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 temperature 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, plotlib. 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.

```python
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

```python
data = pd.read_csv('forestfires.csv')
print(data.shape)
data.head()
```

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

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>month</th>
<th>day</th>
<th>FFMC</th>
<th>DMC</th>
<th>DC</th>
<th>ISI</th>
<th>temp</th>
<th>RH</th>
<th>wind</th>
<th>rain</th>
<th>area</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>7</td>
<td>mar</td>
<td>fri</td>
<td>86.2</td>
<td>26.2</td>
<td>94.3</td>
<td>5.1</td>
<td>8.2</td>
<td>51</td>
<td>6.7</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>oct</td>
<td>tue</td>
<td>90.6</td>
<td>35.4</td>
<td>698.1</td>
<td>6.7</td>
<td>18.0</td>
<td>33</td>
<td>0.9</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td>oct</td>
<td>sat</td>
<td>90.6</td>
<td>43.7</td>
<td>698.9</td>
<td>6.7</td>
<td>14.6</td>
<td>33</td>
<td>1.3</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>mar</td>
<td>fri</td>
<td>91.7</td>
<td>33.3</td>
<td>77.5</td>
<td>9.0</td>
<td>8.3</td>
<td>97</td>
<td>4.0</td>
<td>0.2</td>
<td>0.0</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>mar</td>
<td>sun</td>
<td>89.3</td>
<td>51.3</td>
<td>102.2</td>
<td>9.6</td>
<td>11.4</td>
<td>99</td>
<td>1.8</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

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()`

<table>
<thead>
<tr>
<th></th>
<th>X</th>
<th>Y</th>
<th>FFMC</th>
<th>DMC</th>
<th>DC</th>
<th>ISI</th>
<th>temp</th>
<th>RH</th>
<th>wind</th>
<th>rain</th>
<th>area</th>
</tr>
</thead>
<tbody>
<tr>
<td>count</td>
<td>517</td>
<td>517</td>
<td>517</td>
<td>517</td>
<td>517</td>
<td>517</td>
<td>517</td>
<td>517</td>
<td>517</td>
<td>517</td>
<td>517</td>
</tr>
<tr>
<td>mean</td>
<td>4.068246</td>
<td>4.298968</td>
<td>90.064061</td>
<td>110.92240</td>
<td>347.940038</td>
<td>9.021060</td>
<td>18.069150</td>
<td>44.208221</td>
<td>4.017062</td>
<td>0.021063</td>
<td>12.902282</td>
</tr>
<tr>
<td>std</td>
<td>2.313778</td>
<td>1.229900</td>
<td>5.520111</td>
<td>61.068682</td>
<td>288.868619</td>
<td>4.559477</td>
<td>5.666625</td>
<td>16.317469</td>
<td>1.791853</td>
<td>0.259595</td>
<td>63.053355</td>
</tr>
<tr>
<td>min</td>
<td>0.000000</td>
<td>0.000000</td>
<td>16.000000</td>
<td>1.000000</td>
<td>7.000000</td>
<td>0.000000</td>
<td>2.000000</td>
<td>15.000000</td>
<td>0.400000</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>25%</td>
<td>3.000000</td>
<td>4.000000</td>
<td>40.000000</td>
<td>60.000000</td>
<td>437.700000</td>
<td>5.000000</td>
<td>13.500000</td>
<td>33.000000</td>
<td>2.700000</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>50%</td>
<td>4.000000</td>
<td>4.000000</td>
<td>91.600000</td>
<td>108.300000</td>
<td>664.200000</td>
<td>8.400000</td>
<td>19.300000</td>
<td>42.000000</td>
<td>4.000000</td>
<td>0.000000</td>
<td>0.550000</td>
</tr>
<tr>
<td>75%</td>
<td>7.000000</td>
<td>5.000000</td>
<td>92.900000</td>
<td>142.400000</td>
<td>713.900000</td>
<td>10.800000</td>
<td>22.800000</td>
<td>53.000000</td>
<td>4.900000</td>
<td>0.000000</td>
<td>6.570000</td>
</tr>
</tbody>
</table>

Fig 2: Dataset summarizing the columns

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

Code: `data.isnull().any()`

```python
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()

```

<table>
<thead>
<tr>
<th></th>
<th>month</th>
<th>wind</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>apr</td>
<td>4.600667</td>
</tr>
<tr>
<td>1</td>
<td>aug</td>
<td>4.086413</td>
</tr>
<tr>
<td>2</td>
<td>dec</td>
<td>7.644444</td>
</tr>
<tr>
<td>3</td>
<td>feb</td>
<td>3.755000</td>
</tr>
<tr>
<td>4</td>
<td>jan</td>
<td>2.000000</td>
</tr>
<tr>
<td>5</td>
<td>jul</td>
<td>3.734375</td>
</tr>
<tr>
<td>6</td>
<td>jun</td>
<td>4.135294</td>
</tr>
<tr>
<td>7</td>
<td>mar</td>
<td>4.968519</td>
</tr>
<tr>
<td>8</td>
<td>may</td>
<td>4.450000</td>
</tr>
<tr>
<td>9</td>
<td>nov</td>
<td>4.500000</td>
</tr>
<tr>
<td>10</td>
<td>oct</td>
<td>3.460000</td>
</tr>
<tr>
<td>11</td>
<td>sep</td>
<td>3.557558</td>
</tr>
</tbody>
</table>
```

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

Following code is to groupby the months, wind, temperature 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']

<table>
<thead>
<tr>
<th>month</th>
<th>wind</th>
<th>temp</th>
<th>rain</th>
</tr>
</thead>
<tbody>
<tr>
<td>apr</td>
<td>4.666667</td>
<td>12.044444</td>
<td>0.000000</td>
</tr>
<tr>
<td>aug</td>
<td>4.086413</td>
<td>21.631522</td>
<td>0.058696</td>
</tr>
<tr>
<td>dec</td>
<td>7.6444444</td>
<td>4.522222</td>
<td>0.000000</td>
</tr>
<tr>
<td>feb</td>
<td>3.7550000</td>
<td>9.6350000</td>
<td>0.000000</td>
</tr>
<tr>
<td>jan</td>
<td>2.000000</td>
<td>5.250000</td>
<td>0.000000</td>
</tr>
<tr>
<td>jul</td>
<td>3.734375</td>
<td>22.109375</td>
<td>0.006250</td>
</tr>
<tr>
<td>jun</td>
<td>4.135294</td>
<td>20.494118</td>
<td>0.000000</td>
</tr>
<tr>
<td>mar</td>
<td>4.968519</td>
<td>13.083333</td>
<td>0.003704</td>
</tr>
<tr>
<td>may</td>
<td>4.450000</td>
<td>14.650000</td>
<td>0.000000</td>
</tr>
<tr>
<td>nov</td>
<td>4.500000</td>
<td>11.800000</td>
<td>0.000000</td>
</tr>
<tr>
<td>oct</td>
<td>3.460000</td>
<td>17.093333</td>
<td>0.000000</td>
</tr>
<tr>
<td>sep</td>
<td>3.557558</td>
<td>19.612209</td>
<td>0.000000</td>
</tr>
</tbody>
</table>

Fig 7: Groupby the months ,wind,temperature 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)
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

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

```python
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, 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

```python
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
Fig 12: Count of DMC

```python
]: 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

Fig 13: Count of FFMC

```python
]: 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

Fig 14: Count of FFMC

```python
]: 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
The above code gives the last 5 rows from the dataset.

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>month</th>
<th>day</th>
<th>FFCM</th>
<th>DMC</th>
<th>DC</th>
<th>ISI</th>
<th>temp</th>
<th>RH</th>
<th>wind</th>
<th>rain</th>
<th>area</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>4</td>
<td>3</td>
<td>aug</td>
<td>sun</td>
<td>81.6</td>
<td>56.7</td>
<td>665.6</td>
<td>1.9</td>
<td>27.8</td>
<td>32</td>
<td>2.7</td>
<td>0.0</td>
</tr>
<tr>
<td>513</td>
<td>2</td>
<td>4</td>
<td>aug</td>
<td>sun</td>
<td>81.6</td>
<td>56.7</td>
<td>665.6</td>
<td>1.9</td>
<td>21.9</td>
<td>71</td>
<td>5.8</td>
<td>0.0</td>
</tr>
<tr>
<td>514</td>
<td>7</td>
<td>4</td>
<td>aug</td>
<td>sun</td>
<td>81.6</td>
<td>56.7</td>
<td>665.6</td>
<td>1.9</td>
<td>21.2</td>
<td>70</td>
<td>6.7</td>
<td>0.0</td>
</tr>
<tr>
<td>515</td>
<td>1</td>
<td>4</td>
<td>aug</td>
<td>sat</td>
<td>94.4</td>
<td>146.0</td>
<td>614.7</td>
<td>11.3</td>
<td>25.6</td>
<td>42</td>
<td>4.0</td>
<td>0.0</td>
</tr>
<tr>
<td>516</td>
<td>6</td>
<td>3</td>
<td>nov</td>
<td>tue</td>
<td>79.5</td>
<td>3.0</td>
<td>106.7</td>
<td>1.1</td>
<td>11.8</td>
<td>31</td>
<td>4.5</td>
<td>0.0</td>
</tr>
</tbody>
</table>

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

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

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

<table>
<thead>
<tr>
<th>index</th>
<th>FFMC</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>91.6</td>
</tr>
<tr>
<td>1</td>
<td>92.1</td>
</tr>
<tr>
<td>2</td>
<td>91.0</td>
</tr>
<tr>
<td>3</td>
<td>91.7</td>
</tr>
<tr>
<td>4</td>
<td>93.7</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>101</td>
<td>50.4</td>
</tr>
<tr>
<td>102</td>
<td>82.1</td>
</tr>
<tr>
<td>103</td>
<td>86.3</td>
</tr>
<tr>
<td>104</td>
<td>85.1</td>
</tr>
<tr>
<td>105</td>
<td>87.1</td>
</tr>
</tbody>
</table>

106 rows × 2 columns

Fig 18: 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)
```
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:
```python
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 the features 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
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.

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

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:
```python
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:
```python
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 Temperature predict.

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

Code:
```python
plt.figure(figsize = (18, 6))
sns.FacetGrid(data, hue = 'month', size=5) \n   .map(plt.scatter, 'day', 'temp') \n   .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')
```
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
2. https://www.python.org/