In [1]:

```
import pandas as pd
import numpy as np
import pyod
import matplotlib.pyplot as pyplot
import matplotlib.pyplot as plt
import sklearn.metrics as metrics
from sklearn import preprocessing
from pyod.utils.data import evaluate_print

#importando do arquivo modificado com os prints dos métodos unsupervided
import xgbod as XGBOD

from datetime import datetime

np.set_printoptions(precision=3)
```

In [2]:

```
testing_set = pd.read_csv('../Unbalanced_Samples/Sample_60K.csv',sep=',',header=0)
training_set = pd.read_csv('../Balanced_Samples/Sample_100K.csv',sep=',',header=0)
```

```
c:\python37\lib\site-packages\IPython\core\interactiveshell.py:3044: DtypeWa
rning: Columns (21,22) have mixed types. Specify dtype option on import or s
et low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

In [3]:

```
#transforma dados categóricos em números
for f in testing_set.columns:
    if testing_set[f].dtype=='object':
        label = preprocessing.LabelEncoder()
        label.fit(list(testing_set[f].values))
        testing_set[f] = label.transform(list(testing_set[f].values))

testingSet = testing_set.values
np.random.shuffle(testingSet)

testingSet = testingSet.astype(float)

for i in range (6*10**4-1, 0, -1):
    for j in range(0, 84):
        testingSet[i, j] = float(testingSet[i, j])
        if (np.isinf(testingSet[i, j]) or np.isnan(testingSet[i, j])):
            testingSet = np.delete(testingSet, i, axis=0)

y_test = testingSet[:, 84].astype(int)
```

In [4]:

```
#transforma dados categóricos em números
for f in training_set.columns:
    if training_set[f].dtype=='object':
        label = preprocessing.LabelEncoder()
        label.fit(list(training_set[f].values))
        training_set[f] = label.transform(list(training_set[f].values))

trainingSet = training_set.values
np.random.shuffle(trainingSet)

trainingSet = trainingSet.astype(float)

for i in range (10*10**4-1, 0, -1):
    for j in range(0, 84):
        trainingSet[i, j] = float(trainingSet[i, j])
        if (np.isinf(trainingSet[i, j]) or np.isnan(trainingSet[i, j])):
            trainingSet = np.delete(trainingSet, i, axis=0)

y_train = trainingSet[:, 84].astype(int)
```

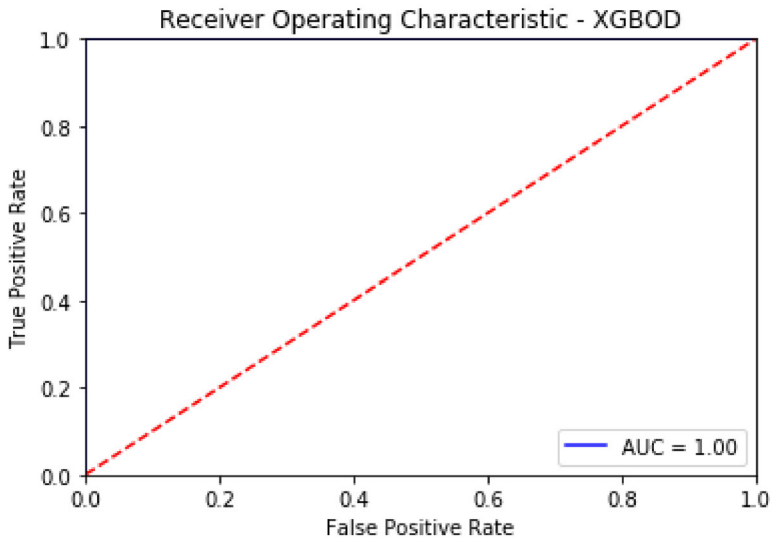
In [5]:

```
outliers_fraction = 0.5
def testMethod(clf, clf_name):
    print("Started fitting: ", datetime.now(), "\n")
    clf.fit(trainingSet, y_train)
    print("Endend fitting: ", datetime.now(), "\n")

    ###########################################################################

    print("Started predicting: ", datetime.now(), "\n")
    y_test_pred = clf.predict(testingSet)  # outlier labels (0 or 1)
    y_test_scores = clf.decision_function(testingSet)  # outlier scores
    print("Ended predicting: ", datetime.now(), "\n")

