In [1]:
```
# import libraries
import pandas as pd
import numpy as np
import pickle

from catboost import CatBoostClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score
```

In [2]:
```
# Load train and test data
```

In [3]:
```
train_data = pd.read_csv('data/train.csv')
```

In [4]:
```
train_data.shape
```

Out[4]:
```
(32769, 10)
```

In [5]:
```
y_train = train_data['ACTION']
```

In [6]:
```
y_train.shape
```

Out[6]:
```
(32769,)
```

In [7]:
```
train_data = train_data.drop(columns=['ACTION'], axis=1)
```

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

Out[8]:
```
(32769, 9)
```

In [9]:
```
test_data = pd.read_csv('data/test.csv')
```

In [10]:
```
test_data.shape
```

Out[10]: (58921, 10)

Function-1

1. Should include entire pipeline, from data preprocessing to ma
king final predictions.
2. It should take in raw data as input.
3. It should return predictions for your input. Here the input c
an be a single point or a set of points.
def final_fun_1(X):
.....
.....
..... # you will use the best model that you found out with your
 experiments
return predictions made on X (Raw Data)

In [11]:
```
def final_fun_1(X):
    '''
    Use pretrained model and make the prediction on single or multiple
 input
    '''
    # Load pre trained catboost model
    model = pickle.load(open('models/catboost_model.pkl', 'rb'))
    test = X.drop(columns=['id'], axis=1)
    return model.predict(test)
```

In [12]:
```
# test final_fun_1 on single and multiple input
```

In [13]:
```
test1 = final_fun_1(test_data[:1])
```

In [14]:
```
test1
```

Out[14]: array([1])

In [15]:
```
test10 = final_fun_1(test_data[:10])
```

In [16]:
```
test10
```

Out[16]:
```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1])
```

In [17]:
```
test90 = final_fun_1(test_data[10:100])
```

In [18]:
```
test90
```

Out[18]:
```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0,
       1, 1, 1])
```

In [19]:
```
test200 = final_fun_1(test_data[100:200])
```

In [20]:
```
test200
```

Out[20]:
```
array([1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
       1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0])
```

Function-2

```
1. Should include entire pipeline, from data preprocessing to ma
king final predictions.
2. It should take in raw data as input along with its target val
ues.
3. It should return the metric value that you are judging your m
odels on.
def final_fun_2(X,Y):
.....
.....
..... # you will use the best model that you found out with your
 experiments
return final_metric computed on X ( Raw Data) and Y (target vari
able)
```

In [21]:
```
params = {
            'loss_function':'Logloss',
            'eval_metric':'AUC',
            'cat_features':list(range(train_data.shape[1])),
            'verbose':100,
            'random_seed':42
        }
```

In [22]:
```
def final_fun_2(X_train, X_test, y_train, y_test):
    '''
    Predict performance metric
    '''
    global params

    clf= CatBoostClassifier(**params)
    clf.fit(X_train,y_train)

    pred = clf.predict_proba(X_test)[:,1]
    return roc_auc_score(y_test,pred)
```

Split the train.csv into train and test data , fit the model using training data and calculate the performance metric using test data

In [23]:
```
X_train, X_test, y_train, y_test = train_test_split(train_data, y_train
, test_size=0.30, stratify=y_train)
```

In [24]:
```
X_train.shape,X_test.shape,y_train.shape,y_test.shape
```

Out[24]:
```
((22938, 9), (9831, 9), (22938,), (9831,))
```

In [25]:
```
auc = final_fun_2(X_train,X_test,y_train,y_test)
```

```
Learning rate set to 0.039255
0:      total: 61.3ms   remaining: 1m 1s
100:    total: 1.53s    remaining: 13.7s
200:    total: 3.63s    remaining: 14.4s
300:    total: 5.69s    remaining: 13.2s
400:    total: 7.8s     remaining: 11.7s
500:    total: 9.93s    remaining: 9.89s
600:    total: 12s      remaining: 7.97s
700:    total: 14s      remaining: 5.98s
800:    total: 16.1s    remaining: 4.01s
900:    total: 18.2s    remaining: 2s
999:    total: 20.4s    remaining: 0us
```

In [26]:
```
print('Performance metric: AUC :', auc)
```

```
Performance metric: AUC : 0.8939861041904884
```

In [ ]:
```

```