

**Objective**

Forest fires help in the natural cycle of woods' growth and replenishment. They Clear dead trees, leaves, and competing vegetation from the forest floor, so new plants can grow. Remove weak or disease-ridden trees, leaving more space and nutrients for stronger trees.

But when fires burn too hot and uncontrollable or when they’re in the “wildland-urban interface” (the places where woodlands and homes or other developed areas meet), they can be damaging and life threatening.

In this kernel, our aim is to predict the burned area (area) of forest fires, in the northeast region of Portugal. Based on the spatial, temporal, and weather variables where the fire is spotted.

This prediction can be used for calculating the forces sent to the incident and deciding the urgency of the situation.

For Further Info, Read: [MyLandPlan](https://mylandplan.org/content/good-and-bad-forest-fires)

target = 'area'

**RMSE**

RMSE is the most popular evaluation metric used in regression problems. It follows an assumption that errors are unbiased and follow a normal distribution.

Further Read: [Analytics Vidhya](https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/)

**Dependencies**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

plt.style.use('ggplot')

import statsmodels.api as sm

from statsmodels.compat import lzip

import statsmodels.stats.api as sms

from statsmodels.formula.api import ols

from scipy.stats import zscore

from statsmodels.stats.stattools import durbin\_watson

from sklearn.model\_selection import train\_test\_split,KFold

from statsmodels.stats.outliers\_influence import variance\_inflation\_factor

from sklearn.metrics import mean\_squared\_error

from sklearn.feature\_selection import RFECV

from mlxtend.feature\_selection import SequentialFeatureSelector as sfs

from mlxtend.plotting import plot\_sequential\_feature\_selection as plot\_sfs

from sklearn.linear\_model import LinearRegression,RidgeCV,LassoCV,ElasticNetCV

**Load and describe data**

path = r'Datasets\forestfires.csv'

df = pd.read\_csv(path)

print(df.shape)

print(df.dtypes)

print(df.describe().T)

Output:

(517, 13)

X          int64

Y          int64

month     object

day       object

FFMC     float64

DMC      float64

DC       float64

ISI      float64

temp     float64

RH         int64

wind     float64

rain     float64

area     float64

dtype: object

      count        mean         std   min    25%     50%     75%      max

X     517.0    4.669246    2.313778   1.0    3.0    4.00    7.00     9.00

Y     517.0    4.299807    1.229900   2.0    4.0    4.00    5.00     9.00

FFMC  517.0   90.644681    5.520111  18.7   90.2   91.60   92.90    96.20

DMC   517.0  110.872340   64.046482   1.1   68.6  108.30  142.40   291.30

DC    517.0  547.940039  248.066192   7.9  437.7  664.20  713.90   860.60

ISI   517.0    9.021663    4.559477   0.0    6.5    8.40   10.80    56.10

temp  517.0   18.889168    5.806625   2.2   15.5   19.30   22.80    33.30

RH    517.0   44.288201   16.317469  15.0   33.0   42.00   53.00   100.00

wind  517.0    4.017602    1.791653   0.4    2.7    4.00    4.90     9.40

rain  517.0    0.021663    0.295959   0.0    0.0    0.00    0.00     6.40

area  517.0   12.847292   63.655818   0.0    0.0    0.52    6.57  1090.84

**Missing Value Treatment**

print(df.isna().sum().sum())

Output:

0

**Exploratory Data Analysis**

* **Univariate**
* **Bivariate**
* **Multivariate**

**Univariate**

**Target variable = “area”**

# Univariate

plt.figure(figsize=(16,5))

print("Skew: {}".format(df[target].skew()))

print("Kurtosis: {}".format(df[target].kurtosis()))

ax = sns.kdeplot(df[target],shade=True,color='g')

plt.xticks([i for i in range(0,1200,50)])

plt.show()

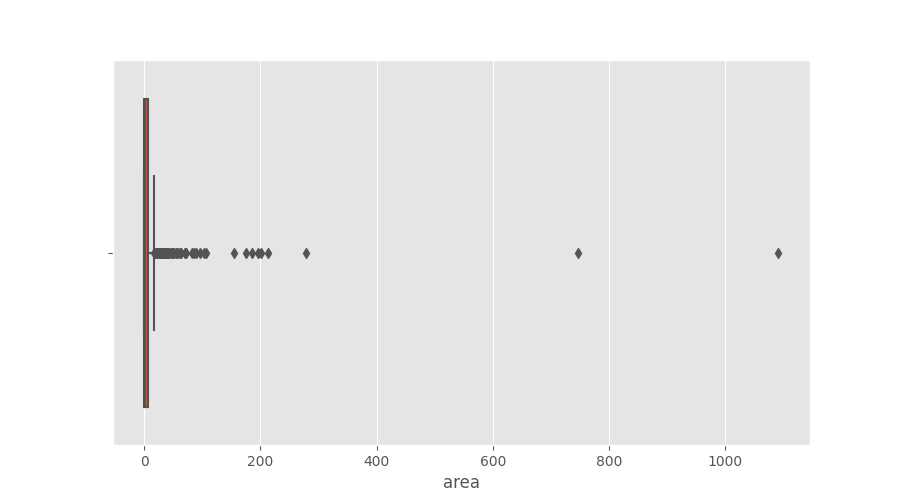
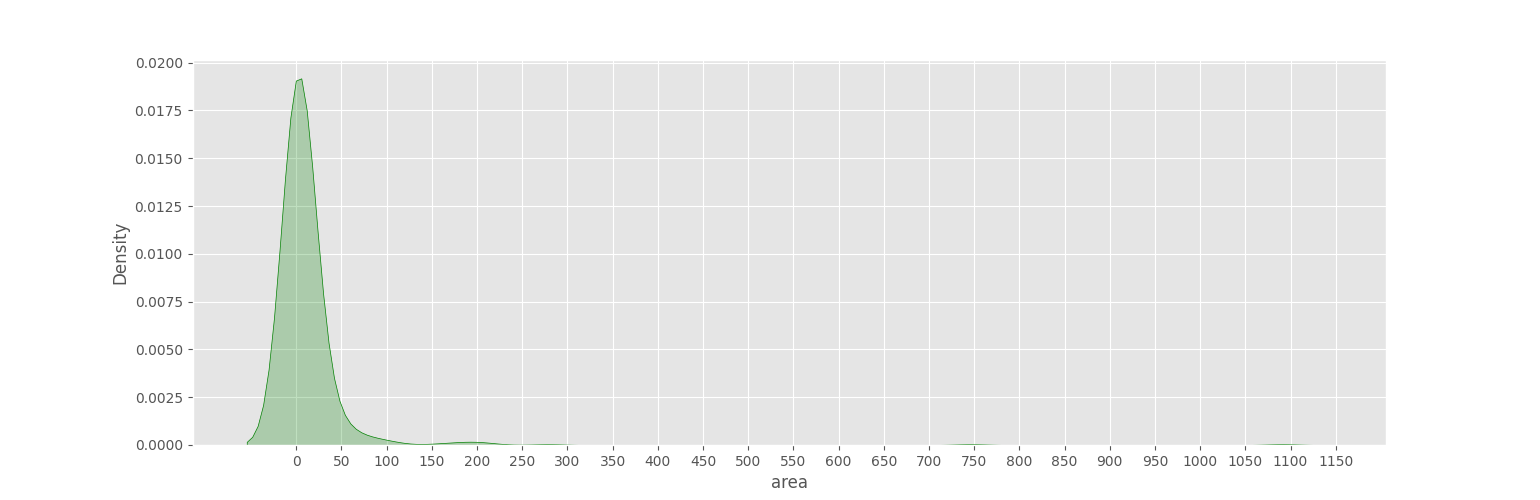
ax = sns.boxplot(df[target])

**Output:**

Skew: 12.846933533934868

Kurtosis: 194.1407210942299

**Graphs:**

****

**To Note:**

1. **Data is Highly Skewed (12.84)**
2. **Huge Kurtosis value of 194**
3. **Majority of the forest fires do not cover a large area, most of the damaged area is under 50 hectares of land.**
4. **We need to transform to fix skew and kurtosis.**
5. **We need to perform Outlier check.**

# Outlier points

y\_outliers = df[abs(zscore(df[target])) >= 3 ]

print(y\_outliers)

**Output:**

     X  Y month  day  FFMC    DMC     DC   ISI  temp  RH  wind  rain     area

237  1  2   sep  tue  91.0  129.5  692.6   7.0  18.8  40   2.2   0.0   212.88

238  6  5   sep  sat  92.5  121.1  674.4   8.6  25.1  27   4.0   0.0  1090.84

415  8  6   aug  thu  94.8  222.4  698.6  13.9  27.5  27   4.9   0.0   746.28

479  7  4   jul  mon  89.2  103.9  431.6   6.4  22.6  57   4.9   0.0   278.53

**Get Columns**

# Independent Columns

dfa = df.drop(columns=target)

cat\_columns = dfa.select\_dtypes(include='object').columns.tolist()

num\_columns = dfa.select\_dtypes(exclude='object').columns.tolist()

print(cat\_columns)

print(num\_columns)

**Output:**

['month', 'day']

['X', 'Y', 'FFMC', 'DMC', 'DC', 'ISI', 'temp', 'RH', 'wind', 'rain']

**Plotting Categorically**

# analyzing categorical columns

plt.figure(figsize=(16,10))

for i,col in enumerate(cat\_columns,1):

    plt.subplot(2,2,i)

    sns.countplot(data=dfa,y=col)

    plt.subplot(2,2,i+2)

    df[col].value\_counts(normalize=True).plot.bar()

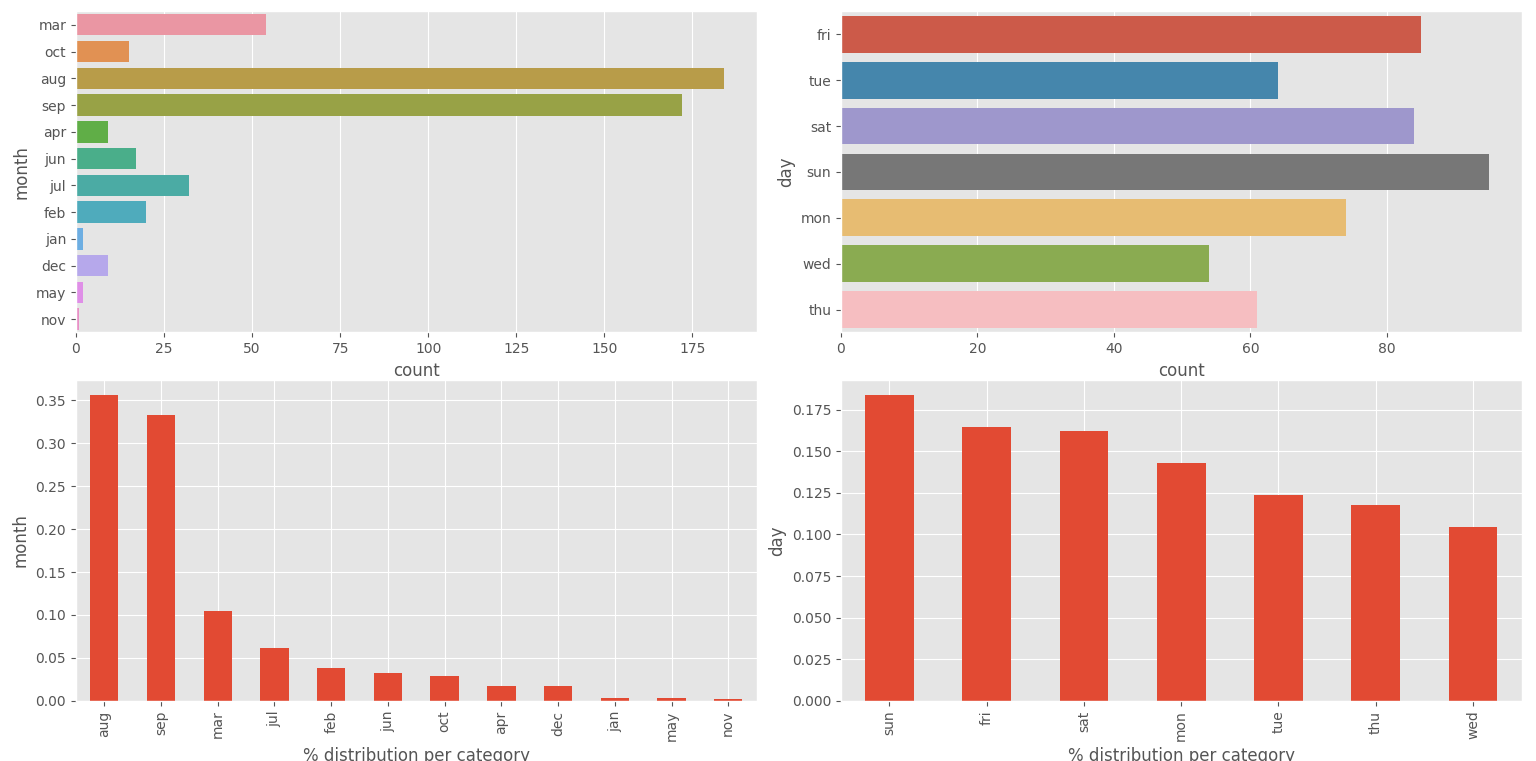
    plt.ylabel(col)

    plt.xlabel('% distribution per category')

plt.tight\_layout()

plt.show()

**Output Graph:**

****

**Observations:**

1. **Abnormally high number of the forest fires occur in the month of August and September.**
2. **No strong indicators but Friday to Monday have higher proportion of cases.**

**Plotting Numerically**

plt.figure(figsize=(18,40))

for i,col in enumerate(num\_columns,1):

    plt.subplot(8,4,i)

    sns.kdeplot(df[col],color='g',shade=True)

    plt.subplot(8,4,i+10)

    df[col].plot.box()

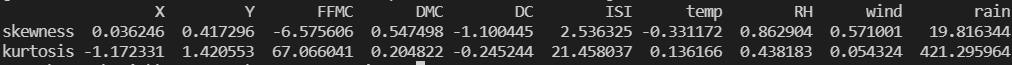
plt.tight\_layout()

plt.show()

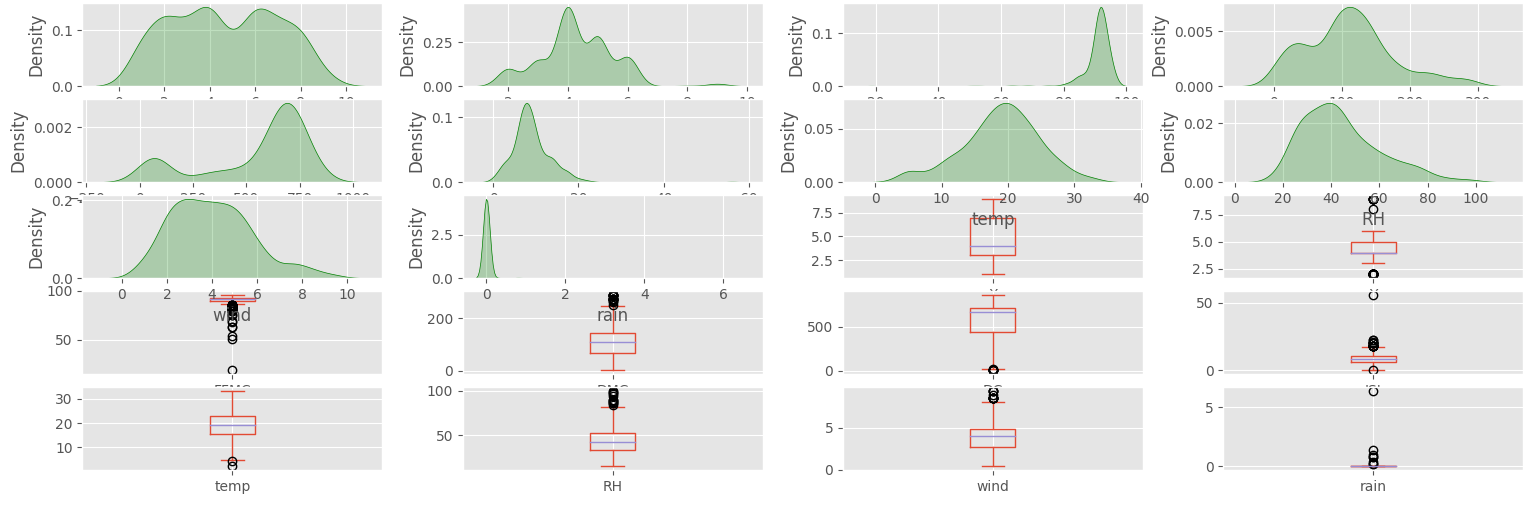
num\_data = df[num\_columns]

print(pd.DataFrame(data=[num\_data.skew(),num\_data.kurtosis()],index=['skewness','kurtosis']))

**Output:**

****

**Graph:**

****