Supplementary File 1

In [1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import classification_report
from sklearn.metrics import average_precision_score
from sklearn.metrics import precision_recall_curve
from sklearn.utils import resample
from imblearn.over_sampling import SMOTE
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt
from imblearn import under_sampling
```

In [2]:
```
#reading the data and set target variable
df = pd.read_csv("C:/Users/silve/Desktop/Project_3/mydata.csv")
df.head()
```

Out[2]:

	Maternal_age	Gestational_diabetes	PROM	Gestational_age	Birthweight	Induction_method	Parity	ANC_visits	Refered_for_delivery	Family_plann
0	37	0	0	40	5.90	1	0	1	0	
1	29	0	1	40	2.50	1	1	2	1	
2	30	0	1	40	3.25	1	1	2	0	
3	26	0	0	40	3.10	1	0	2	0	
4	28	0	0	38	2.40	1	0	2	0	

In [3]:
```
df.shape
```

Out[3]:
```
(7716, 17)
```

In [4]:
```
X=df.iloc[:,0:16]
y=df.iloc[:,16]
```

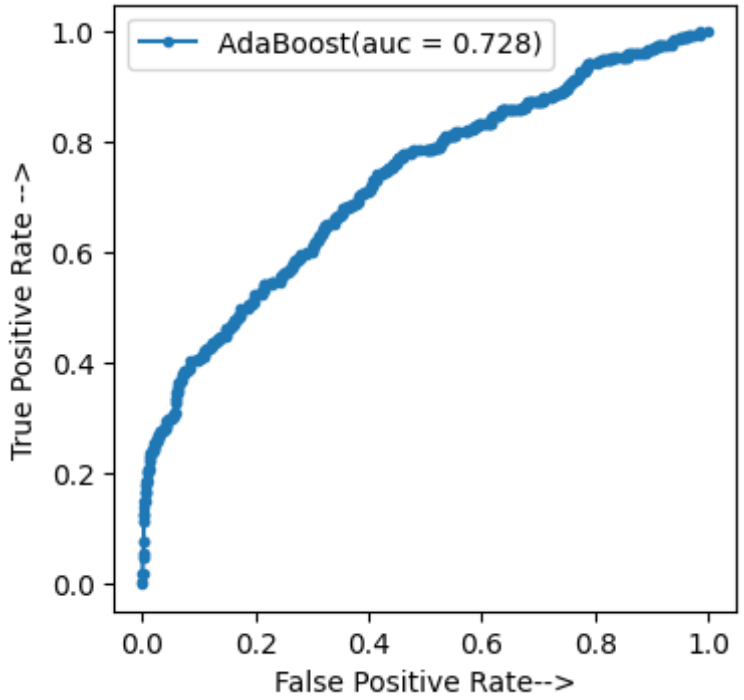
In [5]:
```
#Splitting the dataset into "Training" and "testing set (for internal validation)"
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.3,
                                                    random_state=42)
```

In [6]:
```
#Obtaining the proportion or prevalence of Low-Apgar score
from collections import Counter
counter = Counter(df['Apgar_score'])
print(counter)
```

```
Counter({0: 6983, 1: 733})
```

In [7]:
```
#Building the AdaBoost Model
from sklearn.ensemble import AdaBoostClassifier
from sklearn import datasets

# Create adaboost classifer object
abc = AdaBoostClassifier(n_estimators=150,
                         learning_rate=1)
# Train Adaboost Classifer
model = abc.fit(X_train, y_train)

#Predict the response for test dataset
y_pred = model.predict(X_test)

###Obtaining AUROC metrics
y_pred_roc = model.decision_function(X_test)
Adbst_fpr, Adbst_tpr, threshold = roc_curve(y_test, y_pred_roc)
auc_Adbst = auc(Adbst_fpr, Adbst_tpr)

plt.figure(figsize=(4,4), dpi = 100)
plt.plot(Adbst_fpr, Adbst_tpr, marker='.',label='AdaBoost(auc = %0.3f)' % auc_Adbst)

plt.xlabel('False Positive Rate-->')
plt.ylabel('True Positive Rate -->')

plt.legend()
```

```
plt.show()
```



In [8]:
```
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics
```

In [9]:
```
# Obtaining AdaBoost's performance metrics

print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.9144708423326134
              precision    recall  f1-score   support

           0       0.92      0.99      0.95      2090
           1       0.75      0.18      0.29       225
```

```
    accuracy                           0.91      2315
   macro avg       0.84      0.59      0.62      2315
weighted avg       0.90      0.91      0.89      2315
```

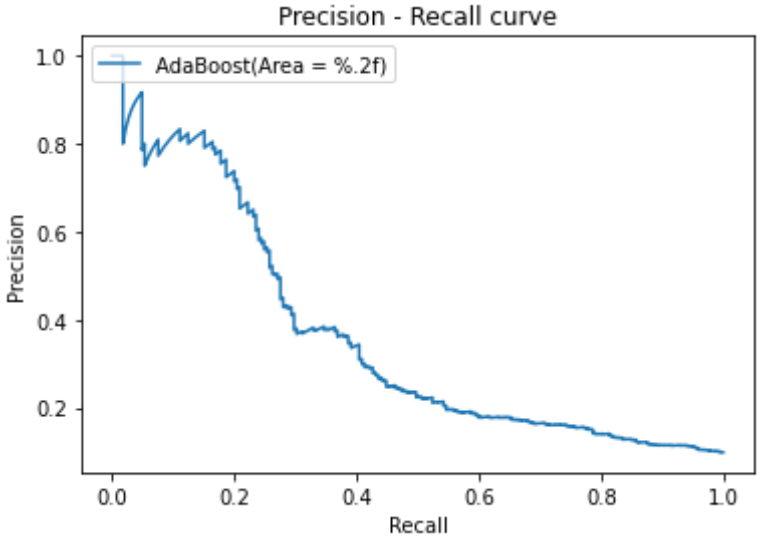
In [10]:
```
confusion_matrix(y_test, y_pred)
```

Out[10]:
```
array([[2077,   13],
       [ 185,   40]], dtype=int64)
```

In [11]:
```
#Obtaining PR-Curve for AdaBoost Classifier

y_pred_PR_curve = model.predict_proba(X_test)
y_pred_PR_curve
#print (type(y_pred_PR_curve))
y_pred_PR_curve = pd.DataFrame(y_pred_PR_curve)
#print(type(y_pred_PR_curve))
y_pred_PR_curve = y_pred_PR_curve.iloc[:,1]
y_pred_PR_curve

precision, recall, thresholds = precision_recall_curve(y_test, y_pred_PR_curve)
```

In [12]:
```
plt.plot(recall, precision, label = "AdaBoost(Area = %.2f)" )
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.legend(loc = "upper left")
plt.title("Precision - Recall curve")
```

Out[12]:
```
Text(0.5, 1.0, 'Precision - Recall curve')
```



In [13]:
```
#Performing SMOTE resampling Prior to AdaBoost Classifier
from sklearn.metrics import average_precision_score
from sklearn.metrics import precision_recall_curve
from sklearn.utils import resample
from imblearn.over_sampling import SMOTE
```

In [14]:
```
smote = SMOTE()
```

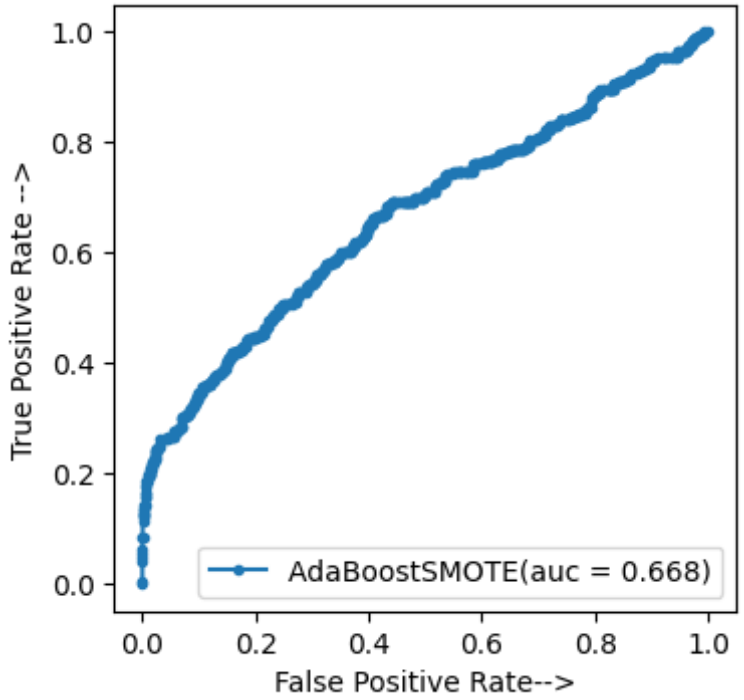
In [15]:
```
X_train_smote, y_train_smote = smote.fit_sample(X_train, y_train)
```

In [16]:
```
# Train Adaboost Classifer
model = abc.fit(X_train_smote, y_train_smote)

#Predict the response for test dataset
y_pred = model.predict(X_test)

#Performance metrics for AdaBoost after SMOTE resampling
print(classification_report(y_test, y_pred))
confusion_matrix(y_test, y_pred)
```

```
              precision    recall  f1-score   support

           0       0.93      0.81      0.87      2090
           1       0.20      0.44      0.28       225

    accuracy                           0.78      2315
   macro avg       0.57      0.63      0.57      2315
weighted avg       0.86      0.78      0.81      2315
```

Out[16]:
```
array([[1700,  390],
       [ 125,  100]], dtype=int64)
```

In [17]:
```
#Obtaining AUROC metrics for AdaBoost_SMOTE
y_pred_roc = model.decision_function(X_test)
AdbstSM_fpr, AdbstSM_tpr, threshold = roc_curve(y_test, y_pred_roc)
auc_AdbstSM = auc(AdbstSM_fpr, AdbstSM_tpr)
plt.figure(figsize=(4,4), dpi = 100)
plt.plot(AdbstSM_fpr, AdbstSM_tpr, marker='.',label='AdaBoostSMOTE(auc = %0.3f)' % auc_AdbstSM)
plt.xlabel('False Positive Rate-->')
plt.ylabel('True Positive Rate -->')

plt.legend()

plt.show()
```

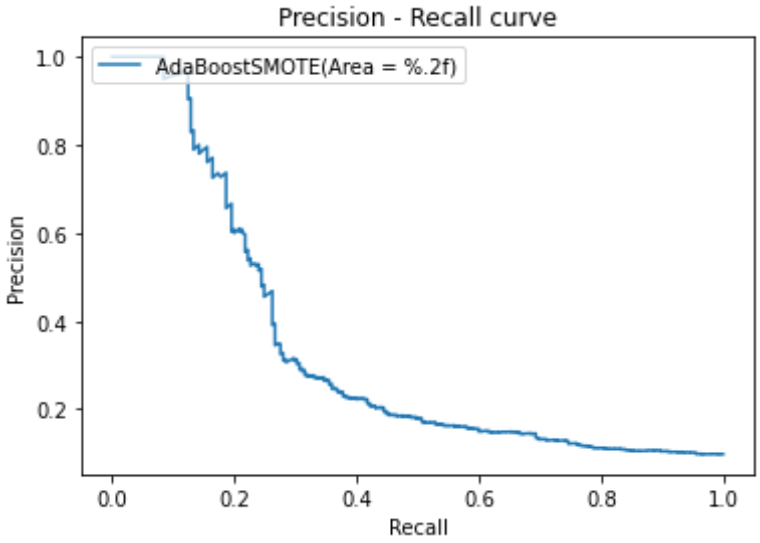


In [18]:

```
#Obtaining PR-curve for AdaBoost_SMOTE

y_pred_PR_curve = model.predict_proba(X_test)
y_pred_PR_curve
print (type(y_pred_PR_curve))
y_pred_PR_curve = pd.DataFrame(y_pred_PR_curve)
print(type(y_pred_PR_curve))
y_pred_PR_curve = y_pred_PR_curve.iloc[:,1]
y_pred_PR_curve

precision, recall, thresholds = precision_recall_curve(y_test, y_pred_PR_curve)

plt.plot(recall, precision, label = "AdaBoostSMOTE(Area = %.2f)" )
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.legend(loc = "upper left")
plt.title("Precision - Recall curve")
```

```
<class 'numpy.ndarray'>
<class 'pandas.core.frame.DataFrame'>
```

Out[18]:
```
Text(0.5, 1.0, 'Precision - Recall curve')
```



In [19]:
```
#Performing Random UnderSampling with AdaBoost Classifier
from imblearn.under_sampling import RandomUnderSampler
rus = RandomUnderSampler(random_state=0)
X_train_rus, y_train_rus = rus.fit_resample(X_train, y_train)
```

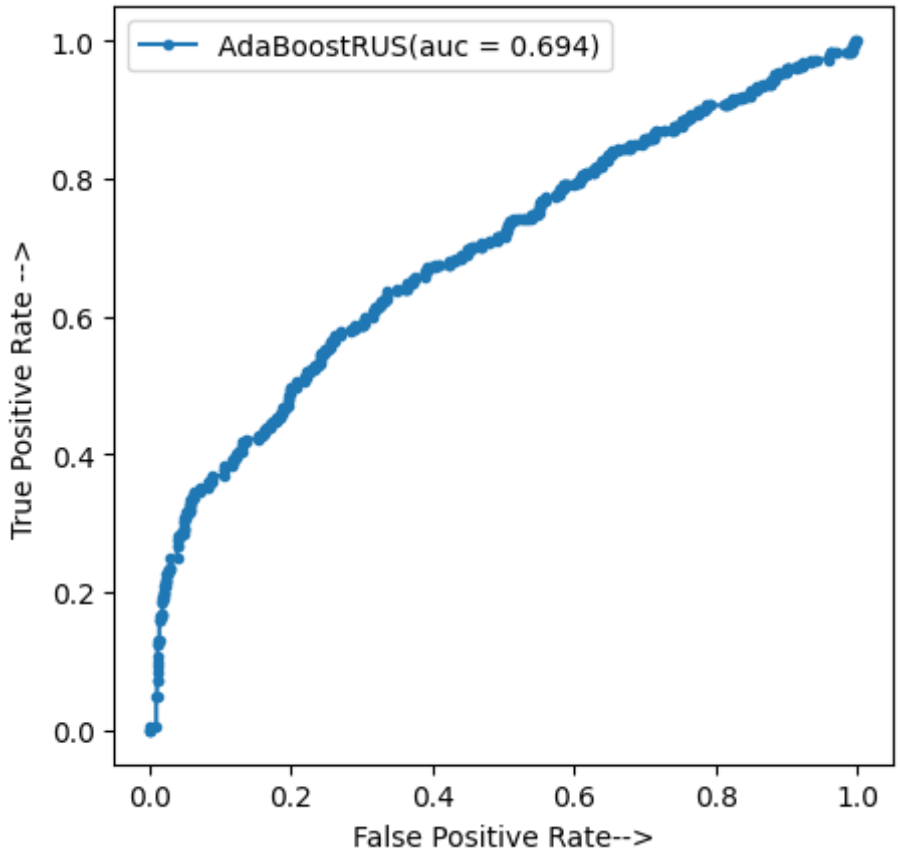
In [20]:
```
model = abc.fit(X_train_rus, y_train_rus)

#Predict the response for test dataset
y_pred = model.predict(X_test)


#Obtaining performance metrics for AdaBoost_RUS
print(classification_report(y_test, y_pred))
confusion_matrix(y_test, y_pred)
```

```
              precision    recall  f1-score   support

           0       0.94      0.76      0.84      2090
```

```
           1       0.19      0.54      0.29       225

    accuracy                           0.74      2315
   macro avg       0.57      0.65      0.56      2315
weighted avg       0.87      0.74      0.79      2315
```

Out[20]:
```
array([[1588,  502],
       [ 104,  121]], dtype=int64)
```

In [21]:
```
#Obtaining AUROC value for AdaBoost_RUS
y_pred_roc = model.decision_function(X_test)
AdbstRUS_fpr, AdbstRUS_tpr, threshold = roc_curve(y_test, y_pred_roc)
auc_AdbstRUS = auc(AdbstRUS_fpr, AdbstRUS_tpr)
plt.figure(figsize=(5,5), dpi = 100)
plt.plot(AdbstRUS_fpr, AdbstRUS_tpr, marker='.',label='AdaBoostRUS(auc = %0.3f)' % auc_AdbstRUS)
plt.xlabel('False Positive Rate-->')
plt.ylabel('True Positive Rate -->')

plt.legend()

plt.show()
```



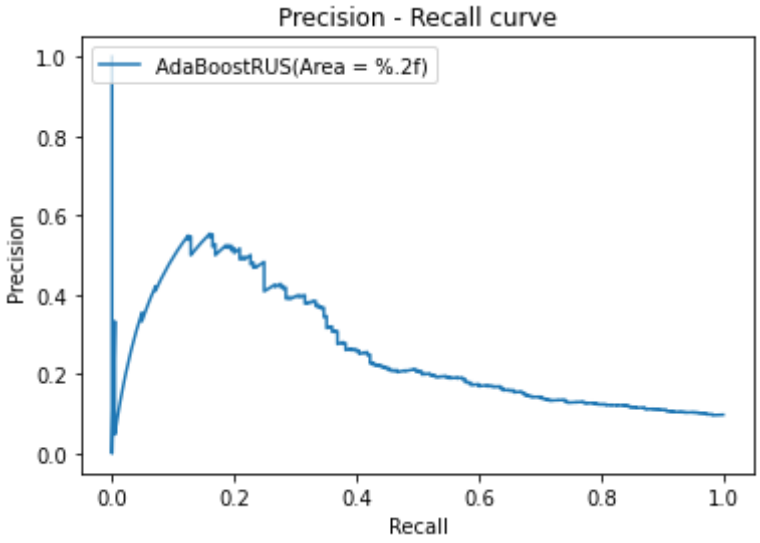
In [26]:

```
#Obtaining PR-curve for AdaBoost with RUS
y_pred_PR_curve = model.predict_proba(X_test)
y_pred_PR_curve
print (type(y_pred_PR_curve))
y_pred_PR_curve = pd.DataFrame(y_pred_PR_curve)
print(type(y_pred_PR_curve))
y_pred_PR_curve = y_pred_PR_curve.iloc[:,1]
y_pred_PR_curve

precision, recall, thresholds = precision_recall_curve(y_test, y_pred_PR_curve)


plt.plot(recall, precision, label = "AdaBoostRUS(Area = %.2f)" )
```

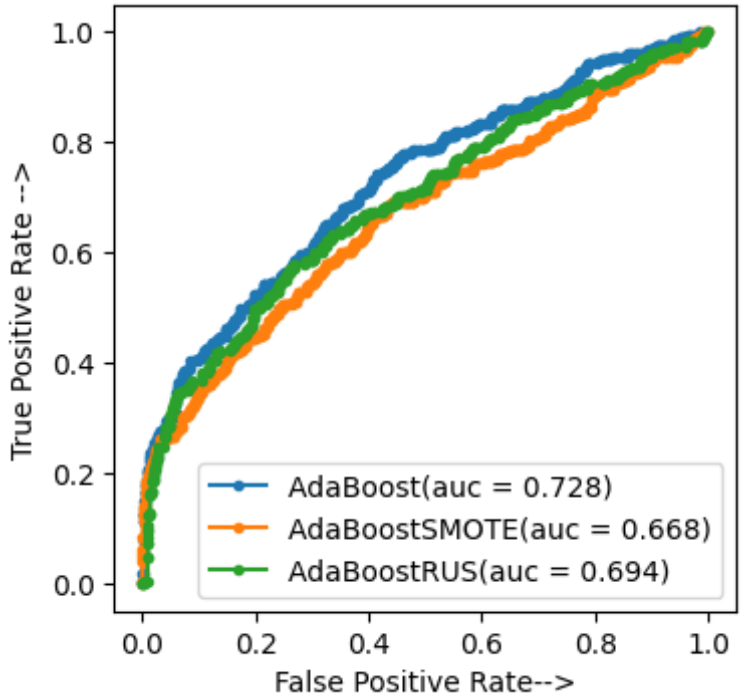
```
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.legend(loc = "upper left")
plt.title("Precision - Recall curve")
```

```
<class 'numpy.ndarray'>
<class 'pandas.core.frame.DataFrame'>
```

Out[26]:
```
Text(0.5, 1.0, 'Precision - Recall curve')
```



In [25]:
```
#Obtaining a Combined AUROC metrics for AdaBoost Classifier

plt.figure(figsize=(4,4), dpi = 100)
plt.plot(Adbst_fpr, Adbst_tpr, marker='.',label='AdaBoost(auc = %0.3f)' % auc_Adbst)
plt.plot(AdbstSM_fpr, AdbstSM_tpr, marker='.',label='AdaBoostSMOTE(auc = %0.3f)' % auc_AdbstSM)
plt.plot(AdbstRUS_fpr, AdbstRUS_tpr, marker='.',label='AdaBoostRUS(auc = %0.3f)' % auc_AdbstRUS)
plt.xlabel('False Positive Rate-->')
plt.ylabel('True Positive Rate -->')

plt.legend()

plt.show()
```



In [27]:
```
                                    # Gradient Boosting Classifier#
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import roc_auc_score
import matplotlib.pylab as plt
```

In [28]:
```
model_gbm = GradientBoostingClassifier(random_state=42)
model_gbm.fit(X_train, y_train)
y_pred_gb = model_gbm.predict(X_test)
```

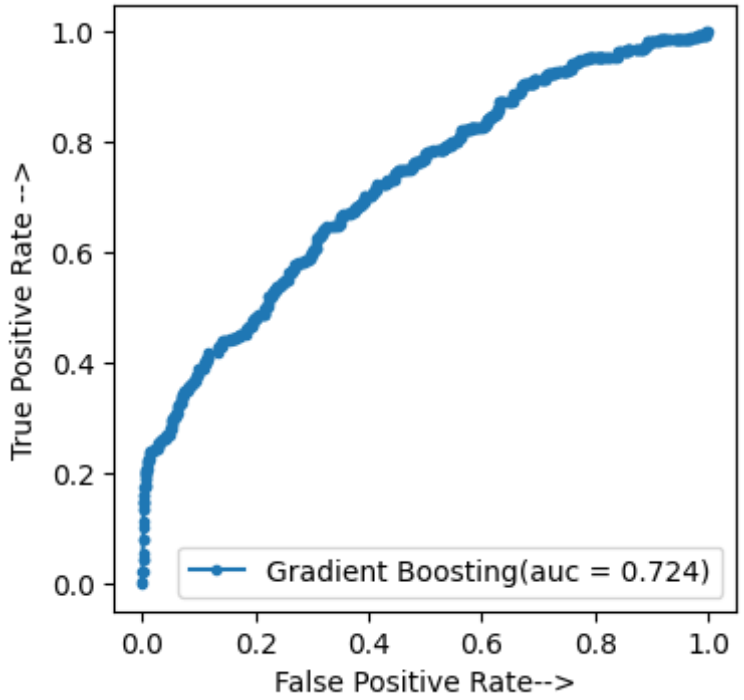
In [31]:
```
#Obtaining performance metrics for baseline GB classifier
print(classification_report(y_test, y_pred_gb))
confusion_matrix(y_test, y_pred_gb)
```

```
              precision    recall  f1-score   support
```

```
              0       0.92      0.99      0.96      2090
              1       0.80      0.19      0.31       225

    accuracy                            0.92      2315
   macro avg       0.86      0.59      0.63      2315
weighted avg       0.91      0.92      0.89      2315
```

Out[31]:
```
array([[2079,   11],
       [ 182,   43]], dtype=int64)
```

In [29]:
```
#Obtaining AUROC metrics for baseline GB classifier
y_pred_roc = model_gbm.decision_function(X_test)
gbm_fpr, gbm_tpr, threshold = roc_curve(y_test, y_pred_roc)
auc_gbm = auc(gbm_fpr, gbm_tpr)
plt.figure(figsize=(4,4), dpi = 100)
plt.plot(gbm_fpr, gbm_tpr, marker='.',label='Gradient Boosting(auc = %0.3f)' % auc_gbm)
plt.xlabel('False Positive Rate-->')
plt.ylabel('True Positive Rate -->')

plt.legend()

plt.show()
```

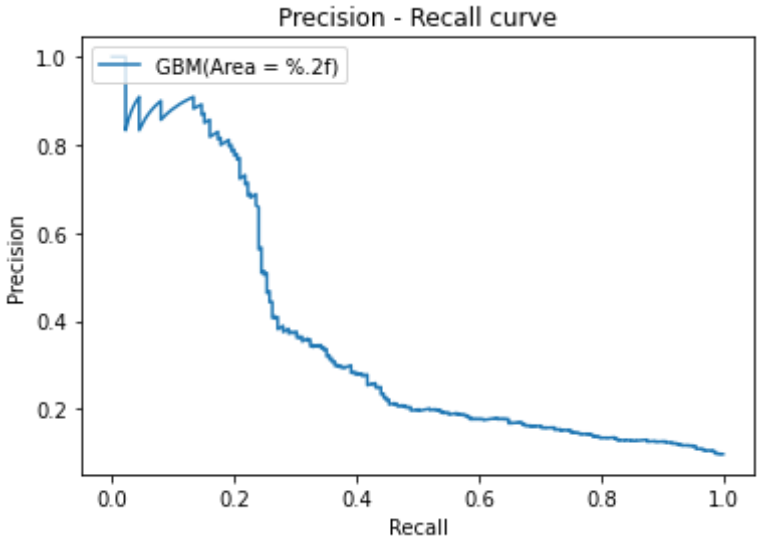


In [33]:

```
#Obtainig  PR-Curve for Baseline GB classifier
y_pred_PR_curve = model_gbm.predict_proba(X_test)
y_pred_PR_curve
print (type(y_pred_PR_curve))
y_pred_PR_curve = pd.DataFrame(y_pred_PR_curve)
print(type(y_pred_PR_curve))
y_pred_PR_curve = y_pred_PR_curve.iloc[:,1]
y_pred_PR_curve

precision, recall, thresholds = precision_recall_curve(y_test, y_pred_PR_curve)


plt.plot(recall, precision, label = "GBM(Area = %.2f)" )
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.legend(loc = "upper left")
plt.title("Precision - Recall curve")
```

```
<class 'numpy.ndarray'>
<class 'pandas.core.frame.DataFrame'>
```

Out[33]: Text(0.5, 1.0, 'Precision - Recall curve')



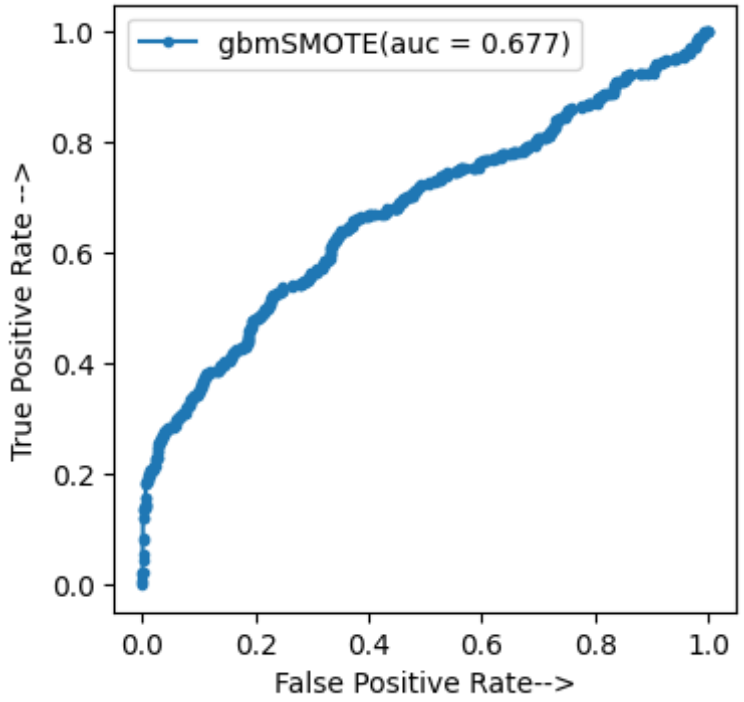
In [34]:
```
#Performing SMOTE with GBM
model_gbm = GradientBoostingClassifier(random_state=42)
model_gbm.fit(X_train_smote, y_train_smote)
y_pred_gb_SM = model_gbm.predict(X_test)
```

In [36]:
```
#Obtaining  performance metrics for GB with SMOTE
print(classification_report(y_test, y_pred_gb_SM))
confusion_matrix(y_test, y_pred_gb_SM)
```

```
              precision    recall  f1-score   support

           0       0.93      0.86      0.89      2090
           1       0.23      0.40      0.29       225

    accuracy                           0.81      2315
   macro avg       0.58      0.63      0.59      2315
weighted avg       0.86      0.81      0.83      2315
```

Out[36]:
```
array([[1790,  300],
       [ 136,   89]], dtype=int64)
```

In [38]:
```
#Obtaining AUROC metrics for GM with SMOTE
y_pred_roc = model_gbm.decision_function(X_test)
gbSM_fpr, gbSM_tpr, threshold = roc_curve(y_test, y_pred_roc)
auc_gbSM = auc(gbSM_fpr, gbSM_tpr)
plt.figure(figsize=(4,4), dpi = 100)
plt.plot(gbSM_fpr, gbSM_tpr, marker='.',label='gbmSMOTE(auc = %0.3f)' % auc_gbSM)
plt.xlabel('False Positive Rate-->')
plt.ylabel('True Positive Rate -->')

plt.legend()

plt.show()
```

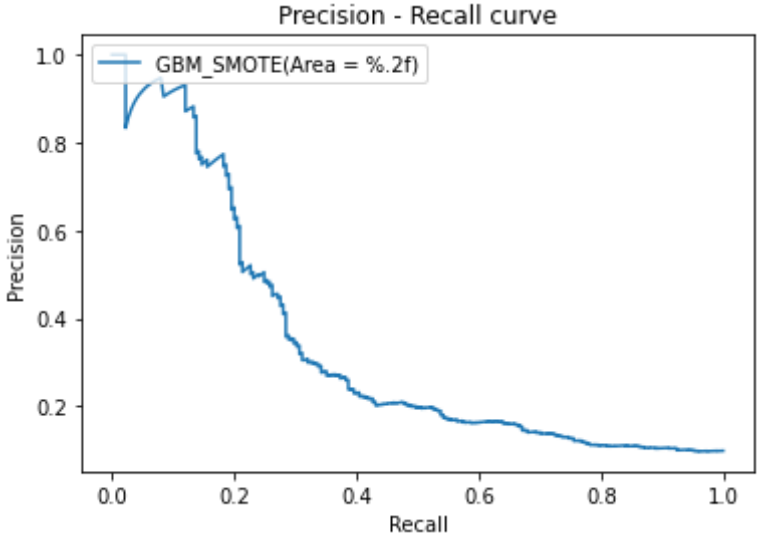


In [40]:
```
#Obtaining PR-Curve for GB with SMOTE
```

```
y_pred_PR_curve = model_gbm.predict_proba(X_test)
y_pred_PR_curve
print (type(y_pred_PR_curve))
y_pred_PR_curve = pd.DataFrame(y_pred_PR_curve)
print(type(y_pred_PR_curve))
y_pred_PR_curve = y_pred_PR_curve.iloc[:,1]
y_pred_PR_curve

precision, recall, thresholds = precision_recall_curve(y_test, y_pred_PR_curve)

plt.plot(recall, precision, label = "GBM_SMOTE(Area = %.2f)" )
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.legend(loc = "upper left")
plt.title("Precision - Recall curve")
```

```
<class 'numpy.ndarray'>
<class 'pandas.core.frame.DataFrame'>
```

Out[40]:
```
Text(0.5, 1.0, 'Precision - Recall curve')
```



In [42]:
```
#Performing GB with Random UnderSampling (RUS)

model_gbm = GradientBoostingClassifier(random_state=42)
```

```
model_gbm.fit(X_train_rus, y_train_rus)
y_pred_gb_RUS = model_gbm.predict(X_test)
```

In [43]:

```
#Obtaining  performance metrics for GB with RUS
print(classification_report(y_test, y_pred_gb_RUS))
confusion_matrix(y_test, y_pred_gb_RUS)
```

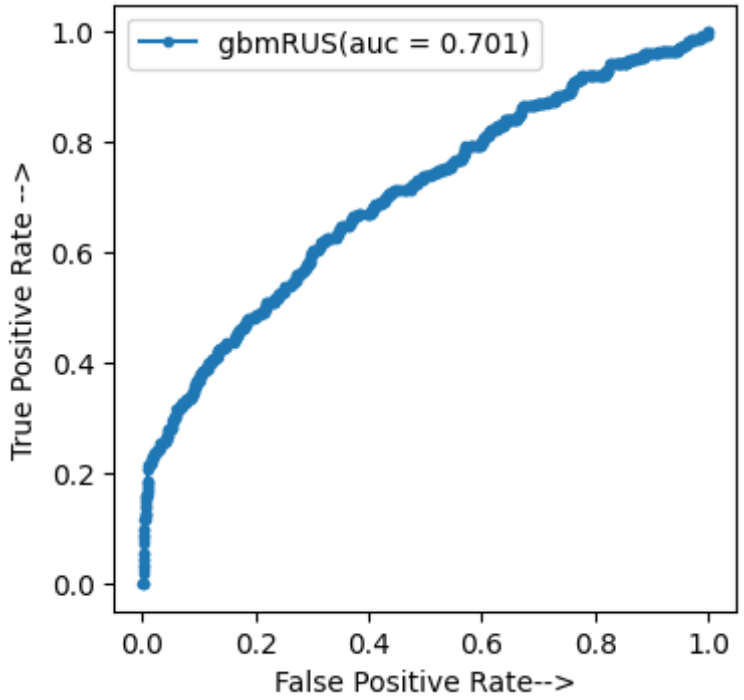
```
              precision    recall  f1-score   support

           0       0.94      0.79      0.85      2090
           1       0.20      0.49      0.28       225

    accuracy                           0.76      2315
   macro avg       0.57      0.64      0.57      2315
weighted avg       0.86      0.76      0.80      2315
```

Out[43]:

```
array([[1645,  445],
       [ 114,  111]], dtype=int64)
```

In [45]:

```
#Obtaining AUROC metrics for GM with RUS
y_pred_roc = model_gbm.decision_function(X_test)
gbRUS_fpr, gbRUS_tpr, threshold = roc_curve(y_test, y_pred_roc)
auc_gbRUS = auc(gbRUS_fpr, gbRUS_tpr)
plt.figure(figsize=(4,4), dpi = 100)
plt.plot(gbRUS_fpr, gbRUS_tpr, marker='.',label='gbmRUS(auc = %0.3f)' % auc_gbRUS)
plt.xlabel('False Positive Rate-->')
plt.ylabel('True Positive Rate -->')

plt.legend()

plt.show()
```

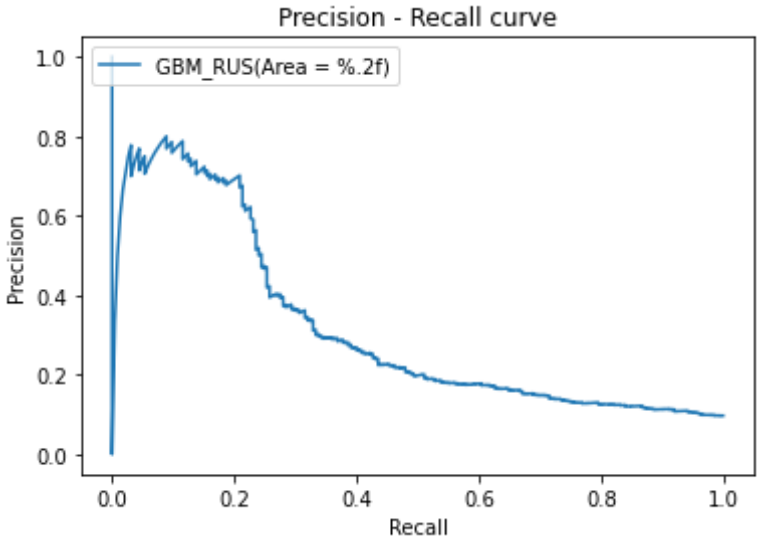


In [47]:

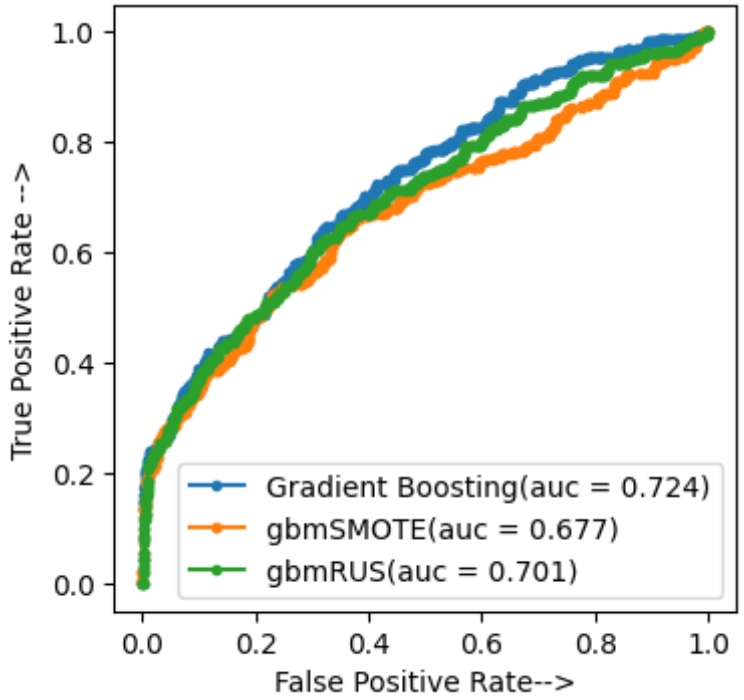
```
#Obtaining PR-Curve for GB with RUS
y_pred_PR_curve = model_gbm.predict_proba(X_test)
y_pred_PR_curve
print (type(y_pred_PR_curve))
y_pred_PR_curve = pd.DataFrame(y_pred_PR_curve)
print(type(y_pred_PR_curve))
y_pred_PR_curve = y_pred_PR_curve.iloc[:,1]
y_pred_PR_curve

precision, recall, thresholds = precision_recall_curve(y_test, y_pred_PR_curve)


plt.plot(recall, precision, label = "GBM_RUS(Area = %.2f)" )
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.legend(loc = "upper left")
plt.title("Precision - Recall curve")
```

```
<class 'numpy.ndarray'>
<class 'pandas.core.frame.DataFrame'>
```

Out[47]:
```
Text(0.5, 1.0, 'Precision - Recall curve')
```



In [46]:
```
#Obtaining a Combined AUROC metrics for Gradient Boosting Classifier
plt.figure(figsize=(4,4), dpi = 100)
plt.plot(gbm_fpr, gbm_tpr, marker='.',label='Gradient Boosting(auc = %0.3f)' % auc_gbm)
plt.plot(gbSM_fpr, gbSM_tpr, marker='.',label='gbmSMOTE(auc = %0.3f)' % auc_gbSM)
plt.plot(gbRUS_fpr, gbRUS_tpr, marker='.',label='gbmRUS(auc = %0.3f)' % auc_gbRUS)
plt.xlabel('False Positive Rate-->')
plt.ylabel('True Positive Rate -->')

plt.legend()

plt.show()
```



In [48]:
```
                                #Performing  XGBoost Classifier
from xgboost import XGBClassifier
XGBClassifier()
model = XGBClassifier(random_state=42)
model.fit(X_train, y_train)
```

```
c:\users\silve\appdata\local\programs\python\python38\lib\site-packages\xgboost\sklearn.py:1146: UserWarning: The use
of label encoder in XGBClassifier is deprecated and will be removed in a future release. To remove this warning, do t
he following: 1) Pass option use_label_encoder=False when constructing XGBClassifier object; and 2) Encode your label
s (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
[16:42:30] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/src/learner.cc:1095: Starting in XGB
oost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logl
oss'. Explicitly set eval_metric if you'd like to restore the old behavior.
```

Out[48]:
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
              importance_type='gain', interaction_constraints='',
```

```
              learning_rate=0.300000012, max_delta_step=0, max_depth=6,
              min_child_weight=1, missing=nan, monotone_constraints='()',
              n_estimators=100, n_jobs=8, num_parallel_tree=1, random_state=42,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
              tree_method='exact', validate_parameters=1, verbosity=None)
```

In [54]:

```
#Obtaining baseline performance metrics for XGBoost
y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred))
confusion_matrix(y_test, y_pred)
```

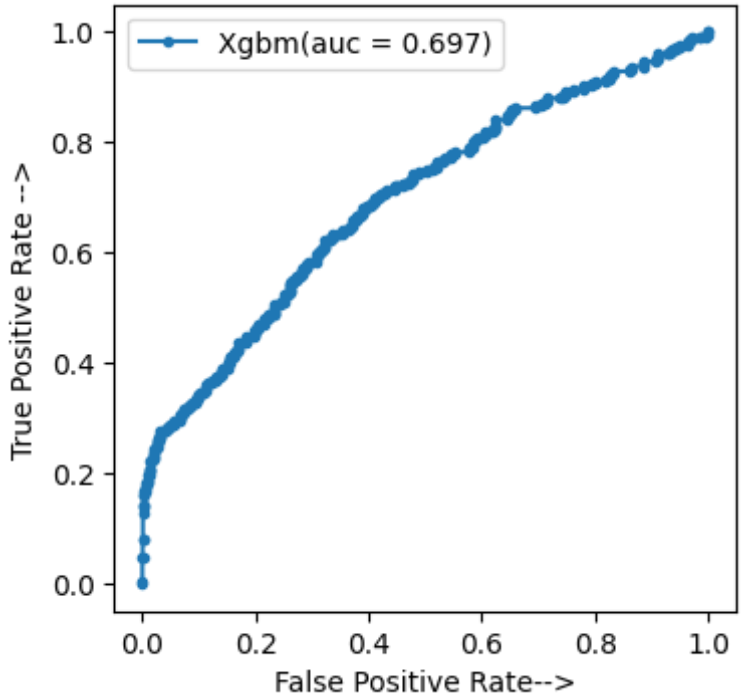
```
              precision    recall  f1-score   support

           0       0.92      0.99      0.95      2090
           1       0.69      0.20      0.30       225

    accuracy                           0.91      2315
   macro avg       0.80      0.59      0.63      2315
weighted avg       0.90      0.91      0.89      2315
```

Out[54]:

```
array([[2070,   20],
       [ 181,   44]], dtype=int64)
```

In [49]:

```
#Obtaining AUROC metrics for XGBoost
y_pred_roc = model.predict_proba(X_test)[:,1]
Xgb_fpr, Xgb_tpr, threshold = roc_curve(y_test, y_pred_roc)
auc_Xgb = auc(Xgb_fpr, Xgb_tpr)
plt.figure(figsize=(4,4), dpi = 100)
plt.plot(Xgb_fpr, Xgb_tpr, marker='.',label='Xgbm(auc = %0.3f)' % auc_Xgb)
plt.xlabel('False Positive Rate-->')
plt.ylabel('True Positive Rate -->')

plt.legend()

plt.show()
```

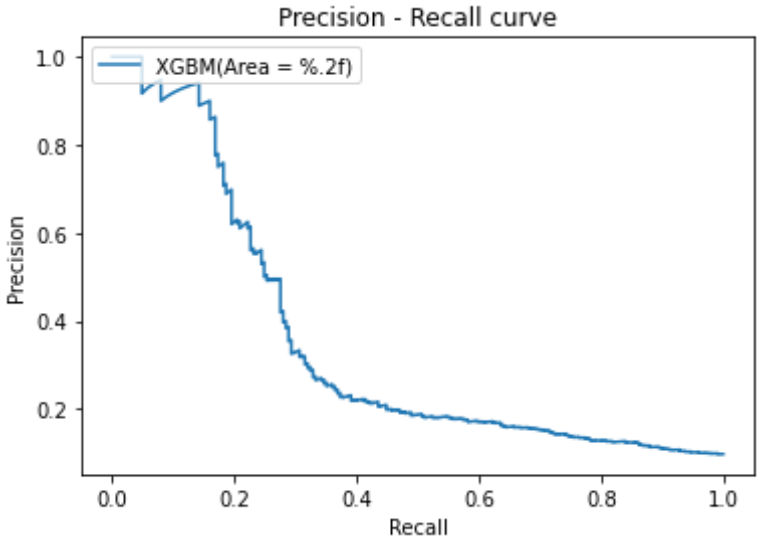


In [57]:

```
#Obtaining PR-Curve for XGBoost
y_pred_PR_curve = model.predict_proba(X_test)
y_pred_PR_curve
print (type(y_pred_PR_curve))
y_pred_PR_curve = pd.DataFrame(y_pred_PR_curve)
print(type(y_pred_PR_curve))
y_pred_PR_curve = y_pred_PR_curve.iloc[:,1]
y_pred_PR_curve

precision, recall, thresholds = precision_recall_curve(y_test, y_pred_PR_curve)

plt.plot(recall, precision, label = "XGBM(Area = %.2f)" )
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.legend(loc = "upper left")
plt.title("Precision - Recall curve")
```

```
<class 'numpy.ndarray'>
<class 'pandas.core.frame.DataFrame'>
```

Out[57]:
```
Text(0.5, 1.0, 'Precision - Recall curve')
```



In [58]:
```
#Performing SMOTE with XGBoost classifier
model = XGBClassifier(random_state=42)
model.fit(X_train_smote, y_train_smote)
y_pred = model.predict(X_test)
```

```
c:\users\silve\appdata\local\programs\python\python38\lib\site-packages\xgboost\sklearn.py:1146: UserWarning: The use
of label encoder in XGBClassifier is deprecated and will be removed in a future release. To remove this warning, do t
he following: 1) Pass option use_label_encoder=False when constructing XGBClassifier object; and 2) Encode your label
s (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
[16:53:32] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/src/learner.cc:1095: Starting in XGB
oost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logl
oss'. Explicitly set eval_metric if you'd like to restore the old behavior.
```

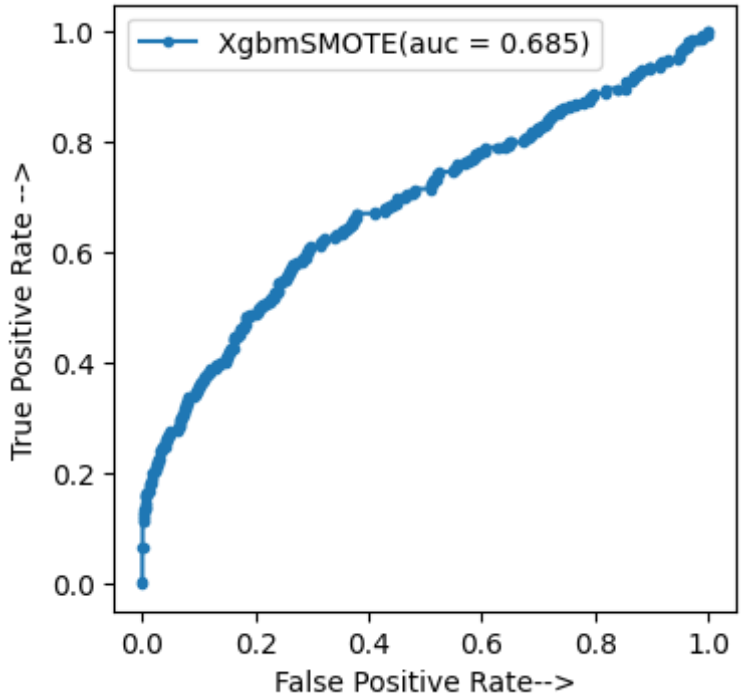
In [59]:
```
#Obtaining performance metrics for XGBoost with SMOTE
print(classification_report(y_test, y_pred))
confusion_matrix(y_test, y_pred)
```

```
              precision    recall  f1-score   support
```

```
              0       0.92      0.95      0.94      2090
              1       0.38      0.27      0.32       225

    accuracy                            0.89      2315
   macro avg       0.65      0.61      0.63      2315
weighted avg       0.87      0.89      0.88      2315
```

Out[59]:
```
array([[1992,   98],
       [ 164,   61]], dtype=int64)
```

In [60]:
```
#Obtaining AUROC metrics for XGBoost with SMOTE
y_pred_roc = model.predict_proba(X_test)[:,1]
XgbSM_fpr, XgbSM_tpr, threshold = roc_curve(y_test, y_pred_roc)
auc_XgbSM = auc(XgbSM_fpr, XgbSM_tpr)
plt.figure(figsize=(4,4), dpi = 100)
plt.plot(XgbSM_fpr, XgbSM_tpr, marker='.',label='XgbmSMOTE(auc = %0.3f)' % auc_XgbSM)
plt.xlabel('False Positive Rate-->')
plt.ylabel('True Positive Rate -->')

plt.legend()

plt.show()
```

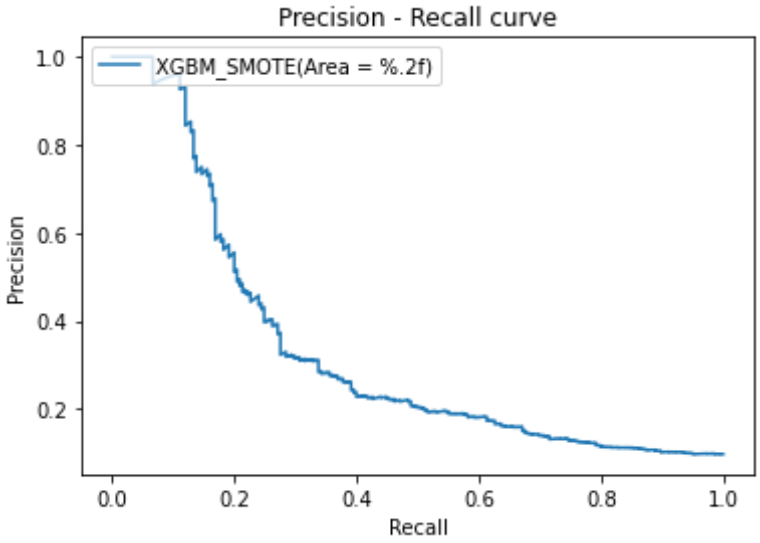


In [69]:

```
#Obtaining PR-Curve for XGBoost with SMOTE
y_pred_PR_curve = model.predict_proba(X_test)
y_pred_PR_curve
print (type(y_pred_PR_curve))
y_pred_PR_curve = pd.DataFrame(y_pred_PR_curve)
print(type(y_pred_PR_curve))
y_pred_PR_curve = y_pred_PR_curve.iloc[:,1]
y_pred_PR_curve

precision, recall, thresholds = precision_recall_curve(y_test, y_pred_PR_curve)

plt.plot(recall, precision, label = "XGBM_SMOTE(Area = %.2f)" )
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.legend(loc = "upper left")
plt.title("Precision - Recall curve")
```

```
<class 'numpy.ndarray'>
<class 'pandas.core.frame.DataFrame'>
```

Out[69]:
```
Text(0.5, 1.0, 'Precision - Recall curve')
```



In [63]:
```
#Performing XGBoost with Random UnderSampling (RUS)
model_XGBoost = XGBClassifier(random_state=42)
model_XGBoost.fit(X_train_rus, y_train_rus)
```

```
c:\users\silve\appdata\local\programs\python\python38\lib\site-packages\xgboost\sklearn.py:1146: UserWarning: The use
of label encoder in XGBClassifier is deprecated and will be removed in a future release. To remove this warning, do t
he following: 1) Pass option use_label_encoder=False when constructing XGBClassifier object; and 2) Encode your label
s (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
[16:53:59] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/src/learner.cc:1095: Starting in XGB
oost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logl
oss'. Explicitly set eval_metric if you'd like to restore the old behavior.
```

Out[63]:
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
              importance_type='gain', interaction_constraints='',
              learning_rate=0.300000012, max_delta_step=0, max_depth=6,
              min_child_weight=1, missing=nan, monotone_constraints='()',
              n_estimators=100, n_jobs=8, num_parallel_tree=1, random_state=42,
```

```
                reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
                tree_method='exact', validate_parameters=1, verbosity=None)
```

In [70]:

```
#Obtaining performance metrics for XGBoost with RUS
y_pred = model_XGBoost.predict(X_test)
print(classification_report(y_test, y_pred))
confusion_matrix(y_test, y_pred)
```

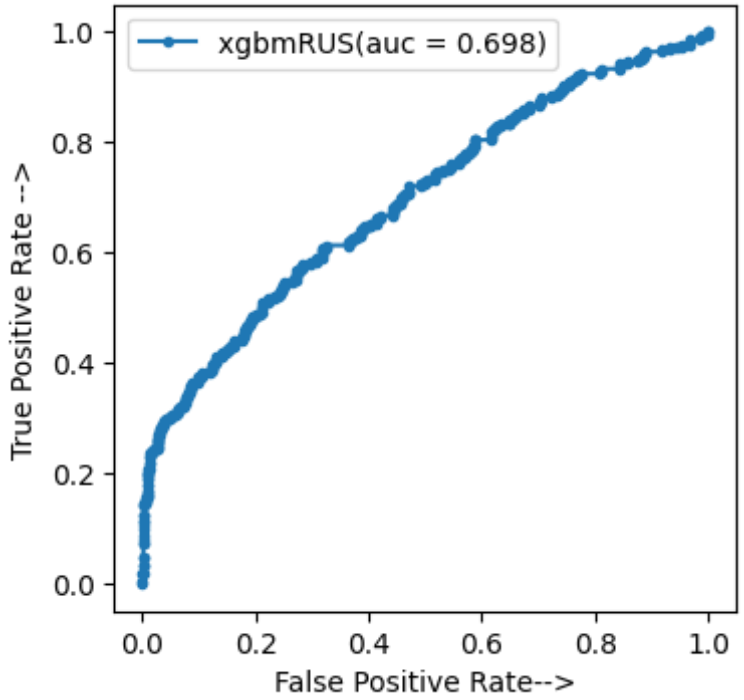
```
              precision    recall  f1-score   support

           0       0.94      0.69      0.79      2090
           1       0.17      0.59      0.26       225

    accuracy                           0.68      2315
   macro avg       0.55      0.64      0.53      2315
weighted avg       0.86      0.68      0.74      2315
```

Out[70]:

```
array([[1434,  656],
       [  92,  133]], dtype=int64)
```

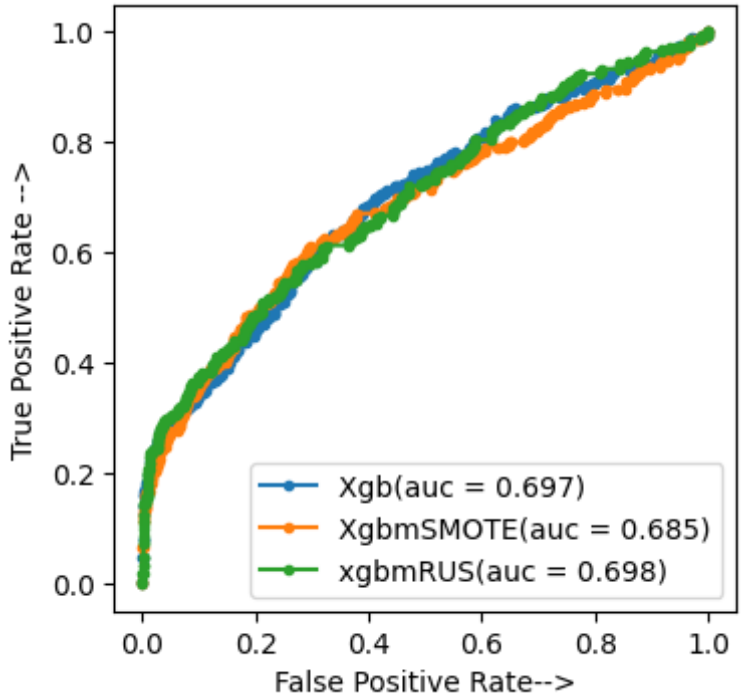
In [71]:

```
#Obtaining AUROC metrics for XGBoost with RUS
y_pred_roc = model_XGBoost.predict_proba(X_test)[:,1]
xgbRUS_fpr, xgbRUS_tpr, threshold = roc_curve(y_test, y_pred_roc)
auc_xgbRUS = auc(xgbRUS_fpr, xgbRUS_tpr)
plt.figure(figsize=(4,4), dpi = 100)
plt.plot(xgbRUS_fpr, xgbRUS_tpr, marker='.',label='xgbmRUS(auc = %0.3f)' % auc_xgbRUS)
plt.xlabel('False Positive Rate-->')
plt.ylabel('True Positive Rate -->')

plt.legend()

plt.show()
```



In [72]:

```
#Obtaining a Combined AUROC metrics for XGBoost Classifier
plt.figure(figsize=(4,4), dpi = 100)
plt.plot(Xgb_fpr, Xgb_tpr, marker='.',label='Xgb(auc = %0.3f)' % auc_Xgb)
plt.plot(XgbSM_fpr, XgbSM_tpr, marker='.',label='XgbmSMOTE(auc = %0.3f)' % auc_XgbSM)
plt.plot(xgbRUS_fpr, xgbRUS_tpr, marker='.',label='xgbmRUS(auc = %0.3f)' % auc_xgbRUS)
plt.xlabel('False Positive Rate-->')
plt.ylabel('True Positive Rate -->')

plt.legend()

plt.show()
```

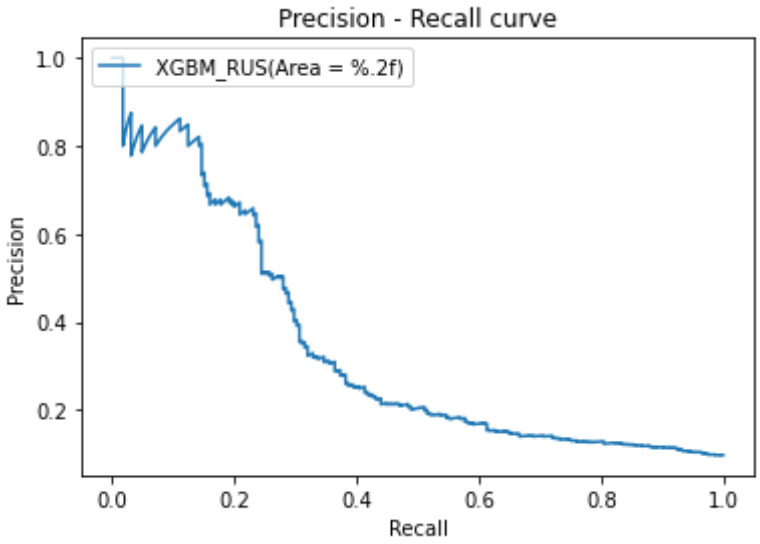


In [74]:

```
#Obtaining PR-Curve for XGBoost with RUS
y_pred_PR_curve = model_XGBoost.predict_proba(X_test)
y_pred_PR_curve
print (type(y_pred_PR_curve))
y_pred_PR_curve = pd.DataFrame(y_pred_PR_curve)
print(type(y_pred_PR_curve))
y_pred_PR_curve = y_pred_PR_curve.iloc[:,1]
y_pred_PR_curve

precision, recall, thresholds = precision_recall_curve(y_test, y_pred_PR_curve)

plt.plot(recall, precision, label = "XGBM_RUS(Area = %.2f)" )
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.legend(loc = "upper left")
plt.title("Precision - Recall curve")
```

```
<class 'numpy.ndarray'>
<class 'pandas.core.frame.DataFrame'>
```

Out[74]:
```
Text(0.5, 1.0, 'Precision - Recall curve')
```



In [75]:
```
#Performing Borderline smote with ADABoost
from imblearn.over_sampling import BorderlineSMOTE
#Transform the dataset
oversample = BorderlineSMOTE(k_neighbors=2)
X = df.copy()
X.pop('Apgar_score')
y = df['Apgar_score']
X, y = oversample.fit_resample(X, y)
```

In [76]:
```
from collections import Counter
counter = Counter(y)
print (counter)
```

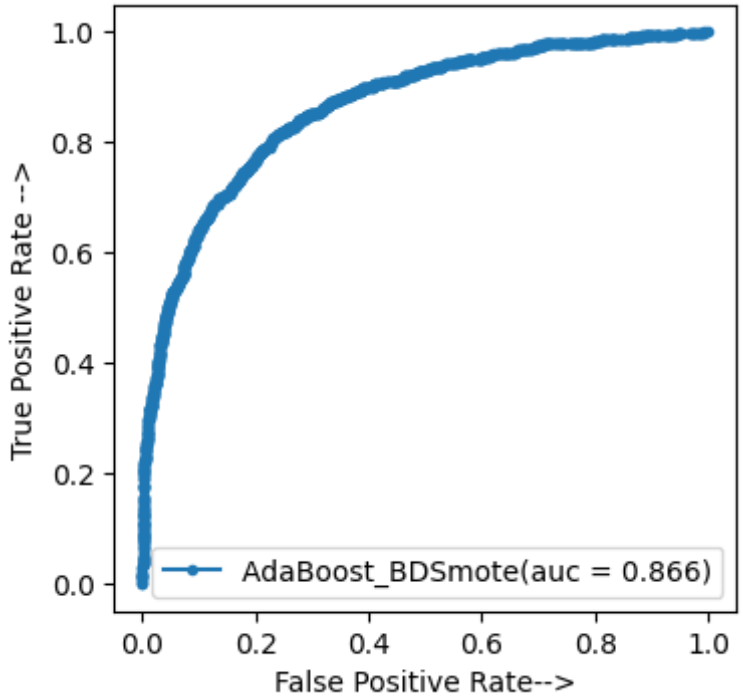
```
Counter({0: 6983, 1: 6983})
```

In [77]:
```
```

```
X_train_bdsm, X_test_bdsm, y_train_bdsm, y_test_bdsm = train_test_split(X, y,
                                                    test_size=0.3,
                                                    random_state=42)
```

In [78]:

```
#Building the AdaBoost Model with BD_SMOTE
# Create adaboost classifer object
abc = AdaBoostClassifier(n_estimators=150,
                         learning_rate=1)
# Train Adaboost Classifer
model = abc.fit(X_train_bdsm, y_train_bdsm)

#Predict the response for test dataset
y_pred = model.predict(X_test_bdsm)

#Trying ROC metrics
y_pred_roc = model.decision_function(X_test_bdsm)
AdbstBDsmote_fpr, AdbstBDsmote_tpr, threshold = roc_curve(y_test_bdsm, y_pred_roc)
auc_Adbst_bdsm = auc(AdbstBDsmote_fpr, AdbstBDsmote_tpr)

plt.figure(figsize=(4,4), dpi = 100)
plt.plot(AdbstBDsmote_fpr, AdbstBDsmote_tpr, marker='.',label='AdaBoost_BDSmote(auc = %0.3f)' % auc_Adbst_bdsm)

plt.xlabel('False Positive Rate-->')
plt.ylabel('True Positive Rate -->')

plt.legend()

plt.show()
```



In [79]:

```
# Classification metrics for AdaBoost with BD_SMOTE
print("Accuracy:",metrics.accuracy_score(y_test_bdsm, y_pred))
print(classification_report(y_test_bdsm, y_pred))
confusion_matrix(y_test_bdsm, y_pred)
```

```
Accuracy: 0.7859188544152744
              precision    recall  f1-score   support

           0       0.78      0.79      0.79      2092
           1       0.79      0.78      0.78      2098

    accuracy                           0.79      4190
   macro avg       0.79      0.79      0.79      4190
weighted avg       0.79      0.79      0.79      4190
```

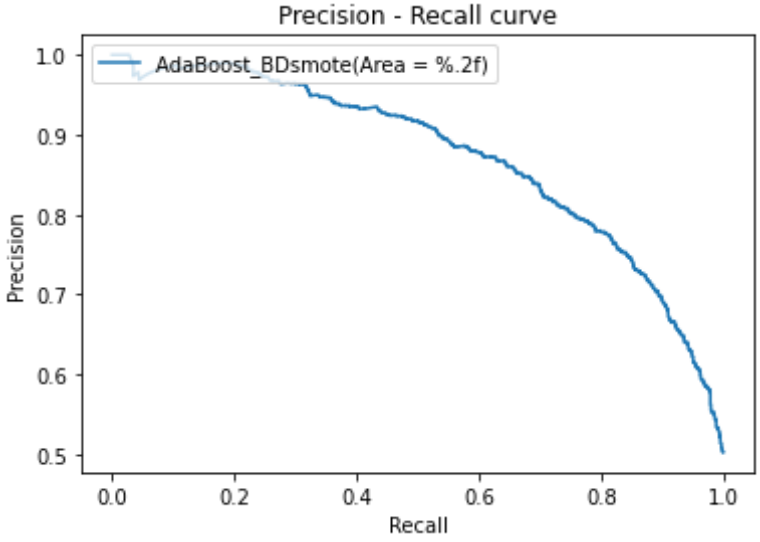
Out[79]:

```
array([[1658,  434],
       [ 463, 1635]], dtype=int64)
```

In [82]:
```
# Obtaining PR-curve for AdaBoost with BD_SMOTE
y_pred_PR_curve = model.predict_proba(X_test_bdsm)
y_pred_PR_curve
print (type(y_pred_PR_curve))
y_pred_PR_curve = pd.DataFrame(y_pred_PR_curve)
print(type(y_pred_PR_curve))
y_pred_PR_curve = y_pred_PR_curve.iloc[:,1]
y_pred_PR_curve

precision, recall, thresholds = precision_recall_curve(y_test_bdsm, y_pred_PR_curve)

plt.plot(recall, precision, label = "AdaBoost_BDsmote(Area = %.2f)" )
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.legend(loc = "upper left")
plt.title("Precision - Recall curve")
```

```
<class 'numpy.ndarray'>
<class 'pandas.core.frame.DataFrame'>
```

Out[82]:
```
Text(0.5, 1.0, 'Precision - Recall curve')
```



In [83]:
```
#Performing Borderline_SMOTE on Gradient Boosting
model_gbm = GradientBoostingClassifier(random_state=42)
```

```
model_gbm.fit(X_train_bdsm, y_train_bdsm)
y_pred_gb = model_gbm.predict(X_test_bdsm)
```

In [84]:

```
#Obtainig performance metrics for Gradient Boosting with Borderline_SMOTE
print(classification_report(y_test_bdsm, y_pred_gb))
confusion_matrix(y_test_bdsm, y_pred_gb)
```

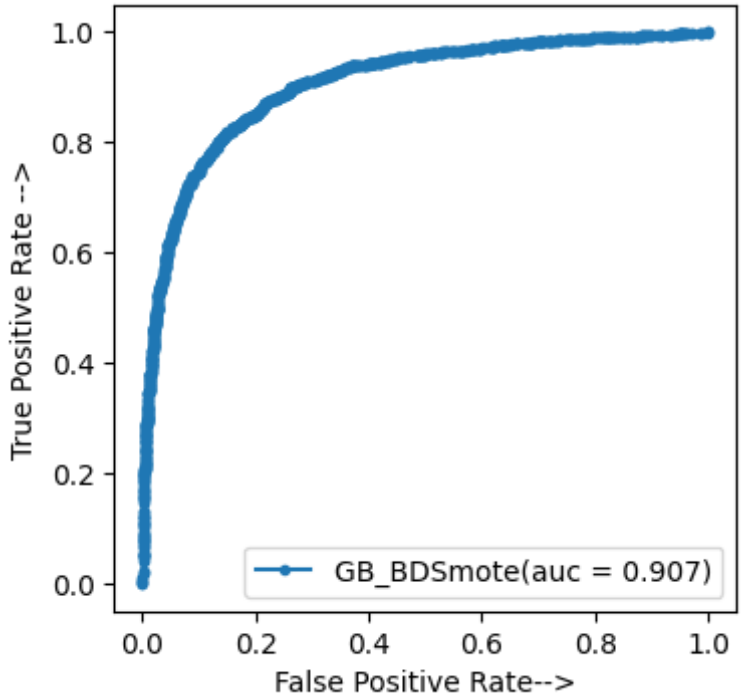
```
              precision    recall  f1-score   support

           0       0.82      0.85      0.83      2092
           1       0.84      0.82      0.83      2098

    accuracy                           0.83      4190
   macro avg       0.83      0.83      0.83      4190
weighted avg       0.83      0.83      0.83      4190
```

Out[84]:

```
array([[1768,  324],
       [ 376, 1722]], dtype=int64)
```

In [85]:

```
#Obtainig AUROC for Gradient Boosting with Borderline_SMOTE
y_pred_roc = model_gbm.decision_function(X_test_bdsm)
gbmBDsmote_fpr, gbmBDsmote_tpr, threshold = roc_curve(y_test_bdsm, y_pred_roc)
auc_gbm_BDsmote = auc(gbmBDsmote_fpr, gbmBDsmote_tpr)
plt.figure(figsize=(4,4), dpi = 100)
plt.plot(gbmBDsmote_fpr, gbmBDsmote_tpr, marker='.',label='GB_BDSmote(auc = %0.3f)' % auc_gbm_BDsmote)
plt.xlabel('False Positive Rate-->')
plt.ylabel('True Positive Rate -->')

plt.legend()

plt.show()
```

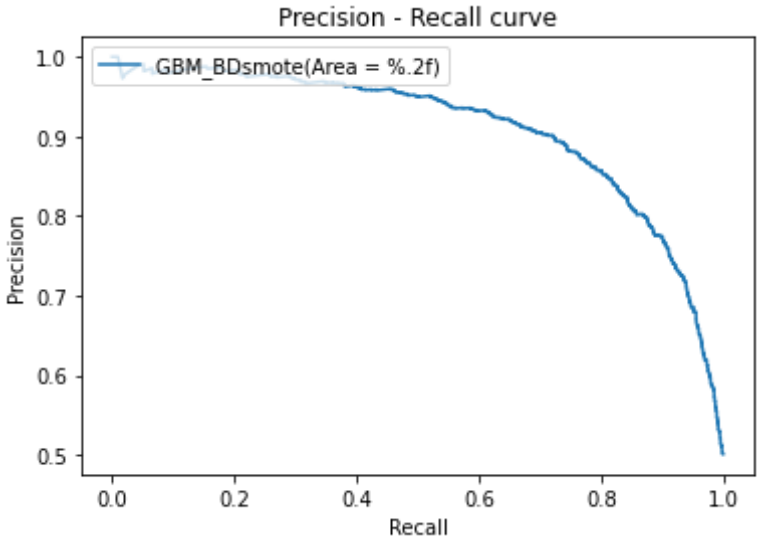


In [87]:

```
## Obtaining PR-curve for Gradient Boosting with BD_SMOTE
y_pred_PR_curve = model_gbm.predict_proba(X_test_bdsm)
y_pred_PR_curve
print (type(y_pred_PR_curve))
y_pred_PR_curve = pd.DataFrame(y_pred_PR_curve)
print(type(y_pred_PR_curve))
y_pred_PR_curve = y_pred_PR_curve.iloc[:,1]
y_pred_PR_curve

precision, recall, thresholds = precision_recall_curve(y_test_bdsm, y_pred_PR_curve)

plt.plot(recall, precision, label = "GBM_BDsmote(Area = %.2f)" )
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.legend(loc = "upper left")
plt.title("Precision - Recall curve")
```

```
<class 'numpy.ndarray'>
<class 'pandas.core.frame.DataFrame'>
```

Out[87]:
```
Text(0.5, 1.0, 'Precision - Recall curve')
```



In [89]:
```
#Performing Borderline_SMOTE on XGBoost Classifier
model = XGBClassifier(random_state=42)
model.fit(X_train_bdsm, y_train_bdsm)
y_pred_xgb_bdsm = model.predict(X_test_bdsm)
```

```
c:\users\silve\appdata\local\programs\python\python38\lib\site-packages\xgboost\sklearn.py:1146: UserWarning: The use
of label encoder in XGBClassifier is deprecated and will be removed in a future release. To remove this warning, do t
he following: 1) Pass option use_label_encoder=False when constructing XGBClassifier object; and 2) Encode your label
s (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
[17:28:45] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/src/learner.cc:1095: Starting in XGB
oost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logl
oss'. Explicitly set eval_metric if you'd like to restore the old behavior.
```

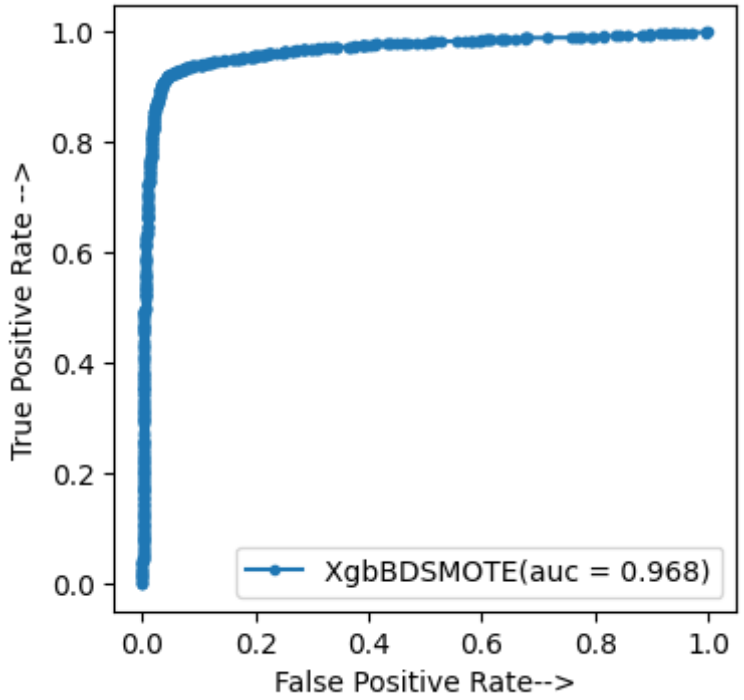
In [91]:
```
#Obtaining performance metrics for XGBoost with SMOTE
print(classification_report(y_test_bdsm,y_pred_xgb_bdsm))
confusion_matrix(y_test_bdsm, y_pred_xgb_bdsm)
```

```
              precision    recall  f1-score   support
```

```
              0       0.93      0.94      0.94      2092
              1       0.94      0.93      0.93      2098

       accuracy                           0.93      4190
      macro avg       0.93      0.93      0.93      4190
   weighted avg       0.93      0.93      0.93      4190
```

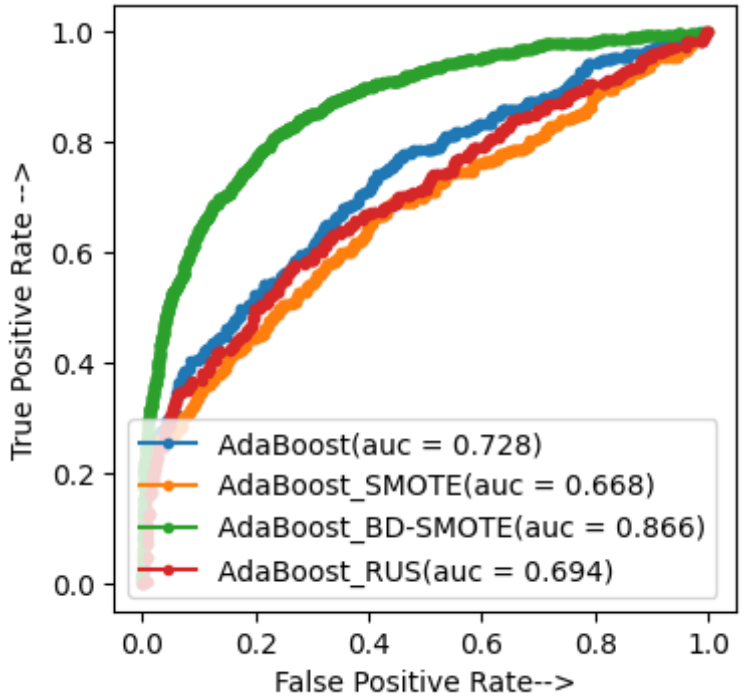
Out[91]:
```
array([[1975,  117],
       [ 156, 1942]], dtype=int64)
```

In [90]:
```
#Obtainig AUROC for XGBoost with Borderline_SMOTE
y_pred_roc = model.predict_proba(X_test_bdsm)[:,1]
XgbBDsmote_fpr, XgbBDsmote_tpr, threshold = roc_curve(y_test_bdsm, y_pred_roc)
auc_XgbBDsmote = auc(XgbBDsmote_fpr, XgbBDsmote_tpr)
plt.figure(figsize=(4,4), dpi = 100)
plt.plot(XgbBDsmote_fpr, XgbBDsmote_tpr, marker='.',label='XgbBDSMOTE(auc = %0.3f)' % auc_XgbBDsmote)
plt.xlabel('False Positive Rate-->')
plt.ylabel('True Positive Rate -->')

plt.legend()

plt.show()
```



In [92]:

```
#Obtaining a Combined AUROC metrics for AdaBoost Classifier
plt.figure(figsize=(4,4), dpi = 100)
plt.plot(Adbst_fpr, Adbst_tpr, marker='.',label='AdaBoost(auc = %0.3f)' % auc_Adbst)
plt.plot(AdbstSM_fpr, AdbstSM_tpr, marker='.',label='AdaBoost_SMOTE(auc = %0.3f)' % auc_AdbstSM)
plt.plot(AdbstBDsmote_fpr, AdbstBDsmote_tpr, marker='.',label='AdaBoost_BD-SMOTE(auc = %0.3f)' % auc_Adbst_bdsm)
plt.plot(AdbstRUS_fpr, AdbstRUS_tpr, marker='.',label='AdaBoost_RUS(auc = %0.3f)' % auc_AdbstRUS)
plt.xlabel('False Positive Rate-->')
plt.ylabel('True Positive Rate -->')

plt.legend()

plt.show()
```



In [94]:

```
#Trying PR-Curve
y_pred_PR_curve = model.predict_proba(X_test_bdsm)
y_pred_PR_curve
print (type(y_pred_PR_curve))
y_pred_PR_curve = pd.DataFrame(y_pred_PR_curve)
print(type(y_pred_PR_curve))
y_pred_PR_curve = y_pred_PR_curve.iloc[:,1]
y_pred_PR_curve

precision, recall, thresholds = precision_recall_curve(y_test_bdsm, y_pred_PR_curve)
```

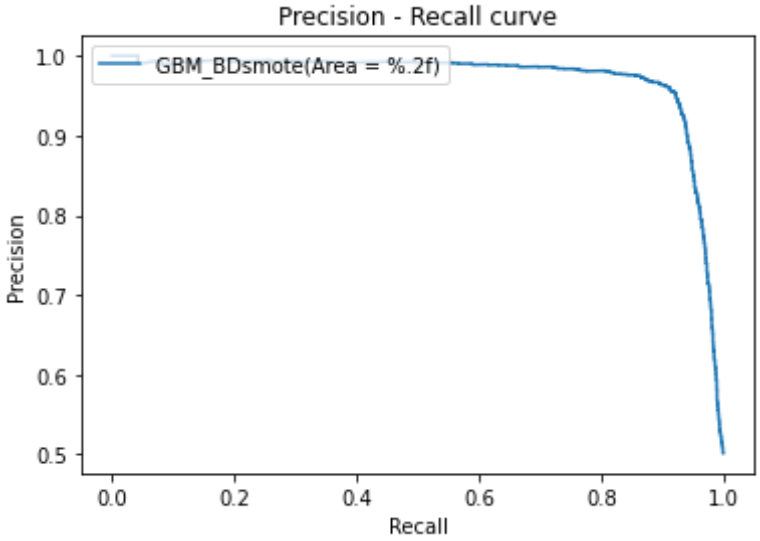
```
<class 'numpy.ndarray'>
<class 'pandas.core.frame.DataFrame'>
```

In [95]:

```
plt.plot(recall, precision, label = "GBM_BDsmote(Area = %.2f)" )
plt.xlabel('Recall')
plt.ylabel('Precision')
```

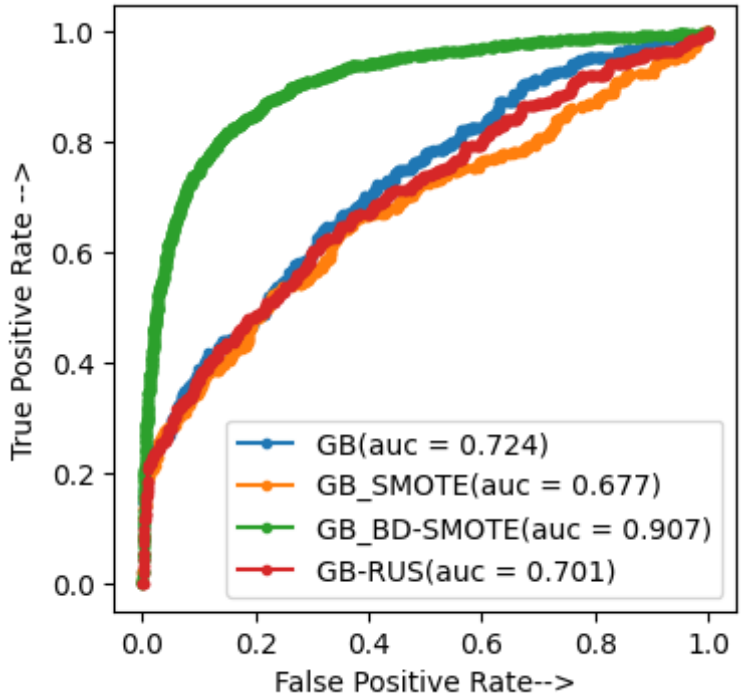
```
plt.legend(loc = "upper left")
plt.title("Precision - Recall curve")
```

Out[95]: Text(0.5, 1.0, 'Precision - Recall curve')



In [96]:

```
#Obtaining a Combined AUROC metrics for Gradient Boosting Classifier
plt.figure(figsize=(4,4), dpi = 100)
plt.plot(gbm_fpr, gbm_tpr, marker='.',label='GB(auc = %0.3f)' % auc_gbm)
plt.plot(gbSM_fpr, gbSM_tpr, marker='.',label='GB_SMOTE(auc = %0.3f)' % auc_gbSM)
plt.plot(gbmBDsmote_fpr, gbmBDsmote_tpr, marker='.',label='GB_BD-SMOTE(auc = %0.3f)' % auc_gbm_BDsmote)
plt.plot(gbRUS_fpr, gbRUS_tpr, marker='.',label='GB-RUS(auc = %0.3f)' % auc_gbRUS)
plt.xlabel('False Positive Rate-->')
plt.ylabel('True Positive Rate -->')

plt.legend()

plt.show()
```



In [97]:

```
#Obtaining a Combined AUROC metrics for XGBoost Classifier
plt.figure(figsize=(4,4), dpi = 100)
plt.plot(Xgb_fpr, Xgb_tpr, marker='.',label='XGb(auc = %0.3f)' % auc_Xgb)
plt.plot(XgbSM_fpr, XgbSM_tpr, marker='.',label='XGb_SMOTE(auc = %0.3f)' % auc_XgbSM)
plt.plot(XgbBDsmote_fpr, XgbBDsmote_tpr, marker='.',label='XGb_BD-SMOTE(auc = %0.3f)' % auc_XgbBDsmote)
plt.plot(xgbRUS_fpr, xgbRUS_tpr, marker='.',label='XGb_RUS(auc = %0.3f)' % auc_xgbRUS)
plt.xlabel('False Positive Rate-->')
plt.ylabel('True Positive Rate -->')

plt.legend()

plt.show()
```