

1. Import necessary libraries

```
In [13]: import numpy as np
import pandas as pd
from sklearn import preprocessing
from sklearn import metrics
import seaborn as sns
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from matplotlib import pyplot as plt
from sklearn.decomposition import PCA
from mlxtend.plotting import plot_decision_regions

import warnings
warnings.filterwarnings('ignore')
```

Problem

Classify the SizeCategorie using SVM

2. Import data

```
In [3]: fire_data = pd.read_csv('forestfires.csv')
fire_data
```

```
Out[3]:
```

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

517 rows × 31 columns

Data understanding

```
In [4]: fire_data.shape
```

```
Out[4]: (517, 31)
```

```
In [5]: fire_data.dtypes
```

```
Out[5]: month          object
day          object
FFMC         float64
DMC          float64
DC           float64
ISI          float64
temp         float64
RH           int64
wind         float64
rain         float64
area         float64
dayfri       int64
daymon       int64
daysat      int64
daysun      int64
daythu      int64
daytue      int64
daywed      int64
monthapr     int64
monthaug    int64
monthdec     int64
monthfeb     int64
monthjan     int64
monthjul     int64
monthjun     int64
monthmar     int64
monthmay     int64
monthnov     int64
monthoct     int64
monthsep     int64
size_category object
dtype: object
```

```
In [7]: fire_data[fire_data.columns[0:11]].describe().T
```

```
Out[7]:
```

	count	mean	std	min	25%	50%	75%	max
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

	count	mean	std	min	25%	50%	75%	max
--	-------	------	-----	-----	-----	-----	-----	-----

```
In [8]: fire_data[fire_data.columns[0:11]].isnull().sum()
```

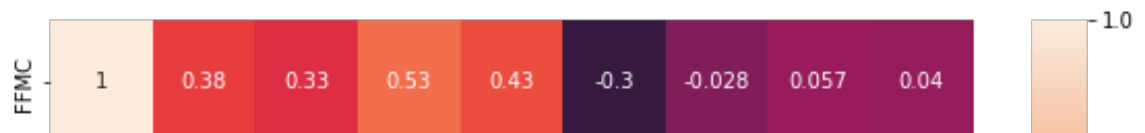
```
Out[8]: month      0
        day        0
        FFMCM      0
        DMC        0
        DC         0
        ISI        0
        temp       0
        RH         0
        wind       0
        rain       0
        area       0
        dtype: int64
```

Finding Correlation

```
In [9]: corr = fire_data[fire_data.columns[0:11]].corr()
```

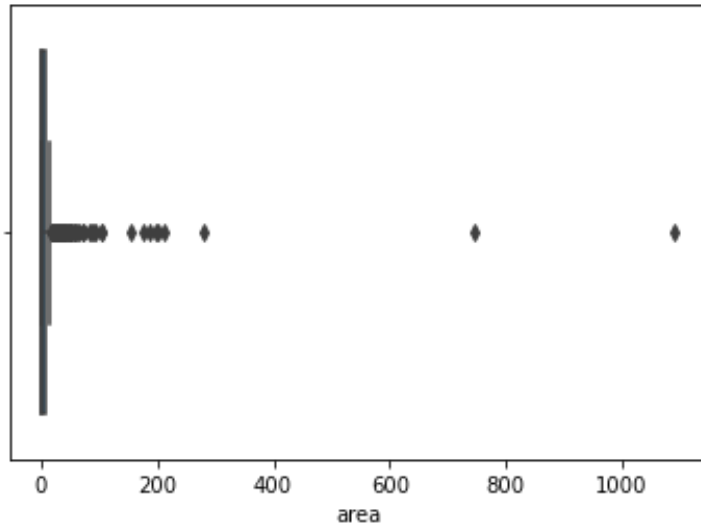
```
In [10]: plt.figure(figsize=(10,10))
         sns.heatmap(corr,annot=True)
```

```
Out[10]: <AxesSubplot:>
```



Outlier Check

```
In [14]: ax = sns.boxplot(fire_data['area'])
```



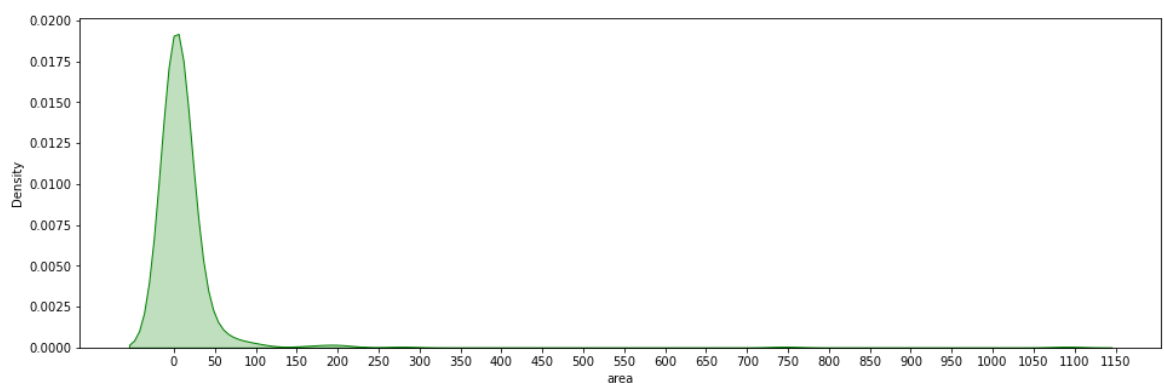
There are 3 Outlier instances in our data

```
In [15]: plt.rcParams["figure.figsize"] = 9,5
```

```
In [17]: plt.figure(figsize=(16,5))
print("Skew: {}".format(fire_data['area'].skew()))
print("Kurtosis: {}".format(fire_data['area'].kurtosis()))
ax = sns.kdeplot(fire_data['area'],shade=True,color='g')
plt.xticks([i for i in range(0,1200,50)])
plt.show()
```

Skew: 12.846933533934868

Kurtosis: 194.1407210942299

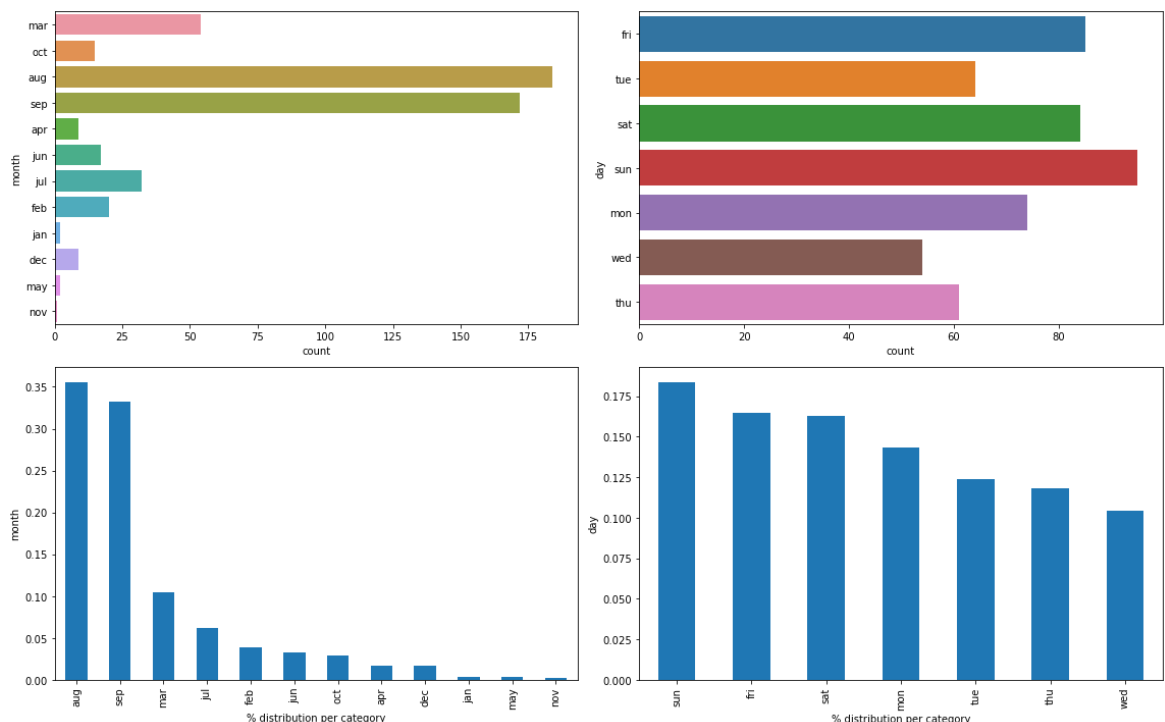


The Data is highly skewed and has large kurtosis value

Majority of the forest fires do not cover a large area, most of the damaged area is under 100 hectares of land

```
In [18]: dfa = fire_data[fire_data.columns[0:10]]
month_column = dfa.select_dtypes(include='object').columns.tolist()
```

```
In [19]: plt.figure(figsize=(16,10))
for i,col in enumerate(month_column,1):
    plt.subplot(2,2,i)
    sns.countplot(data=dfa,y=col)
    plt.subplot(2,2,i+2)
    fire_data[col].value_counts(normalize=True).plot.bar()
    plt.ylabel(col)
    plt.xlabel('% distribution per category')
plt.tight_layout()
plt.show()
```

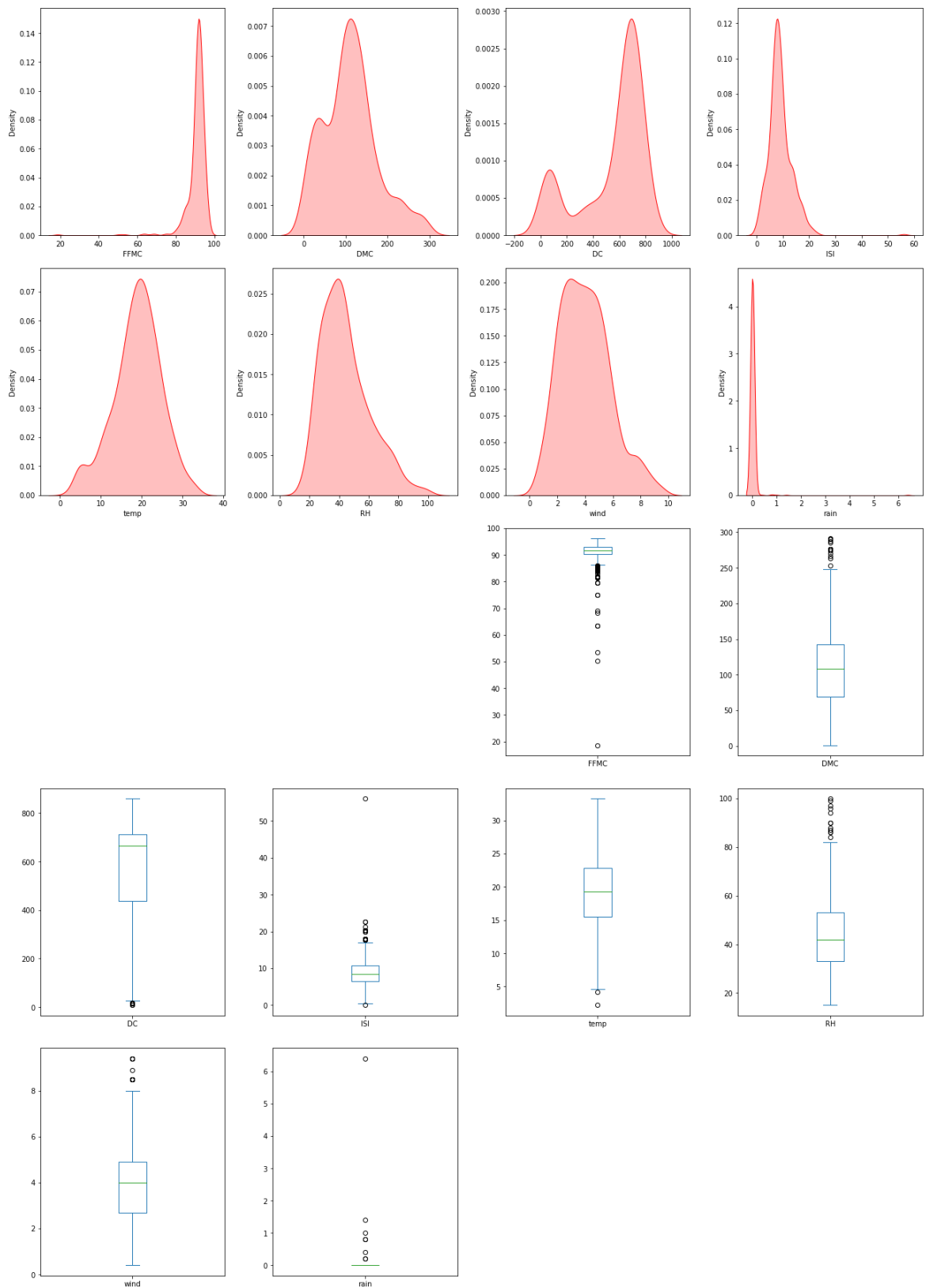


Majority of the fire accors in the month Aug and Sep

For Days Sun and Fri have recorded the most cases

```
In [20]: num_columns = dfa.select_dtypes(exclude='object').columns.tolist()
```

```
In [21]: plt.figure(figsize=(18,40))
for i,col in enumerate(num_columns,1):
    plt.subplot(8,4,i)
    sns.kdeplot(fire_data[col],color='r',shade=True)
    plt.subplot(8,4,i+10)
    fire_data[col].plot.box()
plt.tight_layout()
plt.show()
num_data = fire_data[num_columns]
pd.DataFrame(data=[num_data.skew(),num_data.kurtosis()],index=['skewness','ku
```



Out[21]:

	FFMC	DMC	DC	ISI	temp	RH	wind	ra
skewness	-6.575606	0.547498	-1.100445	2.536325	-0.331172	0.862904	0.571001	19.8163
kurtosis	67.066041	0.204822	-0.245244	21.458037	0.136166	0.438183	0.054324	421.2959

3. SVM

```
In [23]: x = fire_data.iloc[:,2:30]
        y = fire_data.iloc[:,30]
```

```
In [24]: mapping = {'small': 1, 'large': 2}
```

```
In [25]: y = y.replace(mapping)
```

```
In [26]: x_train,x_test,y_train,y_test = train_test_split(X,y,test_size = 0.20, strati
```

3.1 Linear

```
In [27]: model_linear = SVC(kernel = "linear")
        model_linear.fit(x_train,y_train)
        pred_test_linear = model_linear.predict(x_test)
        print("Accuracy:",metrics.accuracy_score(y_test, pred_test_linear))
```

Accuracy: 0.9807692307692307

3.2 Poly

```
In [28]: model_poly = SVC(kernel = "poly")
        model_poly.fit(x_train,y_train)
        pred_test_poly = model_poly.predict(x_test)
        print("Accuracy:",metrics.accuracy_score(y_test, pred_test_poly))
```

Accuracy: 0.7403846153846154

3.3 RBF

```
In [29]: model_rbf = SVC(kernel = "rbf")
        model_rbf.fit(x_train,y_train)
        pred_test_rbf = model_rbf.predict(x_test)
        print("Accuracy:",metrics.accuracy_score(y_test, pred_test_rbf))
```

Accuracy: 0.7403846153846154

3.4 Sigmoid

```
In [30]: model_sigmoid = SVC(kernel = "sigmoid")
        model_sigmoid.fit(x_train,y_train)
        pred_test_sigmoid = model_sigmoid.predict(x_test)
        print("Accuracy:",metrics.accuracy_score(y_test, pred_test_sigmoid))
```

Accuracy: 0.6346153846153846

4 - Conclusion

Linear Model gives the best accuracy

Below is an example on how we can plot the data. I used PCA to select only 2 variables

```
In [31]: yt = y_train.to_numpy()
```

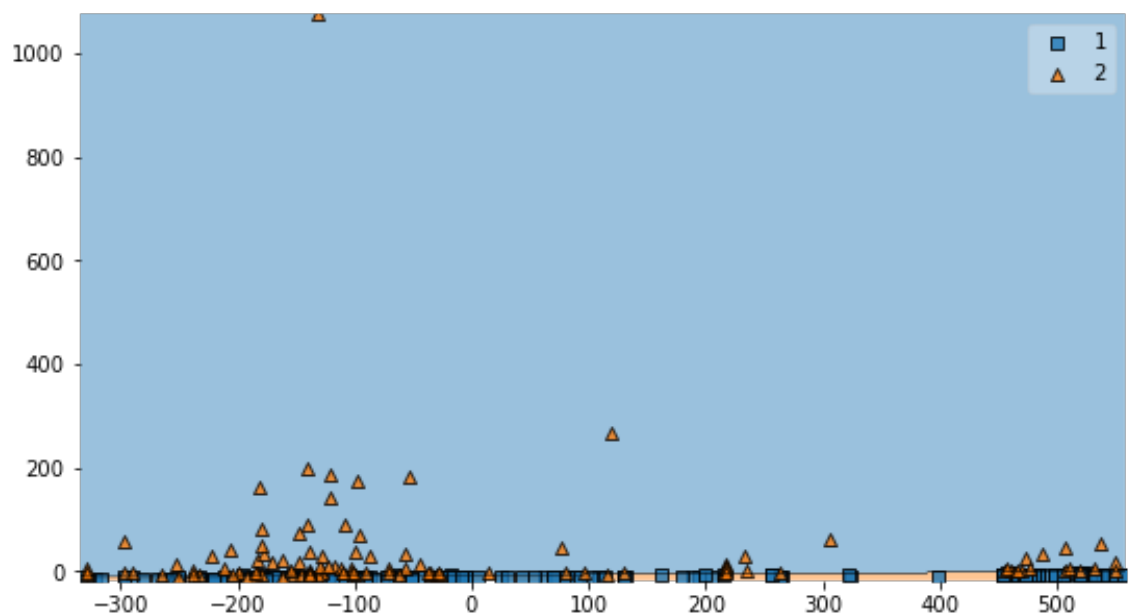
```
In [32]: pca = PCA(n_components = 2)
```

```
In [33]: x_train2 = pca.fit_transform(x_train)
```

```
In [35]: model_linear.fit(x_train2,yt)
```

```
Out[35]: SVC(kernel='linear')
```

```
In [39]: plot_decision_regions(x_train2,yt, clf=model_linear)  
plt.show()
```



```
In [ ]:
```