decision_function

Average anomaly score of X of the base classifiers. The anomaly score of an input sample is computed as the mean anomaly score of the trees in the forest. The measure of normality of an observation given a tree is the depth of the leaf containing this observation, which is equivalent to the number of splittings required to isolate this point. In case of several observations n_left in the leaf, the average path length of a n_left samples isolation tree is added. Parameters: X : array-like or sparse matrix, shape (n_samples, n_features) The input samples. Internally, it will be converted to dtype=np.float32 and if a sparse matrix is provided to a sparse csr_matrix. Returns:
scores : array, shape (n_samples,) The anomaly score of the input samples. The lower, the more abnormal. Negative scores represent outliers, positive scores represent inliers.

predict

Predict if a particular sample is an outlier or not. Parameters: X : array-like or sparse matrix, shape (n_samples, n_features) The input samples. Internally, it will be converted to dtype=np.float32 and if a sparse matrix is provided to a sparse csr_matrix. Returns:
is_inlier : array, shape (n_samples,) For each observation, tells whether or not (+1 or -1) it should be considered as an inlier according to the fitted model.

score_samples

Opposite of the anomaly score defined in the original paper. The anomaly score of an input sample is computed as the mean anomaly score of the trees in the forest. The measure of normality of an observation given a tree is the depth of the leaf containing this observation, which is equivalent to the number of splittings required to isolate this point. In case of several observations n_left in the leaf, the average path length of a n_left samples isolation tree is added. Parameters: X : array-like or sparse matrix, shape (n_samples, n_features) The input samples. Returns:
scores : array, shape (n_samples,) The anomaly score of the input samples. The lower, the more abnormal.

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from time import time
from sklearn.ensemble import IsolationForest
from sklearn.svm import OneClassSVM
from sklearn.neighbors import LocalOutlierFactor
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_auc_score
from matplotlib.mlab import frange
import statistics
from sklearn.metrics import confusion_matrix
from sklearn.metrics import precision_score, recall_score
from sklearn.metrics import f1_score
from sklearn.metrics import roc_curve
from imblearn.metrics import specificity_score
from imblearn.metrics import sensitivity_score
```

```
Using TensorFlow backend.
```

Chargement du jeu de données Shuttle

Shuttle contient 10 attributs dont le dernier est la classe à expliquer. Il y a 46464 observations dont 878 anormales. Avec le dernier attribut, nous avons les classes "o" pour les anomalies et "n" pour les observations normales.

In [2]:

```
data_brut_Shuttle = pd.read_csv('/Users/thesard/Doctorat/These 
2018/ISEP/Developpements/EspaceIntelliJ/LearningAllInPython/Le
arning_IHM_Features/datasets/imported_datasets/shuttle-unsuper
vised-ad_2019-06-14 17:33:02.493755.csv', header=None, index_c
ol=None)
X_Shuttle = data_brut_Shuttle[[0, 1, 2, 3, 4, 5, 6, 7, 8]]
y_brut_Shuttle = data_brut_Shuttle[[9]]
data_brut_Shuttle.describe()
```

Out[2]:

	0	1	2	3	
count	46464.000000	46464.000000	46464.000000	46464.000000	4646
mean	44.775482	-0.712509	84.834108	0.279291	3
std	8.753404	44.218007	8.731175	37.317203	1
min	27.000000	-4475.000000	21.000000	-3939.000000	-18
25%	37.000000	0.000000	79.000000	0.000000	3
50%	43.000000	0.000000	83.000000	0.000000	4
75%	49.000000	0.000000	87.000000	0.000000	4
max	123.000000	1963.000000	149.000000	3830.000000	43

Transformation de la classe à expliquer

L'objectif ici est de remplacer les "o" par -1 et les "n" par 1 afin de faire les matrices de confusion avec la fonction dédiée de scikit-learn.

In [3]:

```
y_transform_Shuttle = y_brut_Shuttle
y_transform_Shuttle = np.where(y_transform_Shuttle=='o',-1,1)
#y_transform
```

Shuttle

Exécution de Isolation Forest sur le jeux de données Shuttle en faisant varier n_estimators

In [4]:

```
func_IF_Shuttle = IsolationForest(behaviour="new")
func_IF_Shuttle.fit(X_Shuttle)
y_pred_IF_Shuttle = func_IF_Shuttle.predict(X_Shuttle)
print("Prediction")
print(y_pred_IF_Shuttle)
y_SS_IF_Shuttle = func_IF_Shuttle.score_samples(X_Shuttle)
print("Score samples")
print(y_SS_IF_Shuttle)
y_DF_IF_Shuttle = func_IF_Shuttle.decision_function(X_Shuttle)
print("Decision Function")
print(y_DF_IF_Shuttle)
```

```
/Users/thesard/anaconda3/lib/python3.7/site-packag
es/sklearn/ensemble/iforest.py:213: FutureWarning:
default contamination parameter 0.1 will change in
version 0.22 to "auto". This will change the predi
ct method behavior.
  FutureWarning)

Prediction
[-1 -1 -1 ...  1  1  1]
Score samples
[-0.65455448 -0.64676144 -0.66125837 ... -0.403324
11 -0.38680677
 -0.44059728]
Decision Function
[-0.18122541 -0.17343237 -0.18792929 ...  0.070004
96  0.08652231
  0.03273179]
```

Matrice de confusion

notre tn = tp de sklearn

notre fp = fn de sklearn

notre fn = fp de sklearn

notre tp = tn de sklearn

In [5]:

```
ttn, tfp, tfn, ttp = confusion_matrix(y_transform_Shuttle, y_p
red_IF_Shuttle).ravel()
tn = ttp
fp = tfn
fn = tfp
tp = ttn
print("True negative === "+str(tn))
print("False negative === "+str(fn))
print("False positive === "+str(fp))
print("True positive === "+str(tp))
```

```
True negative === 41805
False negative === 12
False positive === 3781
True positive === 866
```

Specificity

- Notre Spécificité = Specificity de imblearn = tp/(tp+fn) ==> Rappel des données anormales (negatives pour sklearn)

In [6]:

```
#Specificity own
specificityown_IF_Shuttle = tp/(tp+fn)
print("Specificity own === "+str(specificityown_IF_Shuttle))
# Specificity
specificity_IF_Shuttle = specificity_score(y_transform_Shuttle
, y_pred_IF_Shuttle)
print("Specificity === "+str(specificity_IF_Shuttle))
```

```
Specificity own === 0.9863325740318907
Specificity === 0.9863325740318907
```

Recall

- Notre rappel = Sensitivity de imblearn = Recall de sklearn = tn/(tn+fp) ==> Rappel des données normales (positive pour sklearn)

In [7]:

```
# Recall Own
recallown_IF_Shuttle = tn/(tn+fp)
print("Recall Own === "+str(recallown_IF_Shuttle))
# Recall
recall_IF_Shuttle = recall_score(y_transform_Shuttle, y_pred_I
F_Shuttle)
print("Recall === "+str(recall_IF_Shuttle))
# Sensitivity
sensitivity_IF_Shuttle = sensitivity_score(y_transform_Shuttle
, y_pred_IF_Shuttle)
print("Sensitivity === "+str(sensitivity_IF_Shuttle))
```

```
Recall Own === 0.917057868643882
Recall === 0.917057868643882
Sensitivity === 0.917057868643882
```

Precision

- Notre précision = precision de sklearn = tn/(tn+fn) ===> Précision des données normales (positive pour sklearn)

In [8]:

```
# Precision Own N
precisionown_n_IF_Shuttle = tn/(tn+fn)
print("Normal Precision Own === "+str(precisionown_n_IF_Shuttl
e))
# Precision Own P
precisionown_p_IF_Shuttle = tp/(tp+fp)
print("       Anormal Precision Own === "+str(precisionown_p_I
F_Shuttle))
# Precision
precision_IF_Shuttle = precision_score(y_transform_Shuttle, y_
pred_IF_Shuttle)
print("Precision === "+str(precision_IF_Shuttle))
```

```
Normal Precision Own === 0.9997130353683908
       Anormal Precision Own === 0.186356789326447
17
Precision === 0.9997130353683908
```

ROC CURVE

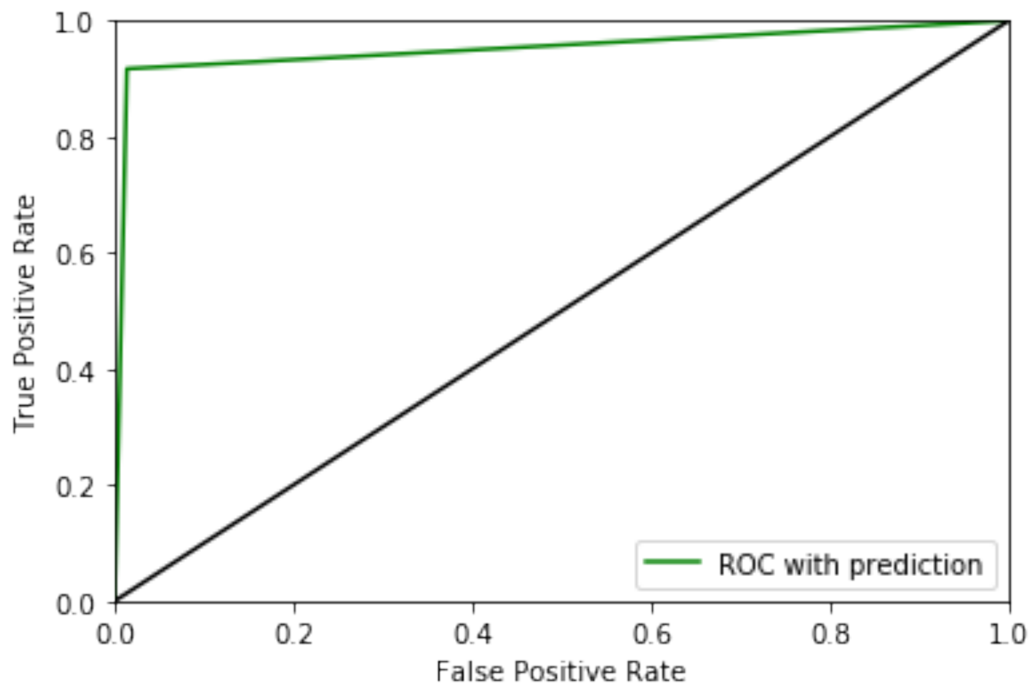
- TPR (abscisse) vs FPR (Ordonnée)
- TPR = Rappel : Taux de données normales bien détectées
- FPR = 1-Spécificité = fn/(tp+fn) : Taux de données normales mal classées
- fpr, tpr, thresholds = roc_curve(y_transform_Shuttle, y_pred_IF_Shuttle)
    - Détermine fpr et tpr en fonction des seuils dans thresholds
    - tpr[i] = taux de données normales bien classées dont le score est supérieur ou égale à thresholds[i]
    - fpr[i] = taux de données normales mal classées dont le score est supérieur ou égale à thresholds[i]
    - Pour les données de prédictions (résultat étant la classe de chaque observation 1(pour normales) ou -1 (pour anormales), thresholds = [2 1 -1]. Ainsi, les classes:
        - = 2 ==> Aucune d'où tpr=fpr=0,
        - = 1 ==> les classes normales d'où rappel et 1-spécificité
        - = -1 ==> toutes les classes d'où tpr=fpr=1

In [9]:

```
# ROC Curve
fpr_pred, tpr_pred, thresholds_pred = roc_curve(y_transform_Sh
uttle, y_pred_IF_Shuttle)
print("ROC Curve with Prediction")
print("fpr ")
print(fpr_pred)
print("tpr ")
print(tpr_pred)
print("thresholds ")
print(thresholds_pred)
plt.plot(fpr_pred, tpr_pred, "g-", label="ROC with prediction"
)
plt.plot([0, 1], [0, 1], 'k-')
plt.axis([0, 1, 0, 1])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc="lower right")

plt.show()
```

```
ROC Curve with Prediction
fpr
[0.         0.01366743 1.        ]
tpr
[0.         0.91705787 1.        ]
thresholds
[ 2  1 -1]
```

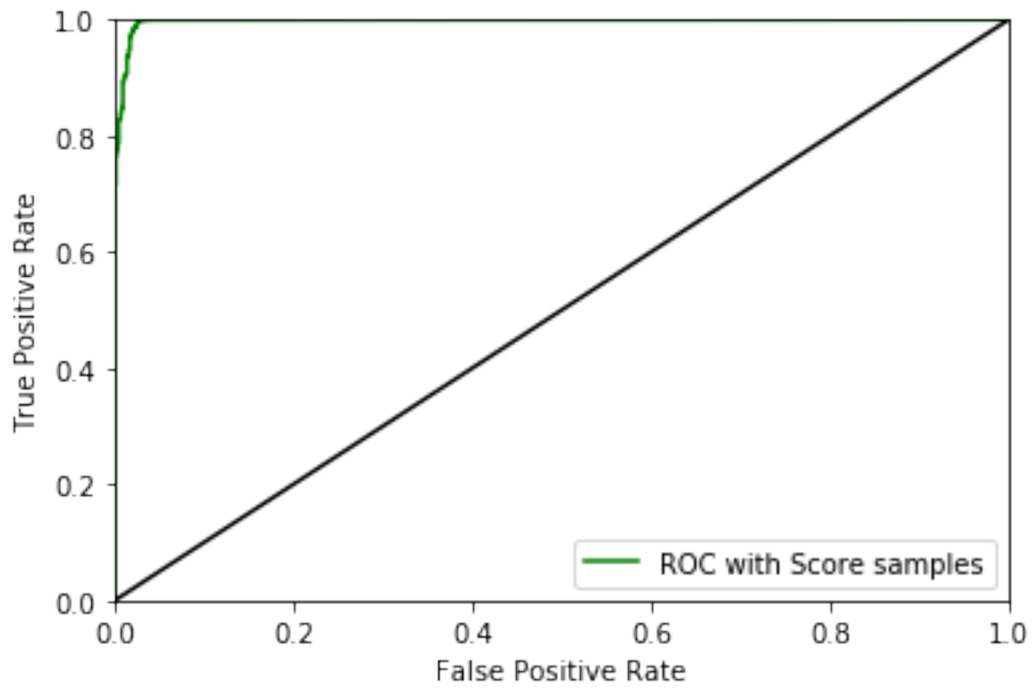


In [11]:

```
# ROC Curve
fpr_SS, tpr_SS, thresholds_SS = roc_curve(y_transform_Shuttle,
y_SS_IF_Shuttle)
print(len(y_SS_IF_Shuttle))
print("ROC Curve with Score samples")
print("fpr ")
print(fpr_SS)
print(len(fpr_SS))
print("tpr ")
print(tpr_SS)
print(len(tpr_SS))
print("thresholds ")
print(thresholds_SS)
print(len(thresholds_SS))
plt.plot(fpr_SS, tpr_SS, "g-", label="ROC with Score samples")
plt.plot([0, 1], [0, 1], 'k-')
plt.axis([0, 1, 0, 1])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc="lower right")

plt.show()
```

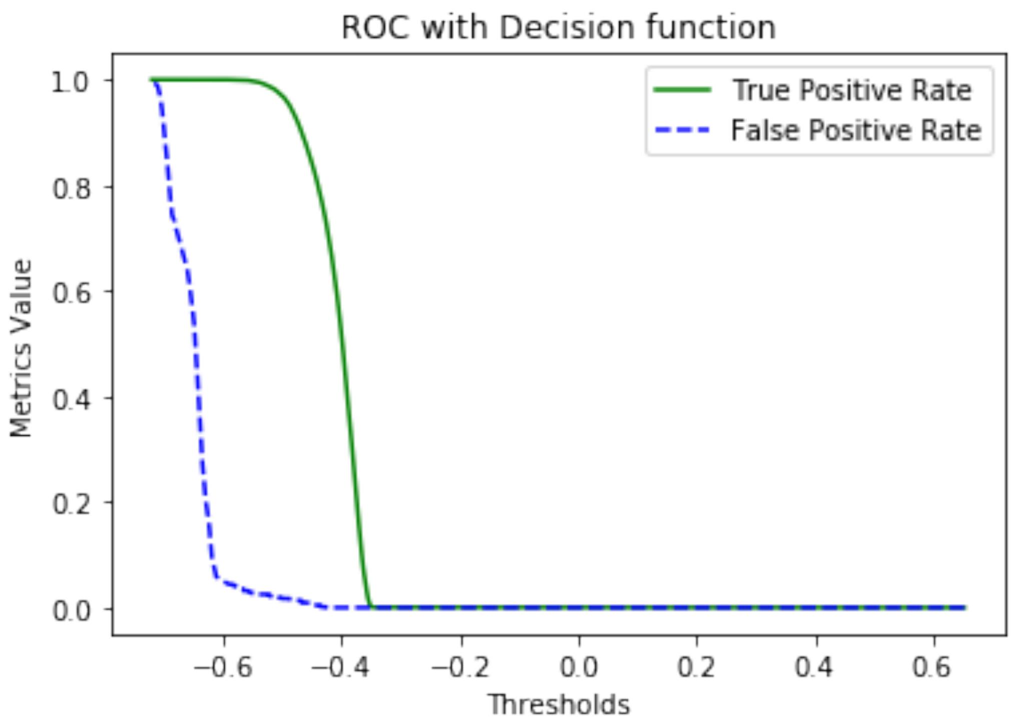
```
46464
ROC Curve with Score samples
fpr
[0.         0.         0.         ... 0.98519362 0
.98747153 1.        ]
5638
tpr
[0.00000000e+00 2.19365595e-05 1.09682797e-04 ...
1.00000000e+00
 1.00000000e+00 1.00000000e+00]
5638
thresholds
[ 0.65355517 -0.34644483 -0.3469929  ... -0.710674
45 -0.71088521
 -0.72115897]
5638
```



In [12]:

```
# Evolution de fpr et tpr en fonction des seuils pour les scor
es
plt.plot(thresholds_SS, tpr_SS, "g-", label="True Positive Rat
e")
plt.plot(thresholds_SS, fpr_SS, "b--", label="False Positive R
ate")
plt.title("ROC with Decision function")
#plt.axis([0, 1, 0, 1])
plt.xlabel('Thresholds')
plt.ylabel('Metrics Value')
plt.legend(loc="best")

plt.show()
```

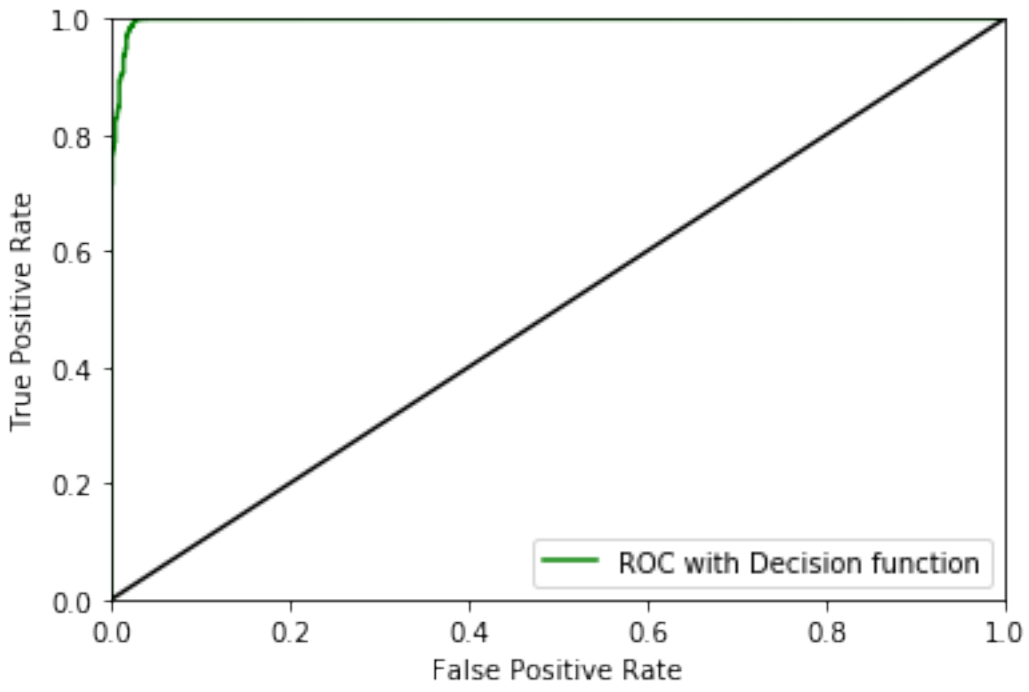


In [13]:

```
# ROC Curve
fpr_DF, tpr_DF, thresholds_DF = roc_curve(y_transform_Shuttle,
y_DF_IF_Shuttle)
print("ROC Curve with Decision function")
print("fpr ")
print(fpr_DF)
print(len(fpr_DF))
print("tpr ")
print(tpr_DF)
print(len(tpr_DF))
print("thresholds ")
print(thresholds_DF)
print(len(thresholds_DF))
plt.plot(fpr_DF, tpr_DF, "g-", label="ROC with Decision functi
on")
plt.plot([0, 1], [0, 1], 'k-')
plt.axis([0, 1, 0, 1])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc="lower right")

