iter_i = _check_optimize_result(

Out[12]: LogisticRegression()
```

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

In [14]:
```
prediction=logr.predict(observation)
```
```
[3]
```

In [15]:

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

In [16]:

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

In [17]:

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']]
```

In [19]:
```
from sklearn.model_selection import train_test_split
```

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

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

In [21]:

Out[21]:
```
7.2739940489914385
```

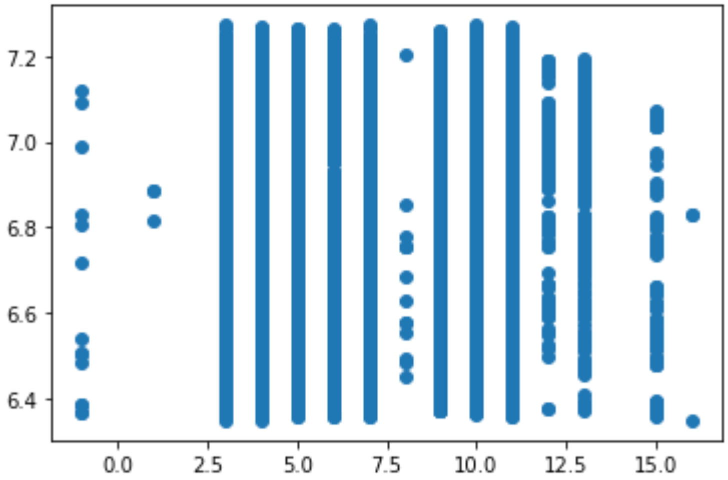
In [22]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
```

Out[22]:

	Co-efficient
row_id	-0.000006
user_id	-0.012404

In [23]:
```
prediction =lr.predict(x_test)
```

Out[23]:
```
<matplotlib.collections.PathCollection at 0x1fe2adf3670>
```



In [24]:

Out[24]:
```
0.006144670121054574
```

In [25]:

Out[25]:
```
0.005164797852675873
```

RANDOM FOREST

In [26]:

Out[26]:
```
 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 [27]:
```
x=df[['row_id', 'user_id']]
y=df[ 'gate_id']
```

In [28]:

```
#g1={ 'TenYearCHD':{'Audi':1, 'BMW':2, 'VW':3, 'ford':4, 'hyundi':5, 'merc':6,
       #'vauxhall':9}}
#df=df.replace(g1)
```

In [29]:

In [30]:

In [31]:

In [32]:

```
rfc=RandomForestClassifier()
```

Out[32]:

```
RandomForestClassifier()
```

In [33]:

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

In [34]:

```
from sklearn.model_selection import GridSearchCV
grid_search =GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring="ac
```

```
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[34]:

```
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 [35]:

Out[35]:

```
0.22595384966872287
```

In [36]:

In [37]:
```
from sklearn.tree import plot_tree

plt.figure(figsize=(80,40))
plot_tree(rfc_best.estimators_[5],feature_names=x.columns,class_names=['a','b'
```

Out[37]:
```
[Text(2349.4736842105262, 1993.2, 'user_id <= 49.5\ngini = 0.871\nsamples = 1
6532\nvalue = [35, 2, 4, 3718, 5726, 3220, 1322, 2075, 34, 2413\n3318, 2841,
503, 837, 0, 212, 2]\nclass = e'),
 Text(1253.0526315789473, 1630.8000000000002, 'user_id <= 16.0\ngini = 0.874\
nsamples = 13791\nvalue = [28, 2, 4, 2678, 4624, 2939, 982, 1717, 34, 2187\n2
824, 2350, 486, 807, 0, 184, 2]\nclass = e'),
 Text(626.5263157894736, 1268.4, 'row_id <= 18502.0\ngini = 0.86\nsamples = 5
488\nvalue = [5, 0, 0, 829, 1914, 1499, 322, 709, 10, 900, 1254\n1012, 49, 17
5, 0, 27, 0]\nclass = e'),
 Text(313.2631578947368, 906.0, 'user_id <= 8.5\ngini = 0.86\nsamples = 2515\
nvalue = [3, 0, 0, 368, 884, 720, 173, 356, 1, 329, 498\n500, 38, 108, 0, 0,
0]\nclass = e'),
 Text(156.6315789473684, 543.5999999999999, 'row_id <= 9107.0\ngini = 0.883\n
samples = 1150\nvalue = [3, 0, 0, 222, 324, 209, 123, 168, 1, 218, 200\n204,
32, 98, 0, 0, 0]\nclass = e'),
 Text(78.3157894736842, 181.19999999999982, 'gini = 0.872\nsamples = 638\nval
ue = [0, 0, 0, 163, 197, 115, 62, 89, 1, 101, 101, 112\n24, 19, 0, 0, 0]\ncla
ss = e'),
 Text(234.9473684210526, 181.19999999999982, 'gini = 0.885\nsamples = 512\nva
lue = [3, 0, 0, 59, 127, 94, 61, 79, 0, 117, 99, 92\n8, 79, 0, 0, 0]\nclass
```