Classify the Size_Categorie using SVM

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
DMC: DMC index from the FWI system: 1.1 to 291.3
DC: DC index from the FWI system: 7.9 to 860.6
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
Size_Categorie: the burned area of the forest (Small , Large)

In [5]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns


%matplotlib inline
sns.set_style('darkgrid')
```

In [3]:

```
forestfires = pd.read_csv("forestfires.csv")
forestfires.head()
```

Out[3]:

	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	...	monthfeb	monthjan	monthjul	monthjun	monthmar	monthmay	mont
0	mar	fri	86.2	26.2	94.3	5.1	8.2	51	6.7	0.0	...	0	0	0	0	1	0	
1	oct	tue	90.6	35.4	669.1	6.7	18.0	33	0.9	0.0	...	0	0	0	0	0	0	
2	oct	sat	90.6	43.7	686.9	6.7	14.6	33	1.3	0.0	...	0	0	0	0	0	0	
3	mar	fri	91.7	33.3	77.5	9.0	8.3	97	4.0	0.2	...	0	0	0	0	1	0	
4	mar	sun	89.3	51.3	102.2	9.6	11.4	99	1.8	0.0	...	0	0	0	0	1	0	

5 rows × 31 columns

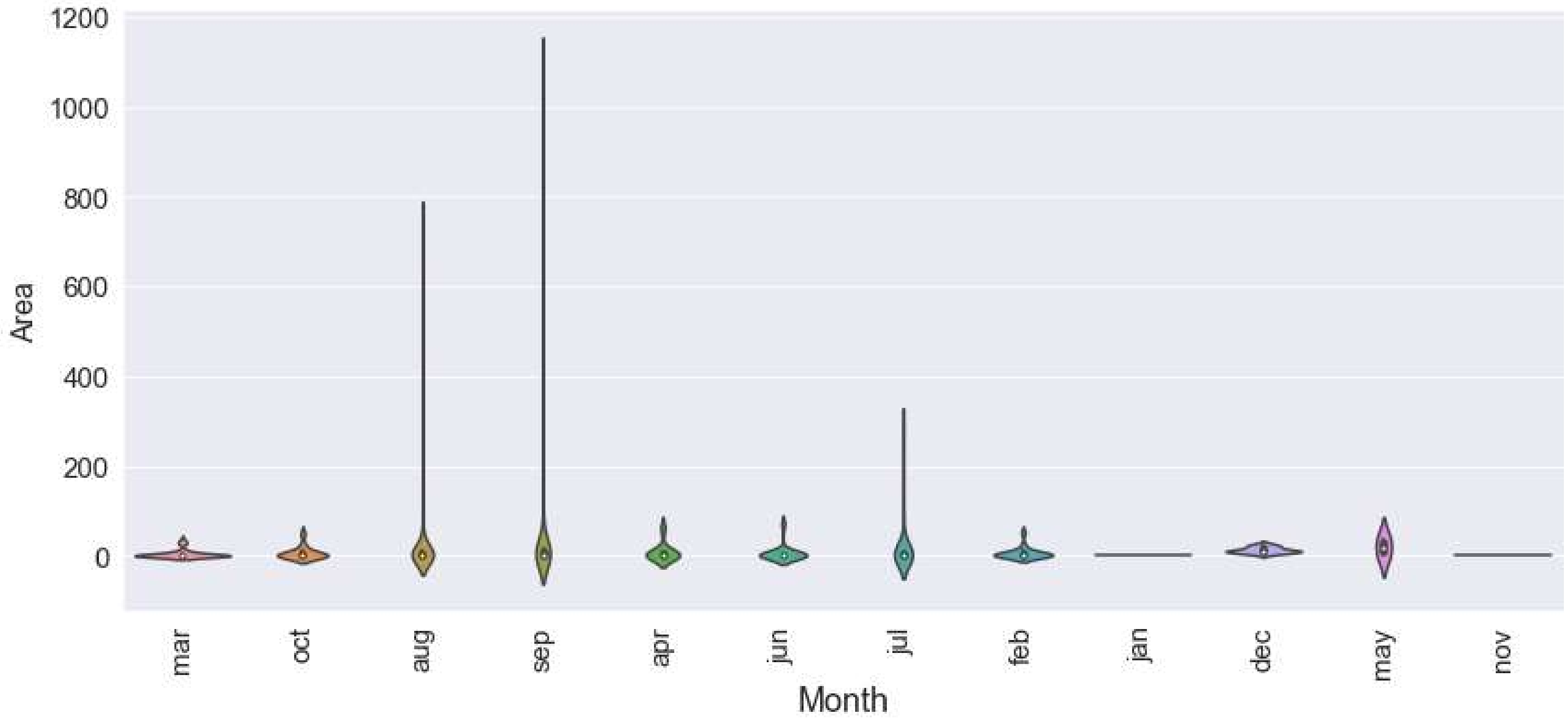
In [4]:

```
forestfires.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 517 entries, 0 to 516
Data columns (total 31 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   month          517 non-null    object 
 1   day            517 non-null    object 
 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   rain           517 non-null    float64
 10  area           517 non-null    float64
 11  dayfri         517 non-null    int64  
 12  daymon         517 non-null    int64  
 13  daysat         517 non-null    int64  
 14  daysun         517 non-null    int64  
 15  daythu         517 non-null    int64  
 16  daytue         517 non-null    int64  
 17  daywed         517 non-null    int64  
 18  monthapr       517 non-null    int64  
 19  monthaug       517 non-null    int64  
 20  monthdec       517 non-null    int64  
 21  monthfeb       517 non-null    int64  
 22  monthjan       517 non-null    int64  
 23  monthjul       517 non-null    int64  
 24  monthjun       517 non-null    int64  
 25  monthmar       517 non-null    int64  
 26  monthmay       517 non-null    int64  
 27  monthnov       517 non-null    int64  
 28  monthoct       517 non-null    int64  
 29  monthsep       517 non-null    int64  
 30  size_category  517 non-null    object 
dtypes: float64(8), int64(20), object(3)
memory usage: 125.3+ KB
```

In [23]:

```
plt.figure(figsize=(14,6))
sns.violinplot(x = 'month', y= "area",data = forestfires)
plt.xticks(rotation = 90, size = 15)
plt.yticks(size = 15)
plt.xlabel('Month',fontsize=18)
plt.ylabel('Area', fontsize=16)

plt.show()
```



In [40]:

```
forestfires['area_km'] = forestfires['area'] / 100

forestfires.head()
```

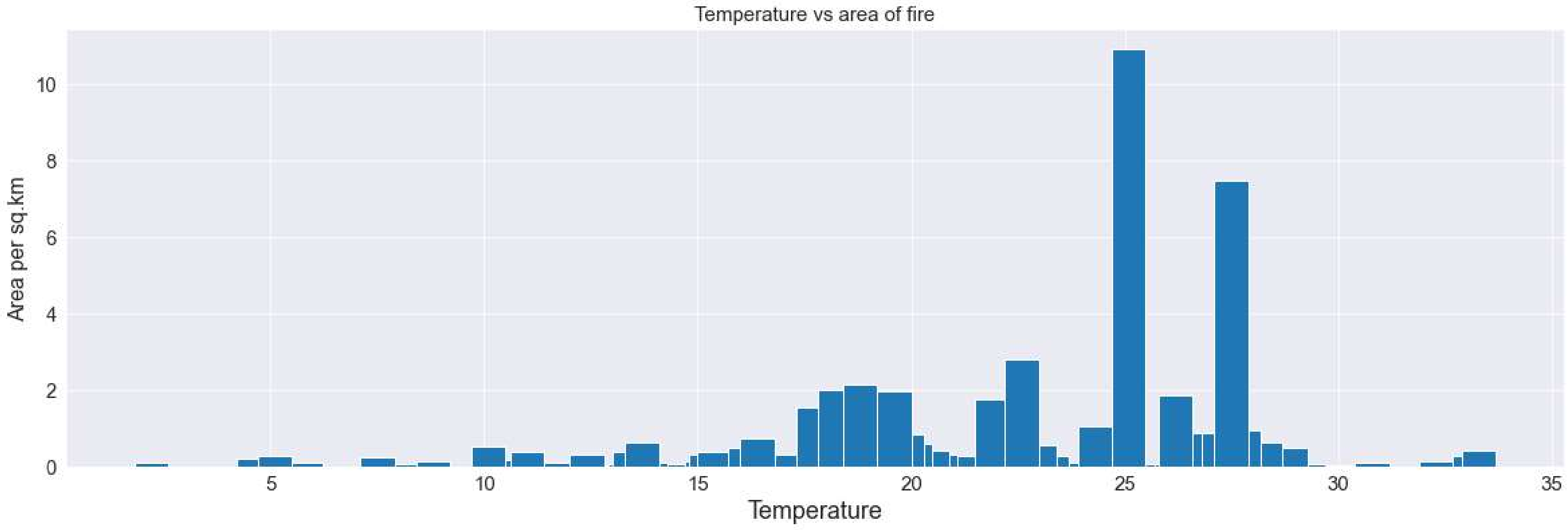
Out[40]:

	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	...	monthjan	monthjul	monthjun	monthmar	monthmay	monthnov	mont
0	mar	fri	86.2	26.2	94.3	5.1	8.2	51	6.7	0.0	...	0	0	0	1	0	0	
1	oct	tue	90.6	35.4	669.1	6.7	18.0	33	0.9	0.0	...	0	0	0	0	0	0	
2	oct	sat	90.6	43.7	686.9	6.7	14.6	33	1.3	0.0	...	0	0	0	0	0	0	
3	mar	fri	91.7	33.3	77.5	9.0	8.3	97	4.0	0.2	...	0	0	0	1	0	0	
4	mar	sun	89.3	51.3	102.2	9.6	11.4	99	1.8	0.0	...	0	0	0	1	0	0	

5 rows × 32 columns

In [41]:

```
highest_fire_area = forestfires.sort_values(by="area_km", ascending=True)

plt.figure(figsize=(20, 6))

plt.title("Temperature vs area of fire" , fontsize=15)
plt.bar(highest_fire_area['temp'], highest_fire_area['area_km'])

plt.xticks(size = 15)
plt.yticks(size = 15)
plt.xlabel('Temperature',fontsize=18)
plt.ylabel('Area per sq.km', fontsize=16)
plt.show()
```



In [38]:

```
numerical_feature = forestfires.describe(include=["int", "float"]).columns

print(list(numerical_feature))
```

```
['FFMC', 'DMC', 'DC', 'ISI', 'temp', 'wind', 'rain', 'area']
```

In [42]:

```
categorical_feature = forestfires.describe(include=["object"]).columns

print(list(categorical_feature))
```

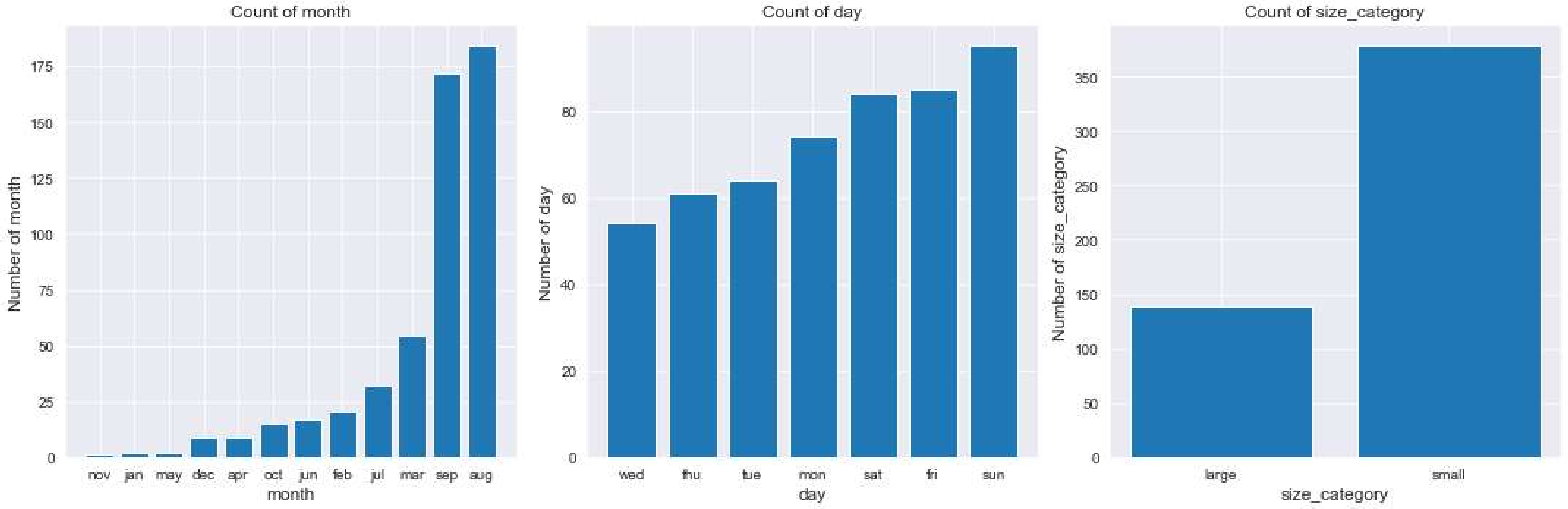
```
['month', 'day', 'size_category']
```

Categorical features

In [47]:

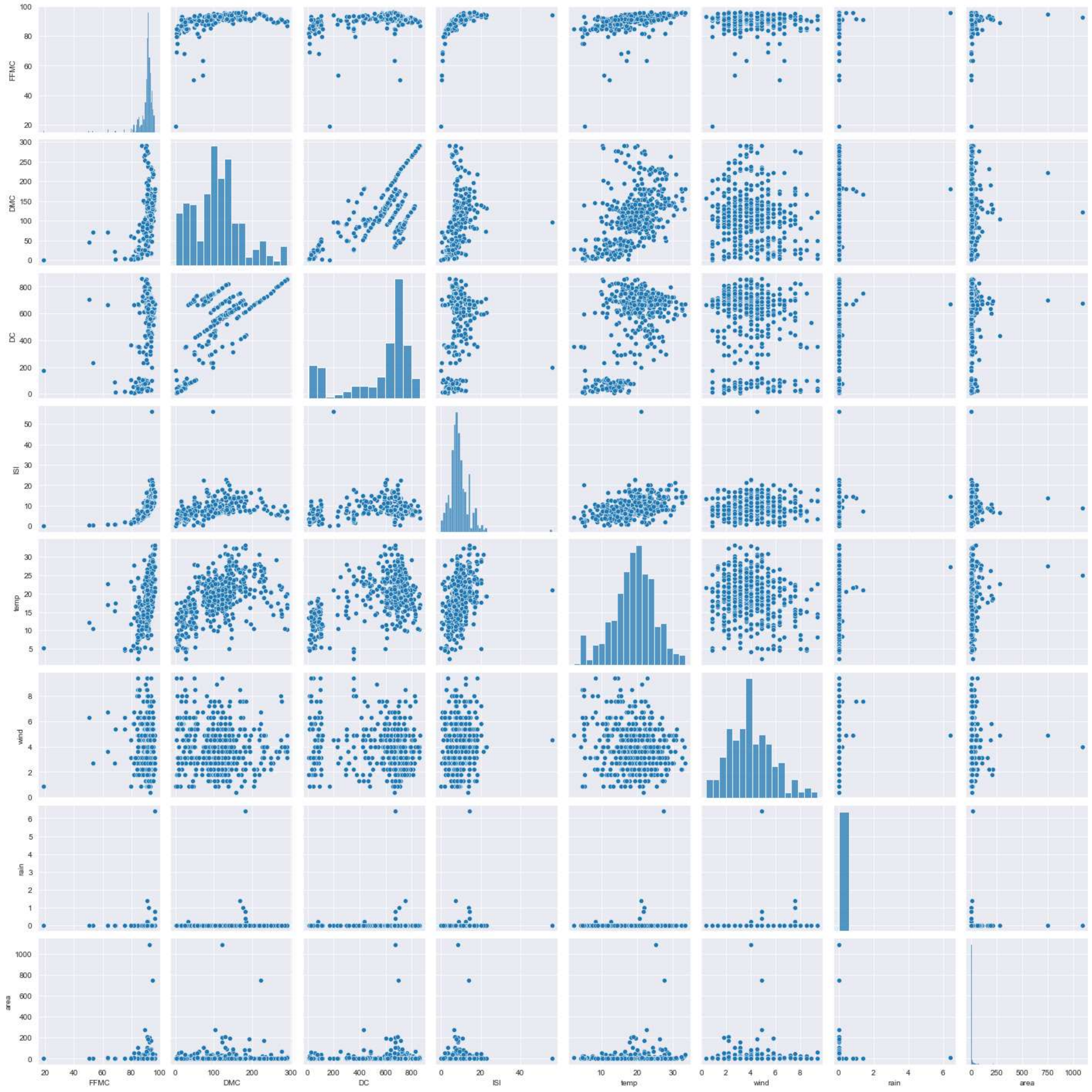
```
print(categorical_feature)


plt.figure(figsize=(15, 5))
for idx, column in enumerate(categorical_feature):
    df = forestfires.copy()
    unique = df[column].value_counts(ascending=True);

    plt.subplot(1, 3, idx+1)
    plt.title("Count of "+ column)
    plt.bar(unique.index, unique.values);

    plt.xlabel(column, fontsize=12)
    plt.ylabel("Number of "+ column, fontsize=12)

plt.tight_layout()
plt.show()
```

```
Index(['month', 'day', 'size_category'], dtype='object')
```



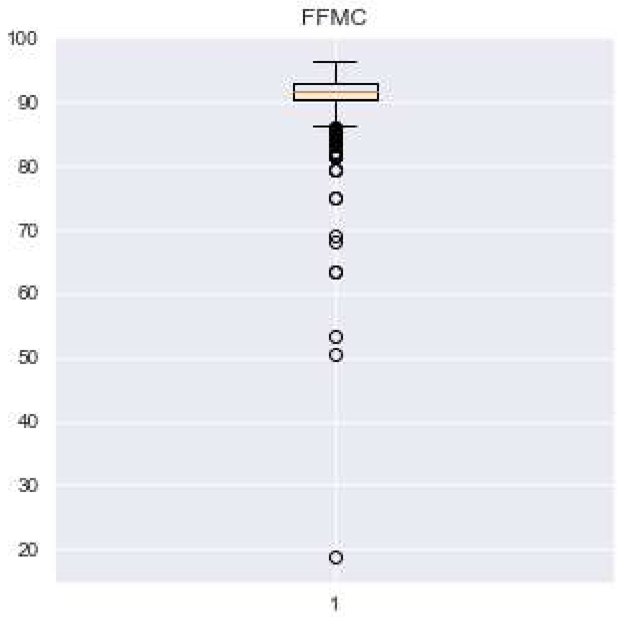
Numerical features

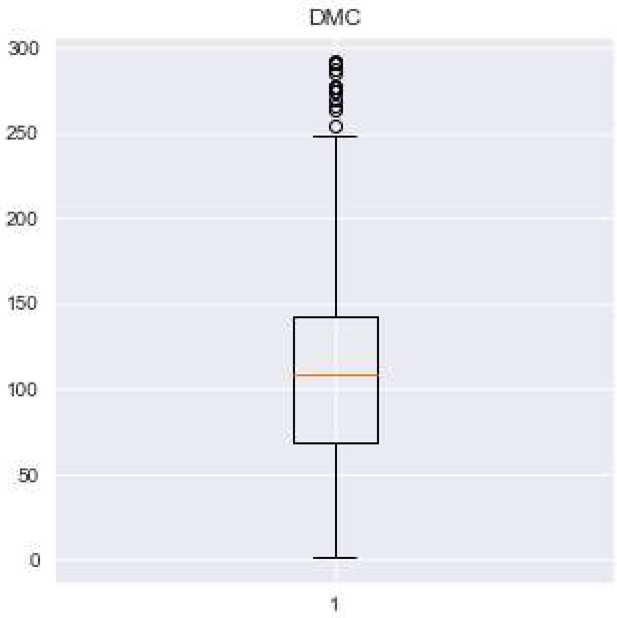
In [49]:
```
sns.set_style('darkgrid')
sns.pairplot(forestfires[numerical_feature])
plt.show()
```

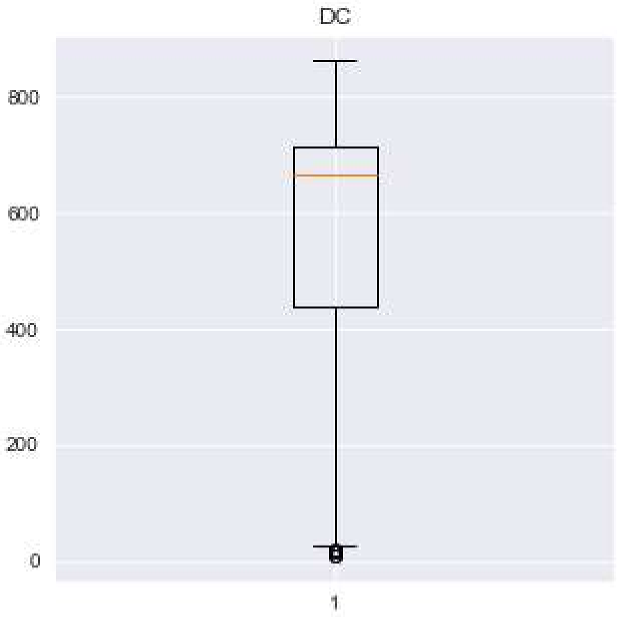


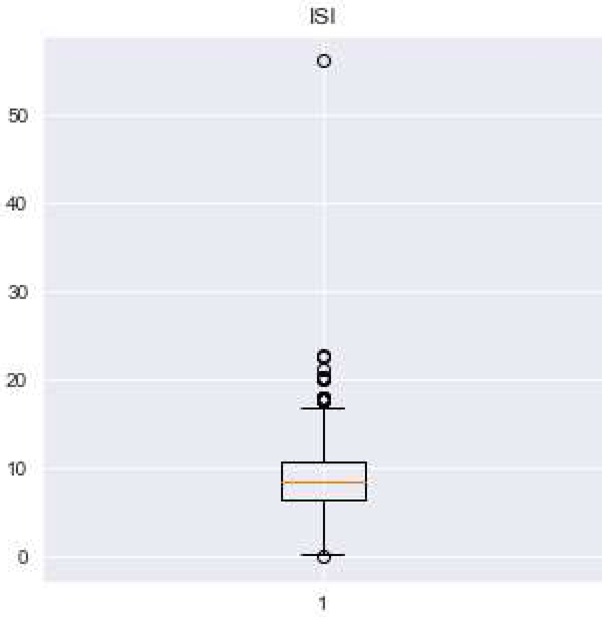
Outliers

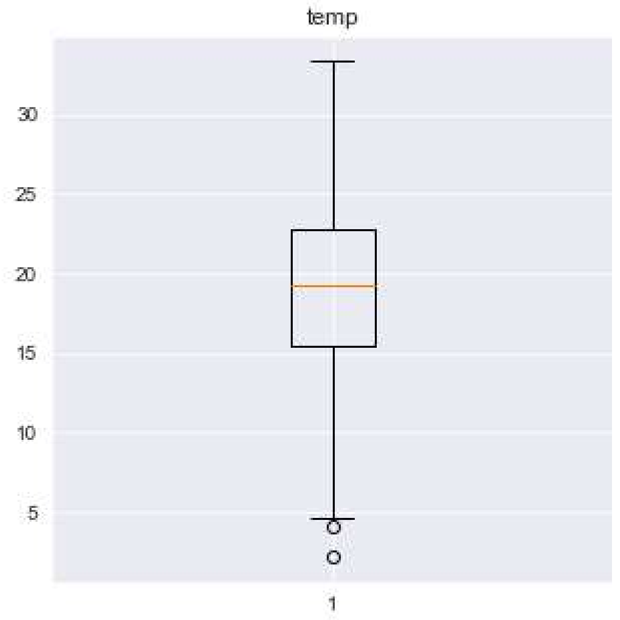
```
In [52]: for idx, col  in enumerate(numerical_feature, 1):
             plt.figure(figsize=(5,5))
             plt.boxplot(forestfires[col])

             plt.title(col)

         #plt.tight_layout()
         plt.show(plt)
```

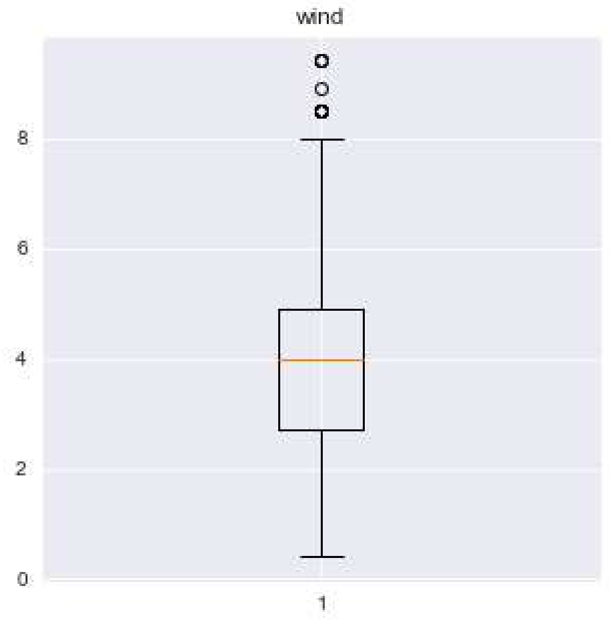


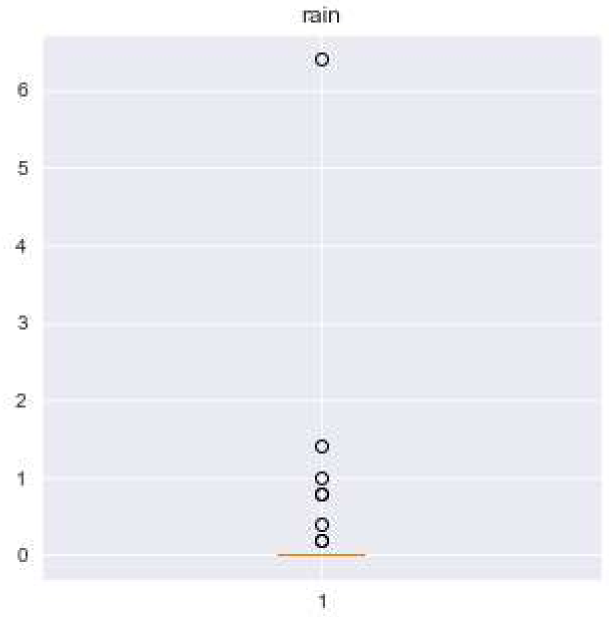


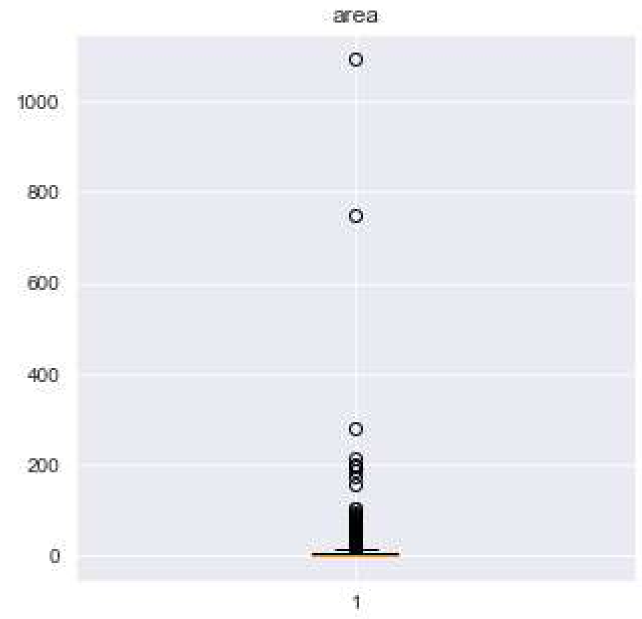






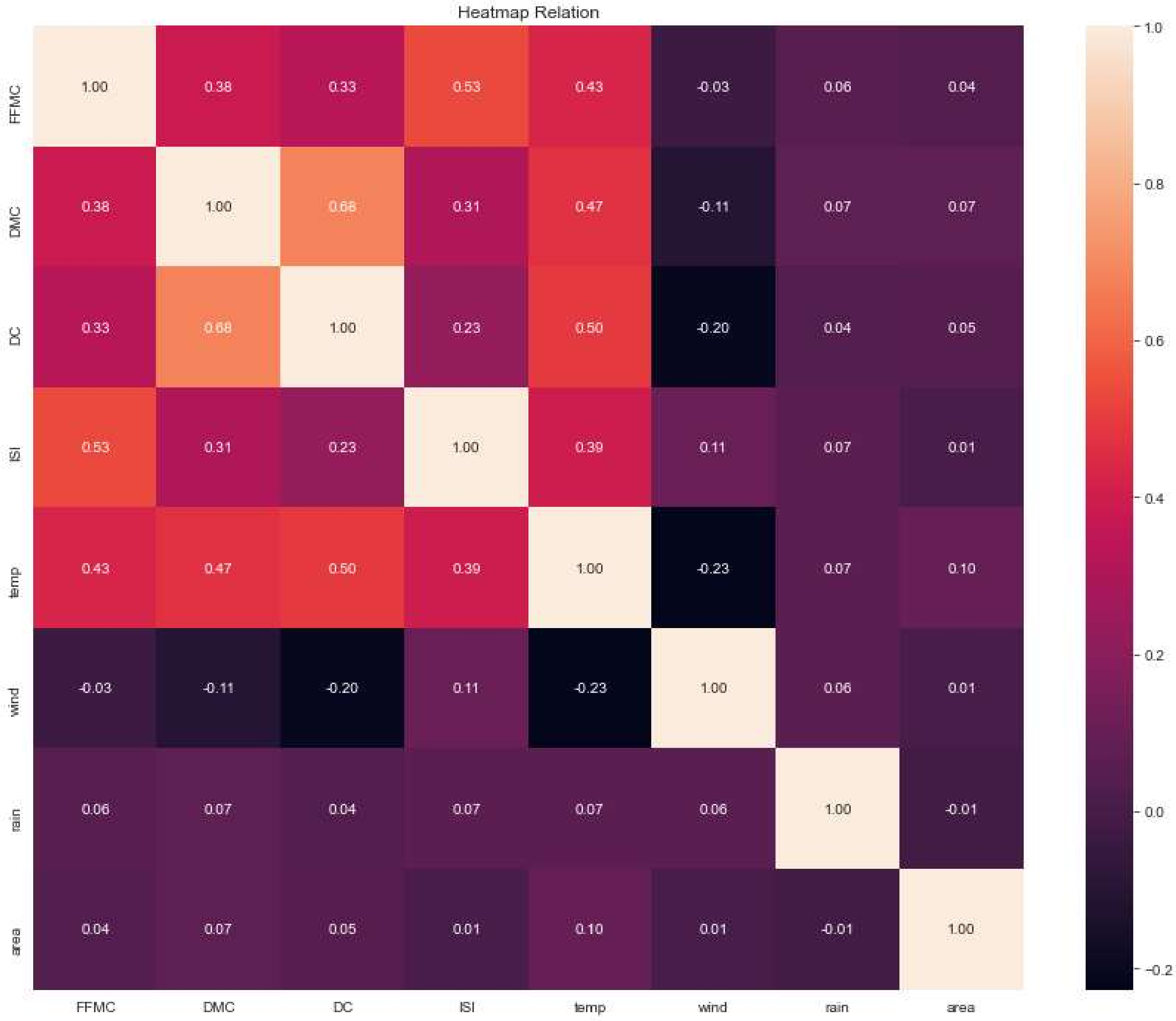






In [53]:

```
#Heatmap

plt.figure(figsize=(15, 12))

plt.title("Heatmap Relation")

sns.heatmap(forestfires[numerical_feature].corr(), annot=True, fmt='.2f');
```



In [59]:

```
#Dropping the month and day columns
forestfires.drop(["month","day"],axis=1,inplace =True)
```

In [62]:

```
X = forestfires.iloc[:,0:28]
y = forestfires.iloc[:,28]
```

In [63]:

```
# Normalizing the data
def norm_func(i):
    x= (i-i.min())/(i.max()-i.min())
    return (x)
```

In [64]:

```
X_ = norm_func(X)
```

In [78]:

```
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split, GridSearchCV

X_train,X_test,y_train,y_test = train_test_split(X_,y,test_size = 0.25, stratify = y)
```

In [74]:

```
model_linear = SVC(kernel = "linear")
model_linear.fit(X_train,y_train)
pred_test = model_linear.predict(X_test)
np.mean(pred_test==y_test)
```

Out[74]:

```
0.7615384615384615
```

In [73]:

```
# kernel = rbf
model_rbf = SVC(kernel = "rbf")
model_rbf.fit(X_train,y_train)
pred_test_rbf = model_rbf.predict(X_test)
np.mean(pred_test_rbf==y_test)
```

Out[73]:

```
0.7538461538461538
```

In [76]:

```
# Kernel = poly
model_poly = SVC(kernel = "poly")
model_poly.fit(X_train,y_train)
pred_test_poly = model_poly.predict(X_test)

np.mean(pred_test_poly==y_test)
```

Out[76]:

```
0.7538461538461538
```

In [77]:

```
#'sigmoid'
model_sig = SVC(kernel = "sigmoid")
model_sig.fit(X_train,y_train)
pred_test_sig = model_rbf.predict(X_test)

np.mean(pred_test_sig==y_test)
```

Out[77]:
```
0.7538461538461538
```

Parameters selection

In [88]:
```
SVMC = SVC(random_state=42)

svc_param_grid = {'kernel': ['rbf', 'sigmoid', 'poly','linear'],
                  'gamma': [1, 0.1, 0.01, 0.001],
                  'C': [1000, 100, 10, 1],
                  'tol':[0.001,0.0008,0.0009,0.0011]}

gsSVMC = GridSearchCV(SVMC, param_grid = svc_param_grid, cv = 5, scoring = "accuracy", n_jobs = 6, verbose = 1)

gsSVMC.fit(X_train,y_train)

svm_best = gsSVMC.best_estimator_

gsSVMC.best_score_
```

```
Fitting 5 folds for each of 256 candidates, totalling 1280 fits
[Parallel(n_jobs=6)]: Using backend LokyBackend with 6 concurrent workers.
[Parallel(n_jobs=6)]: Done  38 tasks      | elapsed:    4.5s
[Parallel(n_jobs=6)]: Done 407 tasks      | elapsed:   15.3s
[Parallel(n_jobs=6)]: Done 1280 out of 1280 | elapsed:   20.6s finished
```

Out[88]:
```
0.938061938061938
```

In [89]:
```
gsSVMC.best_params_
```

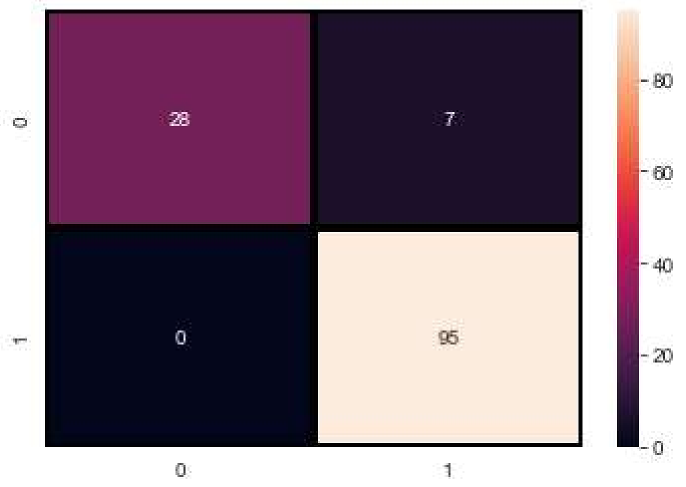
Out[89]:
```
{'C': 1000, 'gamma': 1, 'kernel': 'linear', 'tol': 0.001}
```

In [90]:
```
predict_results = svm_best.predict(X_test)
np.mean(predict_results==y_test)
```

Out[90]:
```
0.9461538461538461
```

In [91]:
```
from sklearn.metrics import confusion_matrix
sns.heatmap(confusion_matrix(y_test, predict_results),annot=True,fmt = "d",linecolor="k",linewidths=3)
```

Out[91]:
```
<AxesSubplot:>
```


28
7
0
95
0
1
0
1
80
60
40
20
0