CatBoost Hyper-parameter.

In [1]:
```python
# Generic Libraries
import warnings
warnings.filterwarnings('ignore')

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import  RobustScaler
import numpy as np
```

In [2]:
```python
# Model Libraries.

# Cross validation
from sklearn.model_selection import cross_val_score

#-------------/ XGBoost /-------------------------
from xgboost import XGBClassifier
import xgboost as xgb

#------------/ AdaBoost /-------------------------
from sklearn.ensemble import AdaBoostClassifier

#------------/ CatBoost /-------------------------

from catboost import CatBoostClassifier
```

In [3]:
```python
# Metric Libraries
from sklearn.metrics import roc_auc_score, accuracy_score, precision_score, recall_score, f1_score,fbeta_score, confusion_matrix

# Grid
from sklearn.model_selection import GridSearchCV
```

In [4]:
```python
# Load dataset.
df = pd.read_csv('creditcard.csv')
df = df.drop("Time", axis = 1)

y= df["Class"]
X = df.drop("Class", axis = 1)
y.shape,X.shape
```

Out[4]:
```
((284807,), (284807, 29))
```

In [5]:
```python
# Separation of the dataset

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state = 42,stratify=y)
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[5]:
```
((227845, 29), (56962, 29), (227845,), (56962,))
```

In [6]:
```python
# Check dataset composition

print(" Fraudulent Count for Full data :  ",np.sum(y))
print(" Fraudulent Count for Train data : ",np.sum(y_train))
print(" Fraudulent Count for Test data :   ",np.sum(y_test))
```

```
 Fraudulent Count for Full data :   492
 Fraudulent Count for Train data :  394
 Fraudulent Count for Test data :    98
```

In [7]:
```python
# Save the testing set for evaluation
X_test_saved = X_test.copy()
y_test_saved = y_test.copy()
print("Saved X_test & y_test")
```

```
Saved X_test & y_test
```

In [8]:
```python
# As PCA is already performed on the dataset from V1 to V28 features, we are scaling only Amount field
scaler = RobustScaler()

# Scaling the train data
X_train[["Amount"]] = scaler.fit_transform(X_train[["Amount"]])

# Transforming the test data
X_test[["Amount"]] = scaler.transform(X_test[["Amount"]])
```

1.- Transformaciones de datos.

Dataset Original

Smote

In [9]:
```python
# Import of specific libraries
from collections import Counter
from imblearn.over_sampling import SMOTE

# Initial situation
print('Original dataset shape %s' % Counter(y_train))

# Calculate OverSampling model
smote = SMOTE(random_state=42)
X_train_smote, y_train_smote = smote.fit_resample(X_train, y_train)

print('Resampled dataset shape %s' % Counter(y_train_smote))
```

```
Original dataset shape Counter({0: 227451, 1: 394})
Resampled dataset shape Counter({0: 227451, 1: 227451})
```

Adasyn

In [10]:
```python
# Import of specific libraries
from imblearn.over_sampling import ADASYN

# Initial situation
print('Original dataset shape %s' % Counter(y_train))

# Calculate OverSampling model
adasyn = ADASYN(random_state=42)
X_train_adasyn, y_train_adasyn = adasyn.fit_resample(X_train, y_train)

print('Resampled dataset shape %s' % Counter(y_train_adasyn))
```

```
Original dataset shape Counter({0: 227451, 1: 394})
Resampled dataset shape Counter({1: 227458, 0: 227451})
```

In [11]:
```python
# LOAD OF MODELS.
# perfom cross validation on the X_train & y_train
from sklearn.model_selection import StratifiedKFold

# Initialize StratifiedKFold cross-validator
# perform cross validation
skf = StratifiedKFold(n_splits=3, random_state=None, shuffle=False)
#  Shuffle is False because we need a constant best model when we use GridSearchCV
```

Power Transformation

Original

In [12]:
```python
# - Apply : preprocessing.PowerTransformer(copy=False) to fit & transform the train & test data

from sklearn import metrics
from sklearn import preprocessing


from sklearn.preprocessing import PowerTransformer

pt= preprocessing.PowerTransformer(method='yeo-johnson', copy=True)  # creates an instance of the PowerTransformer class.
pt.fit(X_train)

X_train_pt = pt.transform(X_train)
X_test_pt = pt.transform(X_test)

y_train_pt = y_train
y_test_pt = y_test
```

Smote

In [13]:
```python
# Import of specific libraries
from collections import Counter
from imblearn.over_sampling import SMOTE

# Initial situation
print('Original dataset shape %s' % Counter(y_train_pt))

# Calculate OverSampling model
smote = SMOTE(random_state=42)
X_train_smote_pt, y_train_smote_pt = smote.fit_resample(X_train_pt, y_train_pt)

print('Resampled dataset shape %s' % Counter(y_train_smote_pt))
```

```
Original dataset shape Counter({0: 227451, 1: 394})
Resampled dataset shape Counter({0: 227451, 1: 227451})
```

Adasyn

In [14]:
```python
# Import of specific libraries
from imblearn.over_sampling import ADASYN

# Initial situation
print('Original dataset shape %s' % Counter(y_train))

# Calculate OverSampling model
adasyn = ADASYN(random_state=42)
X_train_adasyn_pt, y_train_adasyn_pt = adasyn.fit_resample(X_train_pt, y_train_pt)

print('Resampled dataset shape %s' % Counter(y_train_adasyn_pt))
```

```
Original dataset shape Counter({0: 227451, 1: 394})
Resampled dataset shape Counter({1: 227459, 0: 227451})
```

In [15]:
```python
# Original distribution
OR_origin = ['OR origin',X_train, y_train, X_test, y_test]
OR_smote = ['OR smote',X_train_smote, y_train_smote, X_test, y_test]
OR_adasyn = ['OR adasyn', X_train_adasyn, y_train_adasyn, X_test, y_test]

# Power Transformation
PT_origin = ['PT origin',X_train_pt, y_train_pt, X_test_pt, y_test_pt]
PT_smote = ['PT smote',X_train_smote_pt, y_train_smote_pt, X_test_pt, y_test_pt ]
PT_adasyn = ['PT adasyn', X_train_adasyn_pt, y_train_adasyn_pt, X_test_pt, y_test_pt]
```

Preparacion carga de modelos: librerias y funciones

In [16]:
```python
# LOAD OF MODELS.
# perfom cross validation on the X_train & y_train
from sklearn.model_selection import StratifiedKFold

# Initialize StratifiedKFold cross-validator
# perform cross validation
skf = StratifiedKFold(n_splits=3, random_state=None, shuffle=False)
#  Shuffle is False because we need a constant best model when we use GridSearchCV
```

In [17]:
```python
from sklearn.model_selection import cross_val_score
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import cross_val_predict
```

In [49]:
```python
def evaluate_catboost(data_list, params_to_show=None, threshold=0.5, **cat_params):
    '''
    This function trains an XGBoost model and evaluates it with a custom classification threshold.

    Parameters:
        - data_list: List containing [name, X_train, y_train, X_val, y_val].
        - params_to_show: Dictionary with parameters to display (optional).
        - threshold: The classification threshold (default = 0.3).
        - **xgb_params: Additional XGBoost parameters to be passed dynamically.

    Return:
        - A DataFrame with evaluation metrics (Accuracy, Precision, Recall, F1, F2, ROC-AUC, Confusion Matrix).
    '''
    # Diccionario de abreviaturas
    param_abbreviations = {
        'n_estimators': 'n_est',
        'learning_rate': 'lr',
        'max_depth': 'md',
        'threshold': 'th'
    }

    # Unpack the data list
    name = data_list[0]
    X_train, y_train, X_val, y_val = data_list[1:]

    # Define the  model, passing ***ada_params dynamically

    cat_model =  CatBoostClassifier(verbose=0,random_state=42, **cat_params   )

    # Train the model
    cat_model.fit(X_train, y_train)

    # Predict probabilities
    y_prob = cat_model.predict_proba(X_val)[:, 1]  # Probabilities for the positive class (fraud)

    # Adjust predictions based on the threshold
    y_pred = (y_prob > threshold).astype(int)

    # Calculate metrics
    cm = confusion_matrix(y_val, y_pred)
    roc_auc = roc_auc_score(y_val, y_prob)  # Use probabilities to calculate ROC-AUC
    accuracy = accuracy_score(y_val, y_pred)
    precision = precision_score(y_val, y_pred)
    recall = recall_score(y_val, y_pred)
    f1 = f1_score(y_val, y_pred)
    f2 = fbeta_score(y_val, y_pred, beta=2)

    # Create a string with the parameters to show
    # If params_to_show is not provided, show all XGBoost parameters used
    if params_to_show is None:
        params_to_show = {'threshold': threshold}
        params_to_show.update(cat_params)  # Add dynamic XGBoost params to show

    # Crear una versión con abreviaturas
    params_with_abbreviations = {
        param_abbreviations.get(key, key): value for key, value in params_to_show.items()
    }

    # Build the parameter string dynamically
    #params_str = " ".join([f"{key}={value}" for key, value in params_with_abbreviations.items()])
    params_str =[f"{key}={value}" for key, value in params_with_abbreviations.items()]
    # Store the results in a DataFrame
    results_df = pd.DataFrame({
        'Model': ['Catboost'],
        'Description': [data_list[0]],
        'Parameter': [params_str],  # Show abbreviated parameters here
        'ROC-AUC': [roc_auc],
        'Accuracy': [accuracy],
        'Precision': [precision],
        'Recall': [recall],
        'F1 Score': [f1],
        'F2 Score': [f2],
        'Confusion Matrix': [cm]
    })

    # Ajustar para que las celdas no se trunquen
    pd.set_option('display.max_colwidth', None)

    # Mostrar el DataFrame con estilo respetando los saltos de línea
    results_df.style.set_properties(**{'white-space': 'pre-wrap'})

    return results_df
```

In [50]:
```python
# Parámetros para el XGBoost
valores_learning_rate = [ 0.01, 0.05, 0.1]
valores_n_estimators = [100, 150, 200, 250, 300]
valores_max_depth = [3, 5, 7]

total_results = []

# Iterar sobre los parámetros para hacer pruebas combinadas

for learning_rate in valores_learning_rate:
    for n_estimators in valores_n_estimators:
        for max_depth in valores_max_depth:
            # Ejecutar la función con diferentes combinaciones de hiperparámetros
            results = evaluate_catboost(
                OR_origin,
                #thresold = 0.5,
                n_estimators=n_estimators,
                learning_rate=learning_rate,
                max_depth=max_depth
            )
            total_results.append(results)

# Combinar todos los resultados en un único DataFrame para visualizarlo
total_results_df = pd.concat(total_results, ignore_index=True)
```

In [51]:
```python
total_results_df
```

Out[51]:

	Model	Description	Parameter	ROC-AUC	Accuracy	Precision	Recall	F1 Score	F2 Score	Confusion Matrix
0	Catboost	OR origin	[th=0.5, n_est=100, lr=0.01, md=3]	0.966223	0.999280	0.843373	0.714286	0.773481	0.736842	[[56851, 13], [28, 70]]
1	Catboost	OR origin	[th=0.5, n_est=100, lr=0.01, md=5]	0.980782	0.999350	0.850575	0.755102	0.800000	0.772443	[[56851, 13], [24, 74]]
2	Catboost	OR origin	[th=0.5, n_est=100, lr=0.01, md=7]	0.974358	0.999508	0.960526	0.744898	0.839080	0.779915	[[56861, 3], [25, 73]]
3	Catboost	OR origin	[th=0.5, n_est=150, lr=0.01, md=3]	0.967933	0.999333	0.848837	0.744898	0.793478	0.763598	[[56851, 13], [25, 73]]
4	Catboost	OR origin	[th=0.5, n_est=150, lr=0.01, md=5]	0.979097	0.999596	0.962963	0.795918	0.871508	0.824524	[[56861, 3], [20, 78]]
5	Catboost	OR origin	[th=0.5, n_est=150, lr=0.01, md=7]	0.974191	0.999614	0.963415	0.806122	0.877778	0.833333	[[56861, 3], [19, 79]]
6	Catboost	OR origin	[th=0.5, n_est=200, lr=0.01, md=3]	0.974775	0.999368	0.852273	0.765306	0.806452	0.781250	[[56851, 13], [23, 75]]
7	Catboost	OR origin	[th=0.5, n_est=200, lr=0.01, md=5]	0.977959	0.999614	0.963415	0.806122	0.877778	0.833333	[[56861, 3], [19, 79]]
8	Catboost	OR origin	[th=0.5, n_est=200, lr=0.01, md=7]	0.974749	0.999614	0.963415	0.806122	0.877778	0.833333	[[56861, 3], [19, 79]]
9	Catboost	OR origin	[th=0.5, n_est=250, lr=0.01, md=3]	0.976978	0.999403	0.855556	0.785714	0.819149	0.798755	[[56851, 13], [21, 77]]
10	Catboost	OR origin	[th=0.5, n_est=250, lr=0.01, md=5]	0.976264	0.999579	0.962500	0.785714	0.865169	0.815678	[[56861, 3], [21, 77]]
11	Catboost	OR origin	[th=0.5, n_est=250, lr=0.01, md=7]	0.974579	0.999596	0.951807	0.806122	0.872928	0.831579	[[56860, 4], [19, 79]]
12	Catboost	OR origin	[th=0.5, n_est=300, lr=0.01, md=3]	0.975439	0.999421	0.857143	0.795918	0.825397	0.807453	[[56851, 13], [20, 78]]
13	Catboost	OR origin	[th=0.5, n_est=300, lr=0.01, md=5]	0.974549	0.999579	0.962500	0.785714	0.865169	0.815678	[[56861, 3], [21, 77]]
14	Catboost	OR origin	[th=0.5, n_est=300, lr=0.01, md=7]	0.973757	0.999614	0.952381	0.816327	0.879121	0.840336	[[56860, 4], [18, 80]]
15	Catboost	OR origin	[th=0.5, n_est=100, lr=0.05, md=3]	0.974219	0.999508	0.897727	0.806122	0.849462	0.822917	[[56855, 9], [19, 79]]
16	Catboost	OR origin	[th=0.5, n_est=100, lr=0.05, md=5]	0.970847	0.999614	0.952381	0.816327	0.879121	0.840336	[[56860, 4], [18, 80]]
17	Catboost	OR origin	[th=0.5, n_est=100, lr=0.05, md=7]	0.975176	0.999579	0.951220	0.795918	0.866667	0.822785	[[56860, 4], [20, 78]]
18	Catboost	OR origin	[th=0.5, n_est=150, lr=0.05, md=3]	0.973948	0.999526	0.908046	0.806122	0.854054	0.824635	[[56856, 8], [19, 79]]
19	Catboost	OR origin	[th=0.5, n_est=150, lr=0.05, md=5]	0.973888	0.999579	0.930233	0.816327	0.869565	0.836820	[[56858, 6], [18, 80]]
20	Catboost	OR origin	[th=0.5, n_est=150, lr=0.05, md=7]	0.975540	0.999561	0.939759	0.795918	0.861878	0.821053	[[56859, 5], [20, 78]]
21	Catboost	OR origin	[th=0.5, n_est=200, lr=0.05, md=3]	0.971873	0.999526	0.908046	0.806122	0.854054	0.824635	[[56856, 8], [19, 79]]
22	Catboost	OR origin	[th=0.5, n_est=200, lr=0.05, md=5]	0.975000	0.999579	0.940476	0.806122	0.868132	0.829832	[[56859, 5], [19, 79]]
23	Catboost	OR origin	[th=0.5, n_est=200, lr=0.05, md=7]	0.975257	0.999579	0.940476	0.806122	0.868132	0.829832	[[56859, 5], [19, 79]]
24	Catboost	OR origin	[th=0.5, n_est=250, lr=0.05, md=3]	0.973511	0.999561	0.919540	0.816327	0.864865	0.835073	[[56857, 7], [18, 80]]
25	Catboost	OR origin	[th=0.5, n_est=250, lr=0.05, md=5]	0.975727	0.999579	0.940476	0.806122	0.868132	0.829832	[[56859, 5], [19, 79]]
26	Catboost	OR origin	[th=0.5, n_est=250, lr=0.05, md=7]	0.977370	0.999614	0.952381	0.816327	0.879121	0.840336	[[56860, 4], [18, 80]]
27	Catboost	OR origin	[th=0.5, n_est=300, lr=0.05, md=3]	0.975944	0.999579	0.920455	0.826531	0.870968	0.843750	[[56857, 7], [17, 81]]
28	Catboost	OR origin	[th=0.5, n_est=300, lr=0.05, md=5]	0.975663	0.999596	0.951807	0.806122	0.872928	0.831579	[[56860, 4], [19, 79]]
29	Catboost	OR origin	[th=0.5, n_est=300, lr=0.05, md=7]	0.978204	0.999596	0.941176	0.816327	0.874317	0.838574	[[56859, 5], [18, 80]]
30	Catboost	OR origin	[th=0.5, n_est=100, lr=0.1, md=3]	0.977833	0.999579	0.930233	0.816327	0.869565	0.836820	[[56858, 6], [18, 80]]
31	Catboost	OR origin	[th=0.5, n_est=100, lr=0.1, md=5]	0.975536	0.999649	0.953488	0.836735	0.891304	0.857741	[[56860, 4], [16, 82]]
32	Catboost	OR origin	[th=0.5, n_est=100, lr=0.1, md=7]	0.977616	0.999544	0.950000	0.775510	0.853933	0.805085	[[56860, 4], [22, 76]]
33	Catboost	OR origin	[th=0.5, n_est=150, lr=0.1, md=3]	0.978390	0.999579	0.930233	0.816327	0.869565	0.836820	[[56858, 6], [18, 80]]
34	Catboost	OR origin	[th=0.5, n_est=150, lr=0.1, md=5]	0.975121	0.999631	0.952941	0.826531	0.885246	0.849057	[[56860, 4], [17, 81]]
35	Catboost	OR origin	[th=0.5, n_est=150, lr=0.1, md=7]	0.978382	0.999561	0.962025	0.775510	0.858757	0.806794	[[56861, 3], [22, 76]]
36	Catboost	OR origin	[th=0.5, n_est=200, lr=0.1, md=3]	0.977572	0.999596	0.931034	0.826531	0.875676	0.845511	[[56858, 6], [17, 81]]
37	Catboost	OR origin	[th=0.5, n_est=200, lr=0.1, md=5]	0.977246	0.999614	0.952381	0.816327	0.879121	0.840336	[[56860, 4], [18, 80]]
38	Catboost	OR origin	[th=0.5, n_est=200, lr=0.1, md=7]	0.979530	0.999561	0.962025	0.775510	0.858757	0.806794	[[56861, 3], [22, 76]]
39	Catboost	OR origin	[th=0.5, n_est=250, lr=0.1, md=3]	0.978278	0.999579	0.920455	0.826531	0.870968	0.843750	[[56857, 7], [17, 81]]
40	Catboost	OR origin	[th=0.5, n_est=250, lr=0.1, md=5]	0.977892	0.999614	0.952381	0.816327	0.879121	0.840336	[[56860, 4], [18, 80]]
41	Catboost	OR origin	[th=0.5, n_est=250, lr=0.1, md=7]	0.980459	0.999579	0.962500	0.785714	0.865169	0.815678	[[56861, 3], [21, 77]]
42	Catboost	OR origin	[th=0.5, n_est=300, lr=0.1, md=3]	0.977500	0.999561	0.910112	0.826531	0.866310	0.841996	[[56856, 8], [17, 81]]
43	Catboost	OR origin	[th=0.5, n_est=300, lr=0.1, md=5]	0.980207	0.999614	0.952381	0.816327	0.879121	0.840336	[[56860, 4], [18, 80]]
44	Catboost	OR origin	[th=0.5, n_est=300, lr=0.1, md=7]	0.981271	0.999579	0.962500	0.785714	0.865169	0.815678	[[56861, 3], [21, 77]]

In [60]:
```python
total_results_df_sorted = total_results_df.sort_values(by='F2 Score', ascending=False).reset_index(drop=True)
```

In [61]:
```python
total_results_df_sorted
```

Out[61]:

	Model	Description	Parameter	ROC-AUC	Accuracy	Precision	Recall	F1 Score	F2 Score	Confusion Matrix
0	Catboost	OR origin	[th=0.5, n_est=100, lr=0.1, md=5]	0.975536	0.999649	0.953488	0.836735	0.891304	0.857741	[[56860, 4], [16, 82]]
1	Catboost	OR origin	[th=0.5, n_est=150, lr=0.1, md=5]	0.975121	0.999631	0.952941	0.826531	0.885246	0.849057	[[56860, 4], [17, 81]]
2	Catboost	OR origin	[th=0.5, n_est=200, lr=0.1, md=3]	0.977572	0.999596	0.931034	0.826531	0.875676	0.845511	[[56858, 6], [17, 81]]
3	Catboost	OR origin	[th=0.5, n_est=250, lr=0.1, md=3]	0.978278	0.999579	0.920455	0.826531	0.870968	0.843750	[[56857, 7], [17, 81]]
4	Catboost	OR origin	[th=0.5, n_est=300, lr=0.05, md=3]	0.975944	0.999579	0.920455	0.826531	0.870968	0.843750	[[56857, 7], [17, 81]]
5	Catboost	OR origin	[th=0.5, n_est=300, lr=0.1, md=3]	0.977500	0.999561	0.910112	0.826531	0.866310	0.841996	[[56856, 8], [17, 81]]
6	Catboost	OR origin	[th=0.5, n_est=250, lr=0.05, md=7]	0.977370	0.999614	0.952381	0.816327	0.879121	0.840336	[[56860, 4], [18, 80]]
7	Catboost	OR origin	[th=0.5, n_est=300, lr=0.01, md=7]	0.973757	0.999614	0.952381	0.816327	0.879121	0.840336	[[56860, 4], [18, 80]]
8	Catboost	OR origin	[th=0.5, n_est=200, lr=0.1, md=5]	0.977246	0.999614	0.952381	0.816327	0.879121	0.840336	[[56860, 4], [18, 80]]
9	Catboost	OR origin	[th=0.5, n_est=100, lr=0.05, md=5]	0.970847	0.999614	0.952381	0.816327	0.879121	0.840336	[[56860, 4], [18, 80]]
10	Catboost	OR origin	[th=0.5, n_est=250, lr=0.1, md=5]	0.977892	0.999614	0.952381	0.816327	0.879121	0.840336	[[56860, 4], [18, 80]]
11	Catboost	OR origin	[th=0.5, n_est=300, lr=0.1, md=5]	0.980207	0.999614	0.952381	0.816327	0.879121	0.840336	[[56860, 4], [18, 80]]
12	Catboost	OR origin	[th=0.5, n_est=300, lr=0.05, md=7]	0.978204	0.999596	0.941176	0.816327	0.874317	0.838574	[[56859, 5], [18, 80]]
13	Catboost	OR origin	[th=0.5, n_est=100, lr=0.1, md=3]	0.977833	0.999579	0.930233	0.816327	0.869565	0.836820	[[56858, 6], [18, 80]]
14	Catboost	OR origin	[th=0.5, n_est=150, lr=0.1, md=3]	0.978390	0.999579	0.930233	0.816327	0.869565	0.836820	[[56858, 6], [18, 80]]
15	Catboost	OR origin	[th=0.5, n_est=150, lr=0.05, md=5]	0.973888	0.999579	0.930233	0.816327	0.869565	0.836820	[[56858, 6], [18, 80]]
16	Catboost	OR origin	[th=0.5, n_est=250, lr=0.05, md=3]	0.973511	0.999561	0.919540	0.816327	0.864865	0.835073	[[56857, 7], [18, 80]]
17	Catboost	OR origin	[th=0.5, n_est=200, lr=0.01, md=7]	0.974749	0.999614	0.963415	0.806122	0.877778	0.833333	[[56861, 3], [19, 79]]
18	Catboost	OR origin	[th=0.5, n_est=200, lr=0.01, md=5]	0.977959	0.999614	0.963415	0.806122	0.877778	0.833333	[[56861, 3], [19, 79]]
19	Catboost	OR origin	[th=0.5, n_est=150, lr=0.01, md=7]	0.974191	0.999614	0.963415	0.806122	0.877778	0.833333	[[56861, 3], [19, 79]]
20	Catboost	OR origin	[th=0.5, n_est=300, lr=0.05, md=5]	0.975663	0.999596	0.951807	0.806122	0.872928	0.831579	[[56860, 4], [19, 79]]
21	Catboost	OR origin	[th=0.5, n_est=250, lr=0.01, md=7]	0.974579	0.999596	0.951807	0.806122	0.872928	0.831579	[[56860, 4], [19, 79]]
22	Catboost	OR origin	[th=0.5, n_est=250, lr=0.05, md=5]	0.975727	0.999579	0.940476	0.806122	0.868132	0.829832	[[56859, 5], [19, 79]]
23	Catboost	OR origin	[th=0.5, n_est=200, lr=0.05, md=5]	0.975000	0.999579	0.940476	0.806122	0.868132	0.829832	[[56859, 5], [19, 79]]
24	Catboost	OR origin	[th=0.5, n_est=200, lr=0.05, md=7]	0.975257	0.999579	0.940476	0.806122	0.868132	0.829832	[[56859, 5], [19, 79]]
25	Catboost	OR origin	[th=0.5, n_est=150, lr=0.05, md=3]	0.973948	0.999526	0.908046	0.806122	0.854054	0.824635	[[56856, 8], [19, 79]]
26	Catboost	OR origin	[th=0.5, n_est=200, lr=0.05, md=3]	0.971873	0.999526	0.908046	0.806122	0.854054	0.824635	[[56856, 8], [19, 79]]
27	Catboost	OR origin	[th=0.5, n_est=150, lr=0.01, md=5]	0.979097	0.999596	0.962963	0.795918	0.871508	0.824524	[[56861, 3], [20, 78]]
28	Catboost	OR origin	[th=0.5, n_est=100, lr=0.05, md=3]	0.974219	0.999508	0.897727	0.806122	0.849462	0.822917	[[56855, 9], [19, 79]]
29	Catboost	OR origin	[th=0.5, n_est=100, lr=0.05, md=7]	0.975176	0.999579	0.951220	0.795918	0.866667	0.822785	[[56860, 4], [20, 78]]
30	Catboost	OR origin	[th=0.5, n_est=150, lr=0.05, md=7]	0.975540	0.999561	0.939759	0.795918	0.861878	0.821053	[[56859, 5], [20, 78]]
31	Catboost	OR origin	[th=0.5, n_est=250, lr=0.1, md=7]	0.980459	0.999579	0.962500	0.785714	0.865169	0.815678	[[56861, 3], [21, 77]]
32	Catboost	OR origin	[th=0.5, n_est=300, lr=0.1, md=7]	0.981271	0.999579	0.962500	0.785714	0.865169	0.815678	[[56861, 3], [21, 77]]
33	Catboost	OR origin	[th=0.5, n_est=300, lr=0.01, md=5]	0.974549	0.999579	0.962500	0.785714	0.865169	0.815678	[[56861, 3], [21, 77]]
34	Catboost	OR origin	[th=0.5, n_est=250, lr=0.01, md=5]	0.976264	0.999579	0.962500	0.785714	0.865169	0.815678	[[56861, 3], [21, 77]]
35	Catboost	OR origin	[th=0.5, n_est=300, lr=0.01, md=3]	0.975439	0.999421	0.857143	0.795918	0.825397	0.807453	[[56851, 13], [20, 78]]
36	Catboost	OR origin	[th=0.5, n_est=150, lr=0.1, md=7]	0.978382	0.999561	0.962025	0.775510	0.858757	0.806794	[[56861, 3], [22, 76]]
37	Catboost	OR origin	[th=0.5, n_est=200, lr=0.1, md=7]	0.979530	0.999561	0.962025	0.775510	0.858757	0.806794	[[56861, 3], [22, 76]]
38	Catboost	OR origin	[th=0.5, n_est=100, lr=0.1, md=7]	0.977616	0.999544	0.950000	0.775510	0.853933	0.805085	[[56860, 4], [22, 76]]
39	Catboost	OR origin	[th=0.5, n_est=250, lr=0.01, md=3]	0.976978	0.999403	0.855556	0.785714	0.819149	0.798755	[[56851, 13], [21, 77]]
40	Catboost	OR origin	[th=0.5, n_est=200, lr=0.01, md=3]	0.974775	0.999368	0.852273	0.765306	0.806452	0.781250	[[56851, 13], [23, 75]]
41	Catboost	OR origin	[th=0.5, n_est=100, lr=0.01, md=7]	0.974358	0.999508	0.960526	0.744898	0.839080	0.779915	[[56861, 3], [25, 73]]
42	Catboost	OR origin	[th=0.5, n_est=100, lr=0.01, md=5]	0.980782	0.999350	0.850575	0.755102	0.800000	0.772443	[[56851, 13], [24, 74]]
43	Catboost	OR origin	[th=0.5, n_est=150, lr=0.01, md=3]	0.967933	0.999333	0.848837	0.744898	0.793478	0.763598	[[56851, 13], [25, 73]]
44	Catboost	OR origin	[th=0.5, n_est=100, lr=0.01, md=3]	0.966223	0.999280	0.843373	0.714286	0.773481	0.736842	[[56851, 13], [28, 70]]

In [62]:
```python
catboost_hyperparameters = total_results_df_sorted[total_results_df_sorted['F2 Score']>= .85].reset_index(drop=True)
```

In [63]:
```python
catboost_hyperparameters
```

Out[63]:

	Model	Description	Parameter	ROC-AUC	Accuracy	Precision	Recall	F1 Score	F2 Score	Confusion Matrix
0	Catboost	OR origin	[th=0.5, n_est=100, lr=0.1, md=5]	0.975536	0.999649	0.953488	0.836735	0.891304	0.857741	[[56860, 4], [16, 82]]

In [64]:
```python
catboost_hyperparameters.to_csv(r'C:\TFM\06_hyperparameter\catboost.csv', index=False)
```