    ###########################################################################

    print("\nOn Test Data - "+clf_name+":")
    truePositive = 0
    trueNegative = 0
    falsePositive = 0
    falseNegative = 0

    for i in range(y_test.size):
        if(y_test[i] == 1 and y_test_pred[i] == 1):
            truePositive = truePositive+1
        elif (y_test[i] == 0 and y_test_pred[i] == 0):
            trueNegative = trueNegative+1
        elif (y_test[i] == 0 and y_test_pred[i] == 1):
            falsePositive = falsePositive+1
        else:
            falseNegative = falseNegative+1


    print()
    print("              CONFUSION MATRIX")
    print()
    print("                  Actual")
    print()
    print("Predicted     ", truePositive, " | ", falsePositive)
    print("              ",falseNegative, " | ", trueNegative,"\n")

    evaluate_print(clf_name, y_test, y_test_scores)

    ###########################################################################

    fpr, tpr, threshold = metrics.roc_curve(y_test, y_test_scores)
    roc_auc = metrics.auc(fpr, tpr)

    import matplotlib.pyplot as plt
    plt.title('Receiver Operating Characteristic - '+clf_name)
    plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
    plt.legend(loc = 'lower right')
    plt.plot([0, 1], [0, 1],'r--')
    plt.xlim([0, 1])
    plt.ylim([0, 1])
    plt.ylabel('True Positive Rate')
    plt.xlabel('False Positive Rate')
    plt.show()
```

In [6]:

```
testMethod(XGBOD.XGBOD(contamination= outliers_fraction), "XGBOD")
```

```
XGBOD ROC:1.0, precision @ rank n:1.0
```



In [7]:

```
from xgboost import XGBClassifier
from xgboost import plot_importance
from xgboost import plot_tree
```

In [8]:

```
model_name = 'XGBOOST'
model = XGBClassifier()
model.fit(trainingSet, y_train) # matriz(sem label de respostas), vetor(label com respostas
```

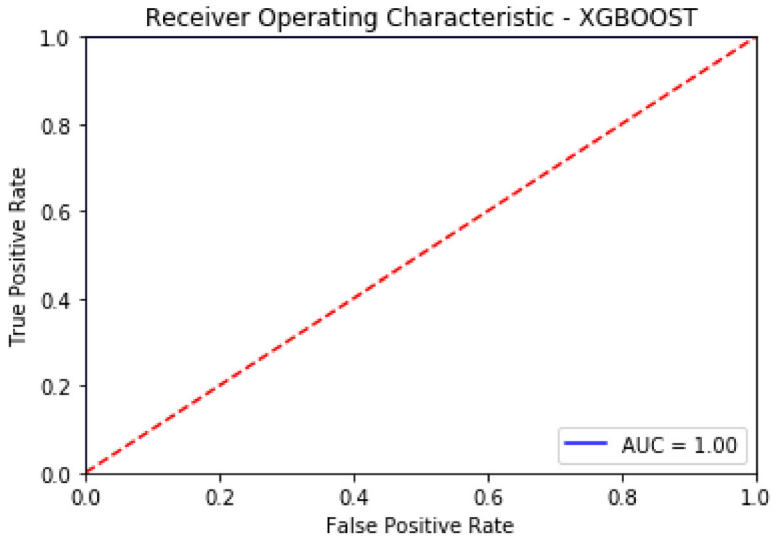
Out[8]:

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0,
              learning_rate=0.1, max_delta_step=0, max_depth=3,
              min_child_weight=1, missing=None, n_estimators=100, n_jobs=1,
              nthread=None, objective='binary:logistic', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
              silent=None, subsample=1, verbosity=1)
```

In [9]:

```
y_pred = model.predict(testingSet)
predictions = [round(value) for value in y_pred]

print("\nOn Test Data - "+model_name+":")
truePositive = 0
trueNegative = 0
falsePositive = 0
falseNegative = 0

for i in range(y_test.size):
    if(y_test[i] == 1 and y_pred[i] == 1):
        truePositive = truePositive+1
    elif (y_test[i] == 0 and y_pred[i] == 0):
        trueNegative = trueNegative+1
    elif (y_test[i] == 0 and y_pred[i] == 1):
        falsePositive = falsePositive+1
    else:
        falseNegative = falseNegative+1
print()
print("              CONFUSION MATRIX")
print()
print("                  Actual")
print()
print("Predicted     ", truePositive, " | ", falsePositive)
print("              ",falseNegative, " | ", trueNegative,"\n")

fpr, tpr, threshold = metrics.roc_curve(y_test, predictions)
roc_auc = metrics.auc(fpr, tpr)

plt.title('Receiver Operating Characteristic - '+model_name)
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```

```
On Test Data - XGBOOST:

              CONFUSION MATRIX

                  Actual

Predicted      10256  |  0
               0  |  49744
```


Receiver Operating Characteristic - XGBOOST
True Positive Rate
False Positive Rate
AUC = 1.00
0.0
0.2
0.4
0.6
0.8
1.0

In [ ]: