In [3]:
```
import os

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


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

In [2]:
```
df=pd.read_csv('forestfires.csv')
df
```

Out[2]:

	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

In [4]:
```
df.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 [5]:
```
df.shape
```

Out[5]:
```
(517, 13)
```

In [6]:
```
df.head(10)
```

Out[6]:

	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	0.0
1	7	4	oct	tue	90.6	35.4	669.1	6.7	18.0	33	0.9	0.0	0.0
2	7	4	oct	sat	90.6	43.7	686.9	6.7	14.6	33	1.3	0.0	0.0
3	8	6	mar	fri	91.7	33.3	77.5	9.0	8.3	97	4.0	0.2	0.0
4	8	6	mar	sun	89.3	51.3	102.2	9.6	11.4	99	1.8	0.0	0.0
5	8	6	aug	sun	92.3	85.3	488.0	14.7	22.2	29	5.4	0.0	0.0
6	8	6	aug	mon	92.3	88.9	495.6	8.5	24.1	27	3.1	0.0	0.0
7	8	6	aug	mon	91.5	145.4	608.2	10.7	8.0	86	2.2	0.0	0.0
8	8	6	sep	tue	91.0	129.5	692.6	7.0	13.1	63	5.4	0.0	0.0
9	7	5	sep	sat	92.5	88.0	698.6	7.1	22.8	40	4.0	0.0	0.0

In [7]:
```
df.describe()
```

Out[7]:

	X	Y	FFMC	DMC	DC	ISI	temp	
count	517.000000	517.000000	517.000000	517.000000	517.000000	517.000000	517.000000	517.00
mean	4.669246	4.299807	90.644681	110.872340	547.940039	9.021663	18.889168	44.28
std	2.313778	1.229900	5.520111	64.046482	248.066192	4.559477	5.806625	16.31
min	1.000000	2.000000	18.700000	1.100000	7.900000	0.000000	2.200000	15.00
25%	3.000000	4.000000	90.200000	68.600000	437.700000	6.500000	15.500000	33.00
50%	4.000000	4.000000	91.600000	108.300000	664.200000	8.400000	19.300000	42.00
75%	7.000000	5.000000	92.900000	142.400000	713.900000	10.800000	22.800000	53.00
max	9.000000	9.000000	96.200000	291.300000	860.600000	56.100000	33.300000	100.00

In [8]:
```
sns.set_theme(style="whitegrid")
df.shape
```

Out[8]:
```
(517, 13)
```

In [9]:
```
Q1 = df.quantile(0.25) #first 25% of the data
Q3 = df.quantile(0.75) #first 75% of the data
IQR = Q3 - Q1 #IQR = InterQuartile Range
scale = 2 #For Normal Distributions, scale = 1.5
lower_lim = Q1 - scale*IQR
upper_lim = Q3 + scale*IQR
lower_outliers = (df[df.columns[2:13]] < lower_lim)
upper_outliers = (df[df.columns[2:13]] > upper_lim)
```

```
C:\Users\Dhruv Kumar\AppData\Local\Temp\ipykernel_18344\3886136910.py:7: FutureWar
ning: Automatic reindexing on DataFrame vs Series comparisons is deprecated and wi
ll raise ValueError in a future version. Do `left, right = left.align(right, axis=
1, copy=False)` before e.g. `left == right`
  lower_outliers = (df[df.columns[2:13]] < lower_lim)
C:\Users\Dhruv Kumar\AppData\Local\Temp\ipykernel_18344\3886136910.py:8: FutureWar
ning: Automatic reindexing on DataFrame vs Series comparisons is deprecated and wi
ll raise ValueError in a future version. Do `left, right = left.align(right, axis=
1, copy=False)` before e.g. `left == right`
  upper_outliers = (df[df.columns[2:13]] > upper_lim)
```