plt.show()
```

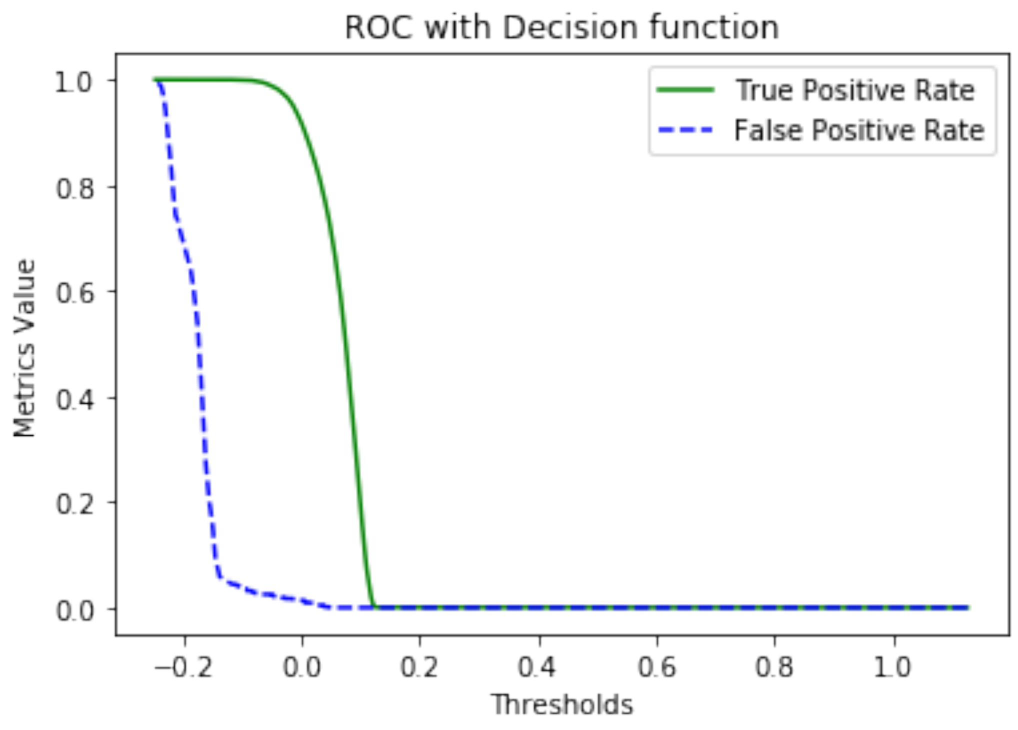
```
ROC Curve with Decision function
fpr
[0.         0.         0.         ... 0.98519362 0
.98747153 1.        ]
5638
tpr
[0.00000000e+00 2.19365595e-05 1.09682797e-04 ...
1.00000000e+00
 1.00000000e+00 1.00000000e+00]
5638
thresholds
[ 1.12688424  0.12688424  0.12633617 ... -0.237345
38 -0.23755614
 -0.2478299 ]
5638
```



In [14]:

```
# Evolution de fpr et tpr en fonction des seuils pour la fonct
ion de décision
plt.plot(thresholds_DF, tpr_DF, "g-", label="True Positive Rat
e")
plt.plot(thresholds_DF, fpr_DF, "b--", label="False Positive R
ate")
plt.title("ROC with Decision function")
#plt.axis([0, 1, 0, 1])
plt.xlabel('Thresholds')
plt.ylabel('Metrics Value')
plt.legend(loc="best")

plt.show()
```



ROC AUC

L'aire sous la courbe ROC est plus grande calculée avec les scores et les function de décision qu'avec les classes prédites.

In [15]:

```
# Avec les classes prédites
auc_pred_IF_Shuttle = roc_auc_score(y_transform_Shuttle, y_pre
d_IF_Shuttle)
auc_pred_IF_Shuttle
```

Out[15]:

```
0.9516952213378863
```

In [16]:

```
# Avec les decision_function
auc_DF_IF_Shuttle = roc_auc_score(y_transform_Shuttle, y_DF_IF
_Shuttle)
auc_DF_IF_Shuttle
```

Out[16]:

```
0.9973161943677109
```

In [17]:

```
# Avec les scores
auc_SS_IF_Shuttle = roc_auc_score(y_transform_Shuttle, y_SS_IF
_Shuttle)
auc_SS_IF_Shuttle
```

Out[17]:

```
0.9973161943677109
```

Conclusion