Xavier Institute of Management Bhubaneswar

<u>Business Analytics with R</u>

A Project Report

on

<u>Predicting Forest Fires</u>

Submitted by: Karan Gupta
UM18092
BAR Section-1

Submitted to: Dr. Sandipan Karmakar
Assistant Professor
Area Coordinator, Data Science

Introduction

A wildfire, wildland fire or rural fire is an uncontrolled fire in an area of combustible vegetation occurring in rural areas. Depending on the type of vegetation present, a wildfire can also be classified more specifically as a brush fire, bushfire (in Australia), desert fire, forest fire, grass fire, hill fire, peat fire, vegetation fire, or veld fire. Many organizations consider wildfire to mean an unplanned and unwanted fire, while wildland fire is a broader term that includes prescribed fire as well as wildland fire use (WFU; these are also called monitored response fires).

Fossil charcoal indicates that wildfires began soon after the appearance of terrestrial plants 420 million years ago. Wildfire's occurrence throughout the history of terrestrial life invites conjecture that fire must have had pronounced evolutionary effects on most ecosystems' flora and fauna. Earth is an intrinsically flammable planet owing to its cover of carbon-rich vegetation, seasonally dry climates, atmospheric oxygen, and widespread lightning and volcanic ignitions.

Wildfires can be characterized in terms of the cause of ignition, their physical properties, the combustible material present, and the effect of weather on the fire. Wildfires can cause damage to property and human life, although naturally occurring wildfires may have beneficial effects on native vegetation, animals, and ecosystems that have evolved with fire.

High-severity wildfire creates complex early seral forest habitat (also called "snag forest habitat"), which often has higher species richness and diversity than unburned old forest. Many plant species depend on the effects of fire for growth and reproduction.[11] Wildfires in ecosystems where wildfire is uncommon or where non-native vegetation has encroached may have strongly negative ecological effects.

Wildfire behavior and severity result from a combination of factors such as available fuels, physical setting, and weather. Analyses of historical meteorological data and national fire records in western North America show the primacy of climate in driving large regional fires via wet periods that create substantial fuels, or drought and warming that extend conducive fire weather.

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 threatning.

In this kernel, our aim is to predict the burned area (`area`) of forest fires, in the northeast region of Portugal. Based on the 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.

In [1]:
```python
target = 'area'
```

Literature Review

Source: https://archive.ics.uci.edu/ml/datasets/forest+fires

Citation Request: This dataset is public available for research. The details are described in [Cortez and Morais, 2007]. Please include this citation if you plan to use this database:

P. Cortez and A. Morais. A Data Mining Approach to Predict Forest Fires using Meteorological Data. In J. Neves, M. F. Santos and J. Machado Eds., New Trends in Artificial Intelligence, Proceedings of the 13th EPIA 2007 - Portuguese Conference on Artificial Intelligence, December, Guimaraes, Portugal, pp. 512-523, 2007. APPIA, ISBN-13 978-989-95618-0-9. Available at: http://www.dsi.uminho.pt/~pcortez/fires.pdf

1. Title: Forest Fires

2. Sources Created by: Paulo Cortez and An�bal Morais (Univ. Minho) @ 2007

3. Past Usage:

   P. Cortez and A. Morais. A Data Mining Approach to Predict Forest Fires using Meteorological Data. In Proceedings of the 13th EPIA 2007 - Portuguese Conference on Artificial Intelligence, December, 2007. (http://www.dsi.uminho.pt/~pcortez/fires.pdf)

   In the above reference, the output "area" was first transformed with a ln(x+1) function. Then, several Data Mining methods were applied. After fitting the models, the outputs were post-processed with the inverse of the ln(x+1) transform. Four different input setups were used. The experiments were conducted using a 10-fold (cross-validation) x 30 runs. Two regression metrics were measured: MAD and RMSE. A Gaussian support vector machine (SVM) fed with only 4 direct weather conditions (temp, RH, wind and rain) obtained the best MAD value: 12.71 +- 0.01 (mean and confidence interval within 95% using a t-student distribution). The best RMSE was attained by the naive mean predictor. An analysis to the regression error curve (REC) shows that the SVM model predicts more examples within a lower admitted error. In effect, the SVM model predicts better small fires, which are the majority.

4. Relevant Information:

   This is a very difficult regression task. It can be used to test regression methods. Also, it could be used to test outlier detection methods, since it is not clear how many outliers are there. Yet, the number of examples of fires with a large burned area is very small.

5. Number of Instances: 517

6. Number of Attributes: 12 + output attribute

   Note: several of the attributes may be correlated, thus it makes sense to apply some sort of feature selection.

7. Attribute information:

1. X - x-axis spatial coordinate within the Montesinho park map: 1 to 9
2. Y - y-axis spatial coordinate within the Montesinho park map: 2 to 9
3. month - month of the year: "jan" to "dec"
4. day - day of the week: "mon" to "sun"
5. FFMC - FFMC index from the FWI system: 18.7 to 96.20
6. DMC - DMC index from the FWI system: 1.1 to 291.3
7. DC - DC index from the FWI system: 7.9 to 860.6
8. ISI - ISI index from the FWI system: 0.0 to 56.10
9. temp - temperature in Celsius degrees: 2.2 to 33.30
10. RH - relative humidity in %: 15.0 to 100
11. wind - wind speed in km/h: 0.40 to 9.40
12. rain - outside rain in mm/m2 : 0.0 to 6.4
13. area - the burned area of the forest (in ha): 0.00 to 1090.84 (this output variable is very skewed towards 0.0, thus it may make sense to model with the logarithm transform).

8. Missing Attribute Values: None

Define the metrics

RMSE

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

Dependencies

In [2]:

```
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

In [3]:

```python
# path = 'forestfires.csv'
path = "../input/forest-fires-data-set/forestfires.csv"
df = pd.read_csv(path)

df.shape
```

Out[3]:

```
(517, 13)
```

In [4]:

```python
df.dtypes
```

Out[4]:

```
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
```

In [5]:

```python
df.describe().T
```

Out[5]:

	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

In [6]:
```
df.isna().sum().sum()
```

Out[6]:
```
0
```

Exploratory Data Analysis

We will try out the following analysis on our dataset

- Univariate
- Bivariate
- Multivariate

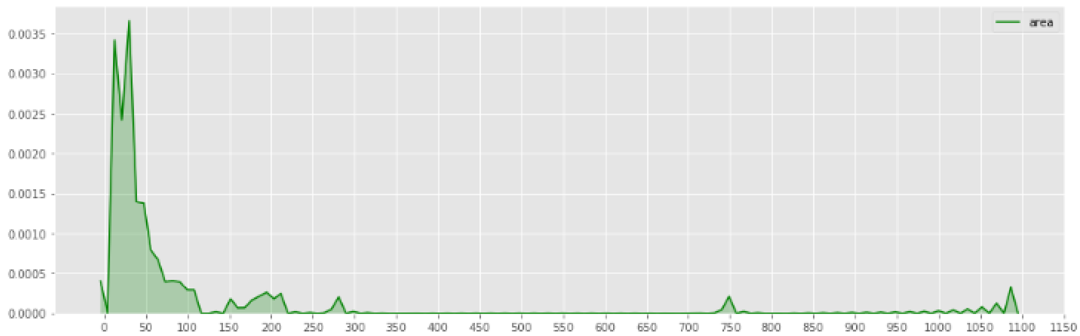
In [7]:
```
plt.rcParams["figure.figsize"] = 9,5
```

Univariate analysis

Let's begin with the target variable, `Area`

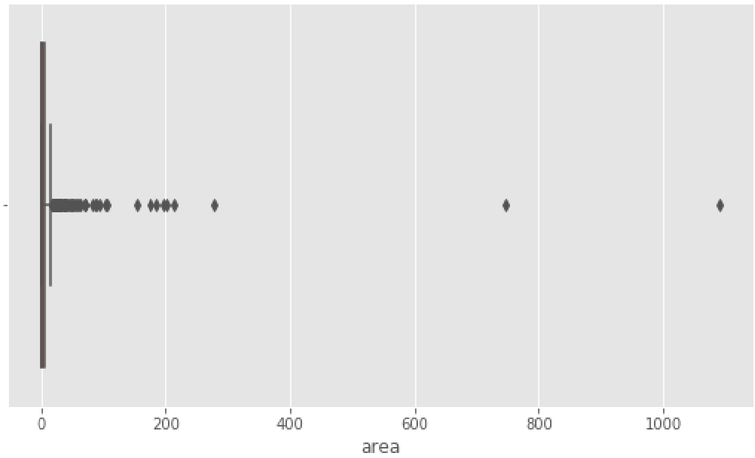
In [8]:
```
plt.figure(figsize=(16,5))
print("Skew: {}".format(df[target].skew()))
print("Kurtosis: {}".format(df[target].kurtosis()))
ax = sns.kdeplot(df[target],shade=True,color='g')
plt.xticks([i for i in range(0,1200,50)])
plt.show()
```

```
Skew: 12.846933533934868
Kurtosis: 194.1407210942299
```



In [9]:

```
ax = sns.boxplot(df[target])
```



Few observations:

- The data is highly skewed with a value of +12.84 and huge kurtosis value of 194.
- It even tells you that majority of the forest fires do not cover a large area, most of the damaged area is under 50 hectares of land.
- We can apply tranformation to fix the skewnesss and kurtosis, however we will have to inverse transform before submitting the output.
- Outlier Check: There are 4 outlier instances in our area columns but the questions is should we drop it or not? (Will get back to this in the outlier treatment step)

In [10]:

```
# Outlier points
y_outliers = df[abs(zscore(df[target])) >= 3 ]
y_outliers
```

Out[10]:

	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

Independent columns

In [11]:

```
dfa = df.drop(columns=target)
cat_columns = dfa.select_dtypes(include='object').columns.tolist()
num_columns = dfa.select_dtypes(exclude='object').columns.tolist()

cat_columns,num_columns
```

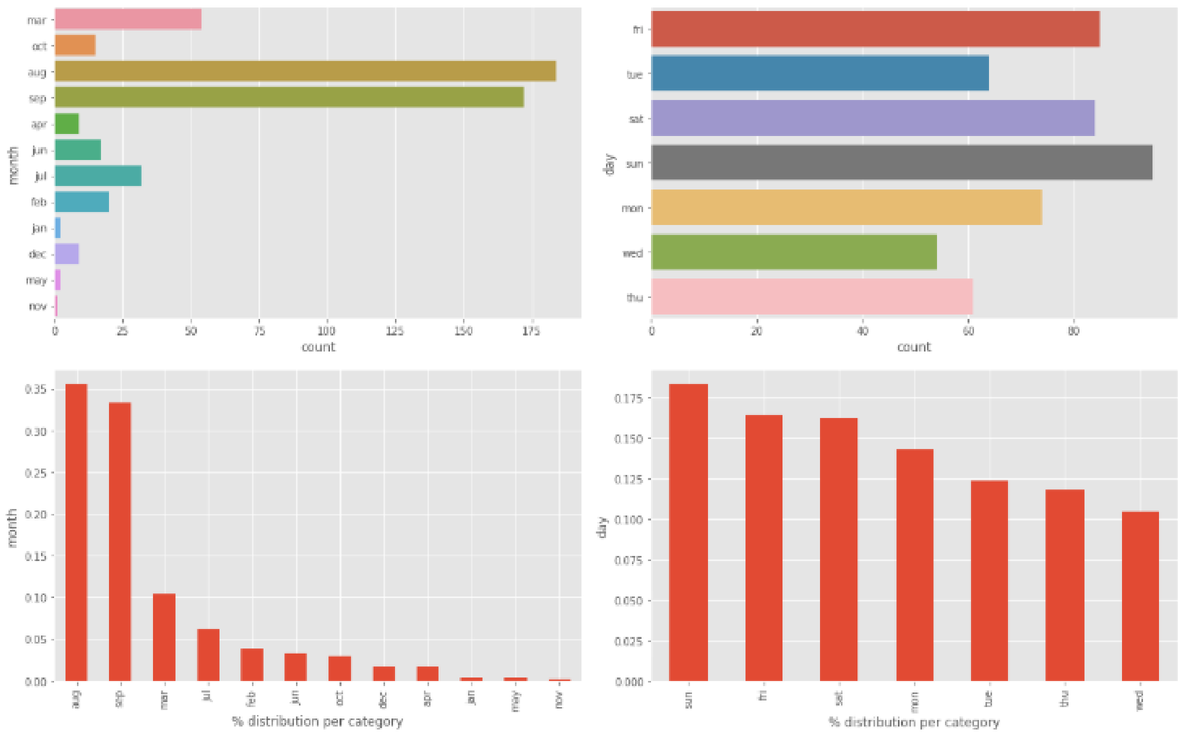
Out[11]:

```
(['month', 'day'],
 ['X', 'Y', 'FFMC', 'DMC', 'DC', 'ISI', 'temp', 'RH', 'wind', 'rain'])
```

Categorical columns

In [12]:

```
# 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()
```

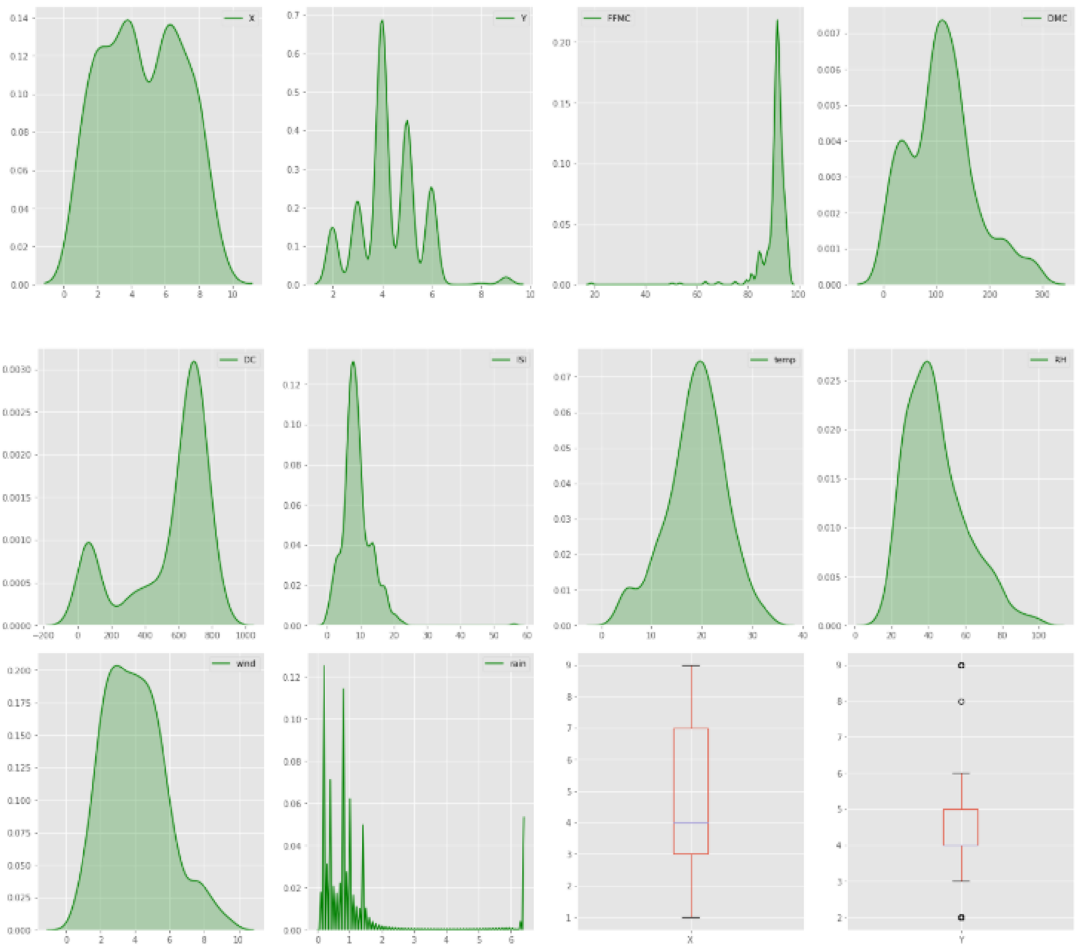


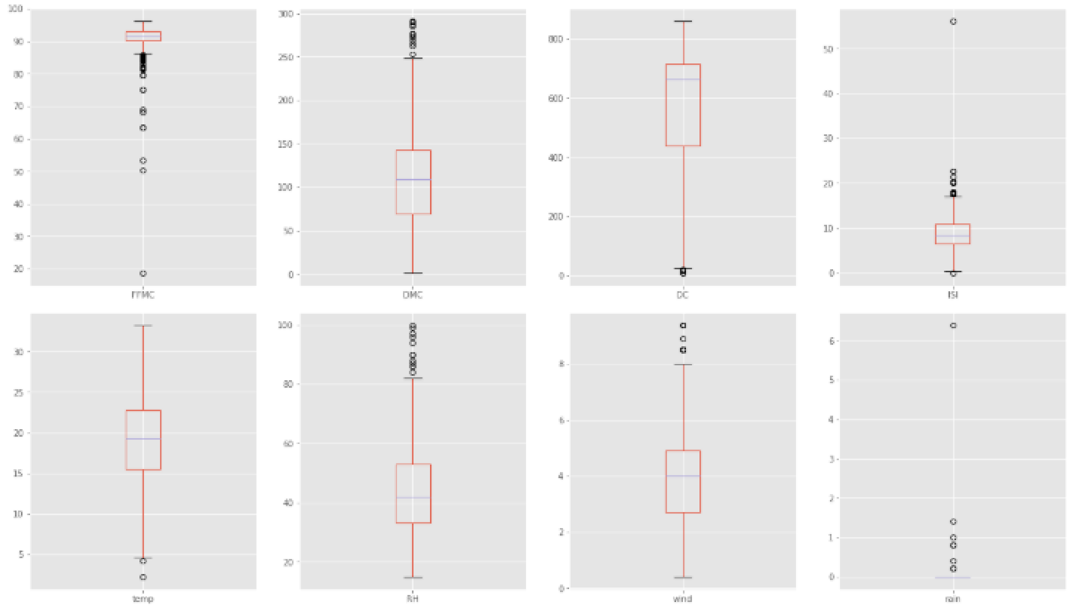
1. It is interesting to see that abnormally high number of the forest fires occur in the month of `August` and `September`.
2. In the case of day, the days `Friday` to `Monday` have higher proportion of cases. (However, no strong indicators)

Numerical Columns

In [13]:

```
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]
pd.DataFrame(data=[num_data.skew(),num_data.kurtosis()],index=['skewness','kurtosis'])
```





Out[13]:

	X	Y	FFMC	DMC	DC	ISI	temp	RH
skewness	0.036246	0.417296	-6.575606	0.547498	-1.100445	2.536325	-0.331172	0.862904
kurtosis	-1.172331	1.420553	67.066041	0.204822	-0.245244	21.458037	0.136166	0.438183

Outliers, Skewness and kurtosis (high positive or negative) was observed in the following columns:

1. FFMC
2. ISI
3. rain

Bivariate analysis with our target variable

In [14]:

```
print(df['area'].describe(),'\n')
print(y_outliers)
```

```
count     517.000000
mean       12.847292
std        63.655818
min         0.000000
25%         0.000000
50%         0.520000
75%         6.570000
max      1090.840000
Name: area, dtype: float64

     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
```

In [15]:

```
# a categorical variable based on forest fire area damage
# No damage, low, moderate, high, very high
def area_cat(area):
    if area == 0.0:
        return "No damage"
    elif area <= 1:
        return "low"
    elif area <= 25:
        return "moderate"
    elif area <= 100:
        return "high"
    else:
        return "very high"

df['damage_category'] = df['area'].apply(area_cat)
df.head()
```

Out[15]:

	X	Y	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area	damage_category
0	7	5	mar	fri	86.2	26.2	94.3	5.1	8.2	51	6.7	0.0	0.0	No damage
1	7	4	oct	tue	90.6	35.4	669.1	6.7	18.0	33	0.9	0.0	0.0	No damage
2	7	4	oct	sat	90.6	43.7	686.9	6.7	14.6	33	1.3	0.0	0.0	No damage
3	8	6	mar	fri	91.7	33.3	77.5	9.0	8.3	97	4.0	0.2	0.0	No damage
4	8	6	mar	sun	89.3	51.3	102.2	9.6	11.4	99	1.8	0.0	0.0	No damage

Categorical columns

In [16]:

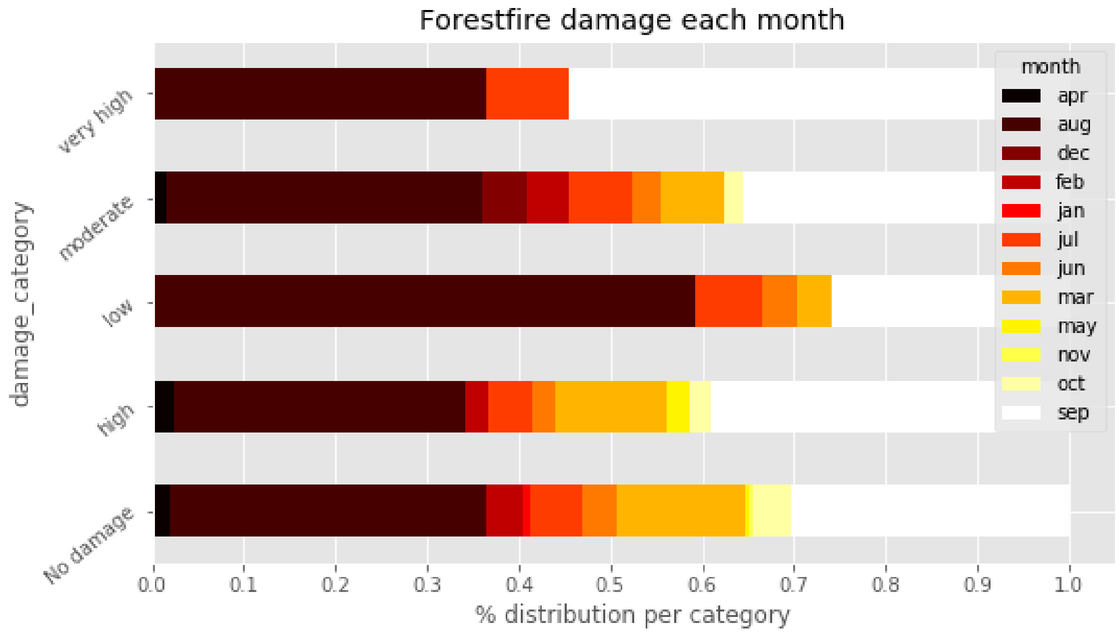
```
cat_columns
```

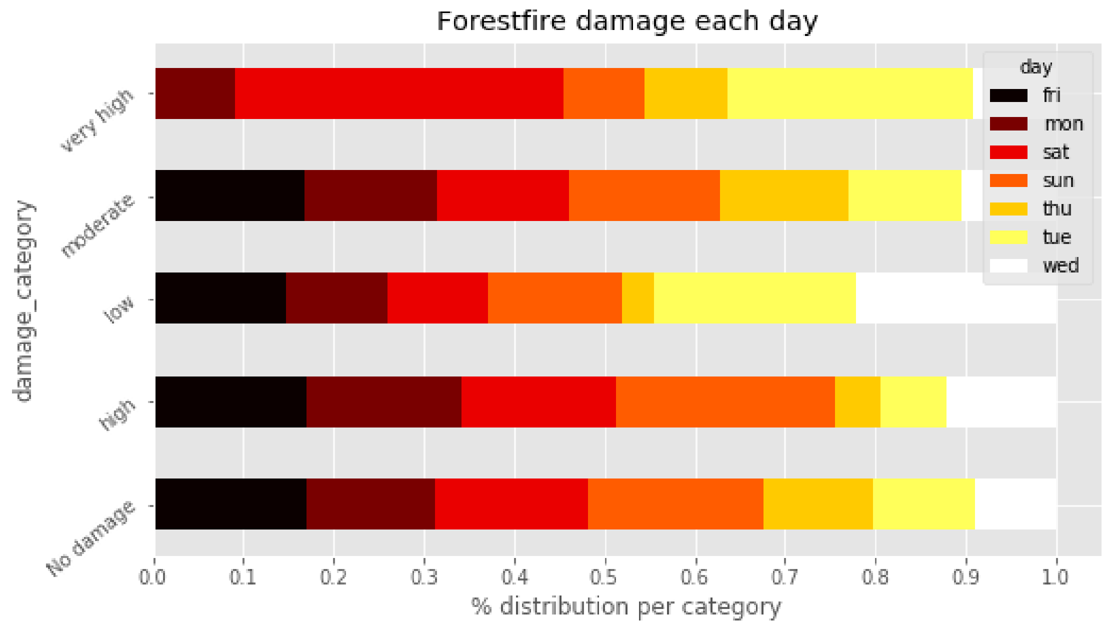
Out[16]:

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

In [17]:

```
for col in cat_columns:
    cross = pd.crosstab(index=df['damage_category'],columns=df[col],normalize='index')
    cross.plot.barh(stacked=True,rot=40,cmap='hot')
    plt.xlabel('% distribution per category')
    plt.xticks(np.arange(0,1.1,0.1))
    plt.title("Forestfire damage each {}".format(col))
plt.show()
```



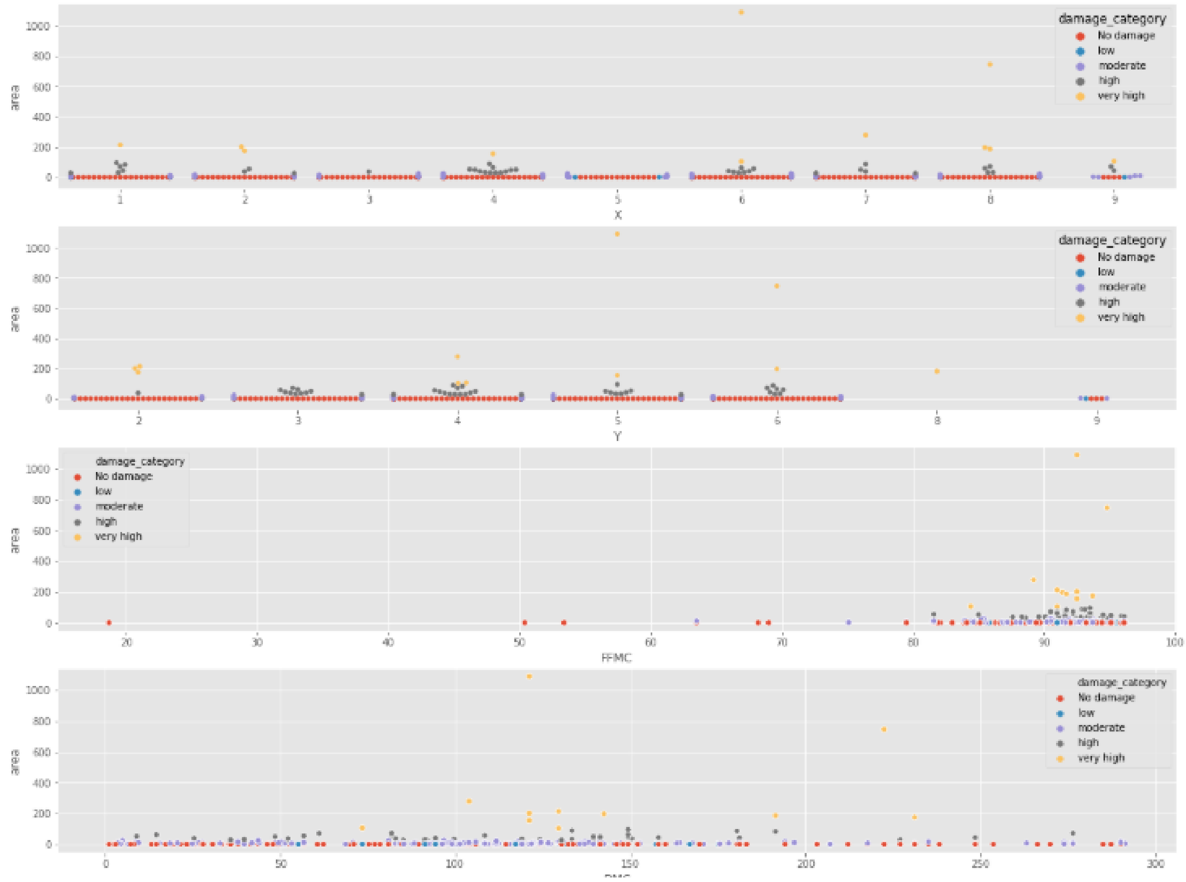


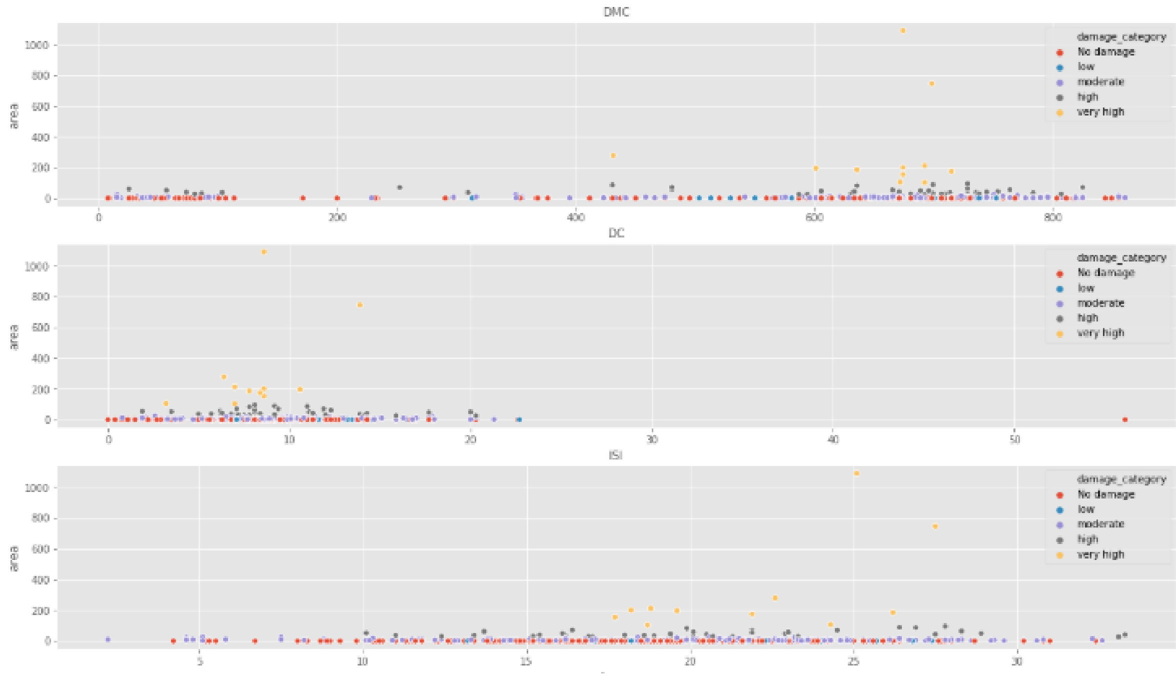
- Previously we had observed that `August` and `September` had the most number of forest fires. And from the above plot of `month`, we can understand few things
  - Most of the fires in August were low (< 1 hectare).
  - The very high damages(>100 hectares) happened in only 3 months - august,july and september.
- Regarding fire damage per day, nothing much can be observed. Except that, there were no `very high` damaging fires on Friday and on Saturdays it has been reported most.

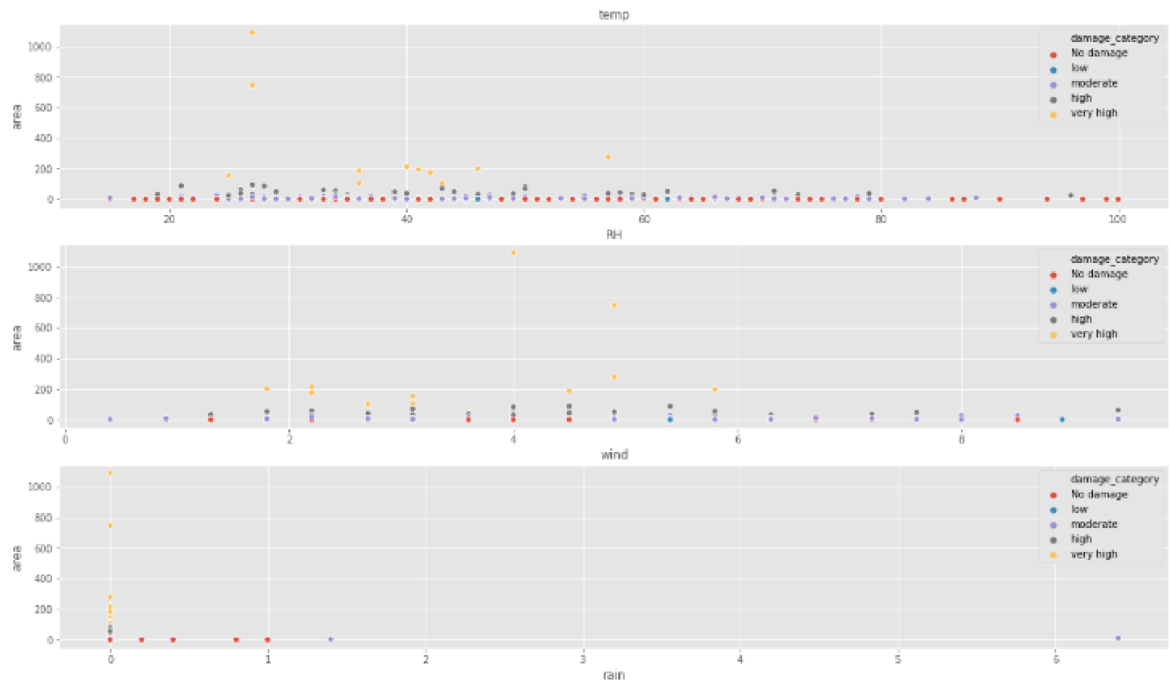
Numerical columns

In [18]:

```
plt.figure(figsize=(20,40))
for i,col in enumerate(num_columns,1):
    plt.subplot(10,1,i)
    if col in ['X','Y']:
        sns.swarmplot(data=df,x=col,y=target,hue='damage_category')
    else:
        sns.scatterplot(data=df,x=col,y=target,hue='damage_category')
plt.show()
```







Multivariate analysis

In [19]:

```
selected_features = df.drop(columns=['damage_category','day','month']).columns
selected_features
```

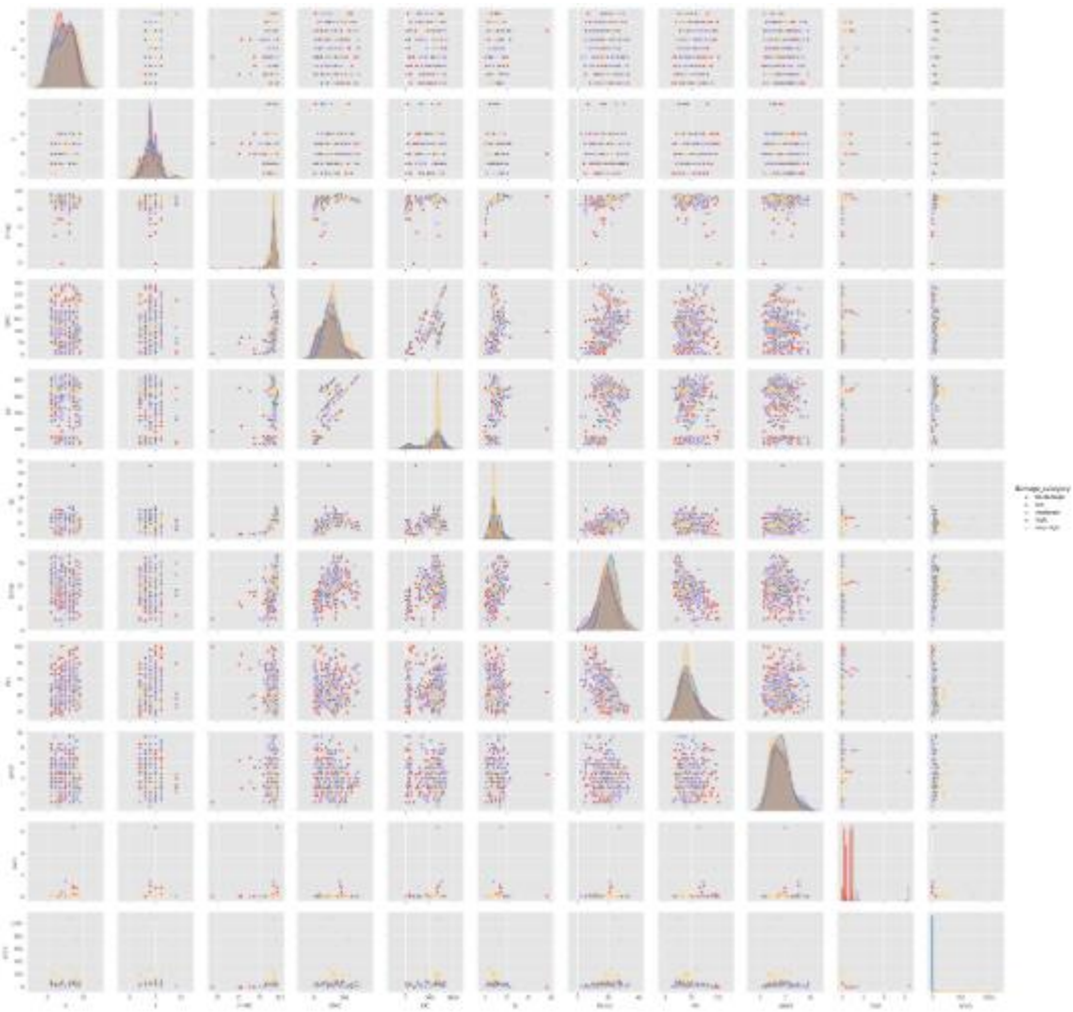
Out[19]:

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

In [20]:

```
sns.pairplot(df,hue='damage_category',vars=selected_features)
plt.show()
```

```
/opt/conda/lib/python3.6/site-packages/statsmodels/nonparametric/kde.py:487: Runtim
eWarning: invalid value encountered in true_divide
  binned = fast_linbin(X, a, b, gridsize) / (delta * nobs)
/opt/conda/lib/python3.6/site-packages/statsmodels/nonparametric/kdetools.py:34: Ru
ntimeWarning: invalid value encountered in double_scalars
  FAC1 = 2*(np.pi*bw/RANGE)**2
```



Outlier treatment

We had observed outliers in the following columns:

1. area
2. FFMC
3. ISI
4. rain

In [21]:
```python
out_columns = ['area','FFMC','ISI','rain']
```

However, the above outliers are not error values so we cannot remove it.

In order to minimize the effect of outliers in our model we will transform the above features.

Ref: https://humansofdata.atlan.com/2018/03/when-delete-outliers-dataset/

Preparing the data for modelling

Thing which we can cover here

- Encoding the categorical columns

In [22]:
```python
df = pd.get_dummies(df,columns=['day','month'],drop_first=True)
```

- Data transformations like `log,root,inverse,exponential`,etc

In [23]:
```python
print(df[out_columns].describe())
np.log1p(df[out_columns]).skew(), np.log1p(df[out_columns]).kurtosis()
```

```
              area        FFMC         ISI        rain
count   517.000000  517.000000  517.000000  517.000000
mean     12.847292   90.644681    9.021663    0.021663
std      63.655818    5.520111    4.559477    0.295959
min       0.000000   18.700000    0.000000    0.000000
25%       0.000000   90.200000    6.500000    0.000000
50%       0.520000   91.600000    8.400000    0.000000
75%       6.570000   92.900000   10.800000    0.000000
max    1090.840000   96.200000   56.100000    6.400000
```

Out[23]:
```
(area     1.217838
 FFMC   -11.675394
 ISI     -0.937218
 rain    14.173028
 dtype: float64, area      0.945668
 FFMC    185.482383
 ISI       2.584588
 rain    234.240025
 dtype: float64)
```

In [24]:

```
# FFMC and rain are still having high skew and kurtosis values,
# since we will be using Linear regression model we cannot operate with such high values
# so for FFMC we can remove the outliers in them using z-score method
mask = df.loc[:,['FFMC']].apply(zscore).abs() < 3

# Since most of the values in rain are 0.0, we can convert it as a categorical column
df['rain'] = df['rain'].apply(lambda x: int(x > 0.0))

df = df[mask.values]
df.shape
```

Out[24]:

```
(510, 29)
```

In [25]:

```
out_columns.remove('rain')
df[out_columns] = np.log1p(df[out_columns])
```

In [26]:

```
df[out_columns].skew()
```

Out[26]:

```
area     1.208492
FFMC    -1.803993
ISI     -0.434372
dtype: float64
```

In [27]:

```
# we will use this dataframe for building our ML model
df_ml = df.drop(columns=['damage_category']).copy()
```

Linear Regression

Difference between statistical and machine learning approach

- Machine learning produces **predictions**. As far as I can tell, it is not very good at drawing conclusions about general principles based on a set of observations.
- Statistical estimation lets the practitioner make **inferences** (conclusions about a larger set of phenomena based on the observation of a smaller set of phenomena.) For example, in a regression model the practitioner can estimate the effect of a one unit change in an independent variable X on a dependent variable y.

Statistical approach

Checking assumptions for linear regression in statistics

1. Linearity of model
2. Normality of residuals
3. Homoscedasticity
4. No Autocorrelation
5. Multicollinearity

In [29]:

```
X_constant = sm.add_constant(X)

# Build OLS model
lin_reg = sm.OLS(y,X_constant).fit()
lin_reg.summary()
```

```
/opt/conda/lib/python3.6/site-packages/numpy/core/fromnumeric.py:2389: FutureWarnin
g: Method .ptp is deprecated and will be removed in a future version. Use numpy.ptp
instead.
  return ptp(axis=axis, out=out, **kwargs)
```

Out[29]:

OLS Regression Results

Dep. Variable:	area	R-squared:	0.077
Model:	OLS	Adj. R-squared:	0.025
Method:	Least Squares	F-statistic:	1.489
Date:	Sun, 01 Dec 2019	Prob (F-statistic):	0.0558
Time:	18:36:15	Log-Likelihood:	-874.85
No. Observations:	510	AIC:	1806.
Df Residuals:	482	BIC:	1924.
Df Model:	27		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.3192	18.275	0.017	0.986	-35.590	36.228
X	0.0532	0.033	1.621	0.106	-0.011	0.118
Y	-0.0115	0.061	-0.187	0.852	-0.132	0.109
FFMC	-0.1061	4.160	-0.025	0.980	-8.280	8.068
DMC	0.0041	0.002	2.166	0.031	0.000	0.008
DC	-0.0019	0.001	-1.440	0.150	-0.004	0.001
ISI	-0.1039	0.290	-0.358	0.720	-0.674	0.466
temp	0.0443	0.023	1.964	0.050	-2.77e-05	0.089
RH	0.0041	0.007	0.624	0.533	-0.009	0.017
wind	0.0678	0.039	1.719	0.086	-0.010	0.145
rain	-0.9272	0.547	-1.695	0.091	-2.002	0.148
day_mon	0.1076	0.230	0.467	0.641	-0.345	0.560
day_sat	0.3312	0.222	1.493	0.136	-0.105	0.767
day_sun	0.1794	0.215	0.836	0.403	-0.242	0.601
day_thu	0.0714	0.243	0.294	0.769	-0.406	0.549
day_tue	0.3483	0.238	1.464	0.144	-0.119	0.816
day_wed	0.1960	0.248	0.791	0.429	-0.291	0.683
month_aug	0.2450	0.847	0.289	0.772	-1.419	1.909
month_dec	2.2223	0.804	2.764	0.006	0.643	3.802
month_feb	0.2217	0.568	0.391	0.696	-0.894	1.337
month_jan	-0.8605	1.483	-0.580	0.562	-3.775	2.054
month_jul	0.0319	0.731	0.044	0.965	-1.405	1.469
month_jun	-0.3316	0.679	-0.488	0.625	-1.666	1.002
month_mar	-0.2613	0.519	-0.503	0.615	-1.282	0.759
month_may	0.6256	1.106	0.565	0.572	-1.548	2.800
month_nov	-1.1779	1.490	-0.790	0.430	-4.106	1.750
month_oct	0.7718	1.005	0.768	0.443	-1.203	2.747
month_sep	0.8978	0.949	0.946	0.345	-0.967	2.762

Omnibus:	76.076	Durbin-Watson:	0.979
Prob(Omnibus):	0.000	Jarque-Bera (JB):	107.018
Skew:	1.074	Prob(JB):	5.77e-24
Kurtosis:	3.652	Cond. No.	1.89e+05

Warnings:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.89e+05. This might indicate that there are
strong multicollinearity or other numerical problems.

1. Linearity of residuals

Linearity can be measured by two methods:

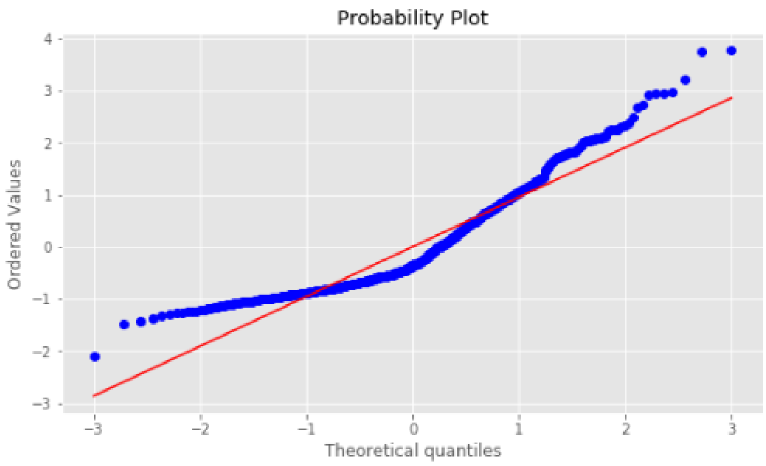
- Plot the observed values Vs predicted values`and plot the`Residual Vs predicted values` and see the linearity of residuals.
- Rainbow test

Rainbow test

In [30]:

```
import scipy.stats as stats
import pylab

# get an instance of Influence with influence and outlier measures
st_resid = lin_reg.get_influence().resid_studentized_internal
stats.probplot(st_resid,dist="norm",plot=pylab)
plt.show()
```



- **Null hypothesis (H0):** The Null hypothesis is that the regression is correctly modeled as linear.
- **Alternate hypothesis(H1)**: The model is non-linear

In [31]:

```
# return fstat and p-value
sm.stats.diagnostic.linear_rainbow(lin_reg)
```

Out[31]:

```
(1.283265916165085, 0.027048891500426765)
```

Expectation Mean of residual is zero

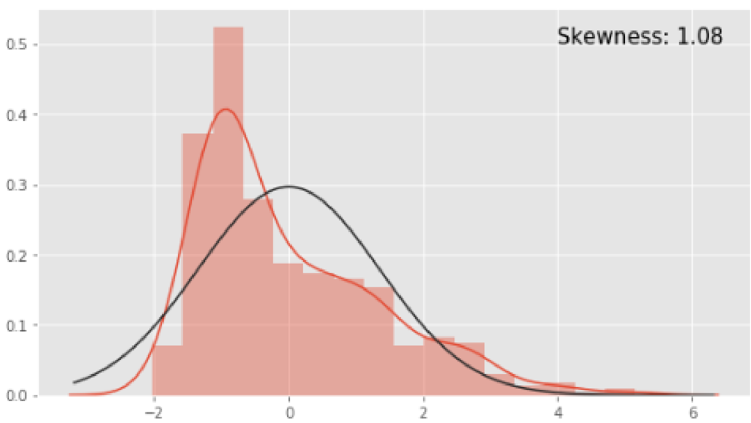
In [32]:
```
# The mean expected value around 0, it implies linearity is preserved
lin_reg.resid.mean()
```
Out[32]:
```
-8.181255237541692e-15
```

The desired outcome of plots is that points are symmetrically distributed around a diagonal line in the former plot or around horizontal line in the latter one.

- By observing the plots the linearity assumption is not there
- Adding new features might result in linearity of model
- Also, transforming the feature from non-linear to linear using various data transformation techniques can help.

2. Normality of the residuals

In [34]:
```
sns.distplot(lin_reg.resid,fit=stats.norm)
plt.text(4,0.5,f"Skewness: {round(lin_reg.resid.skew(),2)}",fontsize=15)
plt.show()
```



Test for normality: Jarque Bera

For a good model, the residuals should be normally distributed. The higher the value of Jarque Bera test, the lesser the residuals are normally distributed.

The Jarque–Bera test is a goodness-of-fit test of whether sample data have the skewness and kurtosis matching a normal distribution.

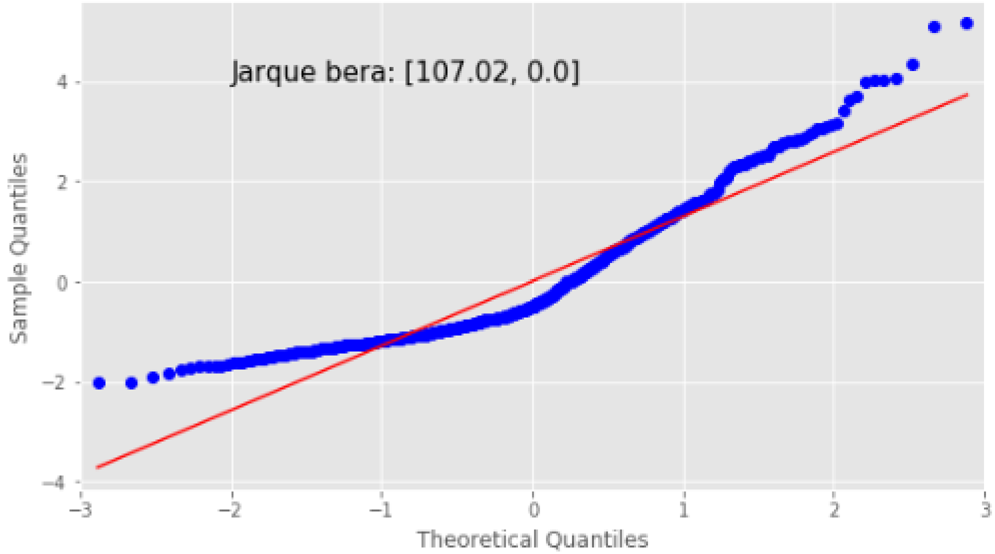
Jarque-Bera (JB): 107.018

The jarque bera test tests whether the sample data has the skewness and kurtosis matching a normal distribution.

Note that this test generally works good for large enough number of data samples(>2000) as the test statistics asymptotically has a chi squared distribution with degrees 2 of freedom.

Our dataframe length, 517

Null hypothesis (H0) - Residuals are normally distributed



The p-value is 0 which simply means we can reject out NULL hypothesis. We can fix that by

- Removing the outliers in the data
- Fixing the Non-linearity in our dependent or target feature
- Removing the bias, the bias might be contributing to the non-normality.

Homoscedasticity

Homoscedacity: If the residuals are symmetrically distributed across the trend , then it is called as homoscedacious.

Heteroscedacity: If the residuals are not symmetric across the trend, then it is called as heteroscedacious.

Goldfeld-Quandt test for Homoscedasticity

H0 = constant variance among residuals (Homoscedacity)

Ha = Heteroscedacity.

```
In [36]: sms.het_goldfeldquandt(lin_reg.resid, lin_reg.model.exog)

Out[36]: (0.900533026862814, 0.7860123901512498, 'increasing')
```

- To identify homoscedasticity in the plots, the placement of the points should be equally distributed, random, no pattern (increase/decrease in values of residuals) should be visible and a flat red line.
- In the plots we can see there are no paticular patterns and P-Values is also greater than 0.05 ,so we can say that there is homoscedasticity.
- Outliers can make it Heteroscedacious, Transforming (log or Box cox, if > 0) the dependent or independent variables can help fix it.

4. No Autocorrelation

Autocorrelation measures the relationship between a variable's current value and its past values.

Test for autocorrelation : Durbin- Watson Test

It's test statistic value ranges from 0-4. If the value is between

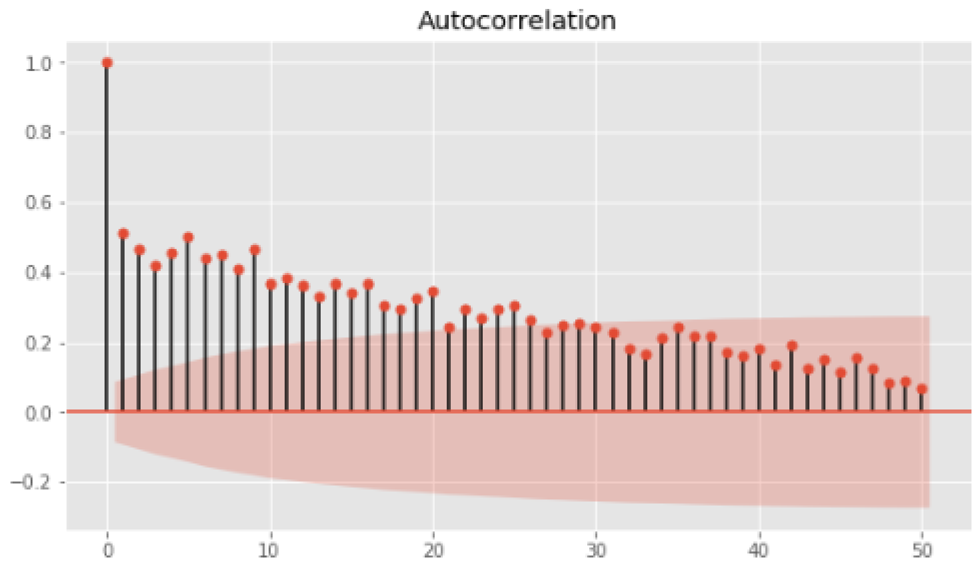
- 0-2, it's known as Positive Autocorrelation.
- 2-4, it is known as Negative autocorrelation.
- exactly 2, it means No Autocorrelation.

For a good linear model, it should have low or no autocorrelation.

```
from statsmodels.stats.stattools import durbin_watson

durbin_watson(lin_reg.resid)
```

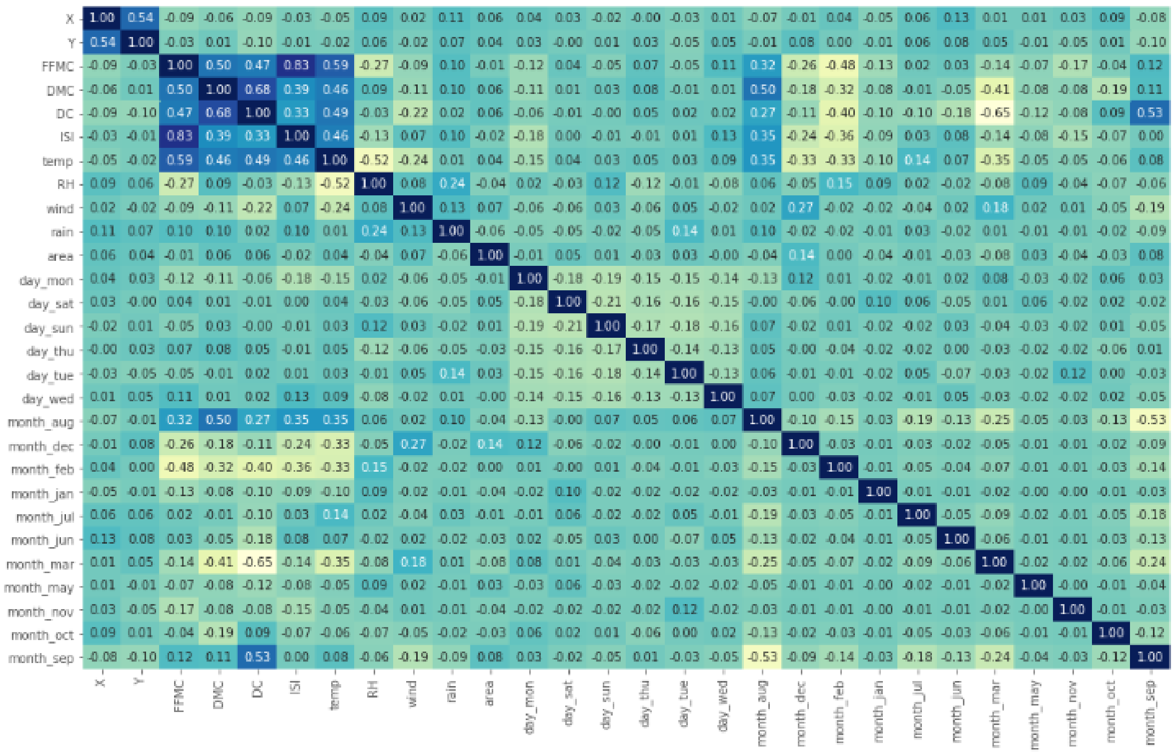
In our case, Durbin-Watson: 0.979



- By observing the above data we can say that there is positive autocorrelation is present , we can reduce it by using fine tuning our parameters
- We can even use Generalize Least Squares (GLS) model

5. Multicollinearity

Multicollineariy arises when one independent variable can be linearly predicted by others with a substantial level of accuracy.



In [40]:

```
from statsmodels.stats.outliers_influence import variance_inflation_factor

vif = [variance_inflation_factor(X_constant.values, i) for i in range(X_constant.shape[1])]
pd.DataFrame({'vif': vif[1:]}, index=X.columns).sort_values(by="vif",ascending=False)
```

Out[40]:

	vif
month_sep	53.307716
month_aug	43.939403
DC	26.792896
month_jul	8.378570
month_oct	7.681340
month_mar	6.694845
FFMC	5.629386
temp	4.535721
ISI	4.107793
DMC	3.978913
month_jun	3.731938
month_feb	3.077978
month_dec	2.984761
RH	2.870672
day_sun	1.829250
day_sat	1.750570
day_mon	1.733149
day_thu	1.654243
day_tue	1.653962
day_wed	1.547815
X	1.539358
Y	1.521475
wind	1.323007
month_may	1.273889
rain	1.231386
month_nov	1.157750
month_jan	1.146912

There is multicollinearity present between some features where vif >5.

- We can even use PCA to reduce features to a smaller set of uncorrelated components.
- To deal with multicollinearity we should iteratively remove features with high values of VIF.

Improving Stats model

Dropping columns to improve accuracy:

By checking high Variance inflation factor and p-value we will decide whether to keep the column or drop it.

R^2 = 1 - SSE(Sum of Square of Residuals)/SST (Sum of square Total)
Just by dropping constant we got a huge bump in adjusted R2 from `2.5%` to `40.6%`.

In [42]:
```
X = df.drop(columns=['area','damage_category'])
y = df['area']
```

In [43]:
```
def check_stats(X,y):
    vif = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
    print(pd.DataFrame({'vif': vif}, index=X.columns).sort_values(by="vif",ascending=False)[:1
0])
    lin_reg = sm.OLS(y,X).fit()
    print(lin_reg.summary())
check_stats(X,y)
```

In [44]:
```
X.drop(columns=['FFMC'],inplace=True)
# check_stats(X,y)
```

In [45]:
```
X.drop(columns=['Y'],inplace=True)
# check_stats(X,y)
```

In [46]:
```
X.drop(columns=['month_jul'],inplace=True)
# check_stats(X,y)
```

In [47]:
```
X.drop(columns=['day_thu'],inplace=True)
# check_stats(X,y)
```

In [48]:
```
X.drop(columns=['day_mon'],inplace=True)
# check_stats(X,y)
```

In [49]:
```
X.drop(columns=['month_aug'],inplace=True)
check_stats(X,y)
```

Similarly, we can continue to optimize the model.

Our Prob (F-statistic) has improved from 0.0558 to 2.20e-48. As the value is less than 0.05, the model becomes more significant.

Conclusion

- The data is highly skewed with a value of +12.84 and huge kurtosis value of 194.
- It even tells you that majority of the forest fires do not cover a large area, most of the damaged area is under 50 hectares of land.
- We can apply tranformation to fix the skewnesss and kurtosis, however we will have to inverse transform before submitting the output.
- We could have done Linear regression using both Statistical and Machine learning approach, but we chose statistical approach because of the following differences:

  ✓ Machine learning produces predictions. It is not very good at drawing conclusions about general principles based on a set of observations.

  ✓ Statistical estimation lets the practitioner make inferences (conclusions about a larger set of phenomena based on the observation of a smaller set of phenomena.) For example, in a regression model the practitioner can estimate the effect of a one unit change in an independent variable X on a dependent variable y.

Bibliography and references:

1. https://archive.ics.uci.edu/ml/datasets/forest+fires
2. https://en.wikipedia.org/wiki/Wildfire
3. https://www.kaggle.com/elikplim/forest-fires-data-set/version/1
4. https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/
5. http://www3.dsi.uminho.pt/pcortez/fires.pdf
6. https://humansofdata.atlan.com/2018/03/when-delete-outliers-dataset/
7. https://www.quora.com/When-do-you-use-machine-learning-vs-statistical-regression
8. https://datascienceplus.com/how-to-detect-heteroscedasticity-and-rectify-it/