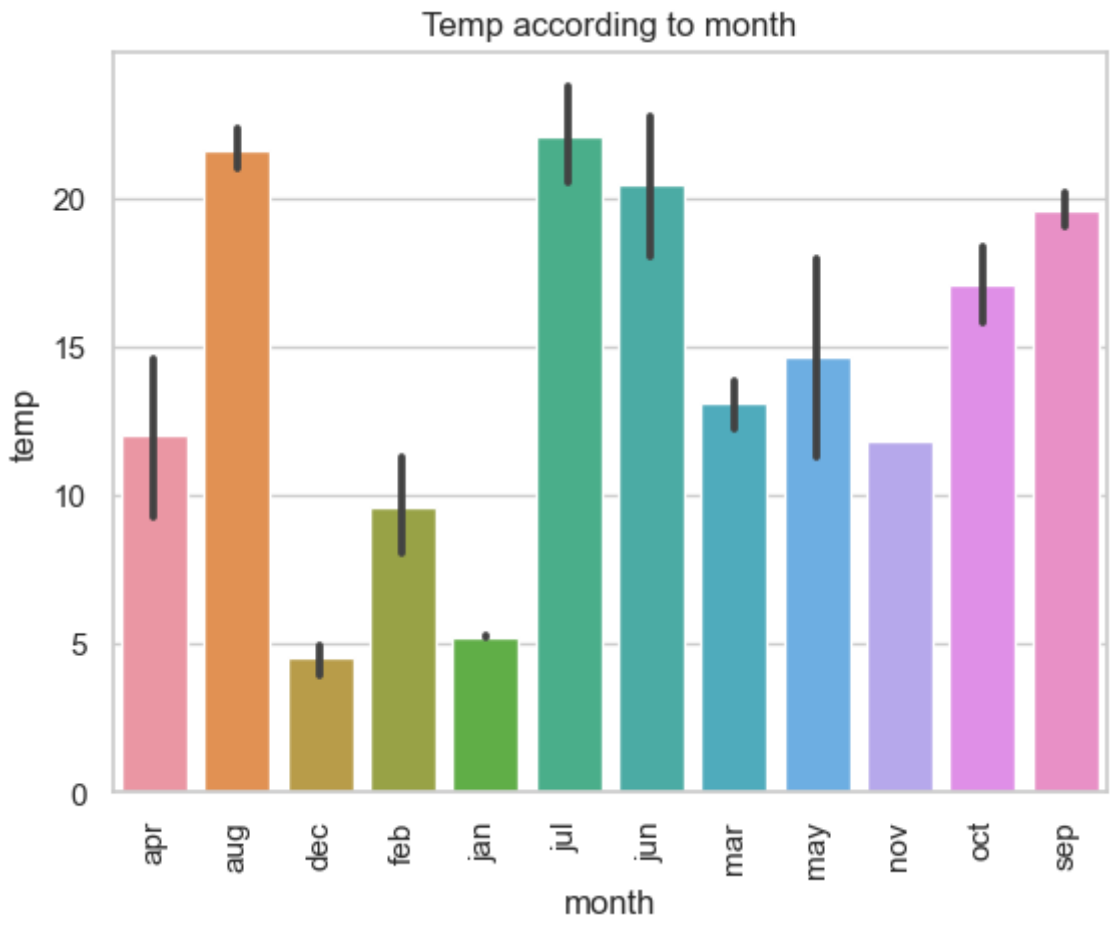
In [10]:
```
df[df.columns[2:13]][(lower_outliers | upper_outliers)].info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 517 entries, 0 to 516
Data columns (total 11 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   month   0 non-null      object 
 1   day     0 non-null      object 
 2   FFMC    37 non-null     float64
 3   DMC     1 non-null      float64
 4   DC      0 non-null      float64
 5   ISI     10 non-null     float64
 6   temp    0 non-null      float64
 7   RH      5 non-null      float64
 8   wind    4 non-null      float64
 9   rain    8 non-null      float64
 10  area    59 non-null     float64
dtypes: float64(9), object(2)
memory usage: 44.6+ KB
```

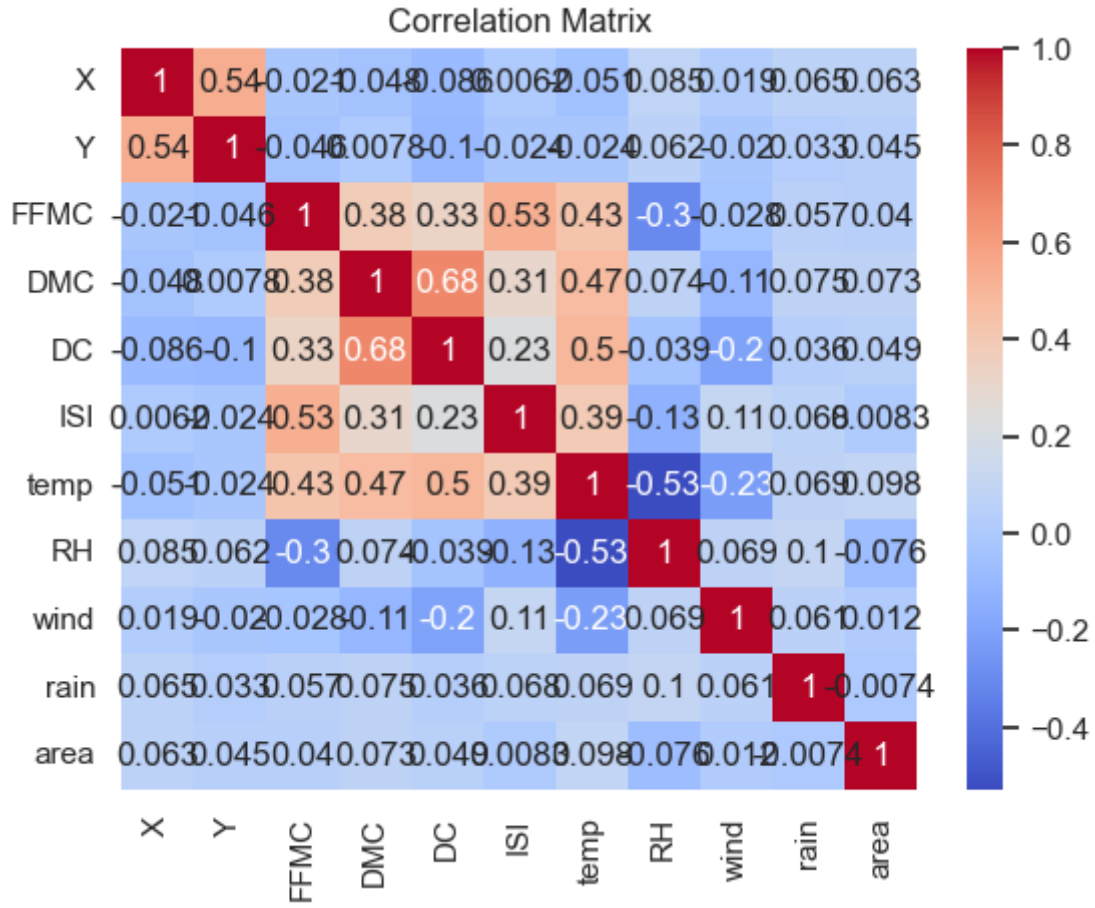
In [13]:
```
df.columns
```

Out[13]:
```
Index(['X', 'Y', 'month', 'day', 'FFMC', 'DMC', 'DC', 'ISI', 'temp', 'RH',
       'wind', 'rain', 'area'],
      dtype='object')
```

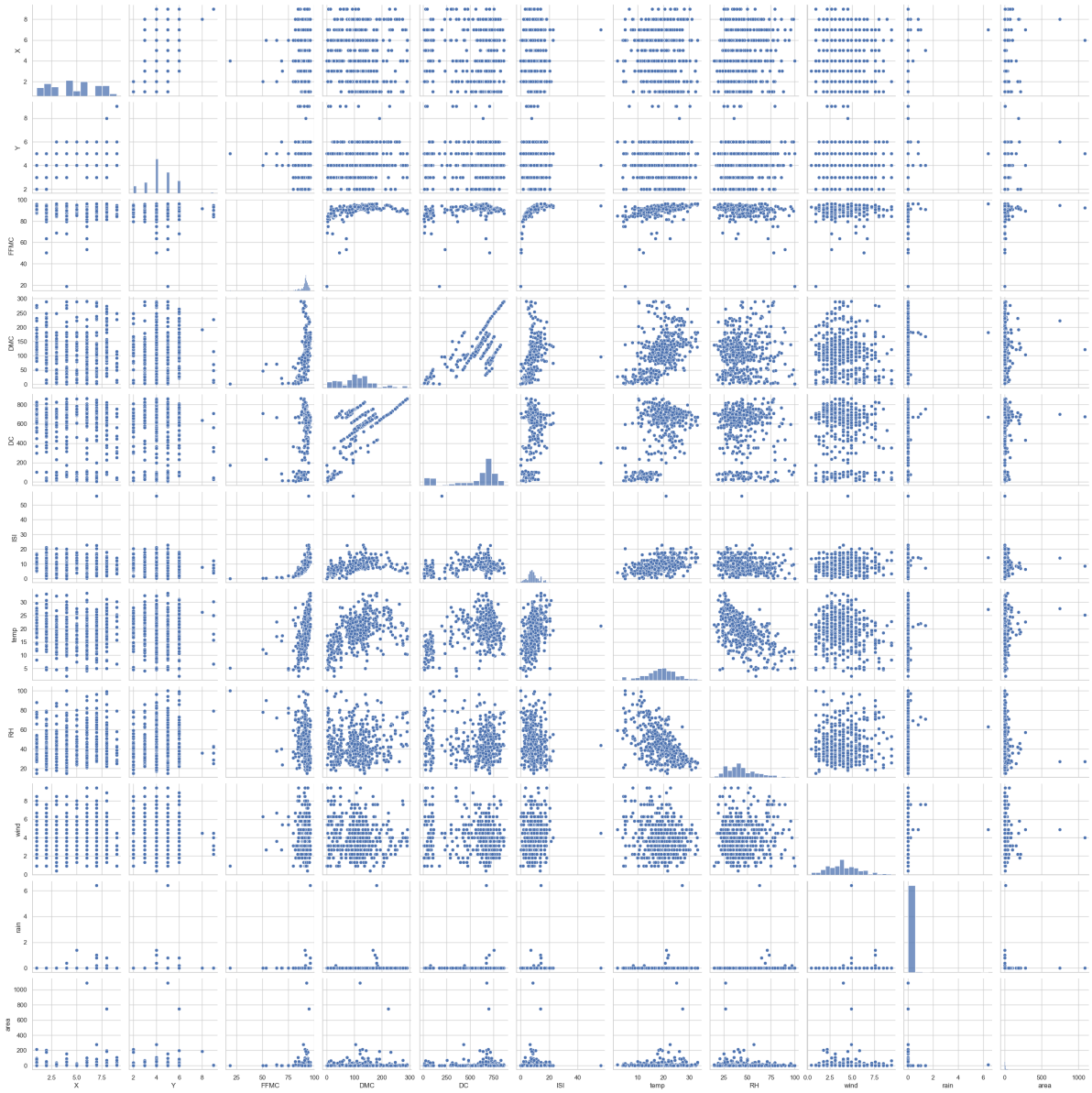
In [22]:
```
sns.barplot(x='month',y='temp', data=df.sort_values('month'))
plt.title('Temp according to month')
plt.xticks(rotation=90)
plt.show()
```



In [23]:
```
sns.heatmap(df.corr(),annot=True,cmap = 'coolwarm')
plt.title('Correlation Matrix')
plt.show()
```



In [24]:
```
sns.pairplot(df)
plt.show()
```



In [26]:
```
from sklearn.model_selection import train_test_split
Y = df['month'] #variável de predição
X = df.drop(['day'], axis=1)
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2)
print(X_train.shape, X_test.shape)
```

```
(413, 12) (104, 12)
```

In [ ]: