Visual Analysis of Forest fires in Portugal using Machine Learning

Karanam Santoshachandra Rao
Asst. Professor, Dept of CSE
Centurion University of Technology and Management, Odisa, India.

Avinash Alugolu
Asst. Professor, Dept of CSE
Centurion University of Technology and Management, Andhra Pradesh, India.

Alibilli Mahesh
Centurion University of Technology and Management, Odisa, India.

Ch.Rani
Centurion University of Technology and Management, Odisa, India.

Sai Kotturu
Centurion University of Technology and Management, Odisa, India.

Deverchetty Sriya
Centurion University of Technology and Management, Odisa, India.

Abstract

Forest fires in Portugal the main aim of the problem was to predict the burned area of a forest in Portugal according to different parameters like Temperature, Humidity and also by considering standard codes like DMC(Duff Moisture Code) and FFMC(Fine Fuel Moisture Code). using machine learning visualization techniques based on some constrains like:1) Mean values of temperature, wind and rain based on months,2) Features of forest on april month,3)Shows temperature in that specified area,4) count of FFMC,5)Shows the mean values of months and dc,6) Shows the mean values of day and dc,7)Finds the relationship between the features of dataset,8)Average of wind with respect to month;9)Day vs temperature violin plot,10)Month wise rain predict,11)Day vs tempetarure predict.

Keywords: Forestfires, Machine Learning, Visualization, Plotting, Python, Packages, Numpy, Pandas, Seaborn, Matplotlib , Plotly.

Introduction

Analysis of mentioned 11 points are based on python best and top libraries like numpy, pandas and visualization libraries like matplotlib, plotll. Generally, Python is high-level general purpose programming and an interpreted language. Developed by Guido van Rossum and primarily released in 1991, Python's design intention emphasizes code readability & understanding with its easy use of significant whitespace or indentation. Its object-oriented approach targets to help developers write unambiguous, very logical code for tiny and large-scale projects.

"Visualization is worth a many thousand words". We are all aware of this expression. This especially applies when we trying to deliver the insight received from the analysis of large datasets. Data visualization plays a major role in the visualization of both tiny and very large-scale data.

One of the important skills of a data scientist or programmer is the ability to express a compelling story, displaying data visually and findings in a possible and stimulating way. Studying how to use a software tool to visualize data-sets will also permits you to fetch information, clear understand the data, and make more perfect decisions.

The main motto of this Data Visualization with Python is to analyze above mentioned 11 points and present that information in the form that makes sense to people. Various techniques have been used for presenting data visually with the help of python libraries namely Matplotlib, Seaborn, and Plotly.

Following python library code is required to import primarily to analyze the data with the help of various function and methods.

```
import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

import plotly.express as px
```

Understanding the dataset

Here, we have considered the data from the dataset "Forest_fires.csv" for analysis. Basic idea on dataset is given by following python code.

Step1:Reads the Forest_fires.csv file

```
data = pd.read_csv('forestfires.csv')

print(data.shape)

data.head()
```

The above code gives the first 5 rows from the dataset.

	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

Fig 1: Dataset shows the first 5 rows

Fig 1 shows the output of the following python code for dataset summarizing the columns.

Code: data.describe()

	X	Y	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area
count	517.000000	517.000000	517.000000	517.000000	517.000000	517.000000	517.000000	517.000000	517.000000	517.000000	517.000000
mean	4.669246	4.299807	90.644681	110.872340	547.940039	9.021663	18.889168	44.288201	4.017602	0.021663	12.862282
std	2.313778	1.229900	5.520111	64.046482	248.066192	4.559477	5.806625	16.317469	1.791653	0.295959	63.653355
min	1.000000	2.000000	18.700000	1.100000	7.900000	0.000000	2.200000	15.000000	0.400000	0.000000	0.000000
25%	3.000000	4.000000	90.200000	68.600000	437.700000	6.500000	15.500000	33.000000	2.700000	0.000000	0.000000
50%	4.000000	4.000000	91.600000	108.300000	664.200000	8.400000	19.300000	42.000000	4.000000	0.000000	0.550000
75%	7.000000	5.000000	92.900000	142.400000	713.900000	10.800000	22.800000	53.000000	4.900000	0.000000	6.570000
max	9.000000	9.000000	96.200000	291.300000	860.600000	56.100000	33.300000	100.000000	9.400000	6.400000	1090.840000

Fig 2: Dataset summarizing the columns

Following python code is used to check for null values in the dataset

Code: data.isnull().any()

```
data.isnull().any()
```

```
X        False
Y        False
month    False
day      False
FFMC     False
DMC      False
DC       False
ISI      False
temp     False
RH       False
wind     False
rain     False
area     False
dtype: bool
```

Fig 3: Identifying null values in the dataset

Following code is to identify shape of the dataset

Code: data.shape

```
In [5]: data.shape

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

Fig 4: Identifying shape of the dataset

Following code is to identify the count of months in the dataset

Code: data['month'].value_counts()

```
[6]: data['month'].value_counts()

[6]: aug    184
     sep    172
     mar     54
     jul     32
     feb     20
     jun     17
     oct     15
     dec      9
     apr      9
     jan      2
     may      2
     nov      1
     Name: month, dtype: int64
```

Fig 5: counts the months of the dataset

Following code is to groupby the months and wind in the dataset

Code: month=data.groupby(['month'])['wind'].mean().reset_index()
month

	month	wind
0	apr	4.666667
1	aug	4.086413
2	dec	7.644444
3	feb	3.755000
4	jan	2.000000
5	jul	3.734375
6	jun	4.135294
7	mar	4.968519
8	may	4.450000
9	nov	4.500000
10	oct	3.460000
11	sep	3.557558

Fig 6: Group by the month and wind of the dataset

Following code is to groupby the months ,wind,temperatue and rain in the dataset

Code: temp=data.groupby(['month'])['temp'].mean().reset_index()
rain=data.groupby(['month'])['rain'].mean().reset_index()
month['temp']=temp['temp']
month['rain']=rain['rain'] month

	month	wind	temp	rain
0	apr	4.666667	12.044444	0.000000
1	aug	4.086413	21.631522	0.058696
2	dec	7.644444	4.522222	0.000000
3	feb	3.755000	9.635000	0.000000
4	jan	2.000000	5.250000	0.000000
5	jul	3.734375	22.109375	0.006250
6	jun	4.135294	20.494118	0.000000
7	mar	4.968519	13.083333	0.003704
8	may	4.450000	14.650000	0.000000
9	nov	4.500000	11.800000	0.000000
10	oct	3.460000	17.093333	0.000000
11	sep	3.557558	19.612209	0.000000

Fig 7: Groupby the months ,wind,temperatue and rain in the dataset

1.Mean values of temperature,wind and rain based on months

This line plot represents to identify the averages of temperature,wind,rain based on the months in the forest fires dataset csv file of Portugal shows different colours for each three lines in the plot.

Code: plt.figure(figsize=(10,8))

```
x=month['month'].tolist()
y1=month['wind'].tolist()
y2=month['temp'].tolist()
y3=month['rain'].tolist()
plt.plot(x,y1,color='gold',label='wind')
plt.plot(x,y2,color='red',label='temp')
plt.plot(x,y3,color='blue',label='rain')
plt.title("mean values of temp,wind and rain based on
months",fontsize=25,fontweight='bold')
plt.xlabel("months",fontsize=20,fontweight='bold')
plt.ylabel("avg of wind,temp & rain",fontsize=20,fontweight='bold')
plt.xticks(x,fontsize=15,fontweight='bold')
plt.yticks(np.arange(0,30,2),fontsize=15)
plt.legend(fontsize=15)
```

```
plt.show()
```

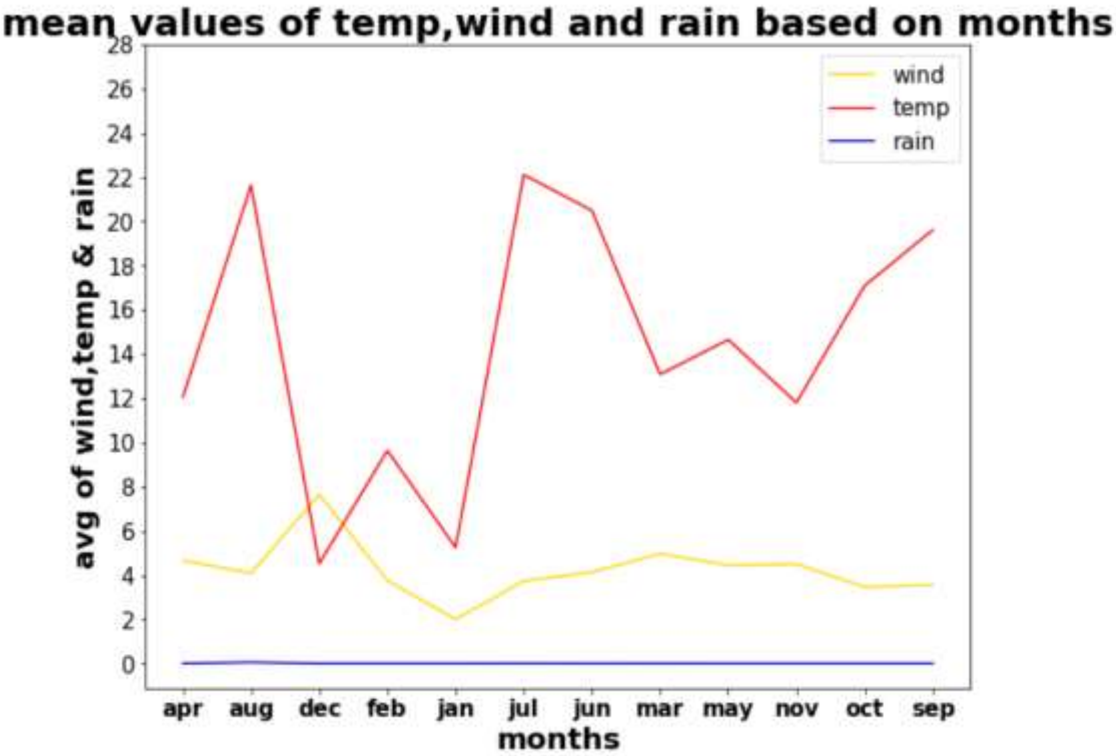

Fig 8: Mean values of temperature,wind and rain based on months

As per the Fig 8 visualization it has been observed that three columns in the dataset shows the average
based of months

code:
```
area=data.groupby(['month'])['area'].mean().reset_index()
rh=data.groupby(['month'])['RH'].mean().reset_index()
dc=data.groupby(['month'])['DC'].mean().reset_index()
ffmc=data.groupby(['month'])['FFMC'].mean().reset_index()
dmc=data.groupby(['month'])['DMC'].mean().reset_index()
isi=data.groupby(['month'])['ISI'].mean().reset_index()
month['area']=area['area']
month['rh']=rh['RH']
month['dc']=dc['DC']
month['ffmc']=ffmc['FFMC']
month['dmc']=dmc['DMC']
month['isi']=isi['ISI']
month
```

	month	wind	temp	rain	area	rh	dc	ffmc	dmc	isi
0	apr	4.666667	12.044444	0.000000	8.891111	46.888889	48.555556	85.788889	15.911111	5.377778
1	aug	4.086413	21.631522	0.058696	12.499674	45.489130	641.077717	92.336957	153.732609	11.072283
2	dec	7.644444	4.522222	0.000000	13.330000	38.444444	351.244444	84.966667	26.122222	3.466667
3	feb	3.755000	9.635000	0.000000	6.275000	55.700000	54.670000	82.905000	9.475000	3.350000
4	jan	2.000000	5.250000	0.000000	0.000000	89.000000	90.350000	50.400000	2.400000	1.450000
5	jul	3.734375	22.109375	0.006250	14.369687	45.125000	450.603125	91.328125	110.387500	9.393750
6	jun	4.135294	20.494118	0.000000	5.841176	45.117647	297.705882	89.429412	93.382353	11.776471
7	mar	4.968519	13.083333	0.003704	4.356667	40.000000	75.942593	89.444444	34.542593	7.107407
8	may	4.450000	14.650000	0.000000	19.240000	67.000000	93.750000	87.350000	26.700000	4.600000
9	nov	4.500000	11.800000	0.000000	5.800000	31.000000	106.700000	79.500000	3.000000	1.100000
10	oct	3.460000	17.093333	0.000000	6.638000	37.466667	681.673333	90.453333	41.420000	7.146667
11	sep	3.557558	19.612209	0.000000	17.942616	42.843023	734.615698	91.243023	120.922674	8.577326

Fig 9:Groupby summerizing columns with months included in columns

2.Features of forest on april month

In this pie plot the plot shows the features of the forest in the april month for some of the columns in the forest fires dataset.

```
Code: y1=month['rh'].tolist()
y2=month['dc'].tolist()
y3=month['ffmc'].tolist()
y4=month['dmc'].tolist()
y5=month['isi'].tolist()

plt.figure(figsize=(10,8))
labels=['rh','dc','ffmc','dmc','isi']
piedata=[y1[0],y2[0],y3[0],y4[0],y5[0]]
colors=['green','b','lawngreen','c','purple']
plt.pie(piedata, colors=colors,
        shadow = False, explode = (0.1,0,0,0,0),
       radius = 1.2, autopct = '%1.2f%%')
plt.legend(labels,fontsize=15)
plt.title("features of forest on april month",fontsize=20,fontweight='bold')
plt.show()
```

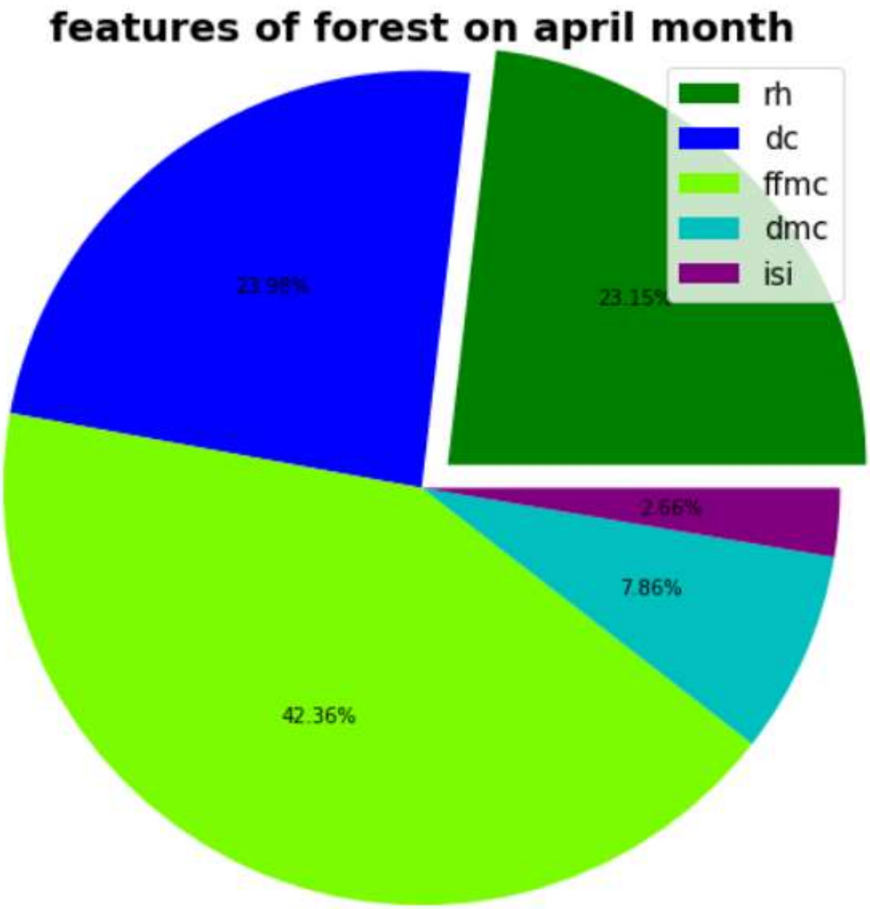

Fig 10: Shows the visualization features of forest in april month highest percentage in ffmc

```
13]: data['day'].value_counts()

13]: sun    95
     fri    85
     sat    84
     mon    74
     tue    64
     thu    61
     wed    54
     Name: day, dtype: int64
```

Fig 11:Count of day

```
]: data['DMC'].value_counts()

]: 99.0     10
   129.5     9
   142.4     8
   231.1     8
   137.0     7
            ..
   4.6       1
   24.9      1
   133.6     1
   96.3      1
   3.2       1
   Name: DMC, Length: 215, dtype: int64
```

Fig 12:Count of DMC

```
]: data['FFMC'].value_counts()

]: 91.6    28
   92.1    28
   91.0    22
   91.7    19
   93.7    16
           ..
   50.4     1
   82.1     1
   86.3     1
   85.1     1
   87.1     1
   Name: FFMC, Length: 106, dtype: int64
```

Fig 13: Count of FFMC

```
: data['DC'].value_counts()

: 745.3    10
  692.6     9
  698.6     8
  601.4     8
  692.3     8
           ..
  730.6     1
  431.6     1
  74.3      1
  313.4     1
  537.4     1
  Name: DC, Length: 219, dtype: int64
```

Fig 14: Count of FFMC

```
data['ISI'].value_counts()
```

```
9.6     23
7.1     21
6.3     20
7.0     17
8.4     17
        ..
7.3      1
12.1     1
14.6     1
56.1     1
22.7     1
Name: ISI, Length: 119, dtype: int64
```

Fig 15: Count of ISI

Code: data1=data.tail(5)
data1

The above code gives the last 5 rows from the dataset

	X	Y	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area
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	1.95
516	6	3	nov	tue	79.5	3.0	106.7	1.1	11.8	31	4.5	0.0	5.80

Fig 16: Dataset shows the last 5 rows

3.Shows temperature in that specified area

This bar graph tells that area vs temp graph which represents area in x-axis and temp in y-axis and describes relation between them in temperature in that specified area of land in forest fires dataset.

Code:
```
plt.figure(figsize = (30,10))
b1=sns.barplot(x = 'area', y = 'temp', data = data1)
for p in b1.patches:
    b1.annotate(format(p.get_height(), '.2f'), (p.get_x() + p.get_width() / 2., p.get_height()), ha = 'center', va = 'center', xytext = (0, 10), textcoords = 'offset points',fontsize=20)
plt.title("area vs temp",fontsize=40,fontweight='bold')
plt.xticks(rotation=70,fontsize=20)
plt.yticks(fontsize=20)
plt.xlabel("area",fontsize=30,fontweight='bold')
plt.ylabel("temp",fontsize=30,fontweight='bold')
```

```
1]: Text(0, 0.5, 'temp')
```

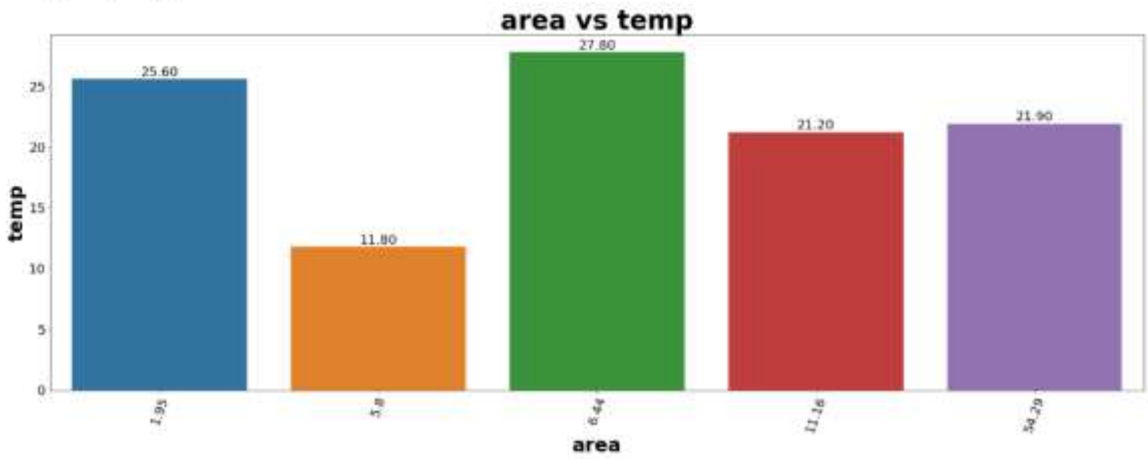

Fig 17: Shows the visualization of temperature in the specified area

value counts chooses unique data items from attribute day counts number of occurences of each type value in the dataset.

```
: d1=data['FFMC'].value_counts().reset_index()
  d1
```

	index	FFMC
0	91.6	28
1	92.1	28
2	91.0	22
3	91.7	19
4	93.7	16
...	...	...
101	50.4	1
102	82.1	1
103	86.3	1
104	85.1	1
105	87.1	1

106 rows × 2 columns

Fig18:Count of FFMC values

. Count of FFMC

This bar graph tells that count of ffmc graph which represents FFMC in x-axis and count in y-axis and describes relation as individual FFMC counts or number of distinct FFMC and counts.

Code:
```
d1=d1.head(35)
plt.figure(figsize = (25,10))
splot=sns.barplot(x = 'index', y = 'FFMC', data = d1)
for p in splot.patches:
    splot.annotate(format(p.get_height(), '.2f'), (p.get_x() + p.get_width() / 2., p.get_height()),
ha = 'center', va = 'center', xytext = (0, 10), textcoords = 'offset points',fontsize=15)
plt.title('count of FFMC',fontsize=40,fontweight='bold')
plt.xticks(rotation=70,fontweight='bold',fontsize=15)
plt.yticks(fontsize=20)
plt.xlabel("FFMC",fontsize=30,fontweight='bold')
plt.ylabel("count",fontsize=30,fontweight='bold')
```

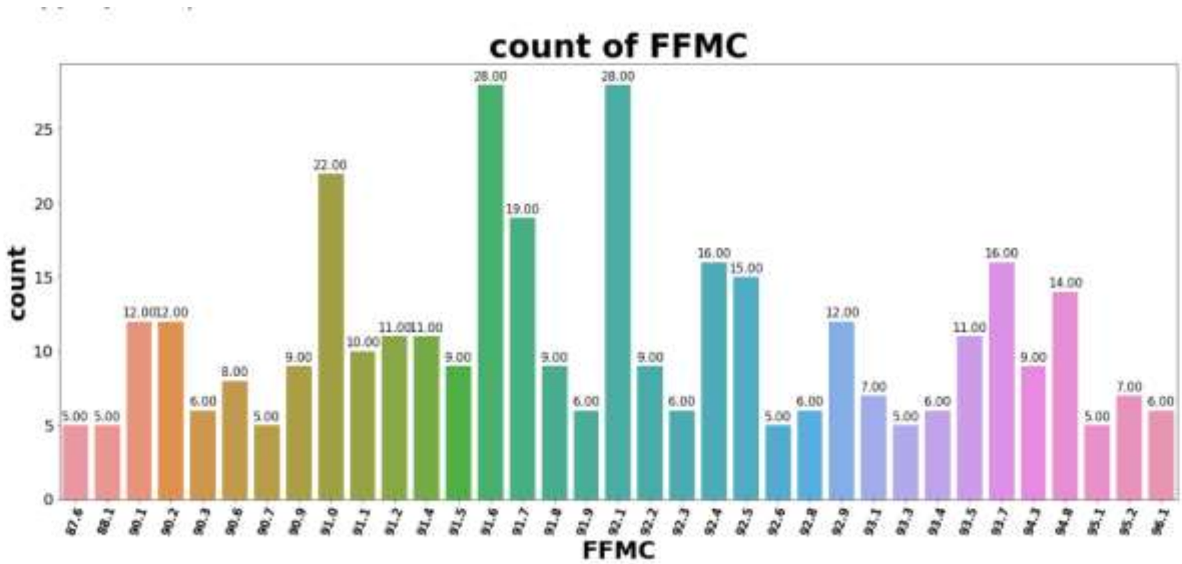

Fig19:Visualization of count of FFMC

5.Shows the mean values of months and DC

This bar graph tells that month vs DC graph which represents month in x-axis and DC y-axis and describes relation between month and DC shows the mean of the data in the two attributes in the dataset.

Code:
```
plt.figure(figsize = (12, 8))
splot=sns.barplot(x = 'month', y = 'DC', data = data)
for p in splot.patches:
    splot.annotate(format(p.get_height(), '.2f'), (p.get_x() + p.get_width() / 2., p.get_height()),
ha = 'center', va = 'center', xytext = (0, 10), textcoords = 'offset points',fontsize=15)
```

```
plt.title('month vs DC',fontsize=40,fontweight='bold')
plt.xticks(fontsize=20,fontweight='bold')
plt.yticks(fontsize=20)
plt.xlabel("month",fontsize=30,fontweight='bold')
plt.ylabel("Dc",fontsize=30,fontweight='bold')
```

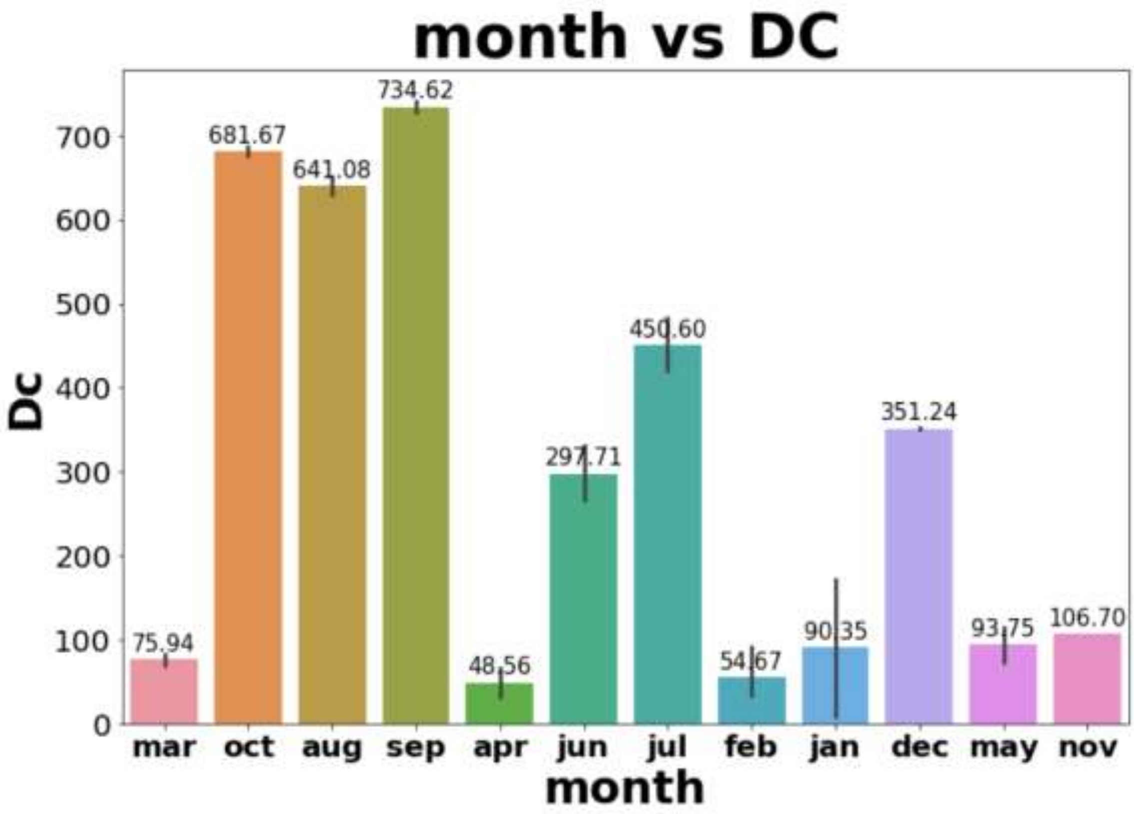

Fig20:Visualization of average in month and DC the forest fires dataset

6. Shows the mean values of day and DC

In this graph days in x-axis and DC in y-axis and it gives the total count of DC for individual days in the data.

Code:
```
plt.figure(figsize = (15,8))
splot=sns.barplot(x = 'day', y = 'DC', data = data)
for p in splot.patches:
    splot.annotate(format(p.get_height(), '.2f'), (p.get_x() + p.get_width() / 2., p.get_height()),
ha = 'center', va = 'center', xytext = (0, 10), textcoords = 'offset points',fontsize=18)
plt.title('day vs dc',fontsize=30,fontweight='bold')
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.xlabel("day",fontsize=20,fontweight='bold')
plt.ylabel("Dc",fontsize=20,fontweight='bold')
```

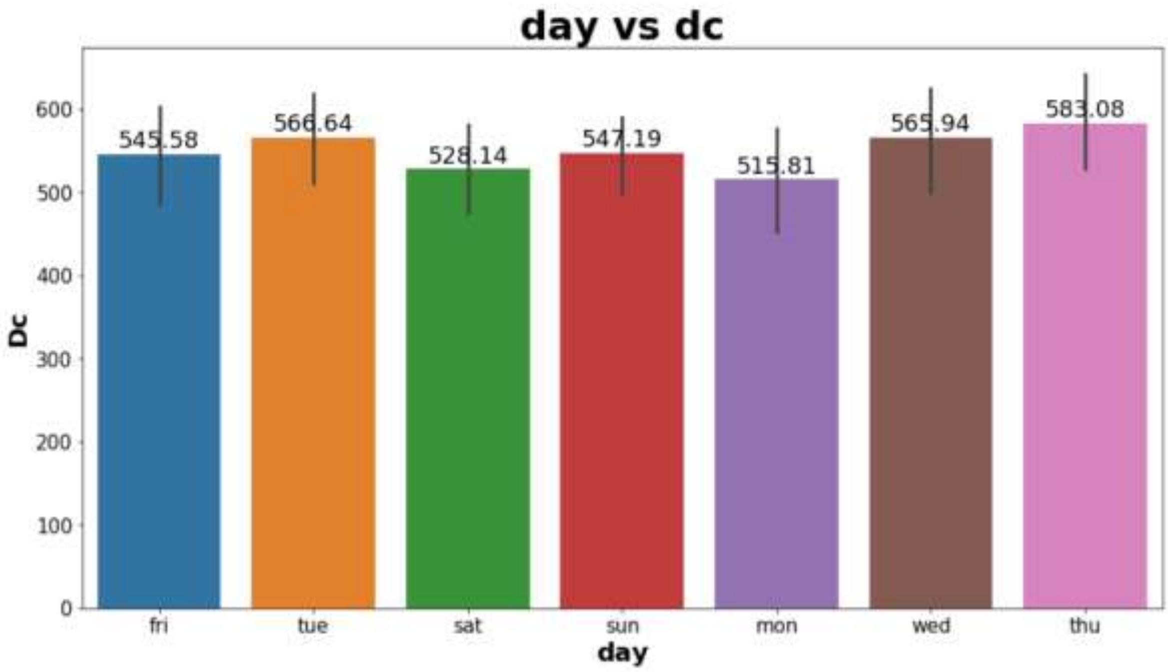

Fig21:Shows the visualization of average in day and DC the forest fires dataset

7. Finds the relationship between the features of dataset

This heatmap finds the relationship between thefeatures of dataset
If it comes in negative values it is worst relationship if it shows positive values that is good relationship.

Code:
```
f, ax = plt.subplots(figsize = (10, 10))
corr = data.corr()
sns.heatmap(corr, mask = np.zeros_like(corr, dtype = np.bool), annot=True,
cmap = sns.diverging_palette(50, 10, as_cmap = True), square = True, ax = ax)
```

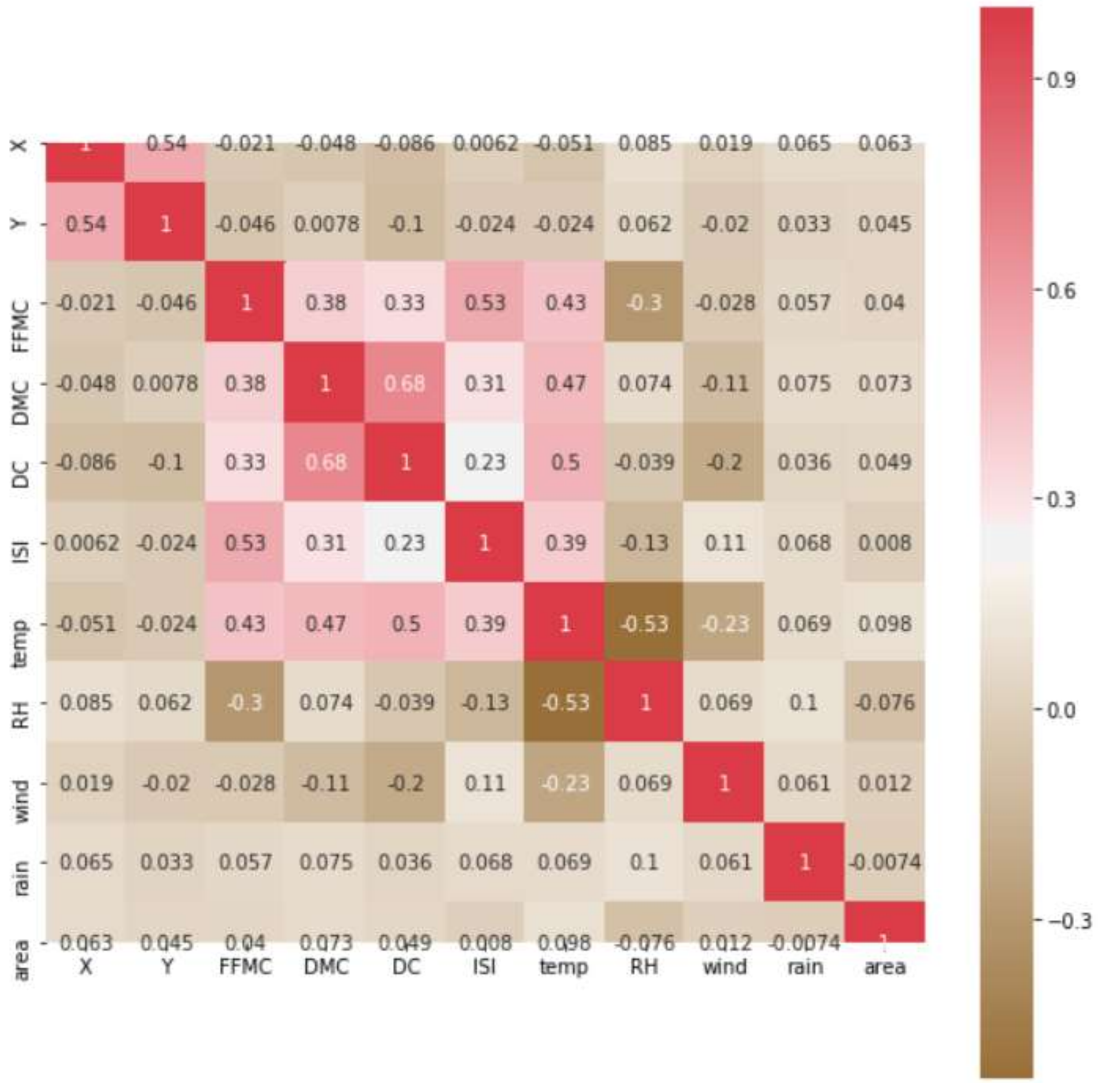

Fig22: Visualized the relationship between the features of dataset

