predicting occurance of forestfire

In [798]:

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_ma
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
```

loading data

In [799]:

```python
data = pd.read_csv("C:\\Users\\Niranjan Bhat\\Downloads\\forestfires (1).csv")
data
```

Out[799]:

```
       X  Y  month  day  FFMC    DMC     DC   ISI  temp  RH  wind  rain   area
  0    7  5    mar  fri  86.2   26.2   94.3   5.1   8.2  51   6.7   0.0   0.00
  1    7  4    oct  tue  90.6   35.4  669.1   6.7  18.0  33   0.9   0.0   0.00
  2    7  4    oct  sat  90.6   43.7  686.9   6.7  14.6  33   1.3   0.0   0.00
  3    8  6    mar  fri  91.7   33.3   77.5   9.0   8.3  97   4.0   0.2   0.00
  4    8  6    mar  sun  89.3   51.3  102.2   9.6  11.4  99   1.8   0.0   0.00
 ...  ... ...  ...  ...   ...    ...    ...   ...   ...  ...  ...   ...    ...
512    4  3    aug  sun  81.6   56.7  665.6   1.9  27.8  32   2.7   0.0   6.44
513    2  4    aug  sun  81.6   56.7  665.6   1.9  21.9  71   5.8   0.0  54.29
514    7  4    aug  sun  81.6   56.7  665.6   1.9  21.2  70   6.7   0.0  11.16
515    1  4    aug  sat  94.4  146.0  614.7  11.3  25.6  42   4.0   0.0   0.00
516    6  3    nov  tue  79.5    3.0  106.7   1.1  11.8  31   4.5   0.0   0.00
```

517 rows × 13 columns

X - x-axis spatial coordinate within the Montesinho park map: 1 to 9 Y - y-axis spatial coordinate within the Montesinho park map: 2 to 9 month - month of the year: "jan" to "dec" day - day of the week: "mon" to "sun" FFMC - FFMC index from the FWI system: 18.7 to 96.20; Fine fuel moisture code representing the moisture content of litter DMC - DMC index from the FWI system: 1.1 to 291.3; Duff moisture code representing the average moisture content of organic layers and woody material DC - DC index from the FWI system: 7.9 to 860.6; Drought moisture code representing the average moisture content of organic layers ISI - ISI index from the FWI system: 0.0 to 56.10 temp - temperature in Celsius degrees: 2.2 to 33.30 RH - relative humidity in %: 15.0 to 100 wind - wind speed in km/h: 0.40 to 9.40 rain - outside rain in mm/m2 : 0.0 to 6.4 area - the burned area of the forest (in hectares (ha)): 0.00 to 1090.84

preprocessing

In [800]:

```
data.isnull().any()
```

Out[800]:

```
X         False
Y         False
month     False
day       False
FFMC      False
DMC       False
DC        False
ISI       False
temp      False
RH        False
wind      False
rain      False
area      False
dtype: bool
```

In [801]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 517 entries, 0 to 516
Data columns (total 13 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   X       517 non-null    int64  
 1   Y       517 non-null    int64  
 2   month   517 non-null    object 
 3   day     517 non-null    object 
 4   FFMC    517 non-null    float64
 5   DMC     517 non-null    float64
 6   DC      517 non-null    float64
 7   ISI     517 non-null    float64
 8   temp    517 non-null    float64
 9   RH      517 non-null    int64  
 10  wind    517 non-null    float64
 11  rain    517 non-null    float64
 12  area    517 non-null    float64
dtypes: float64(8), int64(3), object(2)
memory usage: 52.6+ KB
```

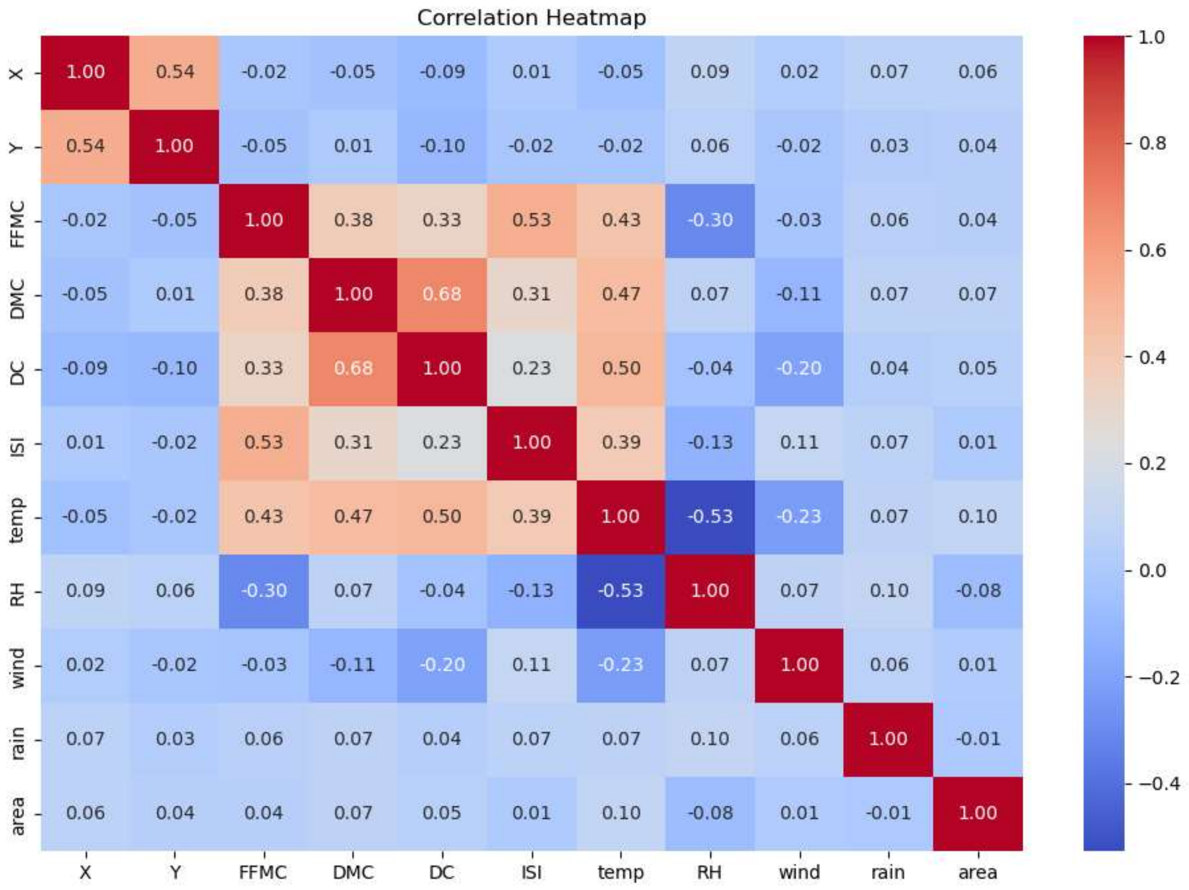
In [802]:
```
data.describe()
```

Out[802]:

	X	Y	FFMC	DMC	DC	ISI	temp	
count	517.000000	517.000000	517.000000	517.000000	517.000000	517.000000	517.000000	517.(
mean	4.669246	4.299807	90.644681	110.872340	547.940039	9.021663	18.889168	44.2
std	2.313778	1.229900	5.520111	64.046482	248.066192	4.559477	5.806625	16.3
min	1.000000	2.000000	18.700000	1.100000	7.900000	0.000000	2.200000	15.(
25%	3.000000	4.000000	90.200000	68.600000	437.700000	6.500000	15.500000	33.(
50%	4.000000	4.000000	91.600000	108.300000	664.200000	8.400000	19.300000	42.(
75%	7.000000	5.000000	92.900000	142.400000	713.900000	10.800000	22.800000	53.(
max	9.000000	9.000000	96.200000	291.300000	860.600000	56.100000	33.300000	100.(

In [803]:
```
correlation_matrix = data.corr()
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap')
plt.show()
```

```
C:\Users\Niranjan Bhat\AppData\Local\Temp\ipykernel_18628\123347052.py:1: Fut
ureWarning: The default value of numeric_only in DataFrame.corr is deprecate
d. In a future version, it will default to False. Select only valid columns o
r specify the value of numeric_only to silence this warning.
  correlation_matrix = data.corr()
```



	X	Y	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area
X	1.00	0.54	-0.02	-0.05	-0.09	0.01	-0.05	0.09	0.02	0.07	0.06
Y	0.54	1.00	-0.05	0.01	-0.10	-0.02	-0.02	0.06	-0.02	0.03	0.04
FFMC	-0.02	-0.05	1.00	0.38	0.33	0.53	0.43	-0.30	-0.03	0.06	0.04
DMC	-0.05	0.01	0.38	1.00	0.68	0.31	0.47	0.07	-0.11	0.07	0.07
DC	-0.09	-0.10	0.33	0.68	1.00	0.23	0.50	-0.04	-0.20	0.04	0.05
ISI	0.01	-0.02	0.53	0.31	0.23	1.00	0.39	-0.13	0.11	0.07	0.01
temp	-0.05	-0.02	0.43	0.47	0.50	0.39	1.00	-0.53	-0.23	0.07	0.10
RH	0.09	0.06	-0.30	0.07	-0.04	-0.13	-0.53	1.00	0.07	0.10	-0.08
wind	0.02	-0.02	-0.03	-0.11	-0.20	0.11	-0.23	0.07	1.00	0.06	0.01
rain	0.07	0.03	0.06	0.07	0.04	0.07	0.07	0.10	0.06	1.00	-0.01
area	0.06	0.04	0.04	0.07	0.05	0.01	0.10	-0.08	0.01	-0.01	1.00

converting continous values of area column into binary values

```
In [804]: data['fire_occurrence'] = data['area'].apply(lambda x: 1 if x > 0 else 0)
```

```
In [805]: data.fire_occurrence
```

```
Out[805]: 0      0
          1      0
          2      0
          3      0
          4      0
                ..
          512    1
          513    1
          514    1
          515    0
          516    0
          Name: fire_occurrence, Length: 517, dtype: int64
```

converting categorial values into binary

```
In [806]: data = pd.get_dummies(data, columns=['month', 'day','rain'])
```

In [807]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 517 entries, 0 to 516
Data columns (total 37 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   X                517 non-null    int64  
 1   Y                517 non-null    int64  
 2   FFMC             517 non-null    float64
 3   DMC              517 non-null    float64
 4   DC               517 non-null    float64
 5   ISI              517 non-null    float64
 6   temp             517 non-null    float64
 7   RH               517 non-null    int64  
 8   wind             517 non-null    float64
 9   area             517 non-null    float64
 10  fire_occurrence  517 non-null    int64  
 11  month_apr        517 non-null    uint8  
 12  month_aug        517 non-null    uint8  
 13  month_dec        517 non-null    uint8  
 14  month_feb        517 non-null    uint8  
 15  month_jan        517 non-null    uint8  
 16  month_jul        517 non-null    uint8  
 17  month_jun        517 non-null    uint8  
 18  month_mar        517 non-null    uint8  
 19  month_may        517 non-null    uint8  
 20  month_nov        517 non-null    uint8  
 21  month_oct        517 non-null    uint8  
 22  month_sep        517 non-null    uint8  
 23  day_fri          517 non-null    uint8  
 24  day_mon          517 non-null    uint8  
 25  day_sat          517 non-null    uint8  
 26  day_sun          517 non-null    uint8  
 27  day_thu          517 non-null    uint8  
 28  day_tue          517 non-null    uint8  
 29  day_wed          517 non-null    uint8  
 30  rain_0.0         517 non-null    uint8  
 31  rain_0.2         517 non-null    uint8  
 32  rain_0.4         517 non-null    uint8  
 33  rain_0.8         517 non-null    uint8  
 34  rain_1.0         517 non-null    uint8  
 35  rain_1.4         517 non-null    uint8  
 36  rain_6.4         517 non-null    uint8  
dtypes: float64(7), int64(4), uint8(26)
memory usage: 57.7 KB
```

In [808]:

```

data.drop('area', axis=1, inplace=True)
```

In [809]:

```
X = data.drop('fire_occurrence', axis=1)
```

In [810]:

```
X.shape
```

Out[810]:

```
(517, 35)
```

In [811]:
```
y = data['fire_occurrence']
```

In [812]:
```
y.shape
```

Out[812]:
```
(517,)
```

splitting data

In [813]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random
```

In [814]:
```
X_train.shape
```

Out[814]:
```
(413, 35)
```

In [815]:
```
X_test.shape
```

Out[815]:
```
(104, 35)
```

In [816]:
```
y_train.shape
```

Out[816]:
```
(413,)
```

In [817]:
```
y_test.shape
```

Out[817]:
```
(104,)
```

Feature scaling

In [818]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Build model

implementation of Randomforest

In [819]:
```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
probabilities = model.predict_proba(X_test)
```

In [820]:
```
#clf = RandomForestClassifier(random_state=42)
```

In [821]:

```
model.fit(X_train, y_train)
```

Out[821]:

```
RandomForestClassifier(random_state=42)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [822]:

```
y_pred = model.predict(X_test)
```

In [823]:

```
accuracy = accuracy_score(y_test, y_pred)
```

In [824]:

```
print("Accuracy:", accuracy)
```

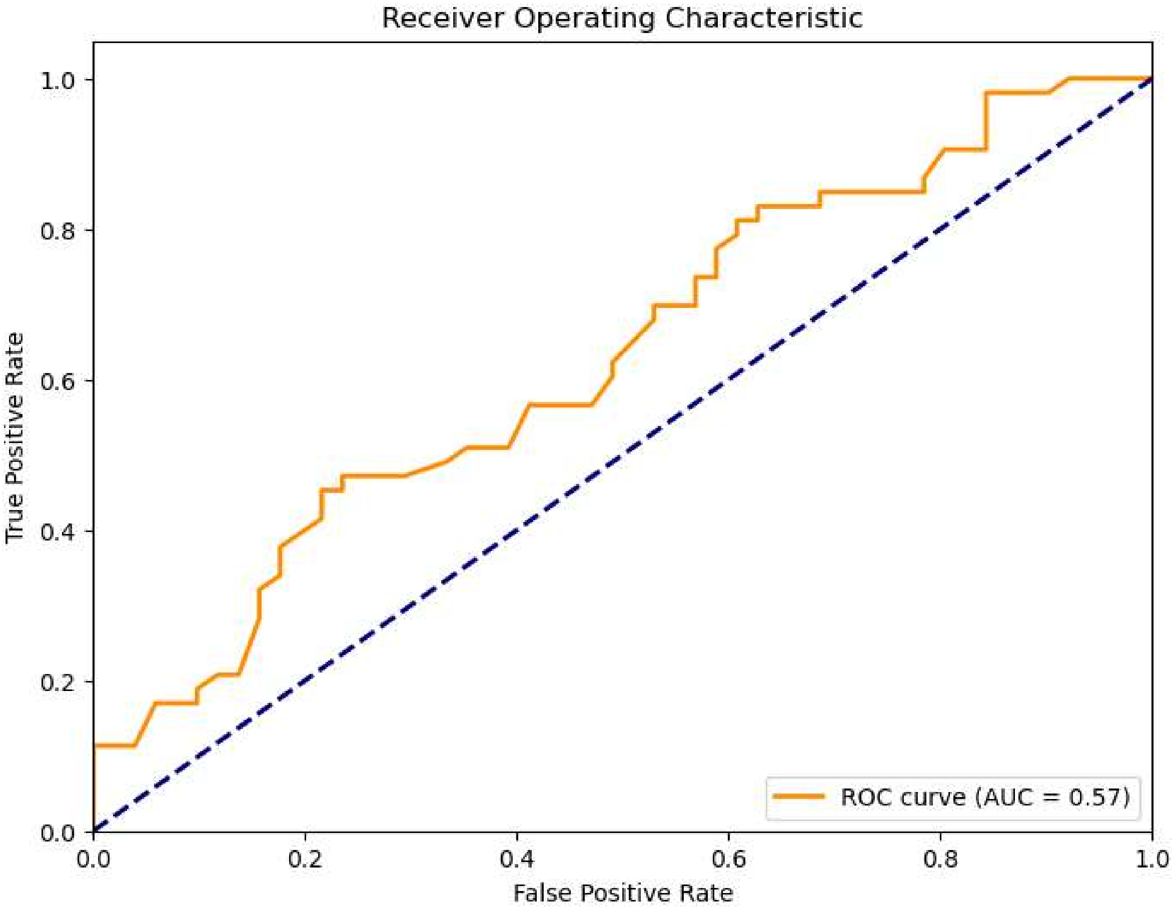
```
Accuracy: 0.5769230769230769
```

In [825]:

```
from sklearn.metrics import roc_curve, roc_auc_score

fpr, tpr, _ = roc_curve(y_test,model.predict_proba(X_test)[:, 1])
roc_auc = roc_auc_score(y_test, y_pred)

plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (AUC = {roc_auc:
plt.plot([0, 1], [0, 1], color='navy', lw=2, 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('Receiver Operating Characteristic')
plt.legend(loc='lower right')
plt.show()
```



In [826]:

```
confusion = confusion_matrix(y_test,y_pred )
```

In [827]:

```
print("Confusion Matrix:\n", confusion)
```

```
Confusion Matrix:
 [[24 27]
 [17 36]]
```

In [828]:

```
classification_rep = classification_report(y_test, y_pred)
```

In [829]:

```
print("Classification Report:\n", classification_rep)
```

```
Classification Report:
               precision    recall  f1-score   support

           0       0.59      0.47      0.52        51
           1       0.57      0.68      0.62        53

    accuracy                           0.58       104
   macro avg       0.58      0.57      0.57       104
weighted avg       0.58      0.58      0.57       104
```

cross validation

In [830]:

```
from sklearn.model_selection import cross_val_score, KFold

model = RandomForestClassifier(n_estimators=100, random_state=42)


num_folds = 15


kf = KFold(n_splits=num_folds, shuffle=True, random_state=42)


scores = cross_val_score(model, X, y, cv=kf, scoring='accuracy')  # Replace X c


print("Cross-Validation Scores:", scores)


mean_score = scores.mean()
std_score = scores.std()

print(f"Mean Accuracy: {mean_score:.2f}")
print(f"Standard Deviation: {std_score:.2f}")
```

```
Cross-Validation Scores: [0.65714286 0.48571429 0.65714286 0.65714286 0.71428
571 0.6
 0.57142857 0.67647059 0.44117647 0.55882353 0.58823529 0.58823529
 0.64705882 0.64705882 0.67647059]
Mean Accuracy: 0.61
Standard Deviation: 0.07
```

implementation of LogisticRegression

In [831]:

```
from sklearn.linear_model import LogisticRegression
clf = LogisticRegression(random_state=42,max_iter=10000)
clf.fit(X_train, y_train)
```

Out[831]:

```
LogisticRegression(max_iter=10000, random_state=42)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [832]:

```
y_pred = clf.predict(X_test)
```

In [833]:

```
accuracy = accuracy_score(y_test, y_pred)
```

In [834]:

```
print("Accuracy:", accuracy)
```

```
Accuracy: 0.5288461538461539
```

In [835]:

```
confusion = confusion_matrix(y_test, y_pred)
```

In [836]:

```
print("Confusion Matrix:\n", confusion)
```

```
Confusion Matrix:
 [[23 28]
 [21 32]]
```

In [837]:

```
classification_rep = classification_report(y_test, y_pred)
print("Classification Report:\n", classification_rep)
```

```
Classification Report:
               precision    recall  f1-score   support

           0       0.52      0.45      0.48        51
           1       0.53      0.60      0.57        53

    accuracy                           0.53       104
   macro avg       0.53      0.53      0.53       104
weighted avg       0.53      0.53      0.53       104
```

implementation of DecisionTreeClassifier

In [844]:

```
from sklearn.tree import DecisionTreeClassifier

# Create a Decision Tree classifier
clf = DecisionTreeClassifier(random_state=42)

# Train the classifier on the training data
clf.fit(X_train, y_train)
```

Out[844]:

```
DecisionTreeClassifier(random_state=42)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [845]:

```
y_pred = clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.49038461538461536
```

In [846]:

```
confusion = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", confusion)
```

```
Confusion Matrix:
 [[31 20]
 [33 20]]
```

In [847]:

```
classification_rep = classification_report(y_test, y_pred)
print("Classification Report:\n", classification_rep)
```

```
Classification Report:
               precision    recall  f1-score   support

           0       0.48      0.61      0.54        51
           1       0.50      0.38      0.43        53

    accuracy                           0.49       104
   macro avg       0.49      0.49      0.48       104
weighted avg       0.49      0.49      0.48       104
```