Python tutorial 7

March 10, 2022

This tutorial aims to introduce the precision-recall (PR) curve for binary classification, for Dr. Xin Tong's DSO530 at the University of Southern California in spring 2022. We will use "2 simulation examples + 3 methods" to illustrate the PR curves. The two examples are Normal distributions in one-dimension and in multi-dimensions. The three methods under consideration are logistic regression, linear discriminant analysis, and random forest (Of course, you are welcome to try other methods). In each setting, we will plot both the ROC curve and the PR curve. In class, we became familiar with the ROC curve. Recall that we have seen the following confusion matrix.

	pred 0	pred 1
real 0	a	b
real 1	c	d

In this table, type I error is $b/(a+b)$ and type II error as $c/(c+d)$. Moreover, the precision is $d/(b+d)$ and the recall is $d/(c+d)$. While ROC curve is widely popular, for imbalanced classification problems, people also draw the precision-recall curve. The reason is, as you learn from the "Drug D user example", when class 1 is a small minority, small type I and type II errors might not guarantee a high precision. That said, if one cares about precision, it would be better to just plot this quantity directly using a precision-recall curve. Like the area under ROC curve (ROC-AUC), people also calculate the area under the precision-recall curve. A method with a higher PR-AUC is considered better. Of course, which area (and which curve) to look at depends on the application. Ideally, a data scientist should talk with the domain experts to make the decision. One assuring message is that if method "A"'s ROC curve strictly dominates method "B"'s ROC curve, then method "A"'s PR curve will also dominate method "B"'s PR curve.

For training data, here we set class 1 sample size $n_1 = 100$ and the imbalance ratio $\frac{n_0}{n_1} = 1, 4, 9$, where n_0 is sample size of class 0. For test data, we set $n_1' = 1{,}000$ and keep the sample size ratio $\frac{n_0'}{n_1'} = 1, 4, 9$ as in the training data.

[1]:
```
## the followiing is to ignore the warnings about version issues, etc., so as␣
 ↪to make the pdf file clearner.
import warnings
warnings.filterwarnings("ignore")

import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve
from sklearn.metrics import auc
```

```
from sklearn.metrics import plot_precision_recall_curve
from sklearn.linear_model import LogisticRegression
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.ensemble import RandomForestClassifier
```

[2]:
```
## As roc curves will be plotted many times, we define a function to plot them␣
 ↪for convinience.
def plot_roc_curve(fpr, tpr, roc_auc):
    plt.figure()
    lw = 2
    plt.figure(figsize=(6,6))
    plt.plot(fpr, tpr, color='darkorange',
             lw=lw, label='ROC curve (area = %0.2f)' % roc_auc)
    plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('ROC Curve')
    plt.legend(loc="lower right")
    plt.show()
```

1 Example 1

In example 1, we use normal distributions $\mathcal{N}(0,1)$ and $\mathcal{N}(\in,\infty)$ to generate the data from class 0 and class 1, separately. In other words, $X|(Y=0) \sim \mathcal{N}(0,1)$ and $X|(Y=1) \sim \mathcal{N}(2,1)$.

[3]:
```
mu_0 = 0
mu_1 = 2
sigma = 1
np.random.seed(0)

n1_train = 100
X1_train = np.random.normal(mu_1, sigma, n1_train)

n1_test = 1000
X1_test = np.random.normal(mu_1, sigma, n1_test)
```

1.1 n_0/n_1 ratio = 1:

[4]:
```
ratio_r1 = 1

n0_train_r1 = n1_train * ratio_r1
X0_train_r1 = np.random.normal(mu_0, sigma, n0_train_r1)

n0_test_r1 = n1_test * ratio_r1
```

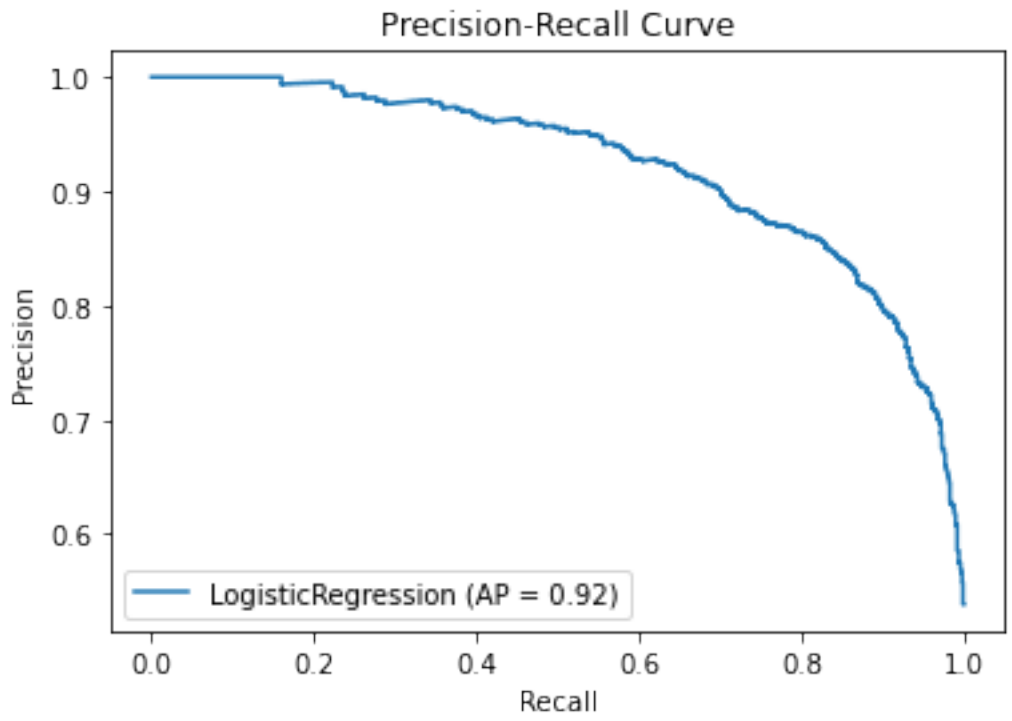
```
X0_test_r1 = np.random.normal(mu_0, sigma, n0_test_r1)

## np.r_ is used to concatenate any number of array slices along row axis
X_train_r1 = np.r_[X1_train, X0_train_r1].reshape(-1, 1)
X_test_r1 = np.r_[X1_test, X0_test_r1].reshape(-1, 1)

## [0]*10 will produce [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
y_train_r1 = np.r_[[1]*n1_train, [0]*n0_train_r1]
y_test_r1 = np.r_[[1]*n1_test, [0]*n0_test_r1]
```

1.1.1 Logistic Regression

[5]:
```
lr_r1 = LogisticRegression(penalty='none').fit(X_train_r1, y_train_r1)
```

[6]:
```
disp = plot_precision_recall_curve(lr_r1, X_test_r1, y_test_r1)
disp.ax_.set_title('Precision-Recall Curve')
```

[6]:
```
Text(0.5, 1.0, 'Precision-Recall Curve')
```

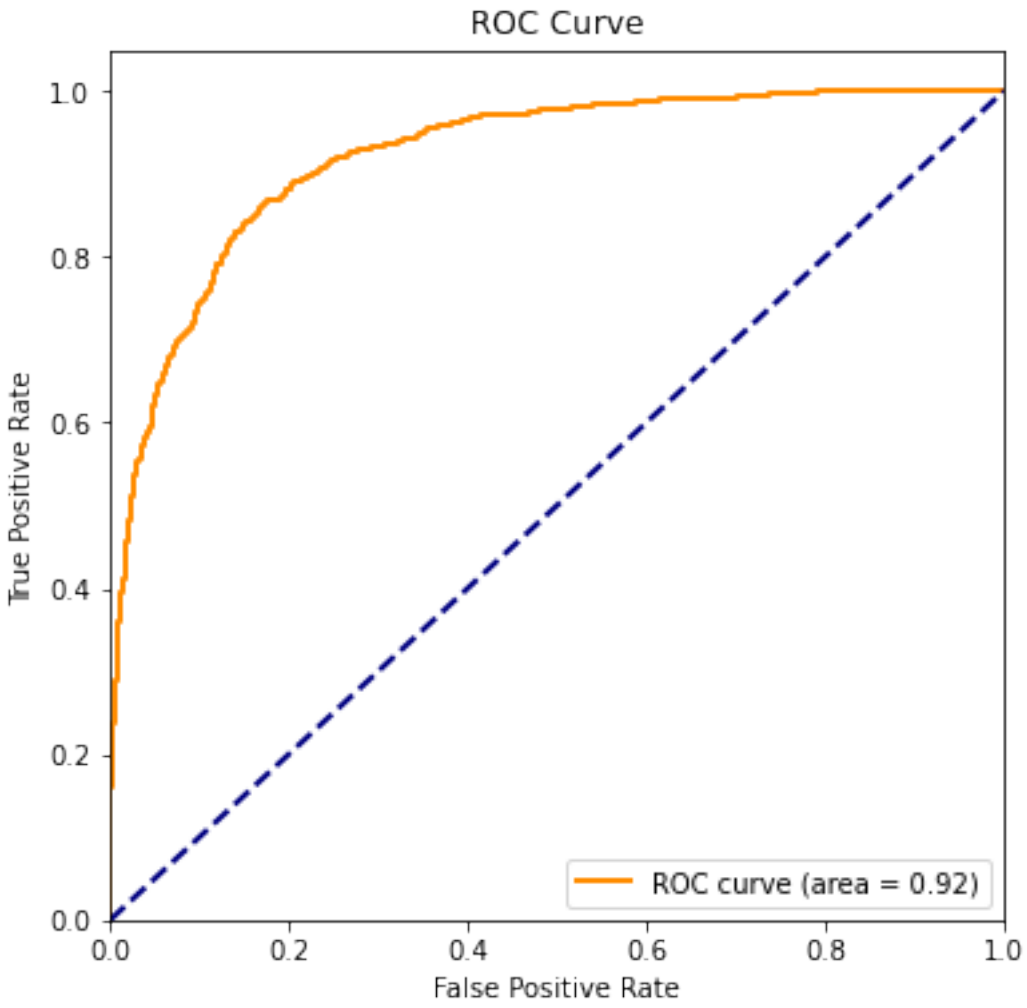


Pause for a moment and look at the scale of the vertial axis of the PR curve.

[7]:
```
y_score_r1_lr = lr_r1.predict_proba(X_test_r1)[:, 1]
```

```
fpr, tpr, thersholds = roc_curve(y_test_r1, y_score_r1_lr)
roc_auc = auc(fpr, tpr)

plot_roc_curve(fpr, tpr, roc_auc)
```

```
<Figure size 432x288 with 0 Axes>
```

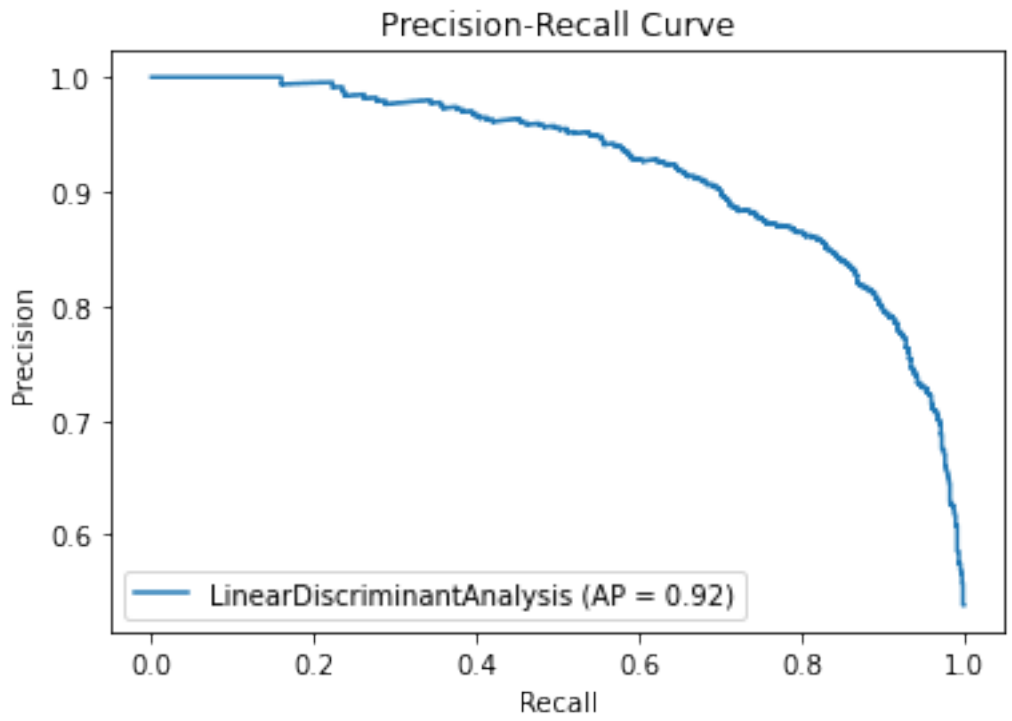


1.1.2 Linear Discriminant Analysis

[8]:
```
lda_r1 = LinearDiscriminantAnalysis().fit(X_train_r1, y_train_r1)
```

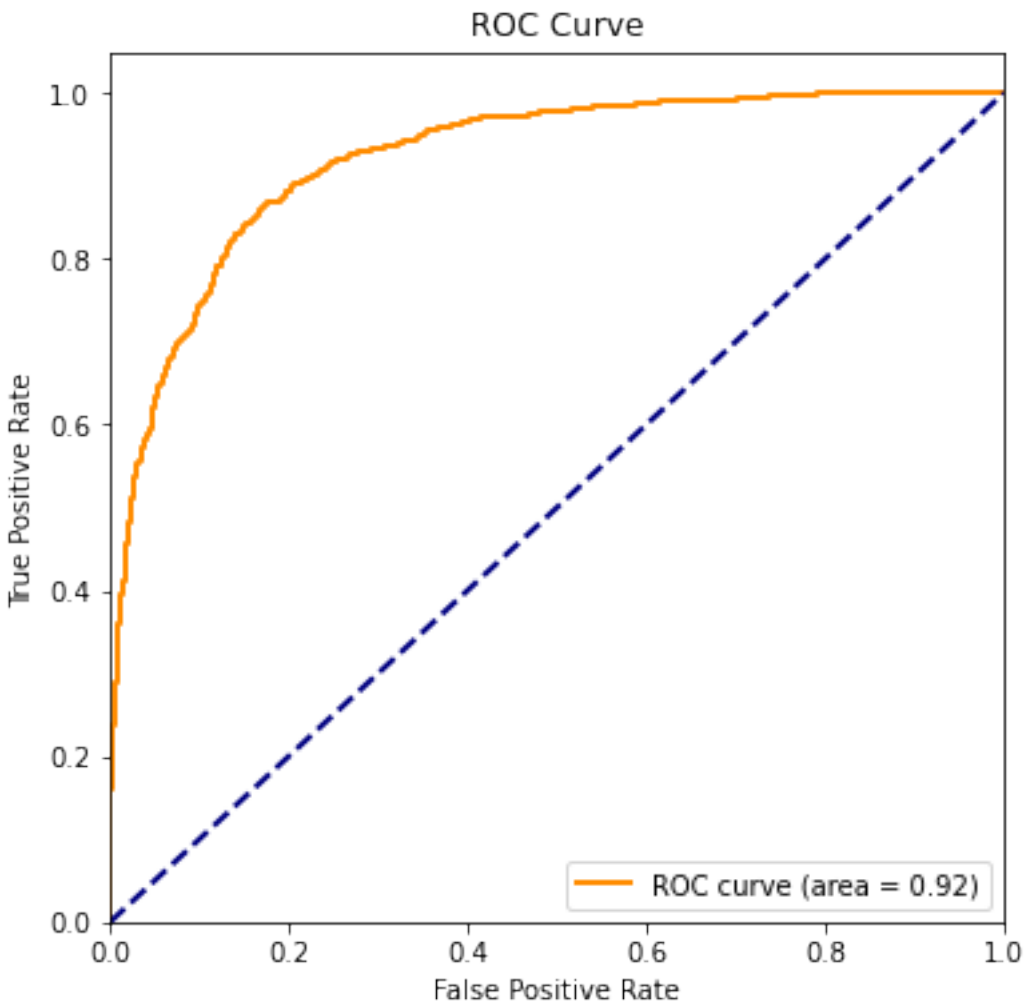
[9]:
```
disp = plot_precision_recall_curve(lda_r1, X_test_r1, y_test_r1)
disp.ax_.set_title('Precision-Recall Curve')
```

[9]:
```
Text(0.5, 1.0, 'Precision-Recall Curve')
```



[10]:
```
y_score_r1_lda = lda_r1.predict_proba(X_test_r1)[:, 1]

fpr, tpr, thersholds = roc_curve(y_test_r1, y_score_r1_lda)
roc_auc = auc(fpr, tpr)

plot_roc_curve(fpr, tpr, roc_auc)
```

```
<Figure size 432x288 with 0 Axes>
```

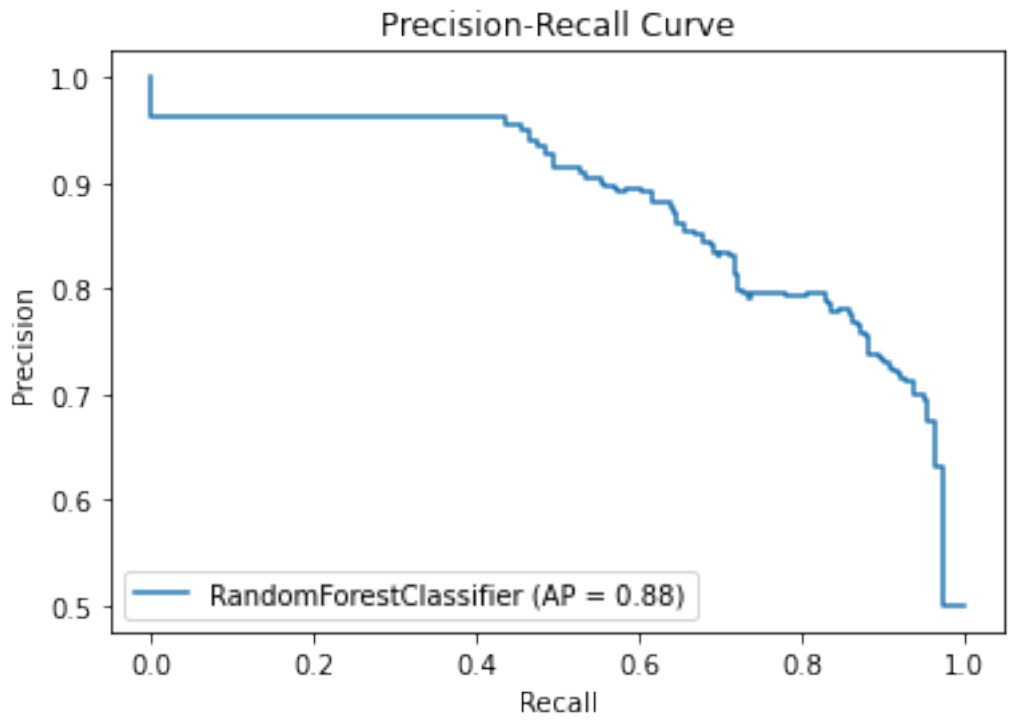


1.1.3 Random Forrest Classifier

[11]:
```
rfc_r1 = RandomForestClassifier().fit(X_train_r1, y_train_r1)
```

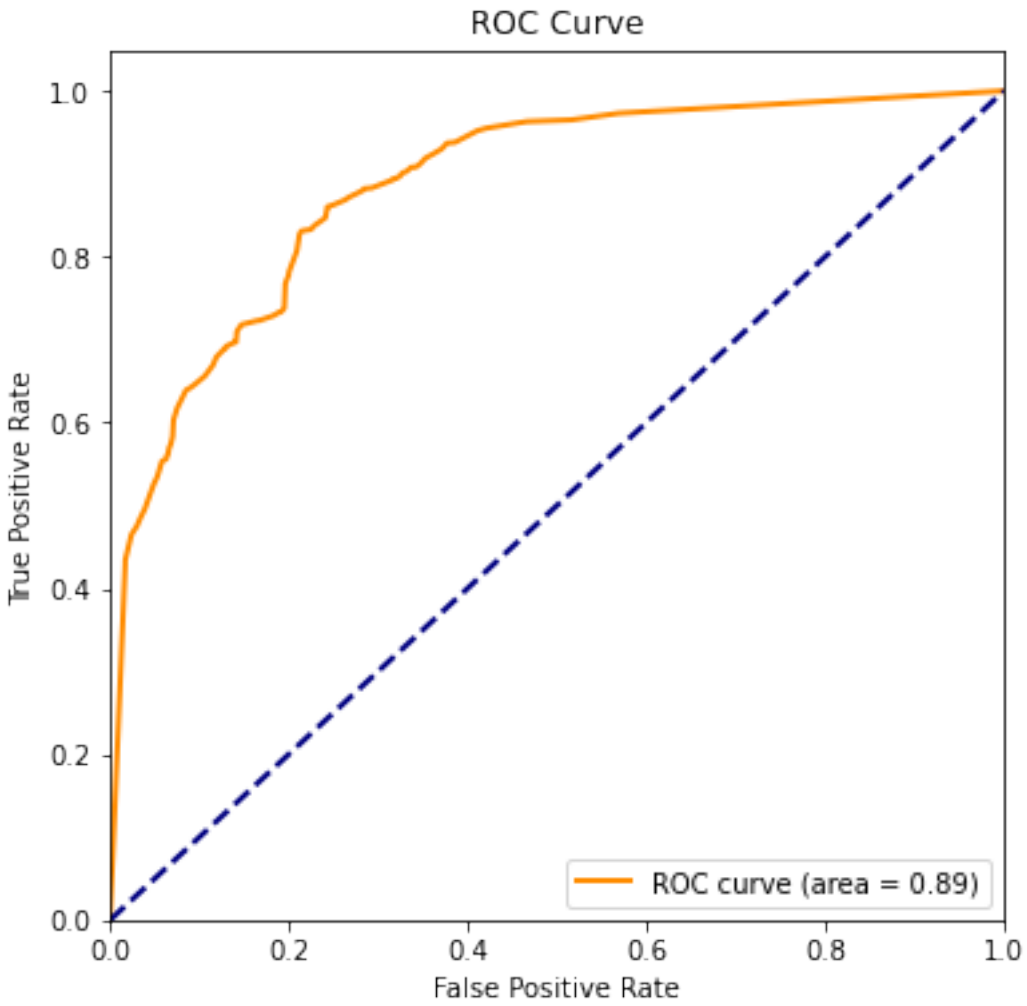
[12]:
```
disp = plot_precision_recall_curve(rfc_r1, X_test_r1, y_test_r1)
disp.ax_.set_title('Precision-Recall Curve')
```

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



[13]:
```
y_score_r1_rfc = rfc_r1.predict_proba(X_test_r1)[:, 1]

fpr, tpr, thersholds = roc_curve(y_test_r1, y_score_r1_rfc)
roc_auc = auc(fpr, tpr)

plot_roc_curve(fpr, tpr, roc_auc)
```

```
<Figure size 432x288 with 0 Axes>
```



1.2 n_0/n_1 ratio = 4:

[14]:
```
ratio_r2 = 4

n0_train_r2 = n1_train * ratio_r2
X0_train_r2 = np.random.normal(mu_0, sigma, n0_train_r2)

n0_test_r2 = n1_test * ratio_r2
X0_test_r2 = np.random.normal(mu_0, sigma, n0_test_r2)

## np.r_ is used to concatenate any number of array slices along row axis
X_train_r2 = np.r_[X1_train, X0_train_r2].reshape(-1, 1)
X_test_r2 = np.r_[X1_test, X0_test_r2].reshape(-1, 1)

## [0]*10 will produce [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

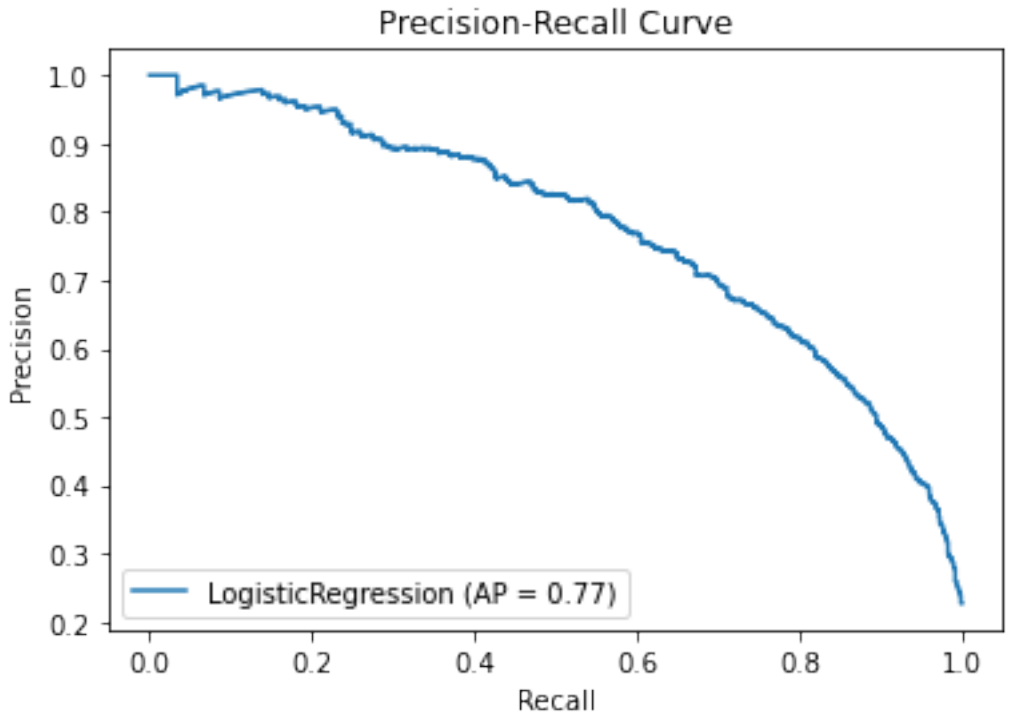
```
y_train_r2 = np.r_[[1]*n1_train, [0]*n0_train_r2]
y_test_r2 = np.r_[[1]*n1_test, [0]*n0_test_r2]
```

1.2.1 Logistic Regression

[15]:
```
lr_r2 = LogisticRegression(penalty='none').fit(X_train_r2, y_train_r2)
```

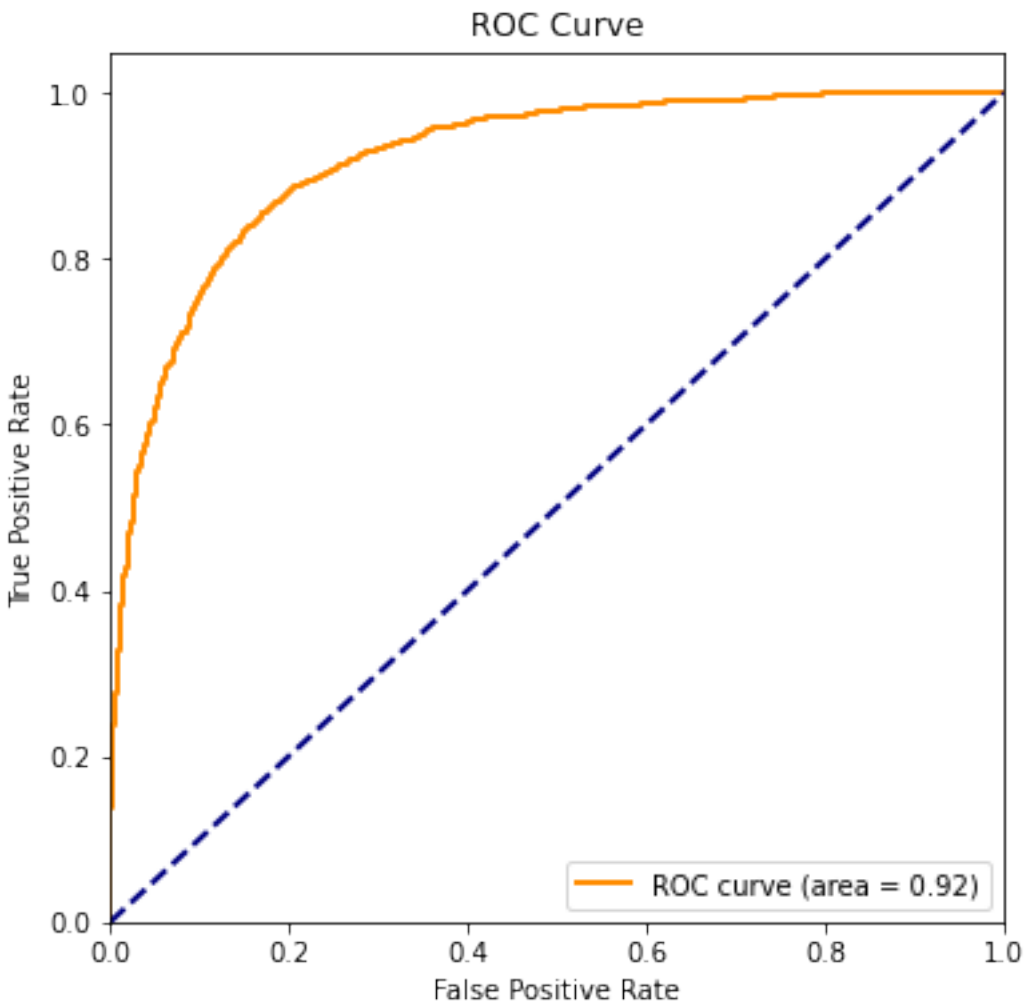
[16]:
```
disp = plot_precision_recall_curve(lr_r2, X_test_r2, y_test_r2)
disp.ax_.set_title('Precision-Recall Curve')
```

[16]:
```
Text(0.5, 1.0, 'Precision-Recall Curve')
```



[17]:
```
y_score_r2_lr = lr_r2.predict_proba(X_test_r2)[:, 1]

fpr, tpr, thersholds = roc_curve(y_test_r2, y_score_r2_lr)
roc_auc = auc(fpr, tpr)

plot_roc_curve(fpr, tpr, roc_auc)
```

```
<Figure size 432x288 with 0 Axes>
```

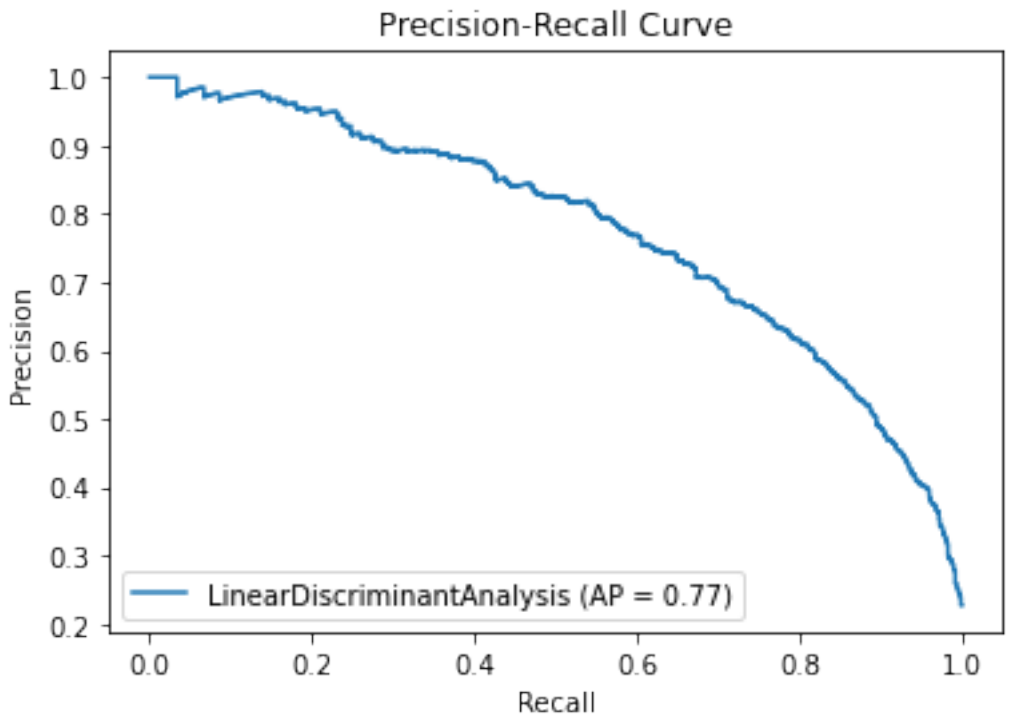


1.2.2 Linear Discriminant Analysis

[18]:
```
lda_r2 = LinearDiscriminantAnalysis().fit(X_train_r2, y_train_r2)
```

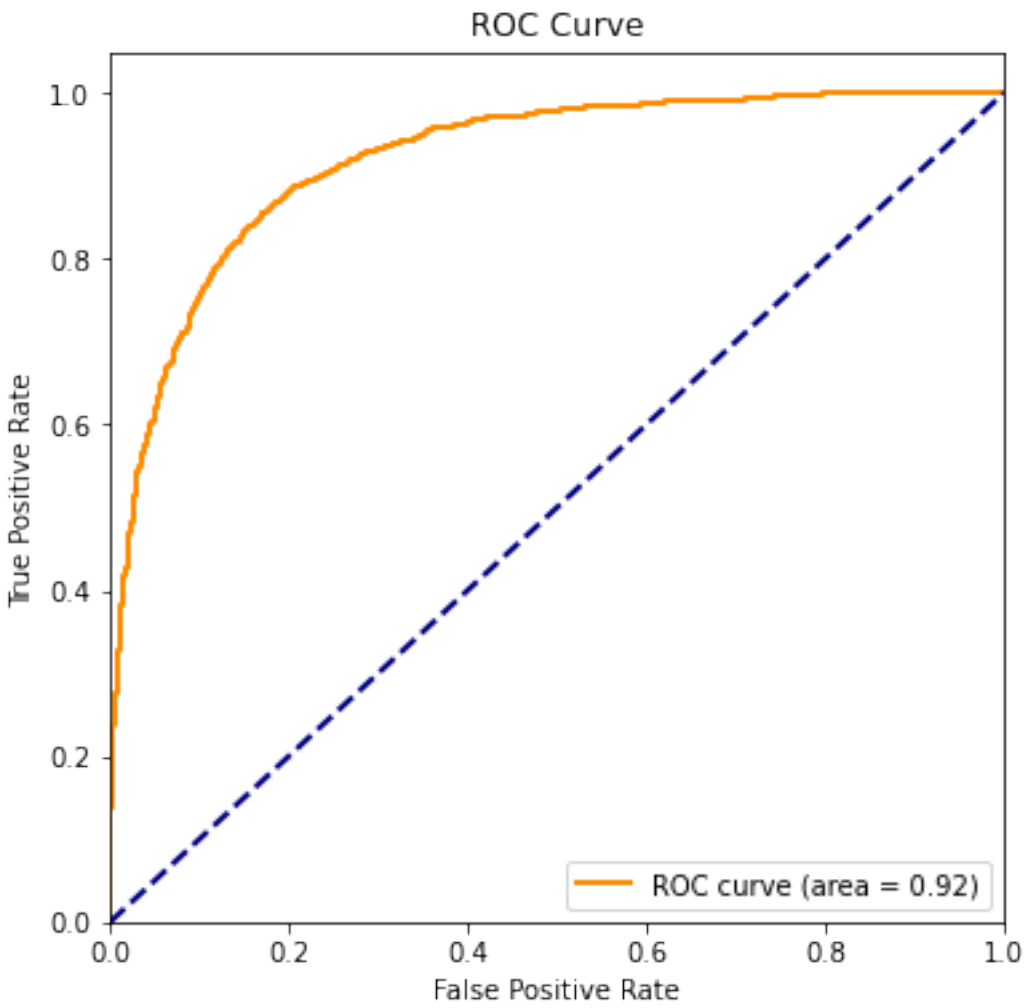
[19]:
```
disp = plot_precision_recall_curve(lda_r2, X_test_r2, y_test_r2)
disp.ax_.set_title('Precision-Recall Curve')
```

[19]:
```
Text(0.5, 1.0, 'Precision-Recall Curve')
```



[20]:
```
y_score_r2_lda = lda_r2.predict_proba(X_test_r2)[:, 1]

fpr, tpr, thersholds = roc_curve(y_test_r2, y_score_r2_lda)
roc_auc = auc(fpr, tpr)

plot_roc_curve(fpr, tpr, roc_auc)
```

```
<Figure size 432x288 with 0 Axes>
```

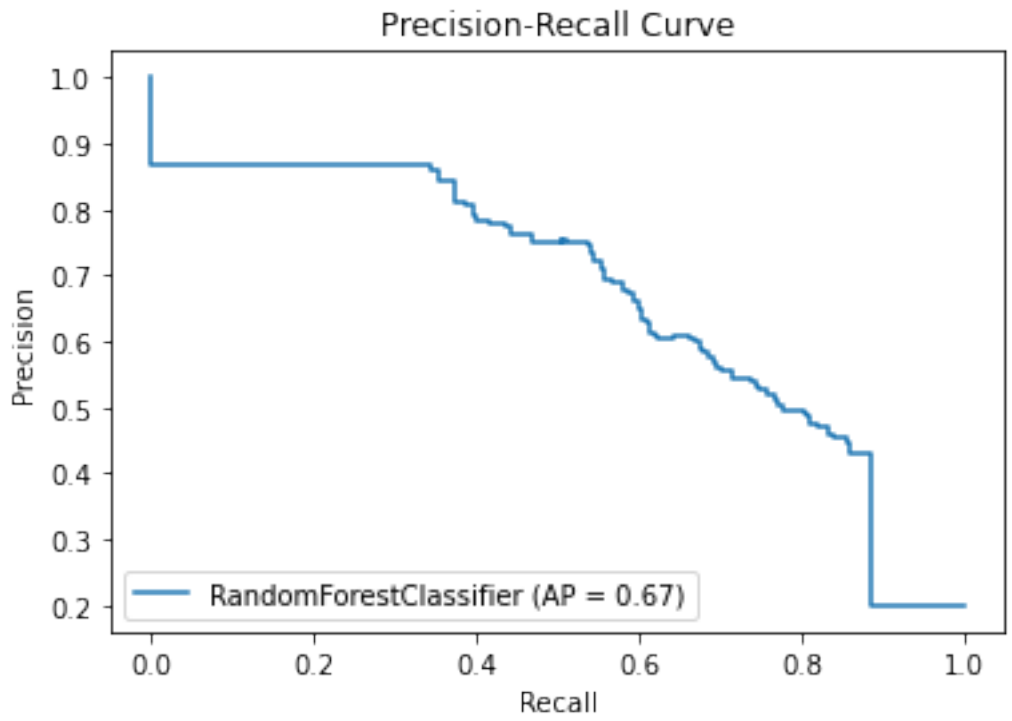


1.2.3 Random Forrest Classifier

[21]:
```
rfc_r2 = RandomForestClassifier().fit(X_train_r2, y_train_r2)
```

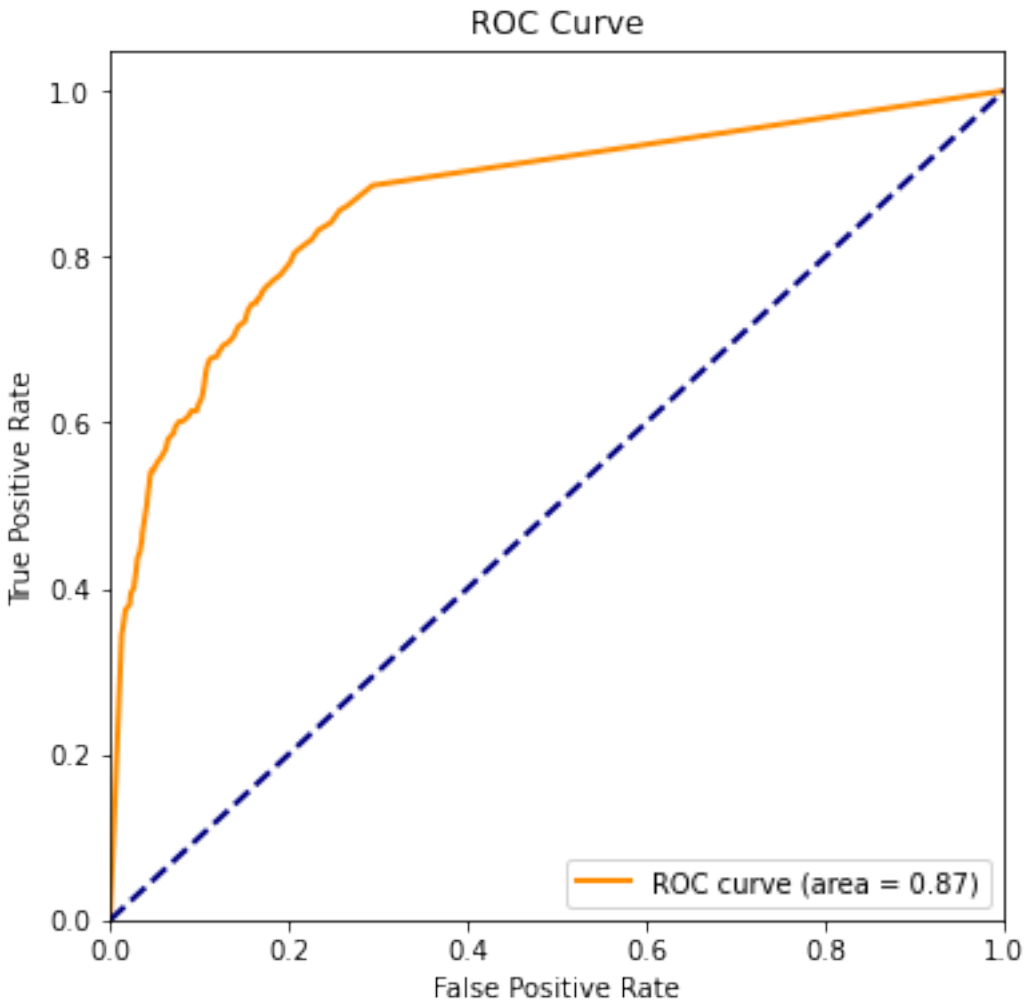
[22]:
```
disp = plot_precision_recall_curve(rfc_r2, X_test_r2, y_test_r2)
disp.ax_.set_title('Precision-Recall Curve')
```

[22]:
```
Text(0.5, 1.0, 'Precision-Recall Curve')
```



[23]:
```
y_score_r2_rfc = rfc_r2.predict_proba(X_test_r2)[:, 1]

fpr, tpr, thersholds = roc_curve(y_test_r2, y_score_r2_rfc)
roc_auc = auc(fpr, tpr)

plot_roc_curve(fpr, tpr, roc_auc)
```

```
<Figure size 432x288 with 0 Axes>
```



1.3 n_0/n_1 ratio = 9:

[24]:
```
ratio_r3 = 9

n0_train_r3 = n1_train * ratio_r3
X0_train_r3 = np.random.normal(mu_0, sigma, n0_train_r3)

n0_test_r3 = n1_test * ratio_r3
X0_test_r3 = np.random.normal(mu_0, sigma, n0_test_r3)

## np.r_ is used to concatenate any number of array slices along row axis
X_train_r3 = np.r_[X1_train, X0_train_r3].reshape(-1, 1)
X_test_r3 = np.r_[X1_test, X0_test_r3].reshape(-1, 1)

## [0]*10 will produce [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

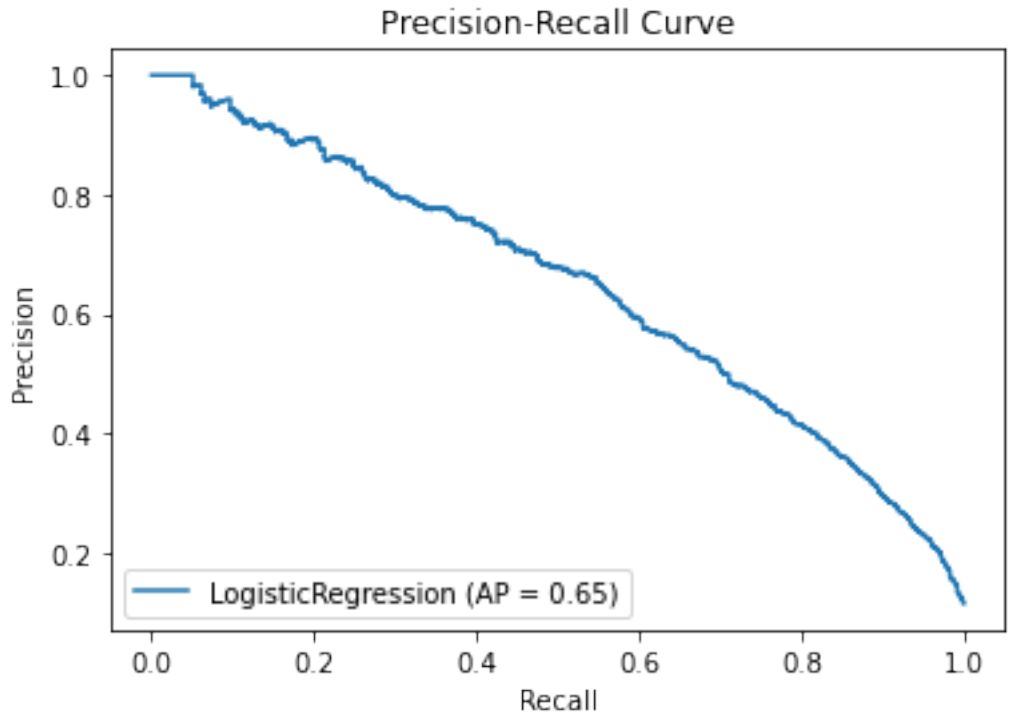
```
y_train_r3 = np.r_[[1]*n1_train, [0]*n0_train_r3]
y_test_r3 = np.r_[[1]*n1_test, [0]*n0_test_r3]
```

1.3.1 Logistic Regression

[25]:
```
lr_r3 = LogisticRegression(penalty='none').fit(X_train_r3, y_train_r3)
```

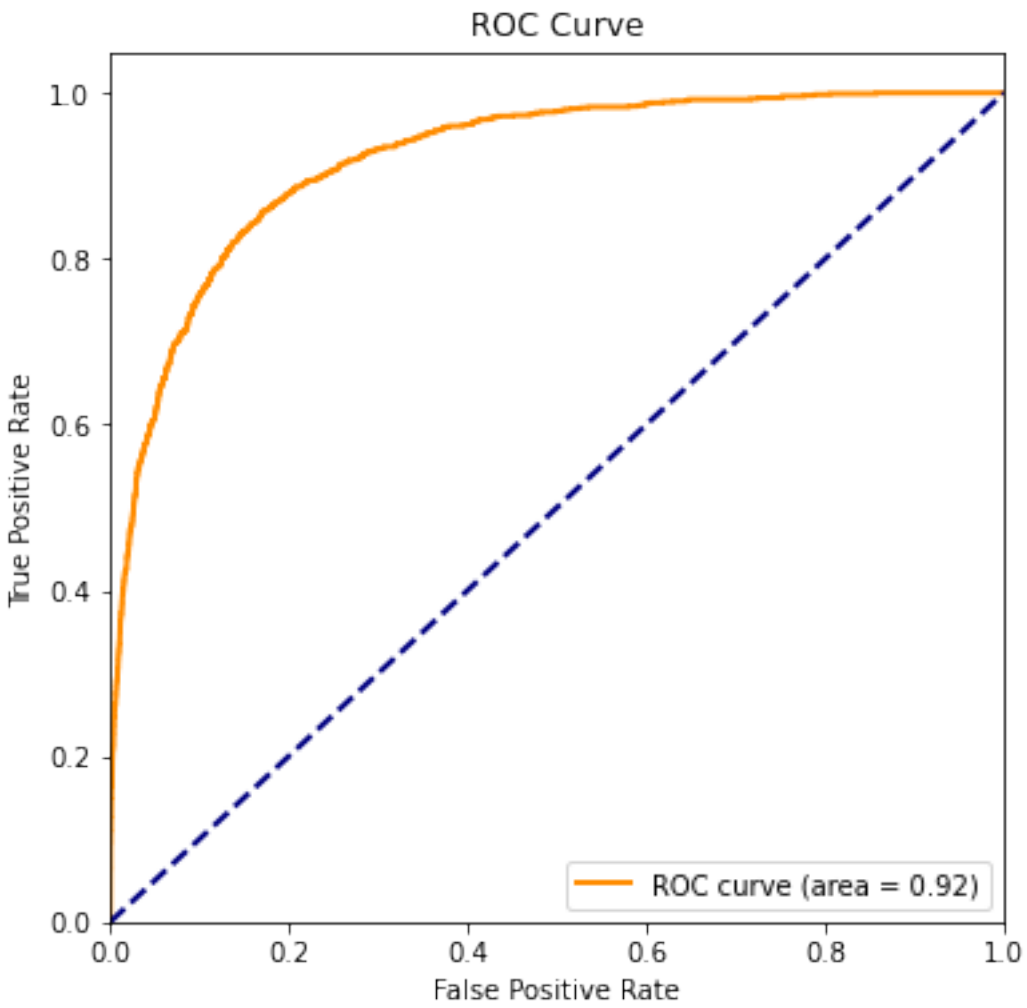
[26]:
```
disp = plot_precision_recall_curve(lr_r3, X_test_r3, y_test_r3)
disp.ax_.set_title('Precision-Recall Curve')
```

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



[27]:
```
y_score_r3_lr = lr_r3.predict_proba(X_test_r3)[:, 1]

fpr, tpr, thersholds = roc_curve(y_test_r3, y_score_r3_lr)
roc_auc = auc(fpr, tpr)

plot_roc_curve(fpr, tpr, roc_auc)
```

```
<Figure size 432x288 with 0 Axes>
```

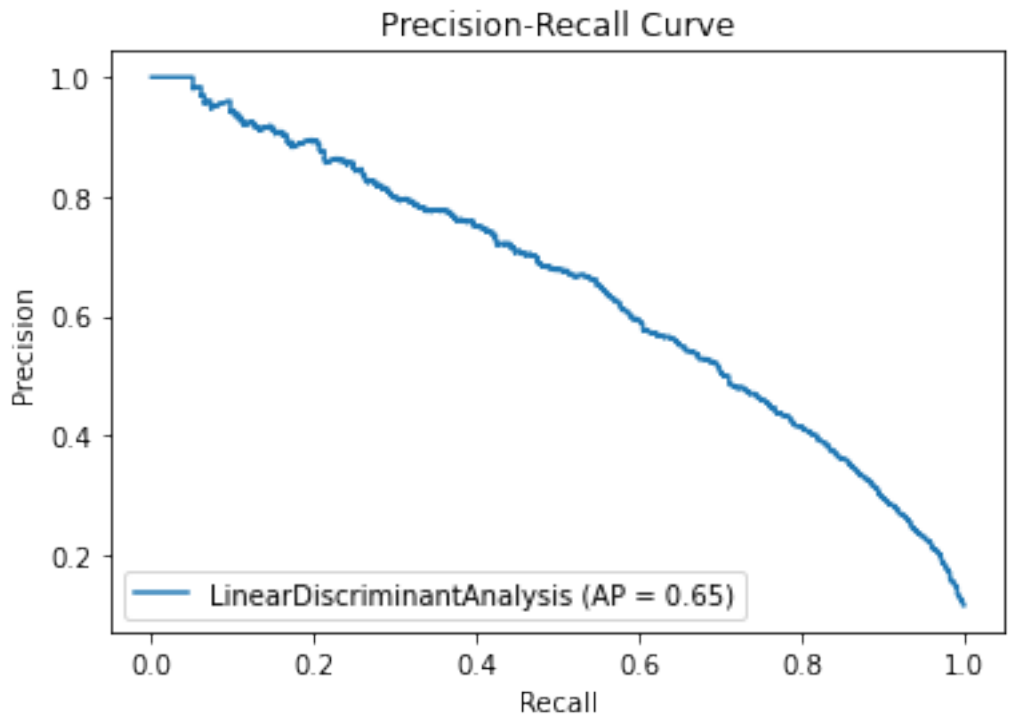


1.3.2 Linear Discriminant Analysis

[28]:
```
lda_r3 = LinearDiscriminantAnalysis().fit(X_train_r3, y_train_r3)
```

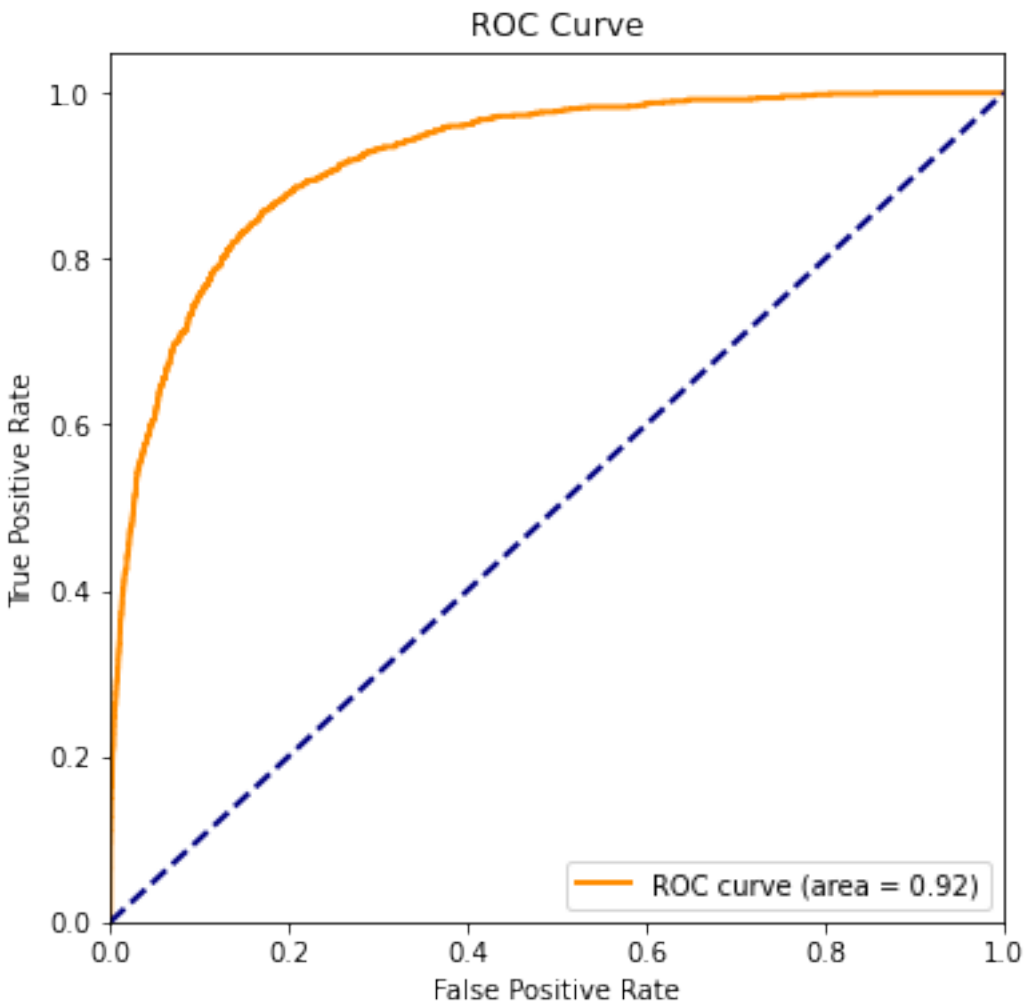
[29]:
```
disp = plot_precision_recall_curve(lda_r3, X_test_r3, y_test_r3)
disp.ax_.set_title('Precision-Recall Curve')
```

[29]:
```
Text(0.5, 1.0, 'Precision-Recall Curve')
```



[30]:
```
y_score_r3_lda = lda_r3.predict_proba(X_test_r3)[:, 1]

fpr, tpr, thersholds = roc_curve(y_test_r3, y_score_r3_lda)
roc_auc = auc(fpr, tpr)

plot_roc_curve(fpr, tpr, roc_auc)
```

```
<Figure size 432x288 with 0 Axes>
```

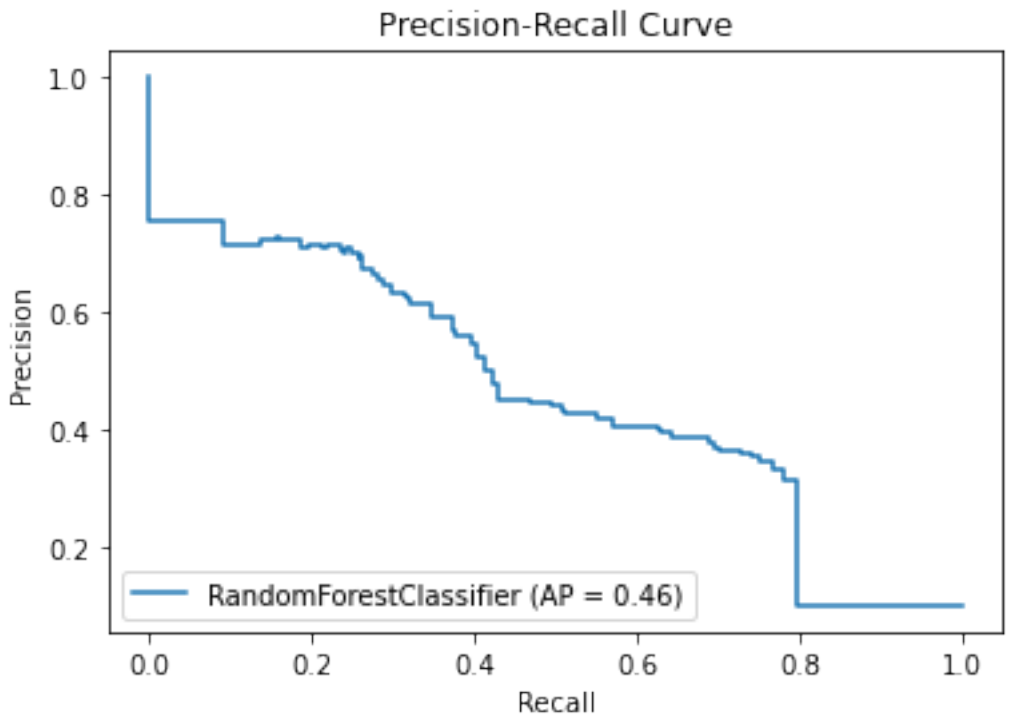


1.3.3 Random Forrest Classifier

[31]:
```
rfc_r3 = RandomForestClassifier().fit(X_train_r3, y_train_r3)
```

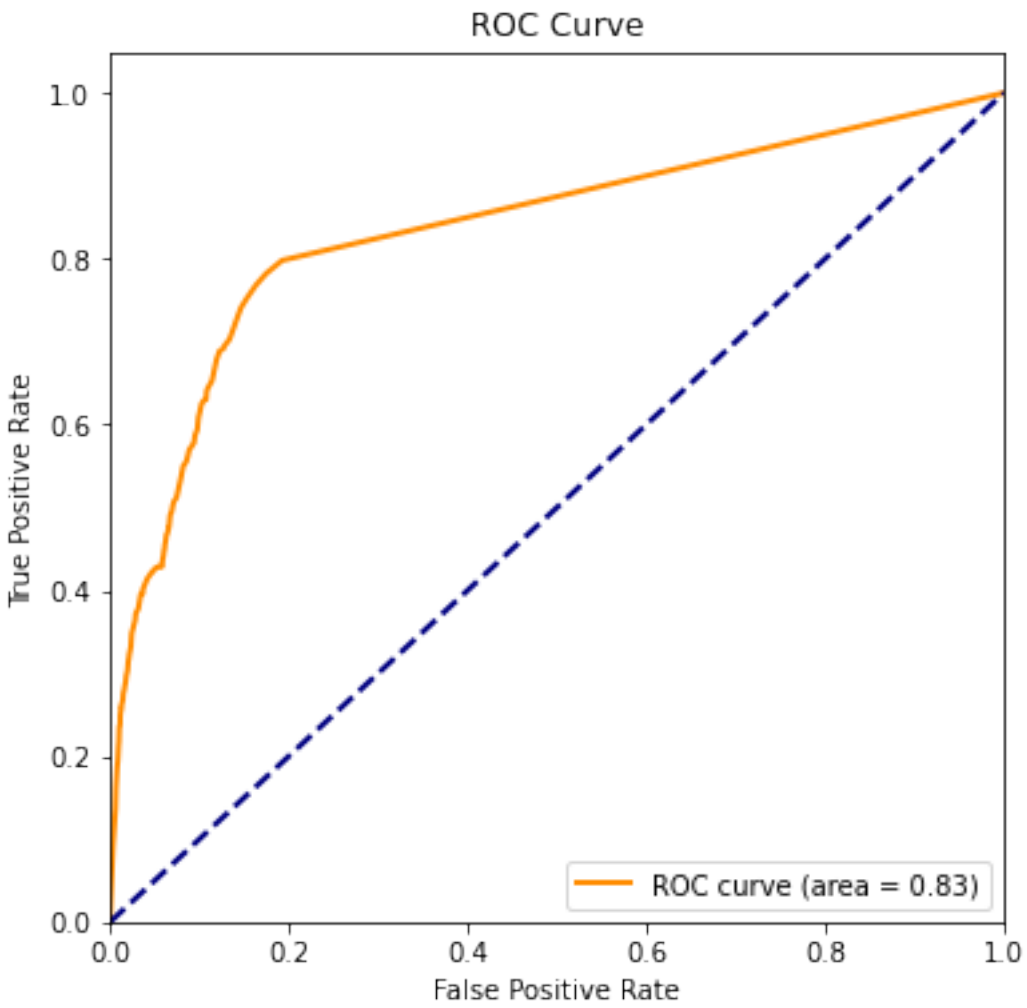
[32]:
```
disp = plot_precision_recall_curve(rfc_r3, X_test_r3, y_test_r3)
disp.ax_.set_title('Precision-Recall Curve')
```

[32]:
```
Text(0.5, 1.0, 'Precision-Recall Curve')
```



[33]:
```
y_score_r3_rfc = rfc_r3.predict_proba(X_test_r3)[:, 1]

fpr, tpr, thersholds = roc_curve(y_test_r3, y_score_r3_rfc)
roc_auc = auc(fpr, tpr)

plot_roc_curve(fpr, tpr, roc_auc)
```

```
<Figure size 432x288 with 0 Axes>
```



2 Example 2

In example 2, we use normal distribution N(μ_0,Σ) and N(μ_1,Σ) to generate the data of class 0 and class 1 separately.

$$\mu_0 = (0,\ 0,\ 0)^T,\ \mu_1 = (1,\ 2,\ 1)^T,\ \Sigma = \begin{pmatrix} 1 & 0.2 & 0 \\ 0.2 & 1 & 0.2 \\ 0 & 0.2 & 1 \end{pmatrix}$$

[34]:
```
mu_0 = [0, 0, 0]
mu_1 = [1, 2, 1]
sigma = [[1, 0.2, 0], [0.2, 1, 0.2], [0, 0.2, 1]]

n1_train = 100
X1_train = np.random.multivariate_normal(mu_1, sigma, n1_train)
```

```
n1_test = 1000
X1_test = np.random.multivariate_normal(mu_1, sigma, n1_test)
```

2.1 n_0/n_1 ratio = 1:

[35]:
```
ratio_r1 = 1

n0_train_r1 = n1_train * ratio_r1
X0_train_r1 = np.random.multivariate_normal(mu_0, sigma, n0_train_r1)

n0_test_r1 = n1_test * ratio_r1
X0_test_r1 = np.random.multivariate_normal(mu_0, sigma, n0_test_r1)

## np.r_ is used to concatenate any number of array slices along row axis
X_train_r1 = np.r_[X1_train, X0_train_r1]
X_test_r1 = np.r_[X1_test, X0_test_r1]

## [0]*10 will produce [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
y_train_r1 = np.r_[[1]*n1_train, [0]*n0_train_r1]
y_test_r1 = np.r_[[1]*n1_test, [0]*n0_test_r1]
```

[36]:
```
X_train_r1.shape
```

[36]:
```
(200, 3)
```

[37]:
```
X_test_r1.shape
```

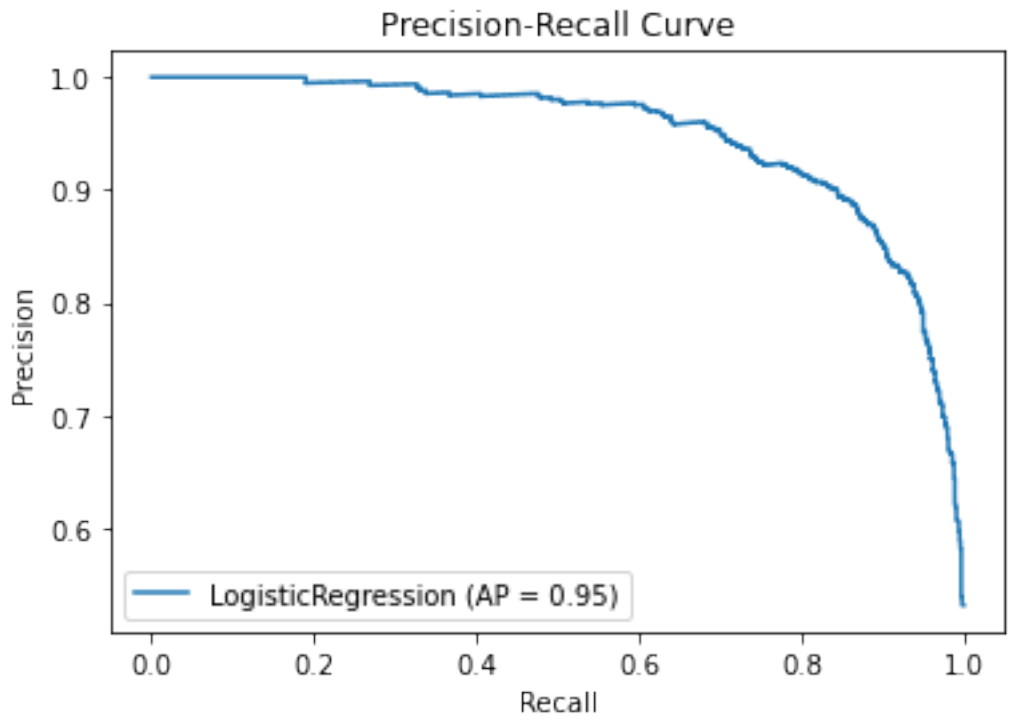
[37]:
```
(2000, 3)
```

2.1.1 Logistic Regression

[38]:
```
lr_r1 = LogisticRegression(penalty='none').fit(X_train_r1, y_train_r1)
```

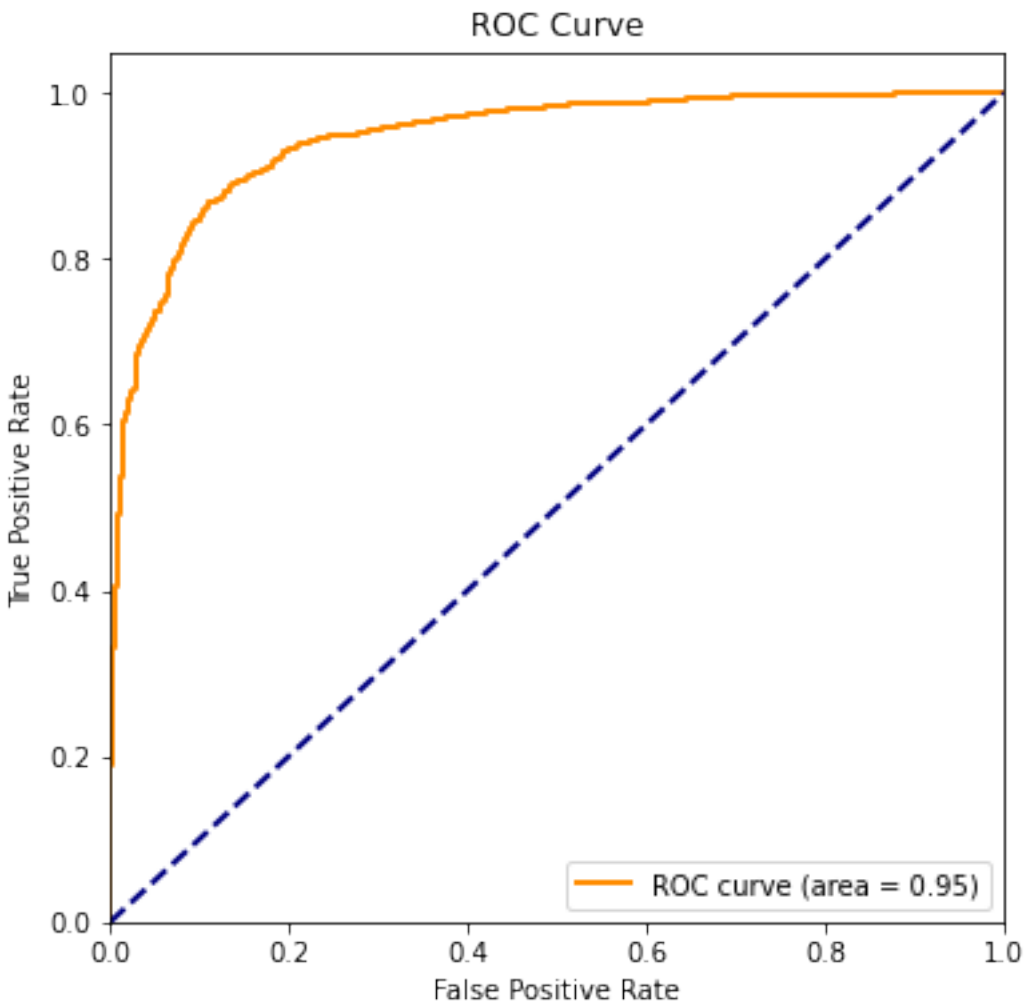
[39]:
```
disp = plot_precision_recall_curve(lr_r1, X_test_r1, y_test_r1)
disp.ax_.set_title('Precision-Recall Curve')
```

[39]:
```
Text(0.5, 1.0, 'Precision-Recall Curve')
```



[40]:
```
y_score_r1_lr = lr_r1.predict_proba(X_test_r1)[:, 1]

fpr, tpr, thersholds = roc_curve(y_test_r1, y_score_r1_lr)
roc_auc = auc(fpr, tpr)

plot_roc_curve(fpr, tpr, roc_auc)
```

```
<Figure size 432x288 with 0 Axes>
```

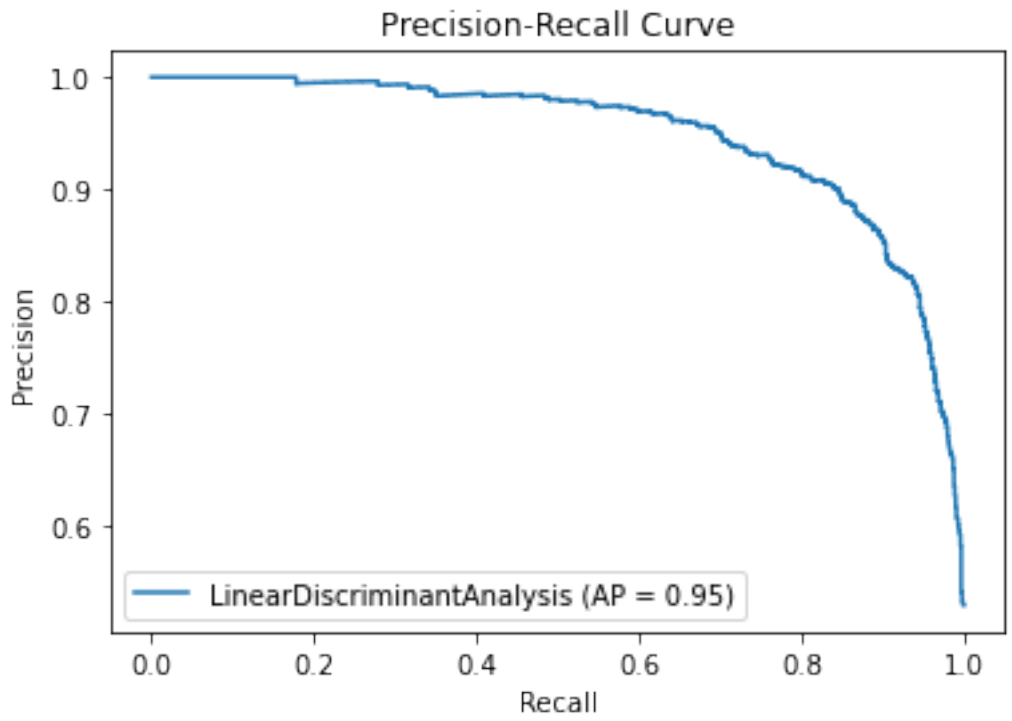


2.1.2 Linear Discriminant Analysis

[41]:
```
lda_r1 = LinearDiscriminantAnalysis().fit(X_train_r1, y_train_r1)
```

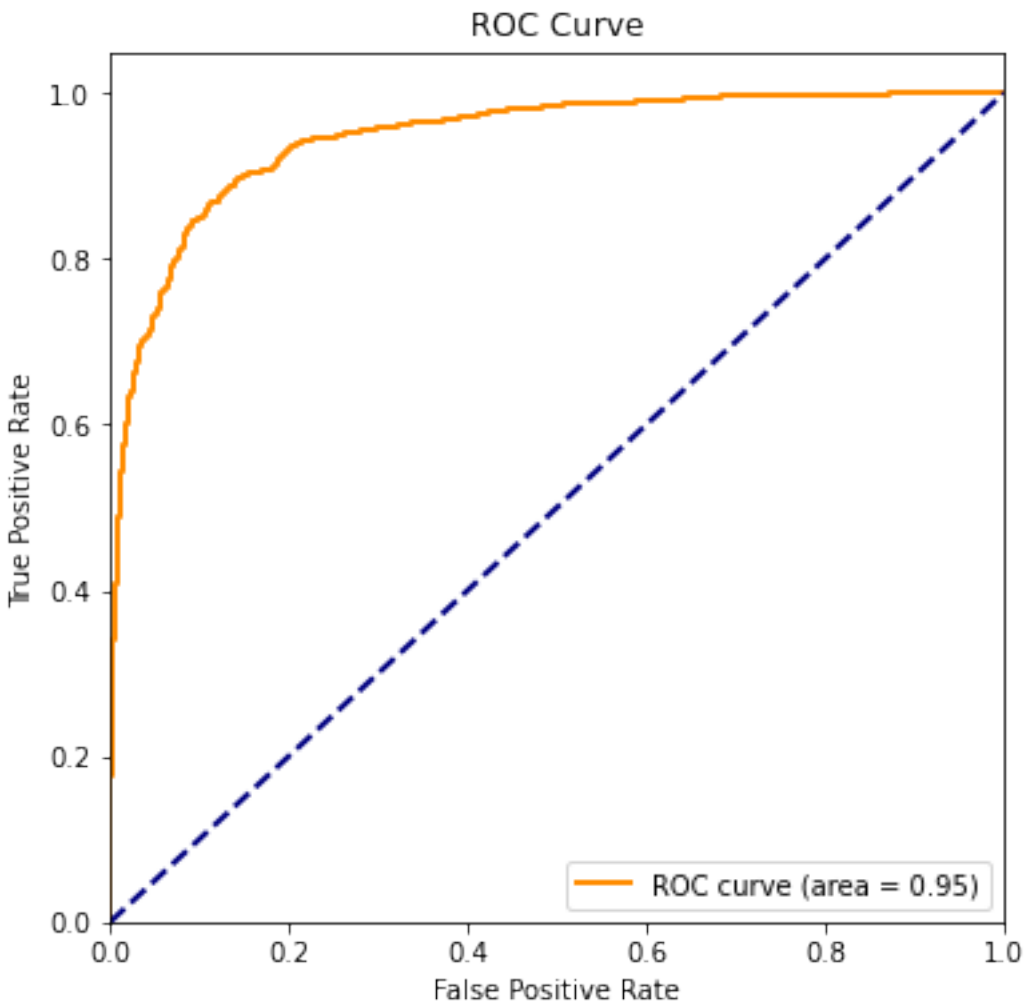
[42]:
```
disp = plot_precision_recall_curve(lda_r1, X_test_r1, y_test_r1)
disp.ax_.set_title('Precision-Recall Curve')
```

[42]:
```
Text(0.5, 1.0, 'Precision-Recall Curve')
```



[43]:
```
y_score_r1_lda = lda_r1.predict_proba(X_test_r1)[:, 1]

fpr, tpr, thersholds = roc_curve(y_test_r1, y_score_r1_lda)
roc_auc = auc(fpr, tpr)

plot_roc_curve(fpr, tpr, roc_auc)
```

```
<Figure size 432x288 with 0 Axes>
```

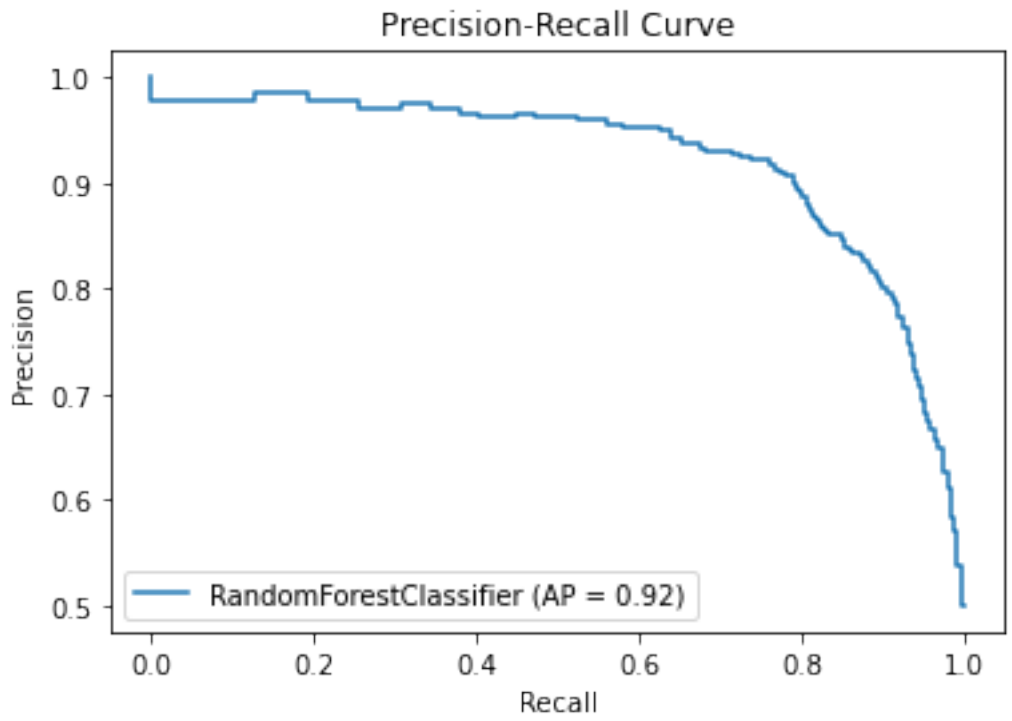


2.1.3 Random Forrest Classifier

[44]:
```
rfc_r1 = RandomForestClassifier().fit(X_train_r1, y_train_r1)
```

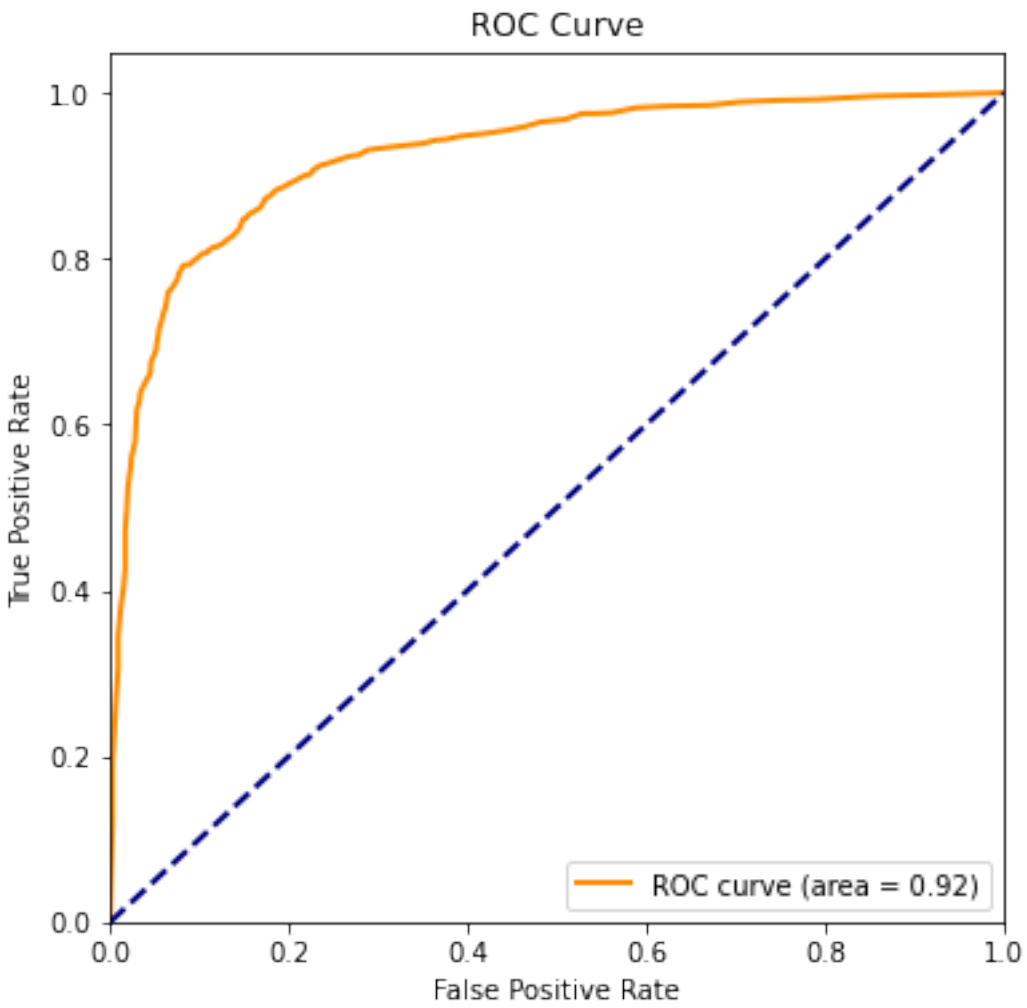
[45]:
```
disp = plot_precision_recall_curve(rfc_r1, X_test_r1, y_test_r1)
disp.ax_.set_title('Precision-Recall Curve')
```

[45]:
```
Text(0.5, 1.0, 'Precision-Recall Curve')
```



[46]:
```
y_score_r1_rfc = rfc_r1.predict_proba(X_test_r1)[:, 1]

fpr, tpr, thersholds = roc_curve(y_test_r1, y_score_r1_rfc)
roc_auc = auc(fpr, tpr)

plot_roc_curve(fpr, tpr, roc_auc)
```

```
<Figure size 432x288 with 0 Axes>
```



2.2 n_0/n_1 ratio = 4:

[47]:
```
ratio_r2 = 4

n0_train_r2 = n1_train * ratio_r2
X0_train_r2 = np.random.multivariate_normal(mu_0, sigma, n0_train_r2)

n0_test_r2 = n1_test * ratio_r2
X0_test_r2 = np.random.multivariate_normal(mu_0, sigma, n0_test_r2)

X_train_r2 = np.r_[X1_train, X0_train_r2]
X_test_r2 = np.r_[X1_test, X0_test_r2]

## [0]*10 will produce [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
y_train_r2 = np.r_[[1]*n1_train, [0]*n0_train_r2]
```

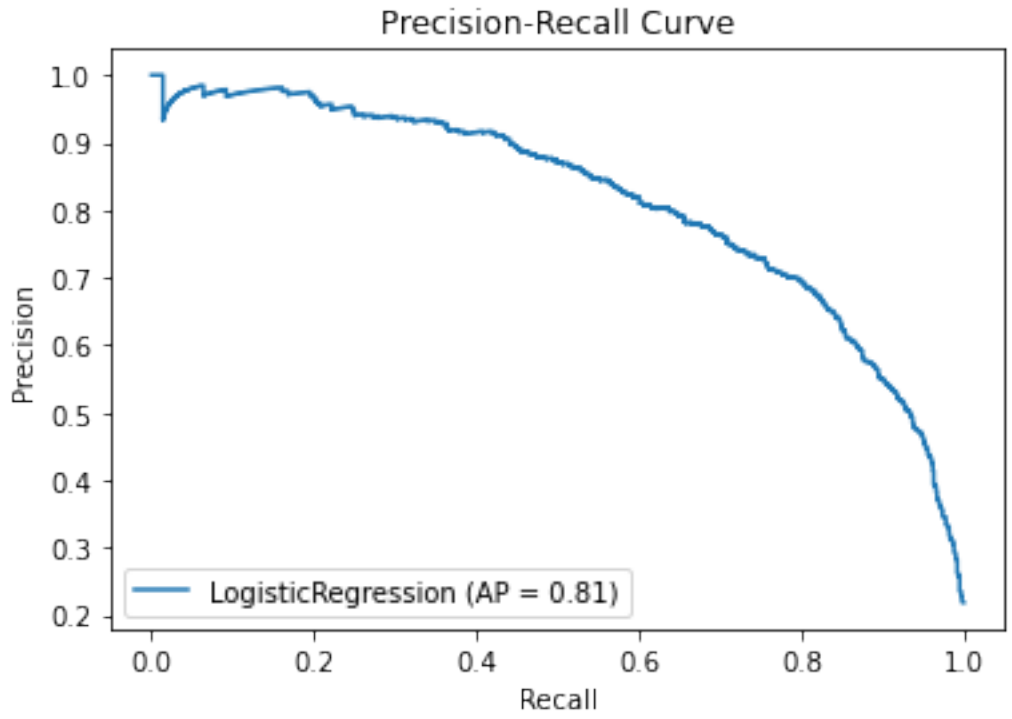
```
y_test_r2 = np.r_[[1]*n1_test, [0]*n0_test_r2]
```

2.2.1 Logistic Regression

[48]:
```
lr_r2 = LogisticRegression(penalty='none').fit(X_train_r2, y_train_r2)
```

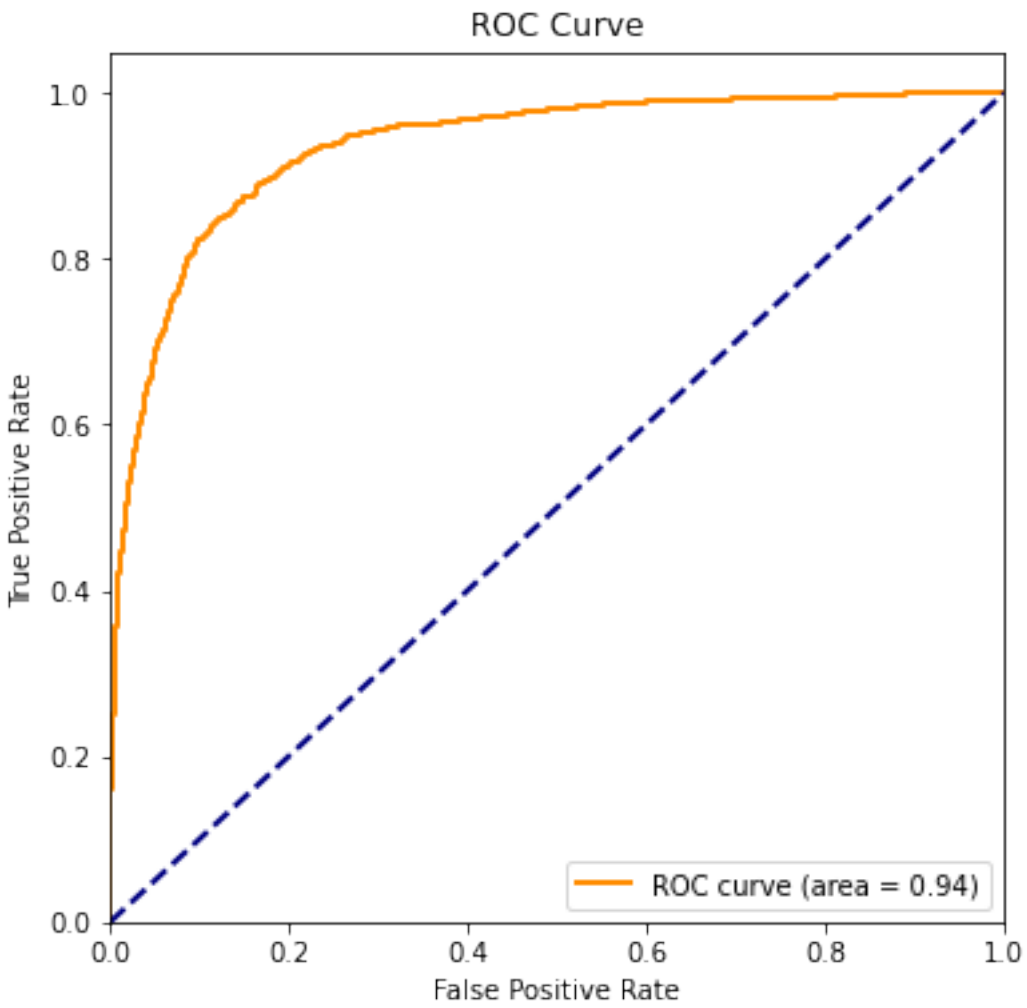
[49]:
```
disp = plot_precision_recall_curve(lr_r2, X_test_r2, y_test_r2)
disp.ax_.set_title('Precision-Recall Curve')
```

[49]:
```
Text(0.5, 1.0, 'Precision-Recall Curve')
```



[50]:
```
y_score_r2_lr = lr_r2.predict_proba(X_test_r2)[:, 1]

fpr, tpr, thersholds = roc_curve(y_test_r2, y_score_r2_lr)
roc_auc = auc(fpr, tpr)

plot_roc_curve(fpr, tpr, roc_auc)
```

```
<Figure size 432x288 with 0 Axes>
```

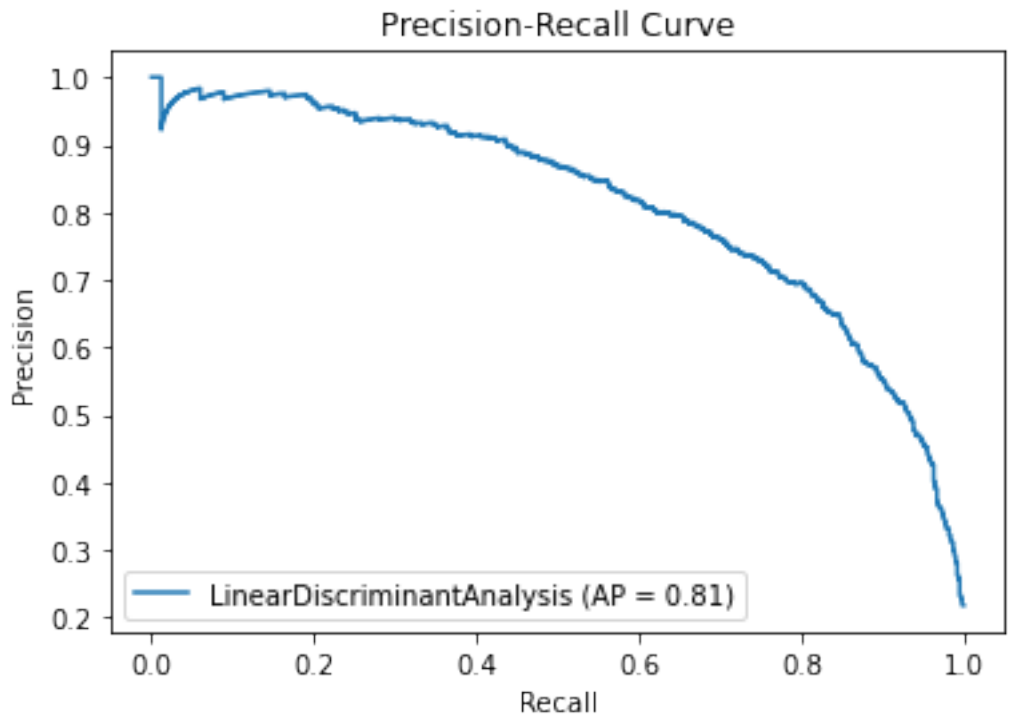


2.2.2 Linear Discriminant Analysis

[51]:
```
lda_r2 = LinearDiscriminantAnalysis().fit(X_train_r2, y_train_r2)
```

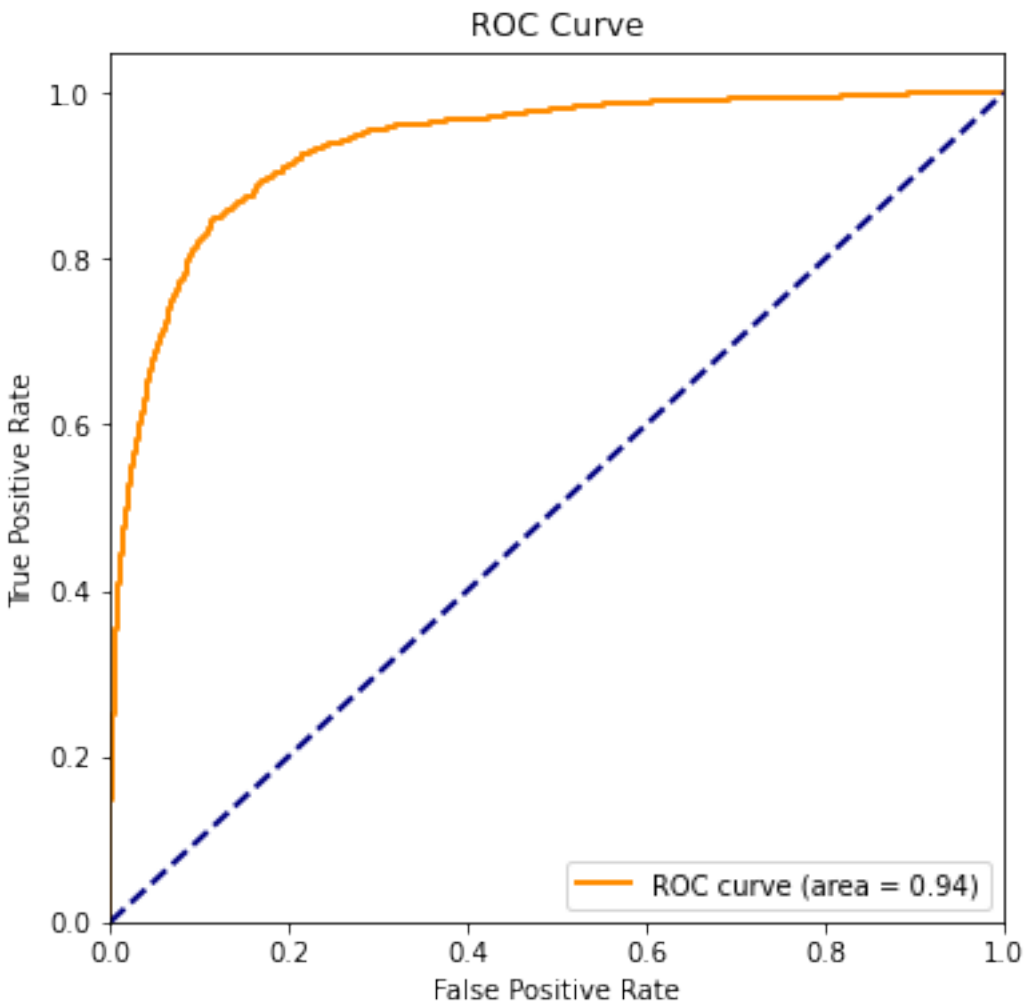
[52]:
```
disp = plot_precision_recall_curve(lda_r2, X_test_r2, y_test_r2)
disp.ax_.set_title('Precision-Recall Curve')
```

[52]:
```
Text(0.5, 1.0, 'Precision-Recall Curve')
```



[53]:
```
y_score_r2_lda = lda_r2.predict_proba(X_test_r2)[:, 1]

fpr, tpr, thersholds = roc_curve(y_test_r2, y_score_r2_lda)
roc_auc = auc(fpr, tpr)

plot_roc_curve(fpr, tpr, roc_auc)
```

```
<Figure size 432x288 with 0 Axes>
```

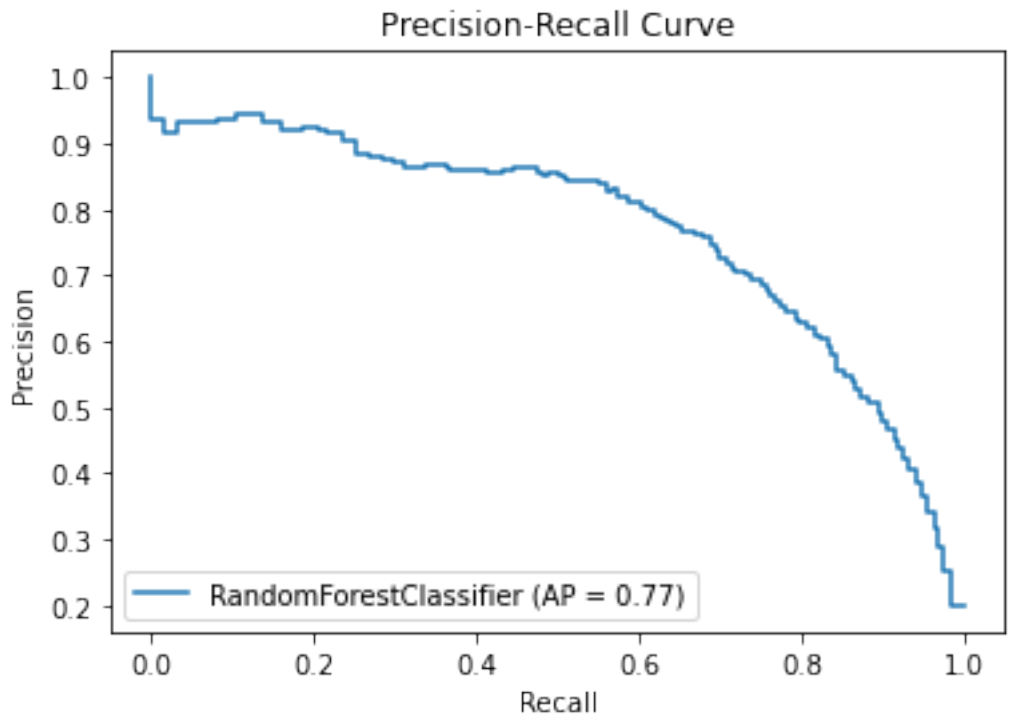


2.2.3 Random Forrest Classifier

[54]:
```
rfc_r2 = RandomForestClassifier().fit(X_train_r2, y_train_r2)
```

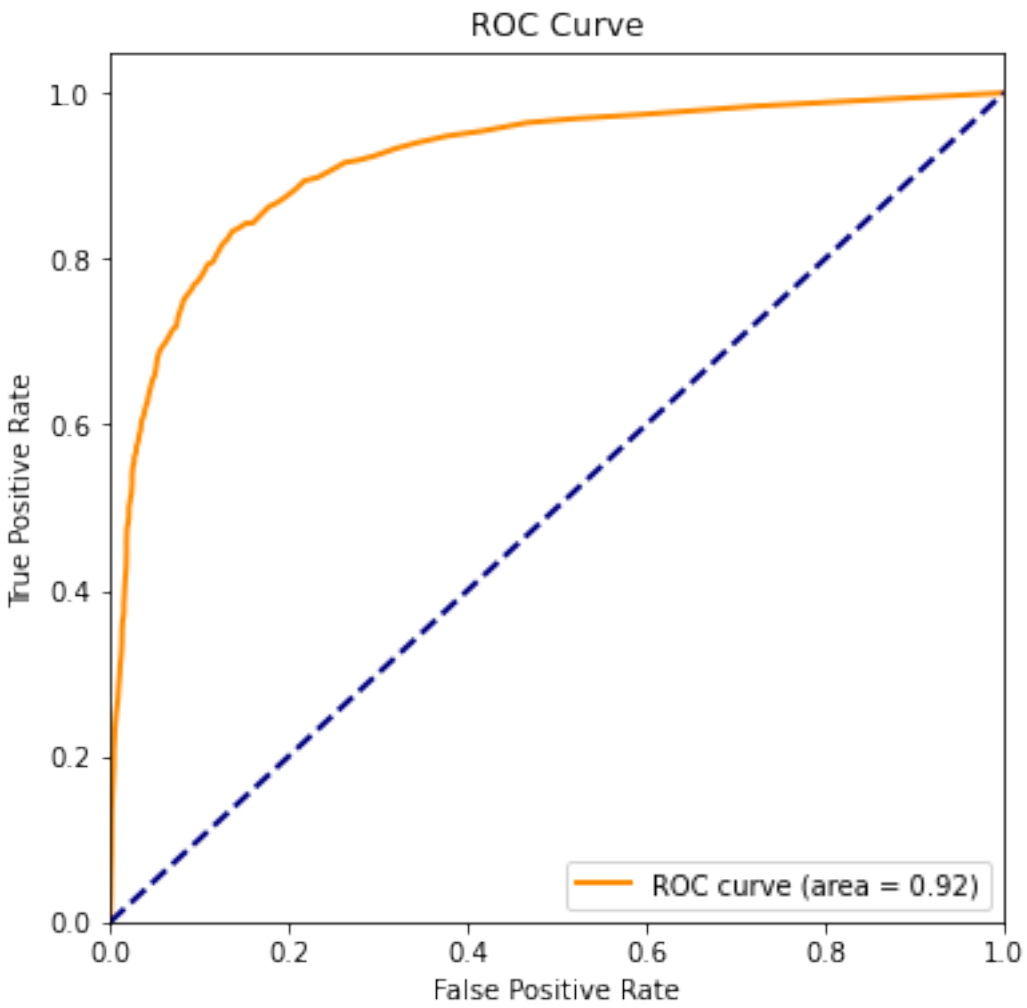
[55]:
```
disp = plot_precision_recall_curve(rfc_r2, X_test_r2, y_test_r2)
disp.ax_.set_title('Precision-Recall Curve')
```

[55]:
```
Text(0.5, 1.0, 'Precision-Recall Curve')
```



[56]:
```
y_score_r2_rfc = rfc_r2.predict_proba(X_test_r2)[:, 1]

fpr, tpr, thersholds = roc_curve(y_test_r2, y_score_r2_rfc)
roc_auc = auc(fpr, tpr)

plot_roc_curve(fpr, tpr, roc_auc)
```

```
<Figure size 432x288 with 0 Axes>
```



2.3 n_0/n_1 ratio = 9:

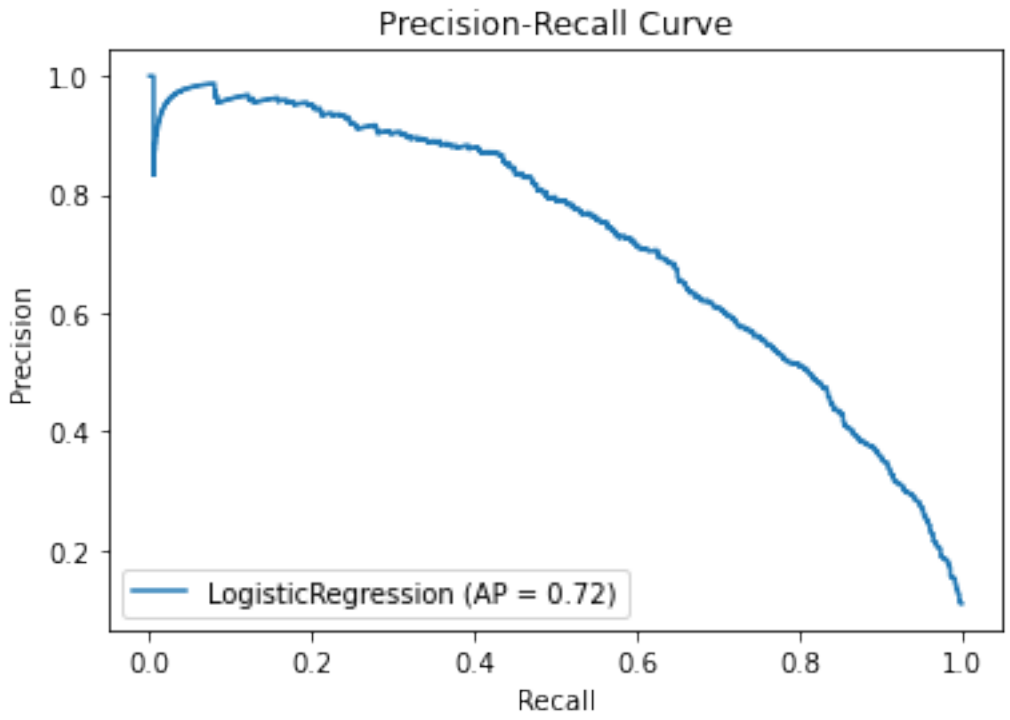
[57]:
```
ratio_r3 = 9

n0_train_r3 = n1_train * ratio_r3
X0_train_r3 = np.random.multivariate_normal(mu_0, sigma, n0_train_r3)

n0_test_r3 = n1_test * ratio_r3
X0_test_r3 = np.random.multivariate_normal(mu_0, sigma, n0_test_r3)

X_train_r3 = np.r_[X1_train, X0_train_r3]
X_test_r3 = np.r_[X1_test, X0_test_r3]

y_train_r3 = np.r_[[1]*n1_train, [0]*n0_train_r3]
y_test_r3 = np.r_[[1]*n1_test, [0]*n0_test_r3]
```

2.3.1 Logistic Regression

[58]:
```
lr_r3 = LogisticRegression(penalty='none').fit(X_train_r3, y_train_r3)
```