```
: month=data.groupby(['month'])['wind'].mean().reset_index()

  month
```

:

	month	wind
0	apr	4.666667
1	aug	4.086413
2	dec	7.644444
3	feb	3.755000
4	jan	2.000000
5	jul	3.734375
6	jun	4.135294
7	mar	4.968519
8	may	4.450000
9	nov	4.500000
10	oct	3.460000
11	sep	3.557558

Fig23:Groupby month and wind

8. Average of wind with respect to month

It gives the relationship as average wind in each month in the data of the forest fires dataset.

Code:
```
plt.figure(figsize=(10,8))
m1=plt.bar(month['month'],month['wind'],color='bg')
plt.title("avg of wing w.r.t month",fontsize=30,fontweight='bold')
plt.xlabel("months",fontsize=20,fontweight='bold')
plt.ylabel("avg wind",fontsize=20,fontweight='bold')
def autolabel(rect1):
  for rect in rect1:
    height = rect.get_height()

    plt.text(rect.get_x() + rect.get_width()/2., height,
        '%d' % int(height),
    ha='center', va='bottom',color='k',fontsize=15)
autolabel(m1)
plt.show()
```

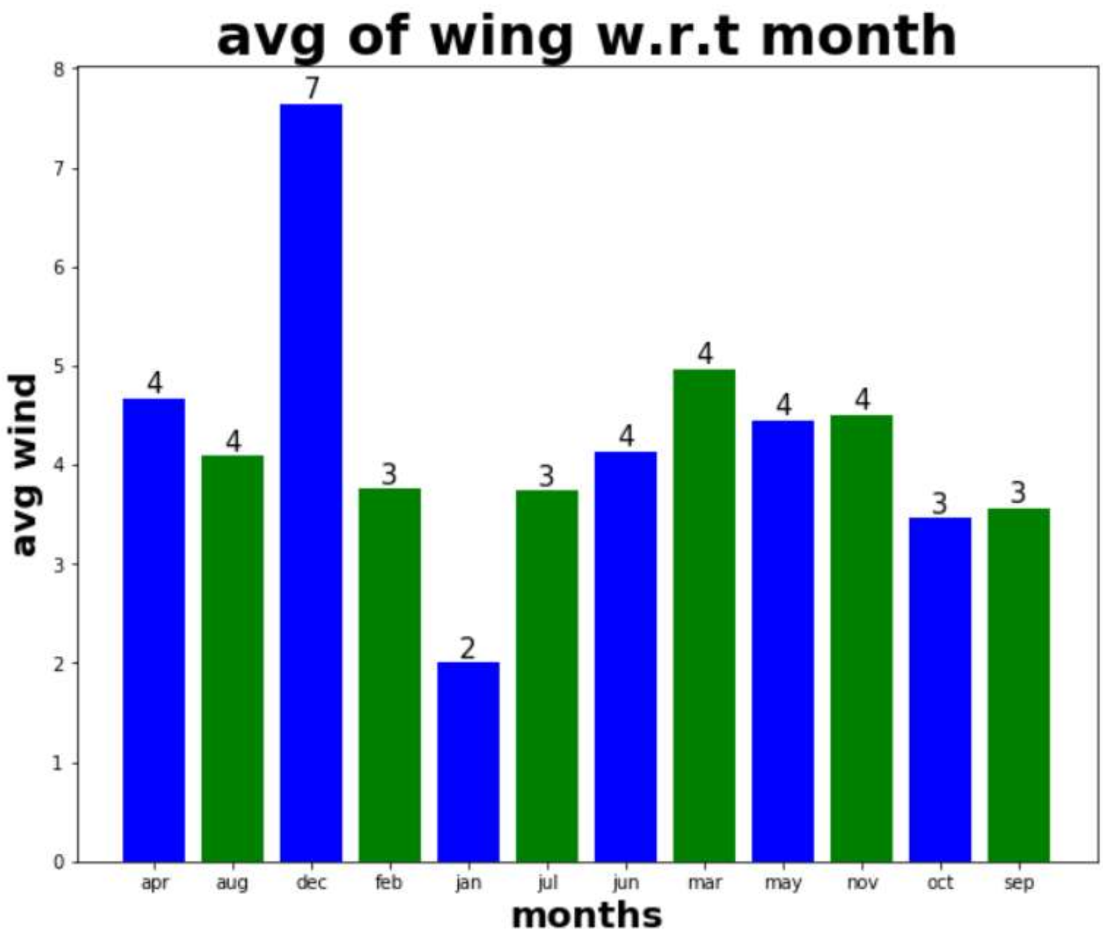

Fig24:Average of wind w.r.t month

9.Day vs temperature violin plot

This plot describes the average temperature on each individual day with day in x-axis and temp in y-axis of the data present in the dataset.

Code:
```
plt.figure(figsize = (18, 6))
sns.violinplot(x = 'day', y = 'temp', data = data)

plt.title('Day vs temp',fontsize=35)
plt.xlabel("day",fontsize=20)
plt.ylabel("temp",fontsize=20)
```

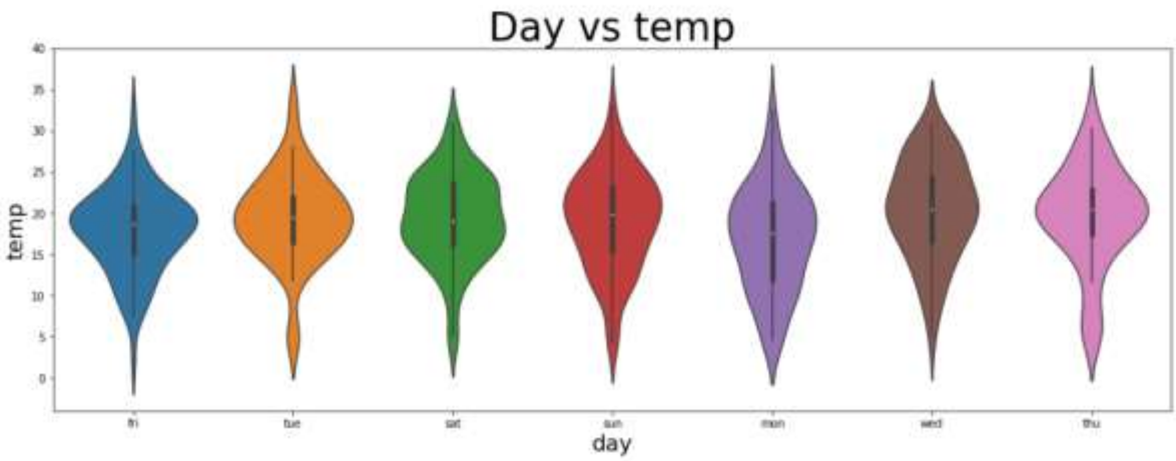

Fig25:Shows the visualization of violinplot of day vs temperature in forest fires dataset features

10. Month wise rain predict

For each individual month given it describes the rate of rain,month in x-axis and rain in y-axis and predicts the rain in each month.

Code:
```
plt.figure(figsize = (10,8))
sns.scatterplot(x = 'month', y = 'rain', data = data,color='maroon')

plt.title('month vs rain',fontsize=20,fontweight='bold')
plt.xlabel("month",fontsize=15,fontweight='bold')
plt.ylabel("rain",fontsize=15,fontweight='bold')
```

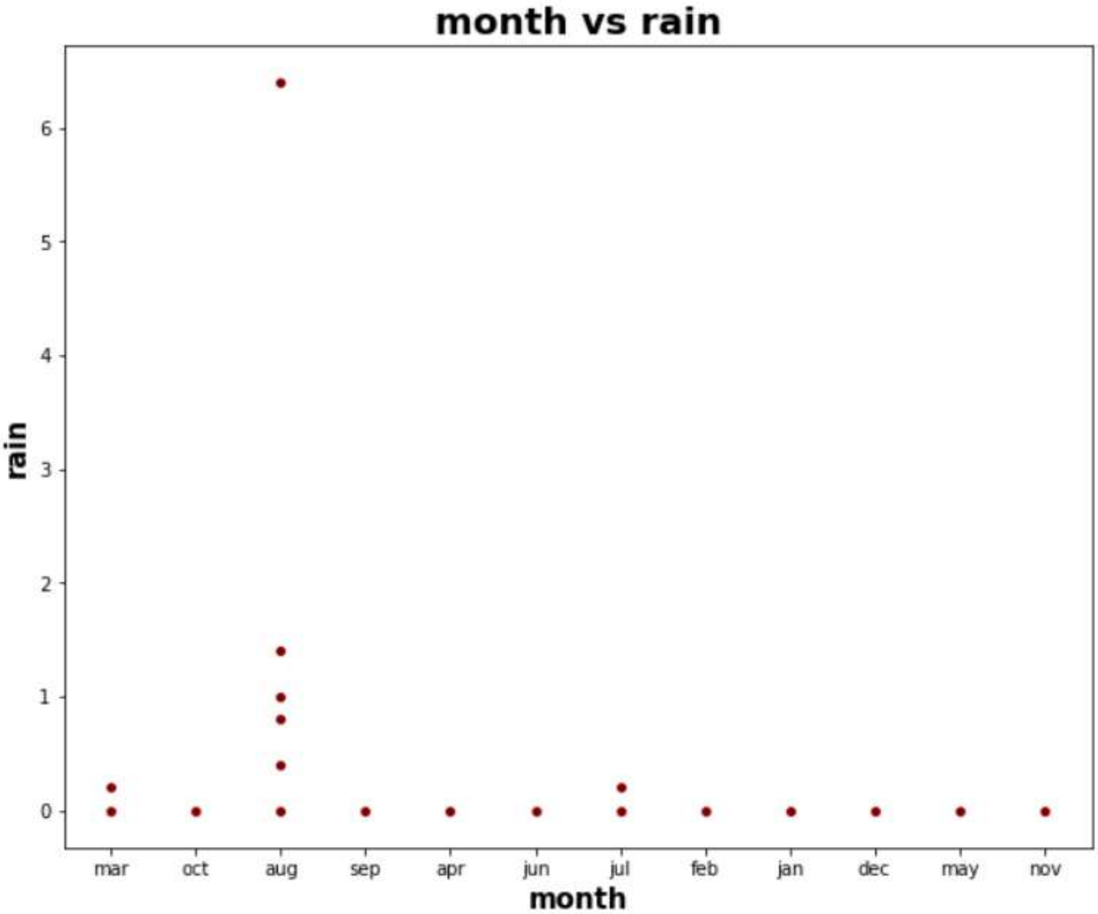

Fig26:Shows the visualization of scatterplot for month wise rain predict

11. Day vs tempetarure predict.

It describes the day vs temp relation but also classify this according to the month given in the scatter plot

Code:
```
plt.figure(figsize = (18, 6))
sns.FacetGrid(data, hue = 'month', size=5) \
   .map(plt.scatter, 'day','temp') \
   .add_legend(fontsize=12)
plt.title("Day vs Temp",fontsize=20,fontweight='bold')
plt.xlabel("day",fontsize=15,fontweight='bold')
plt.ylabel("temp",fontsize=15,fontweight='bold')
```

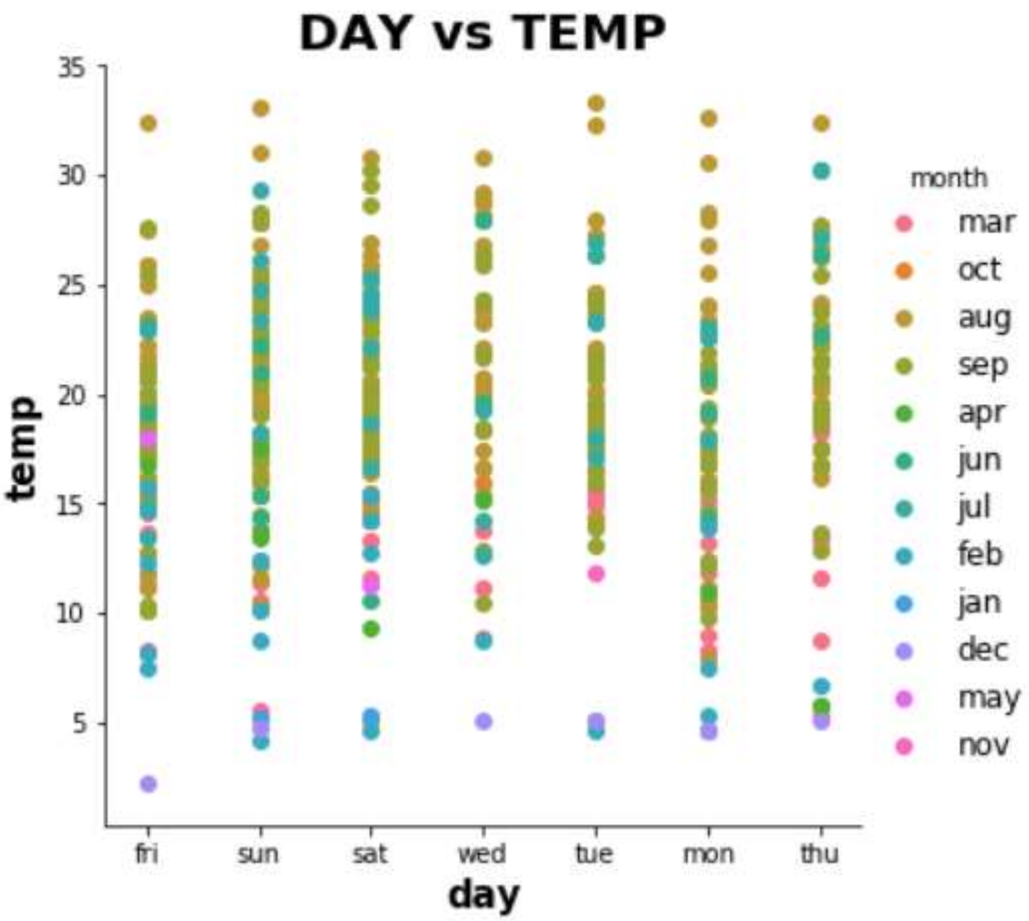

Fig27:Shows the visualization of scatterplot for day vs temperature wise rain predict

Conclusion

Forest fires cause a significant environmental damage while threatening human lives. In the last two decades, a substantial effort was made to build automatic detection tools that could assist Fire Management Systems (FMS). The three major trends are the use of satellite data, infrared/smoke scanners and local sensors (e.g. meteorological). In this work, we propose a Data Mining (DM) approach that uses meteorological data, as detected by local sensors in weather stations, and that is known to influence forest fires. The advantage is that such data can be collected in real-time and with very low costs, when compared with the satellite and scanner approaches. Recent real-world data, from the northeast region of Portugal, was used in the experiments. The database included spatial, temporal, components from the Canadian Fire Weather Index (FWI) and four weather conditions (i.e. temperature, rain, relative humidity and wind speed) is capable of predicting small fires, which constitute the majority of the fire occurrences. The drawback is the lower predictive accuracy for large fires. To our knowledge, this is the first time the burn area is predicted using only meteorological based data and further exploratory research is required. As argued in [18], predicting the size of forest fires is a challenging task. To improve it, we believe that additional information (not available in this study) is required, such as the type of vegetation and firefighting intervention (e.g. time elapsed and firefighting strategy). Nevertheless, the proposed model is still useful

to improve fire fighting resource management. For instance, when small fires are predicted then air tankers could be spared
and small ground crews could be sent. Such management would be particularly advantageous in dramatic fire seasons, when simultaneous fires occur at distinct locations.

References

1. https://en.wikipedia.org/wiki/Category:forest_fires_csv
2. https://www.python.org/
3. van der Walt, Stéfan & Colbert, S. & Varoquaux, Gael. (2011). The NumPy Array: A Structure for Efficient Numerical Computation. Computing in Science & Engineering. 13. 22 - 30. 10.1109/MCSE.2011.37.
4. Barrett, Paul & Hunter, J. & Miller, J.T. & Hsu, J.-C & Greenfield, P.. (2005). matplotlib -- A Portable Python Plotting Package.
5. https://plotly.com/
6. https://packaging.python.org/tutorials/installing-packages/
7. https://statinfer.com/104-2-2-practice-working-with-datasets-in-python/
8. https://www.shanelynn.ie/python-pandas-read_csv-load-data-from-csv-files/
9. https://jakevdp.github.io/PythonDataScienceHandbook/03.03-operations-in-pandas.html
10. https://www.datacamp.com/community/tutorials/matplotlib-tutorial-python