In [75]:
```
# imports
import os
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import sklearn
import scipy.io
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import StratifiedKFold
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.svm import SVC
from xgboost import XGBClassifier
from catboost import CatBoostClassifier
from sklearn import metrics
from sklearn.metrics import roc_auc_score
from sklearn.metrics import RocCurveDisplay
from sklearn.metrics import roc_curve
```

In [76]:
```
# Sub 1
sub1Xtrain = scipy.io.loadmat('bonusSub1Xtrain.mat')
sub1Ytrain = scipy.io.loadmat('bonusSub1Ytrain.mat')
sub1Xtest = scipy.io.loadmat('bonusSub1Xtest.mat')
sub1Ytest = scipy.io.loadmat('bonusSub1Ytest.mat')
xtrain1 = sub1Xtrain['trainData']
#xtrain1 = pd.DataFrame(xtrain1)
ytrain1 = sub1Ytrain['results'].ravel()
#ytrain1 = pd.DataFrame(ytrain1)
xtest1 = sub1Xtest['trainData']
#xtest1 = pd.DataFrame(xtest1)
ytest1 = sub1Ytest['results'].ravel()
# Sub 2
sub2Xtrain = scipy.io.loadmat('bonusSub2Xtrain.mat')
sub2Ytrain = scipy.io.loadmat('bonusSub2Ytrain.mat')
sub2Xtest = scipy.io.loadmat('bonusSub2Xtest.mat')
sub2Ytest = scipy.io.loadmat('bonusSub2Ytest.mat')
xtrain2 = sub2Xtrain['trainData']
#xtrain2 = pd.DataFrame(xtrain2)
ytrain2 = sub2Ytrain['results'].ravel()
#ytrain2 = pd.DataFrame(ytrain2)
xtest2 = sub2Xtest['trainData']
#xtest2 = pd.DataFrame(xtest2)
ytest2 = sub2Ytest['results'].ravel()
# Sub 4
sub4Xtrain = scipy.io.loadmat('bonusSub4Xtrain.mat')
sub4Ytrain = scipy.io.loadmat('bonusSub4Ytrain.mat')
sub4Xtest = scipy.io.loadmat('bonusSub4Xtest.mat')
sub4Ytest = scipy.io.loadmat('bonusSub4Ytest.mat')
xtrain4 = sub4Xtrain['trainData']
#xtrain4 = pd.DataFrame(xtrain4)
ytrain4 = sub4Ytrain['results'].ravel()
#ytrain4 = pd.DataFrame(ytrain4)
xtest4 = sub4Xtest['trainData']
#xtest4 = pd.DataFrame(xtest4)
ytest4 = sub4Ytest['results'].ravel()
# Sub 5
sub5Xtrain = scipy.io.loadmat('bonusSub5Xtrain.mat')
```

```
sub5Ytrain = scipy.io.loadmat('bonusSub5Ytrain.mat')
sub5Xtest = scipy.io.loadmat('bonusSub5Xtest.mat')
sub5Ytest = scipy.io.loadmat('bonusSub5Ytest.mat')
xtrain5 = sub5Xtrain['trainData']
#xtrain5 = pd.DataFrame(xtrain5)
ytrain5 = sub5Ytrain['results'].ravel()
#ytrain5 = pd.DataFrame(ytrain5)
xtest5 = sub5Xtest['trainData']
#xtest5 = pd.DataFrame(xtest5)
ytest5 = sub5Ytest['results'].ravel()
```

In [77]:

```
def TestSetError(Yset, Yp):
    correct = 0
    n = len(Yset)
    for i in range(n):
        if(Yset[i] == Yp[i]):
            correct += 1
    accuracy=correct/len(Yset)
    return accuracy
```

In [78]:

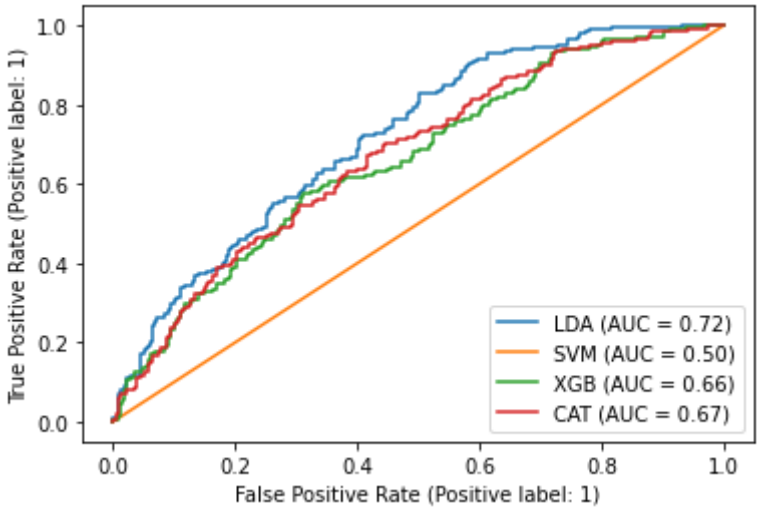
```
def evaluateModel(xtrain,ytrain,xtest,ytest):

    # LDA
    model = LinearDiscriminantAnalysis()
    cv = StratifiedKFold(n_splits=5, shuffle=False, random_state=None)
    # define grid
    grid = dict()
    grid['solver'] = ['svd', 'lsqr', 'eigen']
    # define search
    lda = GridSearchCV(model, grid, scoring='accuracy', cv=cv, n_jobs=-1)
    # perform the search
    results = lda.fit(xtrain, ytrain)
    # summarize
    print("LDA:")
    print('Mean Accuracy: %.3f' % results.best_score_)
    print('Config: %s' % results.best_params_)
    outputLDA = lda.predict(xtest)
    print("test accuracy =", TestSetError(ytest.ravel(), outputLDA))
    # ROC
    ax = plt.gca()
    lda_roc = RocCurveDisplay.from_estimator(lda, xtest, ytest, name = "LDA", ax

    # SVM
    model = SVC(probability=True)
    cv = StratifiedKFold(n_splits=5, shuffle=False, random_state=None)
    # define grid
    parameters = {'C': [1, 10], 'gamma': [0.001, 0.01, 1]}
    # define search
    svc = GridSearchCV(model, param_grid=parameters, scoring='accuracy', cv=cv,
    # perform the search
    results = svc.fit(xtrain, ytrain)
    # summarize
    print("SVM:")
    print('Mean Accuracy: %.3f' % results.best_score_)
    print('Config: %s' % results.best_params_)
    outputSVM = svc.predict(xtest)
    print("test accuracy =", TestSetError(ytest.ravel(), outputSVM))
```

```
# ROC
svm_roc = RocCurveDisplay.from_estimator(svc, xtest, ytest, name = "SVM", ax

# XGB
param_test1 = {
'max_depth':range(3,11,2),
'min_child_weight':range(1,7,2)
}
xgb = GridSearchCV(
        estimator = XGBClassifier(
                        learning_rate = 0.1,
                        n_estimators = 140,
                        gamma = 0,
                        subsample = 0.8,
                        colsample_bytree = 0.8,
                        objective = 'binary:logistic',
                        scale_pos_weight = 1,
                        seed=42),
        param_grid = param_test1,
        scoring='accuracy',
        n_jobs=-1,
        cv=StratifiedKFold(n_splits=5, shuffle=False, random_state=None)
        )

results = xgb.fit(xtrain,ytrain)
print("XGB:")
print(results.best_params_,)
print('Mean Accuracy: %.3f' %  results.best_score_)
outputXGB = xgb.predict(xtest)
print("test accuracy =", TestSetError(ytest.ravel(), outputXGB))
# ROC
xgb_roc = RocCurveDisplay.from_estimator(xgb, xtest, ytest, name = "XGB", ax

# CAT boost
cat_param_test1 = {
'depth':range(1,10,2),
'min_data_in_leaf': range(1,10,2)
}
cat_gsearch1 = GridSearchCV(
            estimator = CatBoostClassifier(
                            learning_rate = 0.1,
                            iterations = 200,
                            loss_function = 'Logloss', # try crossEntropy la
                            border_count = 32,
                            eval_metric="AUC",
                            verbose=False,
                            random_seed=42),
            param_grid = cat_param_test1,
            scoring='accuracy',
            n_jobs=-1,
            # use shuffle to verity validity of hyperparameters
            cv= StratifiedKFold(n_splits=5, shuffle=False, random_state=None
            )

cat_gsearch1.fit(xtrain,ytrain)
print("CATboost:")
print(cat_gsearch1.best_params_)
print('Mean Accuracy: %.3f' % cat_gsearch1.best_score_)
outputCAT = cat_gsearch1.predict(xtest)
print("test accuracy =", TestSetError(ytest.ravel(), outputCAT))
```

```
    # ROC
    cat_roc = RocCurveDisplay.from_estimator(cat_gsearch1, xtest, ytest, name =

    plt.show()
```

In [79]:
```
evaluateModel(xtrain1,ytrain1,xtest1,ytest1)
```

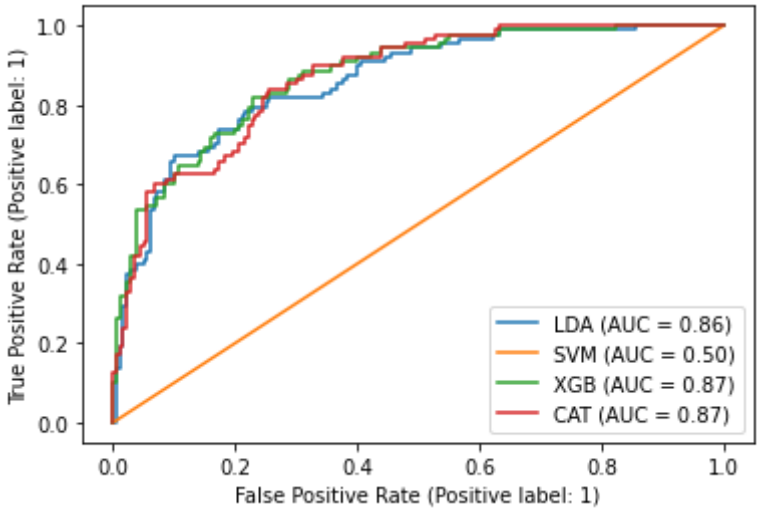
```
LDA:
Mean Accuracy: 0.663
Config: {'solver': 'svd'}
test accuracy = 0.6437768240343348
SVM:
Mean Accuracy: 0.682
Config: {'C': 1, 'gamma': 0.001}
test accuracy = 0.5987124463519313
XGB:
{'max_depth': 7, 'min_child_weight': 1}
Mean Accuracy: 0.656
test accuracy = 0.6330472103004292
CATboost:
{'depth': 9, 'min_data_in_leaf': 1}
Mean Accuracy: 0.661
test accuracy = 0.6223175965665236
```



In [80]:
```
evaluateModel(xtrain2,ytrain2,xtest2,ytest2)
```

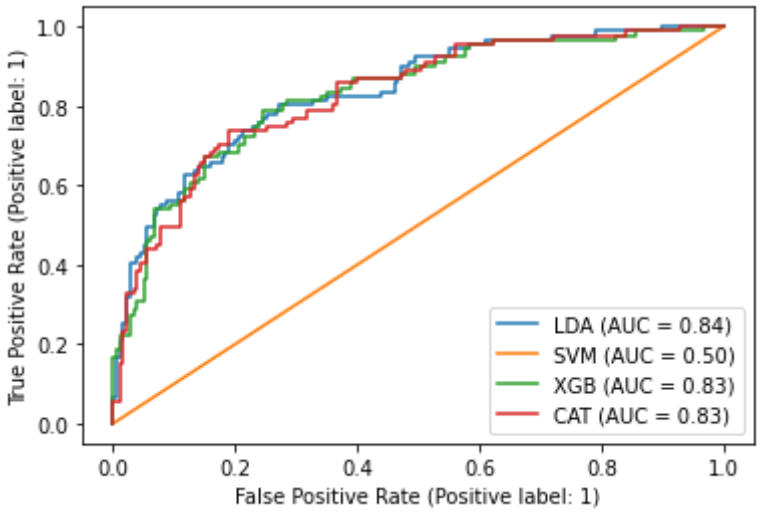
```
LDA:
Mean Accuracy: 0.768
Config: {'solver': 'svd'}
test accuracy = 0.7686567164179104
SVM:
Mean Accuracy: 0.734
Config: {'C': 1, 'gamma': 0.001}
test accuracy = 0.6716417910447762
XGB:
{'max_depth': 5, 'min_child_weight': 1}
Mean Accuracy: 0.778
test accuracy = 0.8097014925373134
CATboost:
{'depth': 1, 'min_data_in_leaf': 1}
```

```
Mean Accuracy: 0.783
test accuracy = 0.7873134328358209
```



In [81]:
```
evaluateModel(xtrain4,ytrain4,xtest4,ytest4)
```

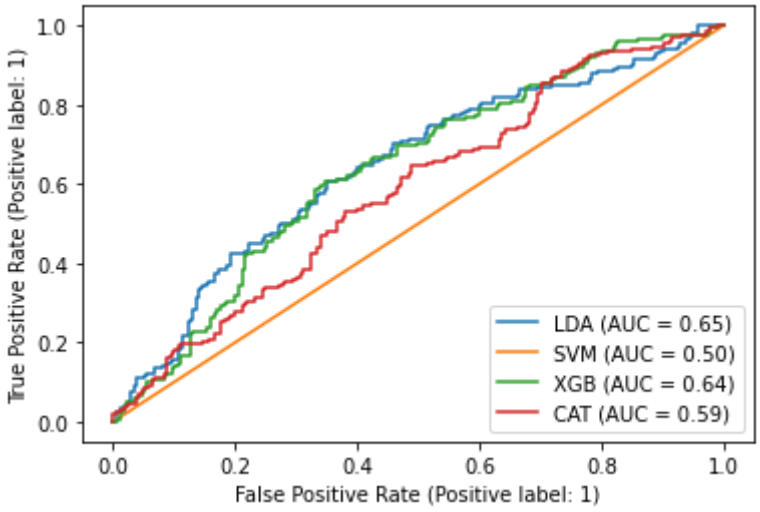
```
LDA:
Mean Accuracy: 0.825
Config: {'solver': 'svd'}
test accuracy = 0.7896678966789668
SVM:
Mean Accuracy: 0.714
Config: {'C': 1, 'gamma': 0.001}
test accuracy = 0.6642066420664207
XGB:
{'max_depth': 9, 'min_child_weight': 1}
Mean Accuracy: 0.812
test accuracy = 0.7785977859778598
CATboost:
{'depth': 1, 'min_data_in_leaf': 1}
Mean Accuracy: 0.822
test accuracy = 0.7638376383763837
```



In [82]:
```
evaluateModel(xtrain5,ytrain5,xtest5,ytest5)
```

```
LDA:
Mean Accuracy: 0.737
```

```
Config: {'solver': 'svd'}
test accuracy = 0.6356968215158925
SVM:
Mean Accuracy: 0.706
Config: {'C': 1, 'gamma': 0.001}
test accuracy = 0.5794621026894865
XGB:
{'max_depth': 3, 'min_child_weight': 5}
Mean Accuracy: 0.720
test accuracy = 0.5941320293398533
CATboost:
{'depth': 7, 'min_data_in_leaf': 1}
Mean Accuracy: 0.719
test accuracy = 0.5916870415647921
```



In [ ]: