使用随机森林预测森林火灾发生概率

利用机器学习对森林火灾的是否会发生做出预测，并分析的模型的错判率。使用从葡萄牙东北部的Montesinho国家公园（517条）、阿尔及利亚东北部Bejaia区域（122条）和阿尔及利亚西北部Sidi Belabbes区域（122条）采集的最新数据预测森林火灾的受灾面积。应用决策树和随机森林对三类指标进行分析(即时间，气象指标和部分FWI系统指标)。将对三类不同性质的指标分别进行基于机器学习的数据分析，如气象指标(即温度，相对湿度，风速和降雨量)与随机森林相结合，能够预测森林火灾是否会发生，构建火灾燃烧等级对未来的火灾防治和消防管理决策是非常有用的。

数据源网站1：https://tianchi.aliyun.com/dataset/dataDetail?dataId=92968

数据源网站2：https://tianchi.aliyun.com/dataset/dataDetail?dataId=103992#1

1.获取数据

In [5]:
```
#引入工具包
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as matplot
import seaborn as sns
%matplotlib inline
```

In [6]:
```
#读入数据
df = pd.read_csv('C:/Users/admin/Desktop/forest_fire.csv',index_col=None)
```

2.数据预处理

In [7]:
```
#检测缺失值并清除
df.isnull().any()
```

Out[7]:
```
temp     False
RH       False
WS       False
RF       False
FFMC     False
DMC      False
DC       False
ISI      False
month    False
fire     False
dtype: bool
```

In [9]:
```
#数据样例
df.head()
```

Out[9]:

	temp	RH	WS	RF	FFMC	DMC	DC	ISI	month	fire
0	8.2	51	6.7	0.0	86.2	26.2	94.3	5.1	3	0
1	18.0	33	0.9	0.0	90.6	35.4	669.1	6.7	10	0
2	14.6	33	1.3	0.0	90.6	43.7	686.9	6.7	10	0
3	8.3	97	4.0	0.2	91.7	33.3	77.5	9.0	3	0
4	11.4	99	1.8	0.0	89.3	51.3	102.2	9.6	3	0

3.分析数据

3.1 数据描述

- 761条数据，每条数据10个特征
- 总森火发生率54.008%，占比大致一半
- 通过分组平均统计感知，降雨量（RF）与是否着火关系较大

In [10]:
```
#数据体量与类型
df.shape
```

Out[10]:
```
(761, 10)
```

In [11]:
```
df.dtypes
```

Out[11]:
```
temp     float64
RH         int64
WS       float64
RF       float64
FFMC     float64
DMC      float64
DC       float64
ISI      float64
month      int64
fire       int64
dtype: object
```

In [12]:
```
fire_rate = df.fire.value_counts()/len(df)
fire_rate
```

Out[12]:
```
1    0.540079
0    0.459921
Name: fire, dtype: float64
```

In [13]:
```
#显示统计数据
df.describe()
```

Out[13]:

	temp	RH	WS	RF	FFMC	DMC	DC	ISI	month	fire
count	761.000000	761.000000	761.000000	761.000000	761.000000	761.000000	761.000000	761.000000	761.000000	761.000000
mean	23.148095	49.947438	7.700526	0.258607	86.554402	80.027989	388.057017	7.659790	7.483574	0.540079
std	8.099219	17.876252	5.786381	1.206968	11.041353	69.659117	311.030407	4.860411	1.978185	0.498719
min	2.200000	15.000000	0.400000	0.000000	18.700000	0.700000	6.900000	0.000000	1.000000	0.000000
25%	17.600000	35.000000	3.100000	0.000000	85.400000	16.500000	50.200000	4.100000	7.000000	0.000000
50%	22.400000	47.000000	4.900000	0.000000	90.700000	73.400000	466.600000	7.500000	8.000000	1.000000
75%	30.000000	64.000000	14.000000	0.000000	92.300000	126.500000	690.000000	9.800000	9.000000	1.000000
max	42.000000	100.000000	29.000000	16.800000	96.200000	291.300000	860.600000	56.100000	12.000000	1.000000

In [14]:
```
#分组进行平均数据统计
fire_summary = df.groupby('fire')
fire_summary.mean()
```

Out[14]:

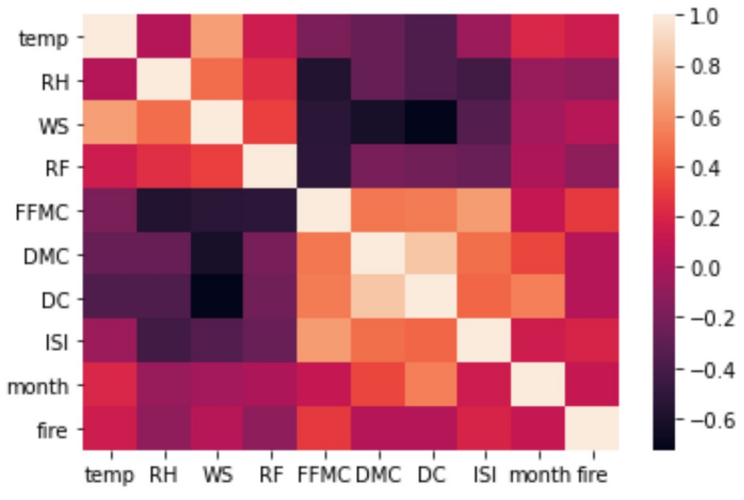
fire	temp	RH	WS	RF	FFMC	DMC	DC	ISI	month
0	21.884857	51.982857	7.370286	0.404286	83.081429	77.207143	375.535714	6.631429	7.245714
1	24.223844	48.214112	7.981752	0.134550	89.511922	82.430170	398.719927	8.535523	7.686131

3.2 相关性分析

In [15]:
```
#相关性矩阵与热力图
corr = df.corr()
sns.heatmap(corr,
            xticklabels = corr.columns.values,
            yticklabels = corr.columns.values)
corr
```

Out[15]:

	temp	RH	WS	RF	FFMC	DMC	DC	ISI	month	fire
temp	1.000000	0.040693	0.656484	0.149654	-0.182074	-0.271531	-0.372716	-0.057690	0.206175	0.144026
RH	0.040693	1.000000	0.471346	0.245347	-0.565281	-0.274507	-0.373618	-0.423187	-0.070981	-0.105142
WS	0.656484	0.471346	1.000000	0.312022	-0.536163	-0.618332	-0.726458	-0.355273	-0.019219	0.052701
RF	0.149654	0.245347	0.312022	1.000000	-0.523760	-0.200112	-0.233409	-0.264070	0.014732	-0.111455
FFMC	-0.182074	-0.565281	-0.536163	-0.523760	1.000000	0.511834	0.525529	0.653401	0.114392	0.290454
DMC	-0.271531	-0.274507	-0.618332	-0.200112	0.511834	1.000000	0.830166	0.474803	0.333650	0.037394
DC	-0.372716	-0.373618	-0.726458	-0.233409	0.525529	0.830166	1.000000	0.442973	0.540461	0.037175
ISI	-0.057690	-0.423187	-0.355273	-0.264070	0.653401	0.474803	0.442973	1.000000	0.143939	0.195376
month	0.206175	-0.070981	-0.019219	0.014732	0.114392	0.333650	0.540461	0.143939	1.000000	0.111033
fire	0.144026	-0.105142	0.052701	-0.111455	0.290454	0.037394	0.037175	0.195376	0.111033	1.000000



In [16]:
```
#比较着火/未着火情况下细小可燃物湿度码(FFMC)和初始蔓延速率(ISI),进行T-Test
fire_ffmc = df['FFMC'][df['fire']==1].mean()
nofire_ffmc = df['FFMC'][df['fire']==0].mean()
fire_isi = df['ISI'][df['fire']==1].mean()
nofire_isi = df['ISI'][df['fire']==0].mean()
import scipy.stats as stats
stats.ttest_1samp(a = df[df['fire']==1]['FFMC'], popmean = nofire_ffmc)
```

Out[16]:
```
Ttest_1sampResult(statistic=23.03028271657713, pvalue=6.471526595566097e-76)
```

In [17]:
```
stats.ttest_1samp(a = df[df['fire']==0]['ISI'],popmean = fire_isi)
```

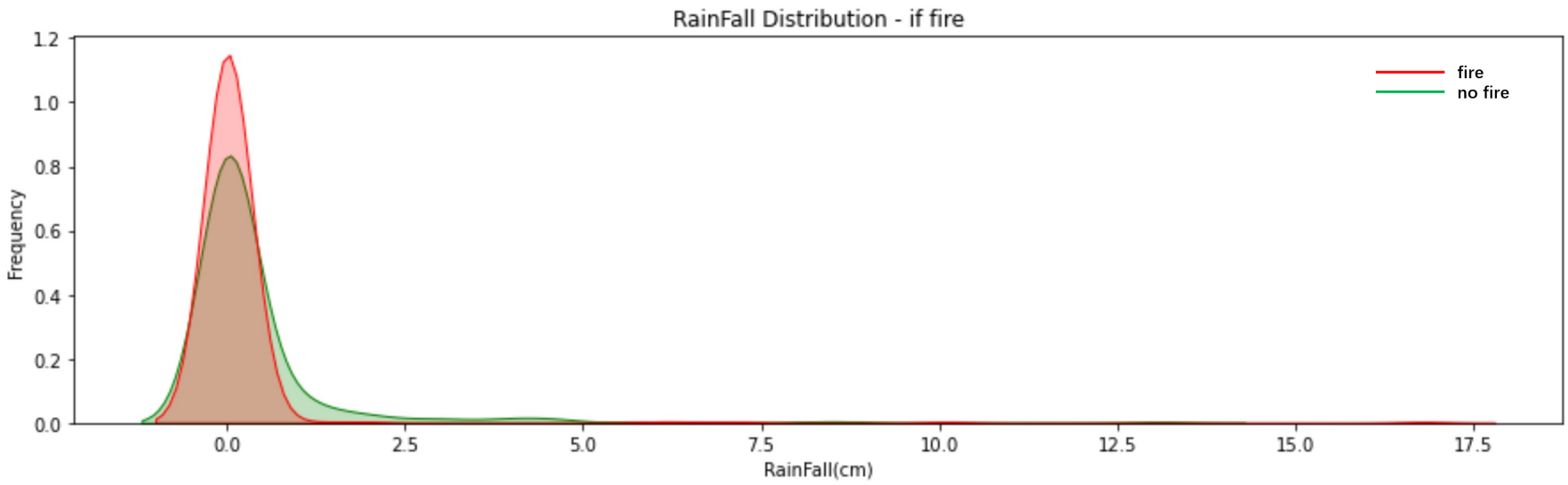
Out[17]:
```
Ttest_1sampResult(statistic=-6.562605398922424, pvalue=1.913492296109212e-10)
```

T-Test显示在是否着火的两组数据中，FFMC和ISI都是有显著差异的。

同时观察平均值时发现降雨量(RF)的均值差距很大，而这一指标与是否着火的相关系数却并不算高（-0.11），考虑进一步使用概率密度分布图分类观察影响着火的因素，认识其数据在着/不着火情况下的分布特征。

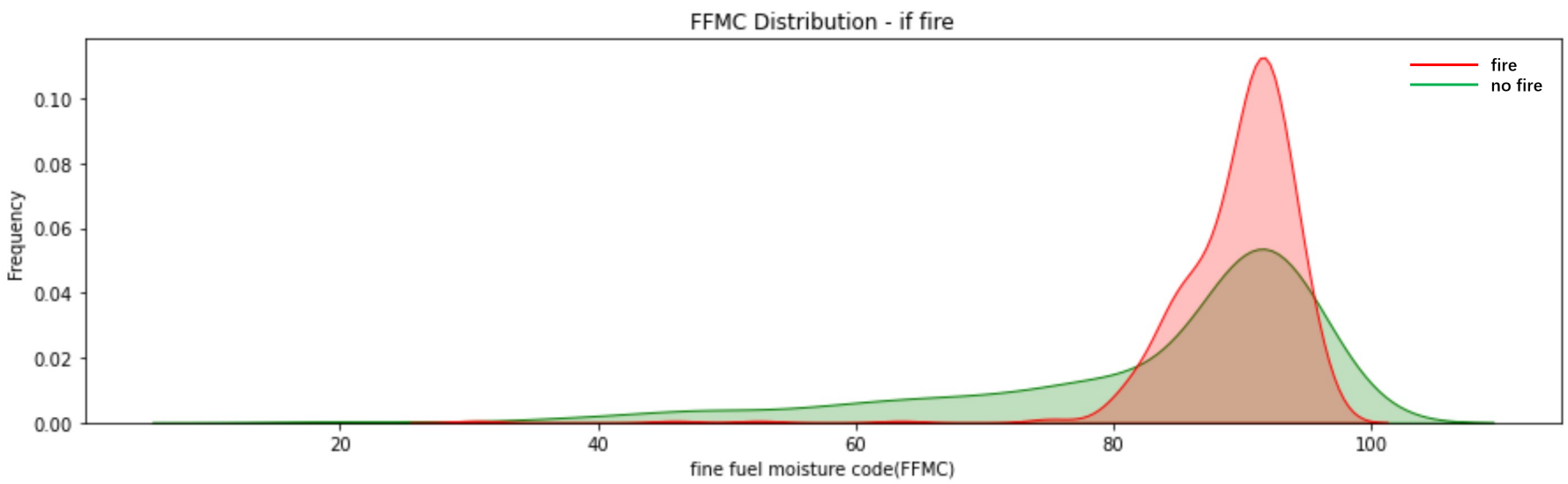
In [18]:
```
fig = plt.figure(figsize = (15,4),)
ax = sns.kdeplot(df.loc[(df['fire']==0),'RF'],color = 'g',shade = True,label = 'no fire')
ax = sns.kdeplot(df.loc[(df['fire']==1),'RF'],color = 'r',shade = True,label = 'fire')
ax.set(xlabel = 'RainFall(cm)',ylabel = 'Frequency')
plt.title('RainFall Distribution - if fire')
```

Out[18]:
```
Text(0.5, 1.0, 'RainFall Distribution - if fire')
```



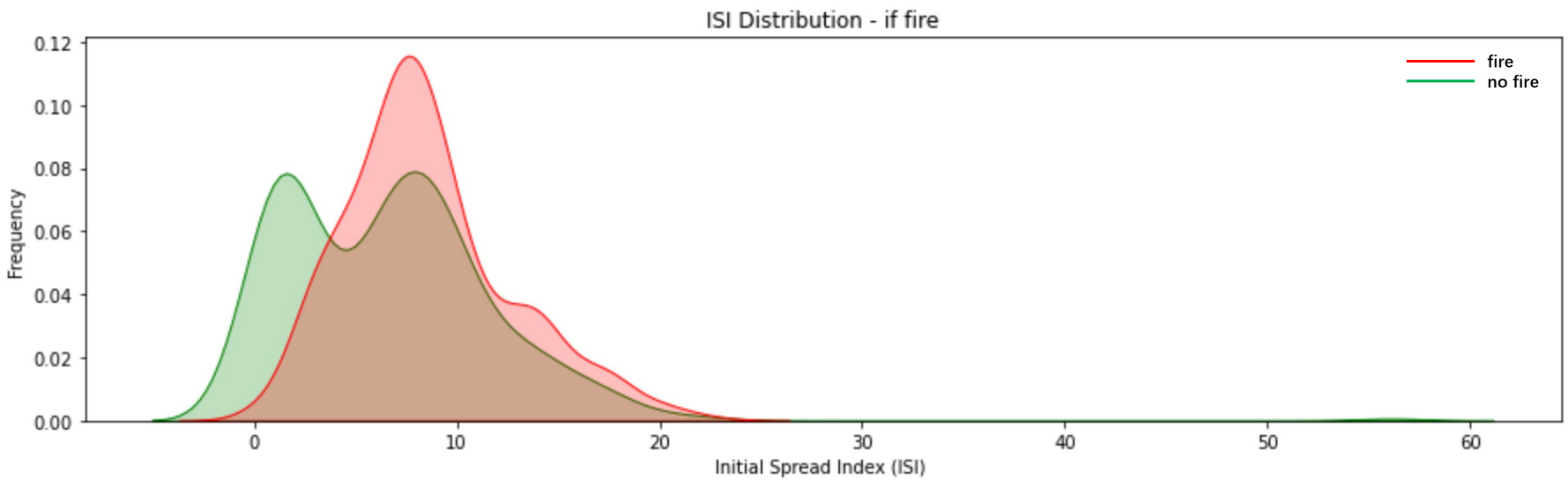
In [19]:
```
fig = plt.figure(figsize = (15,4),)
ax = sns.kdeplot(df.loc[(df['fire']==0),'FFMC'],color = 'g',shade = True,label = 'no fire')
ax = sns.kdeplot(df.loc[(df['fire']==1),'FFMC'],color = 'r',shade = True,label = 'fire')
ax.set(xlabel = 'fine fuel moisture code(FFMC)',ylabel = 'Frequency')
plt.title('FFMC Distribution - if fire')
```

Out[19]:
```
Text(0.5, 1.0, 'FFMC Distribution - if fire')
```



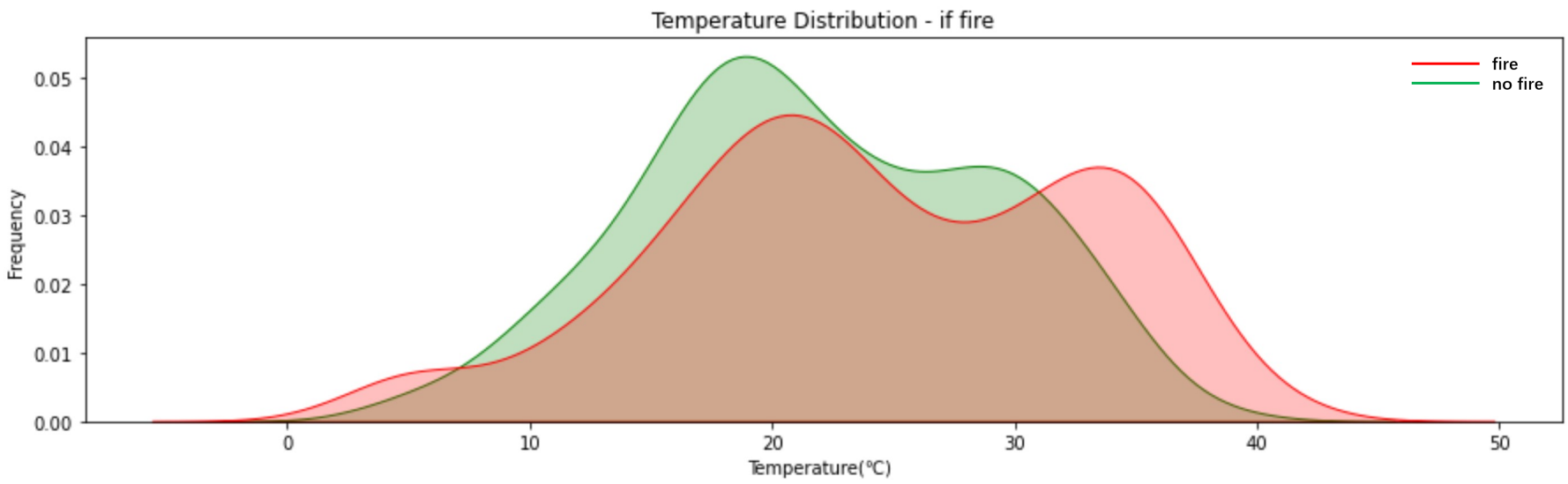
In [27]:
```
fig = plt.figure(figsize = (15,4),)
ax = sns.kdeplot(df.loc[(df['fire']==0),'ISI'],color = 'g',shade = True,label = 'no fire')
ax = sns.kdeplot(df.loc[(df['fire']==1),'ISI'],color = 'r',shade = True,label = 'fire')
ax.set(xlabel = 'Initial Spread Index (ISI) ',ylabel = 'Frequency')
plt.title('ISI Distribution - if fire')
```

Out[27]:
```
Text(0.5, 1.0, 'ISI Distribution - if fire')
```



In [28]:
```
fig = plt.figure(figsize = (15,4),)
ax = sns.kdeplot(df.loc[(df['fire']==0),'temp'],color = 'g',shade = True,label = 'no fire')
ax = sns.kdeplot(df.loc[(df['fire']==1),'temp'],color = 'r',shade = True,label = 'fire')
ax.set(xlabel = 'Temperature(℃) ',ylabel = 'Frequency')
plt.title('Temperature Distribution - if fire')
```

Out[28]:
```
Text(0.5, 1.0, 'Temperature Distribution - if fire')
```



4.决策树（DecisionTree）和随机森林（RandomForest）预测模型的构建

In [29]:
```
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,classification_report,precision_score,recall_score
```

In [38]:
```
#产生x，y；将数据分为训练和测试数据集;stratify = y 意味着在产生训练和测试数据中，
#着火案例的百分比等于原来总的数据中的着火案例的百分比。
target_name = 'fire'
x = df.drop('fire',axis=1)
y = df[target_name]
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.15,random_state=123,stratify = y)
```

In [39]:
```
#采用决策树和随机森林两种建模方法，使用决策树的原因是决策树模型可以可视化
from sklearn.metrics import roc_auc_score
from sklearn.metrics import classification_report
from sklearn.ensemble import RandomForestClassifier
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier

#决策树
dtree = tree.DecisionTreeClassifier(criterion = 'entropy',min_weight_fraction_leaf = 0.01) #叶子节点最少要含1%的样本
dtree = dtree.fit(x_train,y_train)
print('\n\n < 决策树 >')
dt_roc_auc = roc_auc_score(y_test,dtree.predict(x_test))
print('决策树 AUC = %2.2f' % dt_roc_auc)
print(classification_report(y_test,dtree.predict(x_test)))

#随机森林
rf = RandomForestClassifier(
criterion = 'entropy',#使用信息熵计算信息增益，进行分类,也可以采用基尼不纯度'gini'
n_estimators = 1000,#选择构建树的数量
max_depth = None,#不使用设置深度的方式来防止过拟合，因为不明确拟合情况
min_samples_split = 10,#定义至少有10个样本才继续分叉
#min_weight_fraction_leaf = 0.02 #也可以采用定义叶子结点最少需要包含多少百分比的样本来防止过拟合
)
rf.fit(x_train,y_train)
print('\n\n < 随机森林 >')
rf_roc_auc = roc_auc_score(y_test,rf.predict(x_test))
print('随机森林 AUC = %2.2f' % rf_roc_auc)
print(classification_report(y_test,rf.predict(x_test)))
```

```
 < 决策树 >
决策树 AUC = 0.68
              precision    recall  f1-score   support

           0       0.65      0.66      0.65        53
           1       0.70      0.69      0.70        62

    accuracy                           0.68       115
   macro avg       0.68      0.68      0.68       115
weighted avg       0.68      0.68      0.68       115


 < 随机森林 >
随机森林 AUC = 0.72
              precision    recall  f1-score   support

           0       0.71      0.68      0.69        53
           1       0.73      0.76      0.75        62

    accuracy                           0.72       115
   macro avg       0.72      0.72      0.72       115
weighted avg       0.72      0.72      0.72       115
```

5.模型对比与特征选取

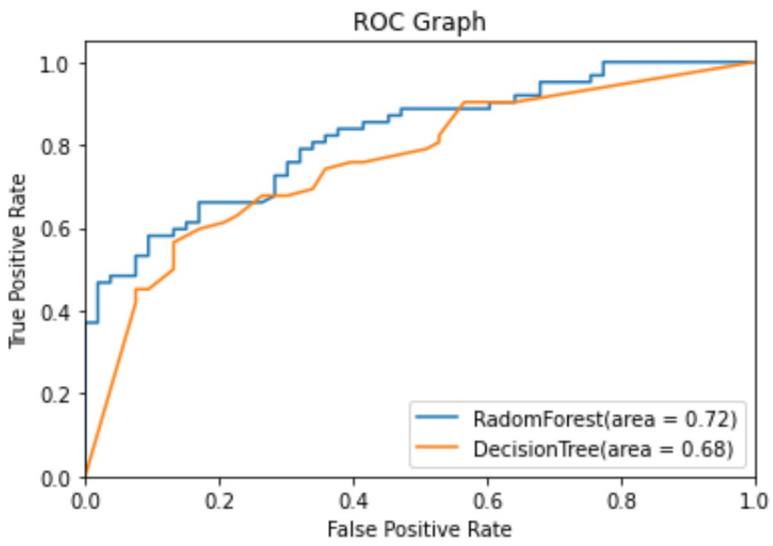
5.1 ROC图

In [40]:
```
from sklearn.metrics import roc_curve
rf_fpr,rf_tpr,rf_thresholds = roc_curve(y_test,rf.predict_proba(x_test)[:,1])
dt_fpr,dt_tpr,dt_thresholds = roc_curve(y_test,dtree.predict_proba(x_test)[:,1])

plt.plot(rf_fpr,rf_tpr,label = 'RadomForest(area = %0.2f)'%rf_roc_auc)
plt.plot(dt_fpr,dt_tpr,label = 'DecisionTree(area = %0.2f)'%dt_roc_auc)

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 Graph')
plt.legend(loc = 'lower right')
plt.show
```

Out[40]: <function matplotlib.pyplot.show(close=None, block=None)>



5.2 选择随机森林辅助特征选择

In [43]:
```
importances = rf.feature_importances_
feat_names = df.drop(['fire'],axis = 1).columns

indices = np.argsort(importances)[::-1]
plt.figure(figsize=(12,6))
plt.title = ('Feature importances by Random Forest')
plt.bar(range(len(indices)),importances[indices],color='darkgreen',align = 'center')
plt.step(range(len(indices)),np.cumsum(importances[indices]),color='green',where = 'mid',label = 'Cumulative')
plt.xticks(range(len(indices)),feat_names[indices],rotation = 'vertical',fontsize = 14)
plt.xlim([-1,len(indices)])
plt.show
```

Out[43]: <function matplotlib.pyplot.show(close=None, block=None)>