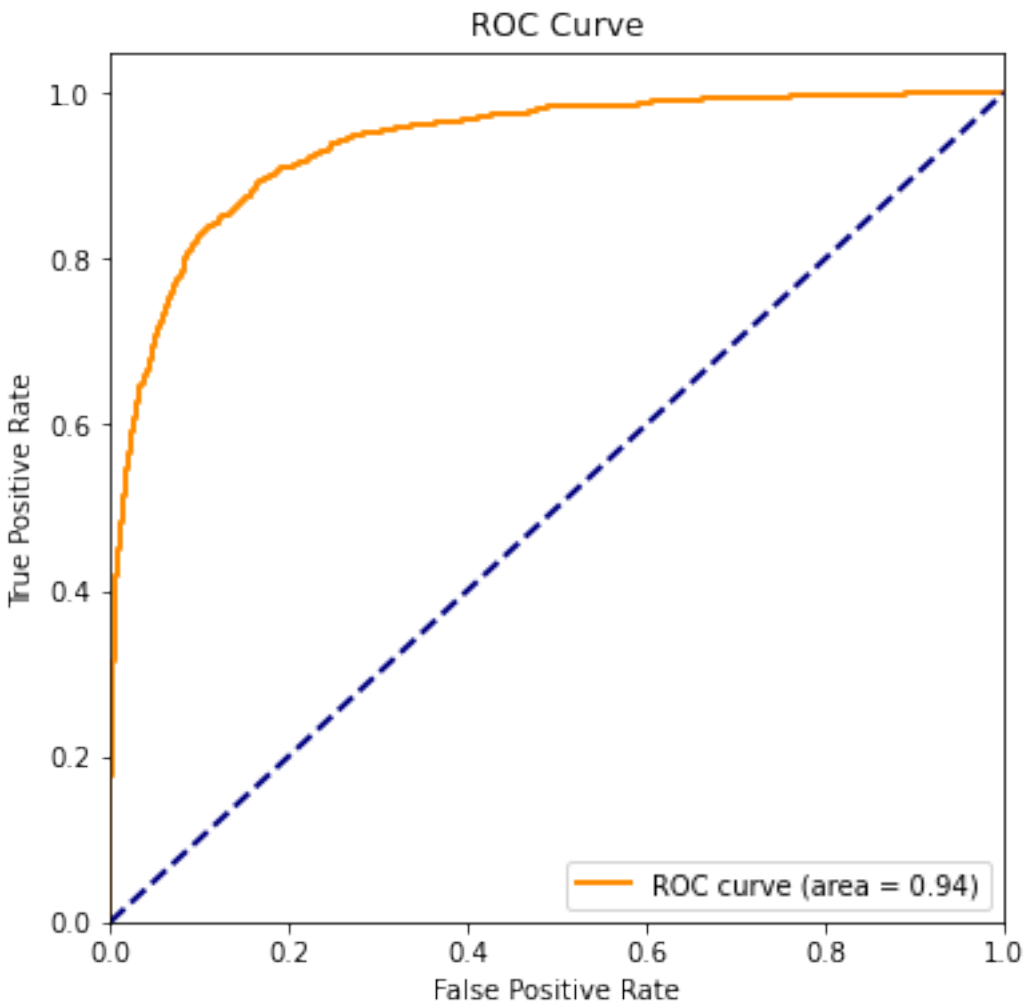
[59]:
```
disp = plot_precision_recall_curve(lr_r3, X_test_r3, y_test_r3)
disp.ax_.set_title('Precision-Recall Curve')
```

[59]:
```
Text(0.5, 1.0, 'Precision-Recall Curve')
```



[60]:
```
y_score_r3_lr = lr_r3.predict_proba(X_test_r3)[:, 1]

fpr, tpr, thersholds = roc_curve(y_test_r3, y_score_r3_lr)
roc_auc = auc(fpr, tpr)

plot_roc_curve(fpr, tpr, roc_auc)
```

```
<Figure size 432x288 with 0 Axes>
```

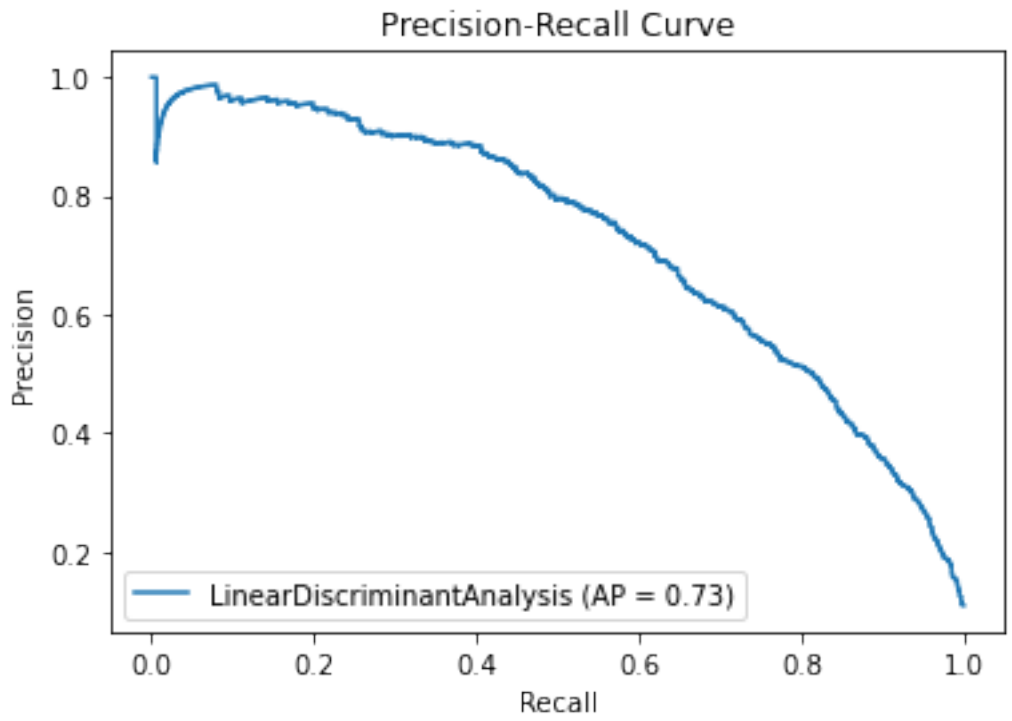


2.3.2 Linear Discriminant Analysis

[61]:
```
lda_r3 = LinearDiscriminantAnalysis().fit(X_train_r3, y_train_r3)
```

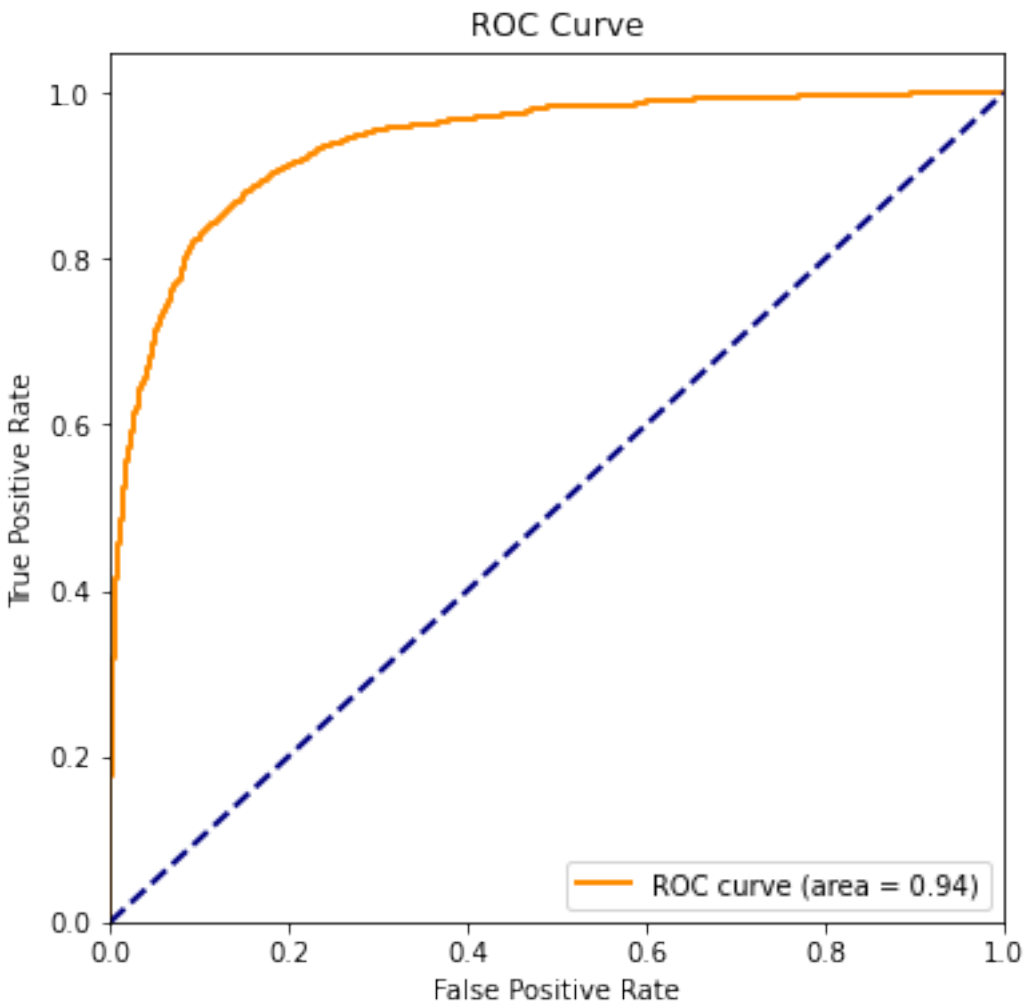
[62]:
```
disp = plot_precision_recall_curve(lda_r3, X_test_r3, y_test_r3)
disp.ax_.set_title('Precision-Recall Curve')
```

[62]:
```
Text(0.5, 1.0, 'Precision-Recall Curve')
```



[63]:
```
y_score_r3_lda = lda_r3.predict_proba(X_test_r3)[:, 1]

fpr, tpr, thersholds = roc_curve(y_test_r3, y_score_r3_lda)
roc_auc = auc(fpr, tpr)

plot_roc_curve(fpr, tpr, roc_auc)
```

```
<Figure size 432x288 with 0 Axes>
```

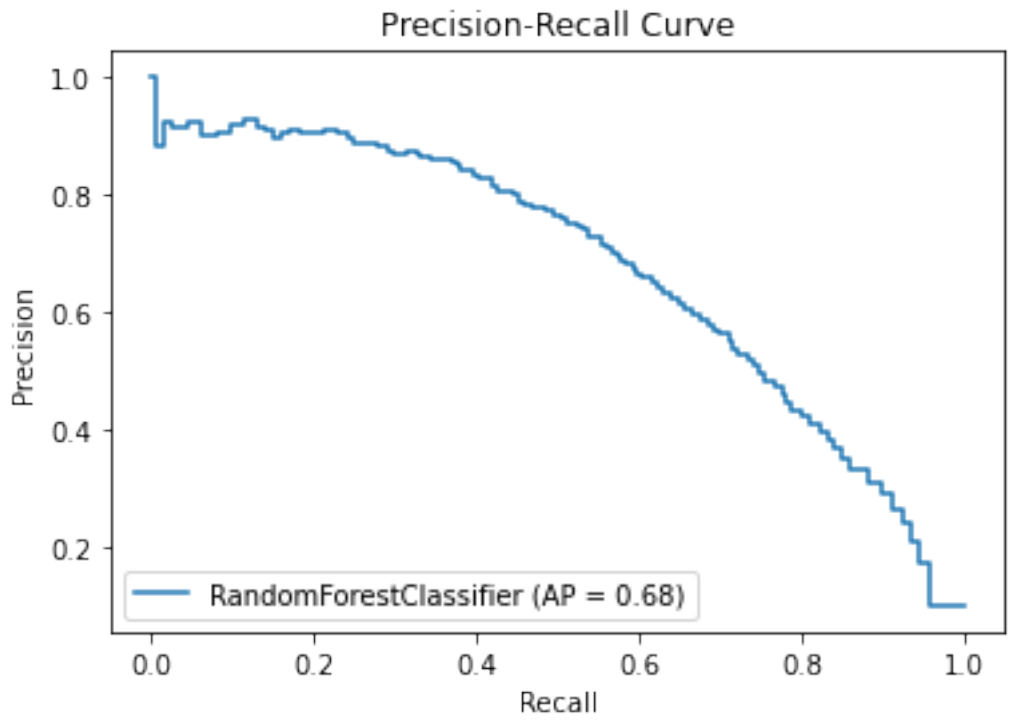


2.3.3 Random Forrest Classifier

[64]:
```
rfc_r3 = RandomForestClassifier().fit(X_train_r3, y_train_r3)
```

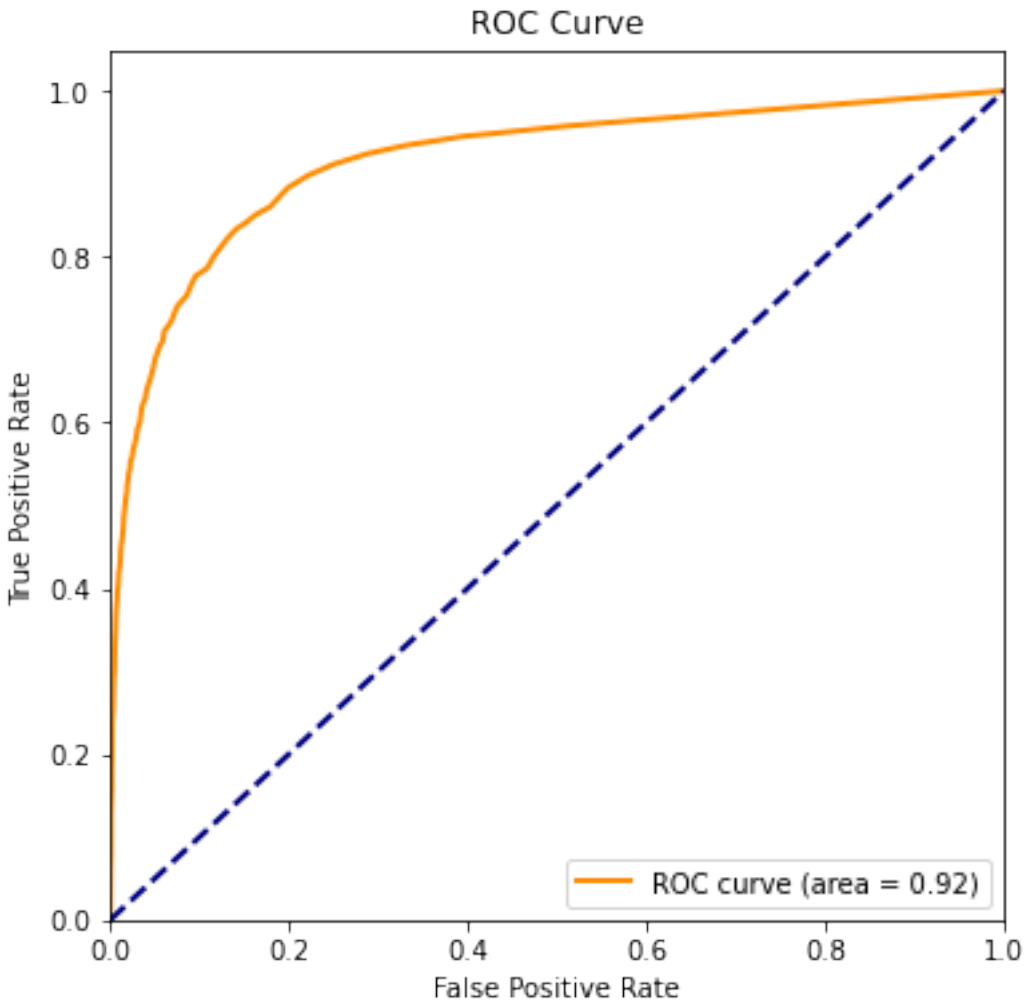
[65]:
```
disp = plot_precision_recall_curve(rfc_r3, X_test_r3, y_test_r3)
disp.ax_.set_title('Precision-Recall Curve')
```

[65]:
```
Text(0.5, 1.0, 'Precision-Recall Curve')
```



[66]:
```
y_score_r3_rfc = rfc_r3.predict_proba(X_test_r3)[:, 1]

fpr, tpr, thersholds = roc_curve(y_test_r3, y_score_r3_rfc)
roc_auc = auc(fpr, tpr)

plot_roc_curve(fpr, tpr, roc_auc)
```

```
<Figure size 432x288 with 0 Axes>
```



Now we have seen that the imbalance ratio impacts the algorithms. In practice, people use various resampling methods, inlcuding under-sampling, over-sampling and hybrid approaches. Which resampling approach to use (or whether one should even consider using a resampling approach) depends on the methods (e.g., logistic regression, random forrest, etc.), the evaluation criterion, as well as the imbalance ratio. When a practicioner has enough time, they might use cross-validation to find a proper resampling approach for their problem at hand.