In [1]:
```
#https://datahub.io/machine-learning/tic-tac-toe-endgame
#https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html
#https://www.youtube.com/watch?v=MRD67WgWonA   (for better understanding of Bagging and Boosting)
print("dataset and its description")
```

```
dataset and its description
```

In [2]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
from sklearn.metrics import classification_report,confusion_matrix ,accuracy_score
from sklearn.metrics import precision_recall_fscore_support,average_precision_score
from sklearn.metrics import precision_score,recall_score,f1_score,precision_recall_curve
```

In [3]:
```
data = pd.read_csv("tictactoe.csv")
```

In [4]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 958 entries, 0 to 957
Data columns (total 10 columns):
top-left-square         958 non-null object
top-middle-square       958 non-null object
top-right-square        958 non-null object
middle-left-square      958 non-null object
middle-middle-square    958 non-null object
middle-right-square     958 non-null object
bottom-left-square      958 non-null object
bottom-middle-square    958 non-null object
bottom-right-square     958 non-null object
Class                   958 non-null object
dtypes: object(10)
memory usage: 74.9+ KB
```

In [5]:
```
data.head()
```

Out[5]:

	top-left-square	top-middle-square	top-right-square	middle-left-square	middle-middle-square	middle-right-square	bottom-left-square	bottom-middle-square	bottom-right-square	Class
0	x	x	x	x	o	o	x	o	o	positive
1	x	x	x	x	o	o	o	x	o	positive
2	x	x	x	x	o	o	o	o	x	positive
3	x	x	x	x	o	o	o	b	b	positive
4	x	x	x	x	o	o	b	o	b	positive

In [6]:
```
data.describe()
```

Out[6]:

	top-left-square	top-middle-square	top-right-square	middle-left-square	middle-middle-square	middle-right-square	bottom-left-square	bottom-middle-square	bottom-right-square	Class
count	958	958	958	958	958	958	958	958	958	958
unique	3	3	3	3	3	3	3	3	3	2
top	x	x	x	x	x	x	x	x	x	positive
freq	418	378	418	378	458	378	418	378	418	626

In [7]:
```
y=data['Class']
y=y.to_frame()
y.head()
```

Out[7]:

	Class
0	positive
1	positive
2	positive
3	positive
4	positive

In [8]:
```
X=data
X= X[['top-left-square','top-middle-square','top-right-square','middle-left-square
','middle-middle-square', 'middle-right-square','bottom-left-square','bottom-middl
e-square','bottom-right-square']]
X.head()
```

Out[8]:

	top-left-square	top-middle-square	top-right-square	middle-left-square	middle-middle-square	middle-right-square	bottom-left-square	bottom-middle-square	bottom-right-square
0	x	x	x	x	o	o	x	o	o
1	x	x	x	x	o	o	o	x	o
2	x	x	x	x	o	o	o	o	x
3	x	x	x	x	o	o	o	b	b
4	x	x	x	x	o	o	b	o	b

In [9]:
```
#Applying Train,Test Split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=32)
```

In [10]:
```
combine=[X_train,X_test]
classmapping={'x':0,'o':1,'b':2}
for dt in combine:
   dt['top-left-square']=data['top-left-square'].map(classmapping)
X_train.head()
```

Out[10]:

	top-left-square	top-middle-square	top-right-square	middle-left-square	middle-middle-square	middle-right-square	bottom-left-square	bottom-middle-square	bottom-right-square
528	2	o	x	x	x	x	b	o	o
744	1	x	x	x	o	o	b	x	o
721	0	b	o	x	o	x	o	b	b
113	0	o	x	x	x	o	x	o	o
822	1	o	o	x	o	x	b	x	x

In [11]:
```
combine=[X_train,X_test]
classmapping={'x':0,'o':1,'b':2}
for dt in combine:
   dt['top-middle-square']=data['top-middle-square'].map(classmapping)
X_train.head()
```

Out[11]:

	top-left-square	top-middle-square	top-right-square	middle-left-square	middle-middle-square	middle-right-square	bottom-left-square	bottom-middle-square	bottom-right-square
528	2	1	x	x	x	x	b	o	o
744	1	0	x	x	o	o	b	x	o
721	0	2	o	x	o	x	o	b	b
113	0	1	x	x	x	o	x	o	o
822	1	1	o	x	o	x	b	x	x

In [12]:
```
combine=[X_train,X_test]
classmapping={'x':0,'o':1,'b':2}
for dt in combine:
   dt['top-right-square']=data['top-right-square'].map(classmapping)
X_train.head()
```

Out[12]:

	top-left-square	top-middle-square	top-right-square	middle-left-square	middle-middle-square	middle-right-square	bottom-left-square	bottom-middle-square	bottom-right-square
528	2	1	0	x	x	x	b	o	o
744	1	0	0	x	o	o	b	x	o
721	0	2	1	x	o	x	o	b	b
113	0	1	0	x	x	o	x	o	o
822	1	1	1	x	o	x	b	x	x

In [13]:
```
combine=[X_train,X_test]
classmapping={'x':0,'o':1,'b':2}
for dt in combine:
   dt['middle-left-square']=data['middle-left-square'].map(classmapping)
X_train.head()
```

Out[13]:

	top-left-square	top-middle-square	top-right-square	middle-left-square	middle-middle-square	middle-right-square	bottom-left-square	bottom-middle-square	bottom-right-square
528	2	1	0	0	x	x	b	o	o
744	1	0	0	0	o	o	b	x	o
721	0	2	1	0	o	x	o	b	b
113	0	1	0	0	x	o	x	o	o
822	1	1	1	0	o	x	b	x	x

In [14]:
```
combine=[X_train,X_test]
classmapping={'x':0,'o':1,'b':2}
for dt in combine:
   dt['middle-middle-square']=data['middle-middle-square'].map(classmapping)
X_train.head()
```

Out[14]:

	top-left-square	top-middle-square	top-right-square	middle-left-square	middle-middle-square	middle-right-square	bottom-left-square	bottom-middle-square	bottom-right-square
528	2	1	0	0	0	x	b	o	o
744	1	0	0	0	1	o	b	x	o
721	0	2	1	0	1	x	o	b	b
113	0	1	0	0	0	o	x	o	o
822	1	1	1	0	1	x	b	x	x

In [15]:
```
combine=[X_train,X_test]
classmapping={'x':0,'o':1,'b':2}
for dt in combine:
   dt['middle-right-square']=data['middle-right-square'].map(classmapping)
X_train.head()
```

Out[15]:

	top-left-square	top-middle-square	top-right-square	middle-left-square	middle-middle-square	middle-right-square	bottom-left-square	bottom-middle-square	bottom-right-square
528	2	1	0	0	0	0	b	o	o
744	1	0	0	0	1	1	b	x	o
721	0	2	1	0	1	0	o	b	b
113	0	1	0	0	0	1	x	o	o
822	1	1	1	0	1	0	b	x	x

In [16]:
```
combine=[X_train,X_test]
classmapping={'x':0,'o':1,'b':2}
for dt in combine:
   dt['middle-right-square']=data['middle-right-square'].map(classmapping)
X_train.head()
```

Out[16]:

	top-left-square	top-middle-square	top-right-square	middle-left-square	middle-middle-square	middle-right-square	bottom-left-square	bottom-middle-square	bottom-right-square
528	2	1	0	0	0	0	b	o	o
744	1	0	0	0	1	1	b	x	o
721	0	2	1	0	1	0	o	b	b
113	0	1	0	0	0	1	x	o	o
822	1	1	1	0	1	0	b	x	x

In [17]:
```
combine=[X_train,X_test]
classmapping={'x':0,'o':1,'b':2}
for dt in combine:
   dt['bottom-left-square']=data['bottom-left-square'].map(classmapping)
X_train.head()
```

Out[17]:

	top-left-square	top-middle-square	top-right-square	middle-left-square	middle-middle-square	middle-right-square	bottom-left-square	bottom-middle-square	bottom-right-square
528	2	1	0	0	0	0	2	o	o
744	1	0	0	0	1	1	2	x	o
721	0	2	1	0	1	0	1	b	b
113	0	1	0	0	0	1	0	o	o
822	1	1	1	0	1	0	2	x	x

In [18]:
```
combine=[X_train,X_test]
classmapping={'x':0,'o':1,'b':2}
for dt in combine:
   dt['bottom-middle-square']=data['bottom-middle-square'].map(classmapping)
X_train.head()
```

Out[18]:

	top-left-square	top-middle-square	top-right-square	middle-left-square	middle-middle-square	middle-right-square	bottom-left-square	bottom-middle-square	bottom-right-square
528	2	1	0	0	0	0	2	1	o
744	1	0	0	0	1	1	2	0	o
721	0	2	1	0	1	0	1	2	b
113	0	1	0	0	0	1	0	1	o
822	1	1	1	0	1	0	2	0	x

In [19]:
```
combine=[X_train,X_test]
classmapping={'x':0,'o':1,'b':2}
for dt in combine:
   dt['bottom-right-square']=data['bottom-right-square'].map(classmapping)
X_train.head()
```

Out[19]:

	top-left-square	top-middle-square	top-right-square	middle-left-square	middle-middle-square	middle-right-square	bottom-left-square	bottom-middle-square	bottom-right-square
528	2	1	0	0	0	0	2	1	1
744	1	0	0	0	1	1	2	0	1
721	0	2	1	0	1	0	1	2	2
113	0	1	0	0	0	1	0	1	1
822	1	1	1	0	1	0	2	0	0

In [20]:
```
combine=[y_train,y_test]
classmapping={'positive':0,'negative':1}
for dt in combine:
   dt['Class']=data['Class'].map(classmapping)
y_train.head()
```

Out[20]:

	Class
528	0
744	1
721	1
113	0
822	1

In [21]:
```
#AdaBoost
clf =  AdaBoostClassifier(DecisionTreeClassifier(max_depth=1),
                          algorithm="SAMME",
                          n_estimators=200)
clf.fit(X_train, y_train)
```

Out[21]:
```
AdaBoostClassifier(algorithm='SAMME',
          base_estimator=DecisionTreeClassifier(class_weight=None, criterion='gi
ni', max_depth=1,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best'),
          learning_rate=1.0, n_estimators=200, random_state=None)
```

In [22]:
```
y_pred = clf.predict(X_test)
```

In [23]:
```
y_pred
```

Out[23]:
```
array([0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1,
       1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1,
       1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0,
       1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1,
       0, 0])
```

In [24]:
```
y_test.head()
```

Out[24]:

	Class
260	0
11	0
250	0
416	0
932	1

In [25]:
```
print(confusion_matrix(y_test, y_pred))
```

```
[[182   4]
 [ 39  63]]
```

In [26]:
```
print(classification_report(y_test, y_pred))
```

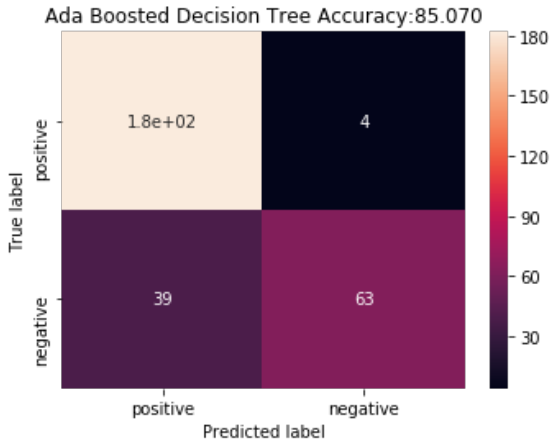
```
              precision    recall  f1-score   support

           0       0.82      0.98      0.89       186
           1       0.94      0.62      0.75       102

   micro avg       0.85      0.85      0.85       288
   macro avg       0.88      0.80      0.82       288
weighted avg       0.86      0.85      0.84       288
```

In [27]:
```
accuracy_test_ada=round(clf.score(X_test,y_test)*100,2)
accuracy_train_ada=round(clf.score(X_train,y_train)*100,2)
accuracy_ada=round(accuracy_score(y_test, y_pred)*100,2)
print('Training accuracy of Ada Boosted Decision Tree',accuracy_train_ada)
print('Testing accuracy of Ada Boosted Decision Tree',accuracy_test_ada)
print('Accuracy of Ada Boosted Decision Tree:',accuracy_ada)
```

```
Training accuracy of Ada Boosted Decision Tree 85.67
Testing accuracy of Ada Boosted Decision Tree 85.07
Accuracy of Ada Boosted Decision Tree: 85.07
```

In [28]:
```
cm=confusion_matrix(y_test, y_pred)
cm_df = pd.DataFrame(cm,
                     index = ['positive','negative'],
                     columns = ['positive','negative'])
```

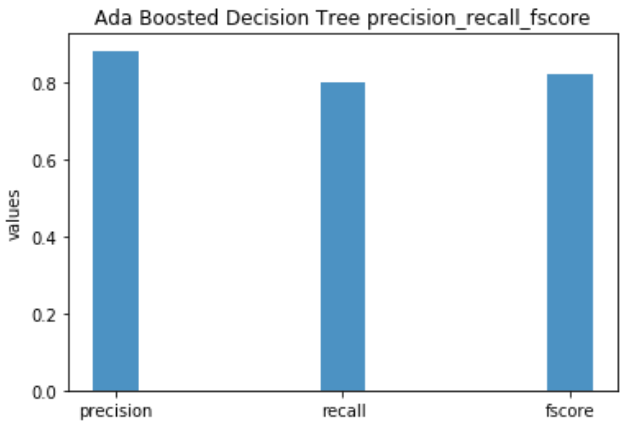
In [29]:
```
plt.figure(figsize=(5.5,4))
sns.heatmap(cm_df, annot=True)
plt.title('Ada Boosted Decision Tree Accuracy:{0:.3f}'.format(accuracy_test_ada))
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.show()
```



In [30]:
```
pprf = precision_recall_fscore_support(y_test, y_pred, average='macro')
print("Ada Boosted Decision Tree precision_recall_fscore_support ", pprf)
pps = precision_score(y_test, y_pred, labels=None, pos_label=1, average='macro', s
ample_weight=None)
print("Ada Boosted Decision Tree precision_score -> %.2f"%pps)
prs = recall_score(y_test, y_pred, labels=None, pos_label=1, average='macro', samp
le_weight=None)
print("Ada Boosted Decision Tree recall_score -> %.2f"%prs)
pf1=f1_score(y_test, y_pred, labels=None, pos_label=1, average='macro', sample_wei
ght=None)
print("f1_score",f1_score(y_test, y_pred, labels=None, pos_label=1, average='macro
', sample_weight=None))
print('Ada Boosted Decision Tree f1 score -> %.2f'%pf1)
```

```
Ada Boosted Decision Tree precision_recall_fscore_support  (0.8819139596136962,
0.7980708412397217, 0.8199555122632046, None)
Ada Boosted Decision Tree precision_score -> 0.88
Ada Boosted Decision Tree recall_score -> 0.80
f1_score 0.8199555122632046
Ada Boosted Decision Tree f1 score -> 0.82
```

In [31]:
```
plt.bar(['precision','recall','fscore'], [pps,prs,pf1], align='center', alpha=0.8,
width=.2)
plt.ylabel('values')
plt.title('Ada Boosted Decision Tree precision_recall_fscore')
plt.show()
```



In [32]:
```
clf = RandomForestClassifier(n_estimators=25)
clf.fit(X_train, y_train)
```

Out[32]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=25, n_jobs=None,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

In [33]:
```
y_pred = clf.predict(X_test)
```

In [34]:
```
y_pred
```

Out[34]:
```
array([0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0,
       0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1,
       1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1,
       0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1,
       1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0,
       1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1,
       0, 0])
```

In [35]:
```
y_test.head()
```

Out[35]:

	Class
260	0
11	0
250	0
416	0
932	1

In [36]:
```
print(confusion_matrix(y_test, y_pred))
```

```
[[186   0]
 [ 20  82]]
```

In [37]:
```
print(classification_report(y_test, y_pred))
```

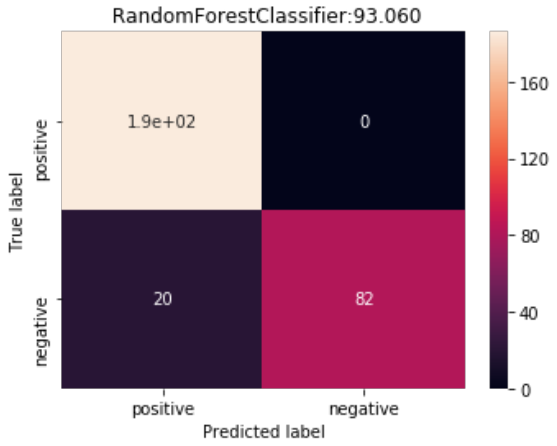
```
              precision    recall  f1-score   support

           0       0.90      1.00      0.95       186
           1       1.00      0.80      0.89       102

   micro avg       0.93      0.93      0.93       288
   macro avg       0.95      0.90      0.92       288
weighted avg       0.94      0.93      0.93       288
```

In [38]:
```
accuracy_test_rf=round(clf.score(X_test,y_test)*100,2)
accuracy_train_rf=round(clf.score(X_train,y_train)*100,2)
accuracy_rf=round(accuracy_score(y_test, y_pred)*100,2)
print('Training accuracy of RandomForestClassifier:',accuracy_train_rf)
print('Testing accuracy of RandomForestClassifier:',accuracy_test_rf)
print('Accuracy of RandomForestClassifier:',accuracy_rf)
```

```
Training accuracy of RandomForestClassifier: 100.0
Testing accuracy of RandomForestClassifier: 93.06
Accuracy of RandomForestClassifier: 93.06
```

In [39]:
```
cm=confusion_matrix(y_test, y_pred)
cm_df = pd.DataFrame(cm,
                     index = ['positive','negative'],
                     columns = ['positive','negative'])
```

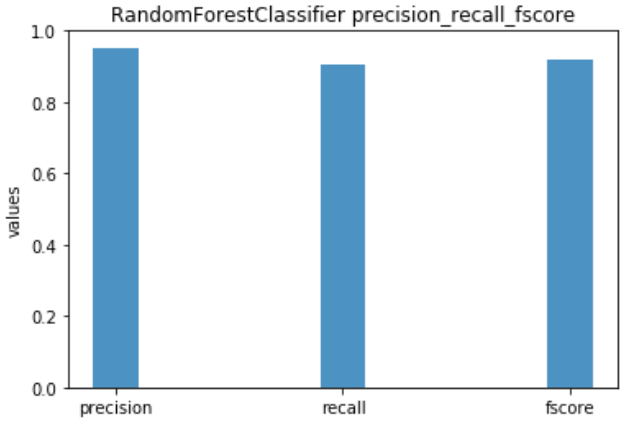
In [40]:
```
plt.figure(figsize=(5.5,4))
sns.heatmap(cm_df, annot=True)
plt.title(' RandomForestClassifier:{0:.3f}'.format(accuracy_test_rf))
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.show()
```



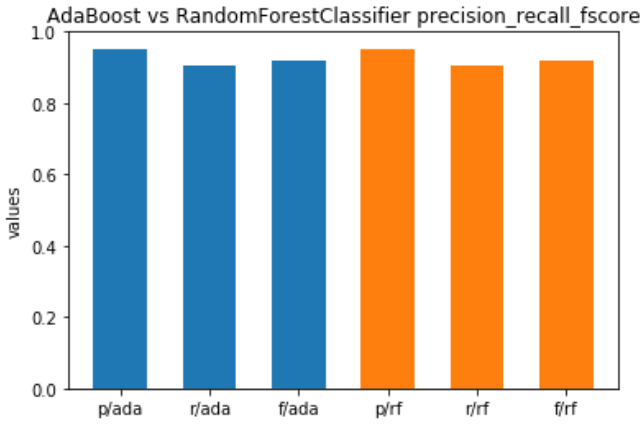
In [43]:
```
pprf1 = precision_recall_fscore_support(y_test, y_pred, average='macro')
print("RandomForestClassifier precision_recall_fscore_support ", pprf)
pps1 = precision_score(y_test, y_pred, labels=None, pos_label=1, average='macro',
sample_weight=None)
print("RandomForestClassifier precision_score -> %.2f"%pps)
prs1 = recall_score(y_test, y_pred, labels=None, pos_label=1, average='macro', sam
ple_weight=None)
print("RandomForestClassifier recall_score -> %.2f"%prs)
pf2=f1_score(y_test, y_pred, labels=None, pos_label=1, average='macro', sample_wei
ght=None)
print("f1_score",f1_score(y_test, y_pred, labels=None, pos_label=1, average='macro
', sample_weight=None))
print('RandomForestClassifier f1 score -> %.2f'%pf1)
```

```
RandomForestClassifier precision_recall_fscore_support  (0.9514563106796117, 0.9
019607843137255, 0.9201419698314108, None)
RandomForestClassifier precision_score -> 0.95
RandomForestClassifier recall_score -> 0.90
f1_score 0.9201419698314108
RandomForestClassifier f1 score -> 0.92
```

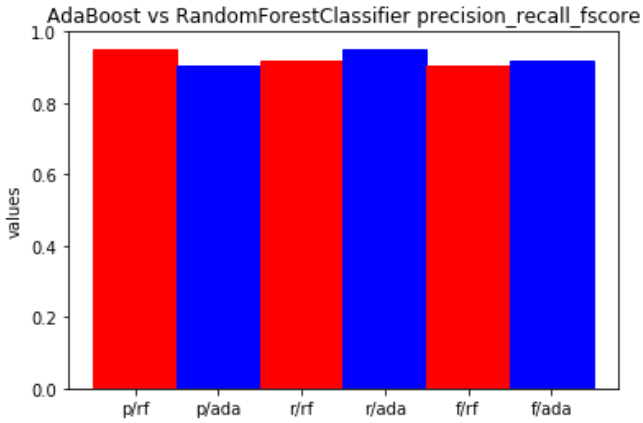
In [44]:
```
plt.bar(['precision','recall','fscore'], [pps,prs,pf2], align='center', alpha=0.8,
width=.2)
plt.ylabel('values')
plt.title('RandomForestClassifier precision_recall_fscore')
plt.show()
```



In [46]:
```
plt.bar(['p/rf','r/rf','f/rf'], [pps1,prs1,pf2], align='center', alpha=1.0,width=.
6)
plt.bar(['p/ada','r/ada','f/ada'],[pps,prs,pf1], align='center', alpha=1.0,width=.
6)
plt.ylabel('values')
plt.title('AdaBoost vs RandomForestClassifier precision_recall_fscore')
plt.show()
```



In [55]:
```
barlist=plt.bar(['p/rf','p/ada','r/rf','r/ada','f/rf','f/ada'], [pps1,prs1,pf2,pps
,prs,pf1], align='center', alpha=1,width=1)
#plt.bar([],[pps,prs,pf1], align='center', alpha=0.1,width=.6)
barlist[0].set_color('r')
barlist[2].set_color('r')
barlist[4].set_color('r')
barlist[1].set_color('b')
barlist[3].set_color('b')
barlist[5].set_color('b')
plt.ylabel('values')
plt.title('AdaBoost vs RandomForestClassifier precision_recall_fscore')
plt.show()
```



In [ ]: