Algerian Forest

March 31, 2023

1 All Regression Models

1.1 Import all necessary Libraries

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

1.2 Import the Dataset

[2]:
```
df=pd.read_csv("Algerian_forest_fires_dataset_UPDATE.csv",header=1)
```

[3]:
```
df.head()
```

[3]:
```
   day month  year Temperature  RH  Ws Rain   FFMC  DMC    DC  ISI  BUI  FWI  \
0  01    06  2012          29  57  18     0  65.7  3.4   7.6  1.3  3.4  0.5
1  02    06  2012          29  61  13   1.3  64.4  4.1   7.6    1  3.9  0.4
2  03    06  2012          26  82  22  13.1  47.1  2.5   7.1  0.3  2.7  0.1
3  04    06  2012          25  89  13   2.5  28.6  1.3   6.9    0  1.7    0
4  05    06  2012          27  77  16     0  64.8    3  14.2  1.2  3.9  0.5

    Classes
0  not fire
1  not fire
2  not fire
3  not fire
4  not fire
```

1.3 Check for Null Values

[4]:
```
df[df.isnull().any(axis=1)]
```

[4]:
```
                              day month  year Temperature   RH   Ws Rain   \
122  Sidi-Bel Abbes Region Dataset   NaN   NaN         NaN  NaN  NaN   NaN
```

```
167                            14    07  2012          37   37   18   0.2

     FFMC   DMC      DC   ISI   BUI      FWI Classes
122   NaN   NaN     NaN   NaN   NaN      NaN       NaN
167  88.9  12.9  14.6 9  12.5  10.4  fire          NaN
```

1.3.1 Observation:

- There were two Data Regions in given Dataset
- Let's divide them

[5]:
```
df.loc[:122,"region"]=0
df.loc[122:,"region"]=1
```

[6]:
```
df.head()
```

[6]:
```
   day month  year Temperature  RH  Ws Rain   FFMC  DMC    DC  ISI  BUI  FWI  \
0  01    06  2012          29  57  18     0  65.7  3.4   7.6  1.3  3.4  0.5
1  02    06  2012          29  61  13   1.3  64.4  4.1   7.6    1  3.9  0.4
2  03    06  2012          26  82  22  13.1  47.1  2.5   7.1  0.3  2.7  0.1
3  04    06  2012          25  89  13   2.5  28.6  1.3   6.9    0  1.7    0
4  05    06  2012          27  77  16     0  64.8    3  14.2  1.2  3.9  0.5

     Classes    region
0  not fire        0.0
1  not fire        0.0
2  not fire        0.0
3  not fire        0.0
4  not fire        0.0
```

[7]:
```
df[df.isnull().any(axis=1)]
```

[7]:
```
                              day month  year Temperature   RH   Ws Rain   \
122  Sidi-Bel Abbes Region Dataset   NaN   NaN         NaN  NaN  NaN   NaN
167                              14    07  2012          37   37   18   0.2

     FFMC   DMC      DC   ISI   BUI      FWI Classes    region
122   NaN   NaN     NaN   NaN   NaN      NaN       NaN     1.0
167  88.9  12.9  14.6 9  12.5  10.4  fire          NaN     1.0
```

1.4 Drop row 122 and 167, i.e,, Null values

[8]:
```
df.drop(index=[122,167],inplace=True)
```

[9]:
```
df[df.isnull().any(axis=1)]
```

[9]:
```
Empty DataFrame
Columns: [day, month, year, Temperature,  RH,  Ws, Rain , FFMC, DMC, DC, ISI,
BUI, FWI, Classes  , region]
Index: []
```

1.5 Now reset the indexes

[10]:
```
df.reset_index(drop=True,inplace=True)
```

[11]:
```
df.head()
```

[11]:

	day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	Classes	region
0	01	06	2012	29	57	18	0	65.7	3.4	7.6	1.3	3.4	0.5	not fire	0.0
1	02	06	2012	29	61	13	1.3	64.4	4.1	7.6	1	3.9	0.4	not fire	0.0
2	03	06	2012	26	82	22	13.1	47.1	2.5	7.1	0.3	2.7	0.1	not fire	0.0
3	04	06	2012	25	89	13	2.5	28.6	1.3	6.9	0	1.7	0	not fire	0.0
4	05	06	2012	27	77	16	0	64.8	3	14.2	1.2	3.9	0.5	not fire	0.0

1.6 Check for Duplicate Rows

[12]:
```
df.duplicated().sum()
```

[12]:
```
0
```

1.7 Summarize the Data

[13]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   day          244 non-null    object
 1   month        244 non-null    object
 2   year         244 non-null    object
 3   Temperature  244 non-null    object
 4    RH          244 non-null    object
 5    Ws          244 non-null    object
 6   Rain         244 non-null    object
```

```
 7   FFMC         244 non-null    object
 8   DMC          244 non-null    object
 9   DC           244 non-null    object
 10  ISI          244 non-null    object
 11  BUI          244 non-null    object
 12  FWI          244 non-null    object
 13  Classes      244 non-null    object
 14  region       244 non-null    float64
dtypes: float64(1), object(14)
memory usage: 28.7+ KB
```

1.8 Check the Column Names

[14]:
```
df.columns
```

[14]:
```
Index(['day', 'month', 'year', 'Temperature', ' RH', ' Ws', 'Rain ', 'FFMC',
       'DMC', 'DC', 'ISI', 'BUI', 'FWI', 'Classes  ', 'region'],
      dtype='object')
```

1.9 Remove empty spaces in column names

[15]:
```
df.columns=df.columns.str.strip()
```

[16]:
```
df.columns
```

[16]:
```
Index(['day', 'month', 'year', 'Temperature', 'RH', 'Ws', 'Rain', 'FFMC',
       'DMC', 'DC', 'ISI', 'BUI', 'FWI', 'Classes', 'region'],
      dtype='object')
```

1.10 Change the Datatypes

[17]:
```
df.dtypes
```

[17]:
```
day             object
month           object
year            object
Temperature     object
RH              object
Ws              object
Rain            object
FFMC            object
DMC             object
DC              object
ISI             object
BUI             object
FWI             object
Classes         object
```

```
region         float64
dtype: object
```

1.11 Drop header values in Dataset

[18]:
```
df[df["day"]=="day"]
```

[18]:
```
     day  month  year  Temperature   RH   Ws   Rain  FFMC  DMC  DC  ISI  BUI  \
122  day  month  year  Temperature   RH   Ws  Rain   FFMC  DMC  DC  ISI  BUI

     FWI    Classes  region
122  FWI  Classes       1.0
```

[19]:
```
df.drop(index=[122],inplace=True)
```

[20]:
```
df[df["day"]=="day"]
```

[20]:
```
Empty DataFrame
Columns: [day, month, year, Temperature, RH, Ws, Rain, FFMC, DMC, DC, ISI, BUI,
FWI, Classes, region]
Index: []
```

1.12 Now convert the object columns to Numerical's

[21]:
```
df[['month','day','year','Temperature','RH','Ws',"region"]]=df[['month','day','year','Temperat
 ↪astype(int)
```

[22]:
```
df[["Rain","FFMC","DMC","DC","ISI","BUI","FWI"]]=df[["Rain","FFMC","DMC","DC","ISI","BUI","FWI
 ↪astype(float)
```

[23]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 243 entries, 0 to 243
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   day          243 non-null    int64
 1   month        243 non-null    int64
 2   year         243 non-null    int64
 3   Temperature  243 non-null    int64
 4   RH           243 non-null    int64
 5   Ws           243 non-null    int64
 6   Rain         243 non-null    float64
 7   FFMC         243 non-null    float64
 8   DMC          243 non-null    float64
 9   DC           243 non-null    float64
```

```
 10  ISI          243 non-null    float64
 11  BUI          243 non-null    float64
 12  FWI          243 non-null    float64
 13  Classes      243 non-null    object
 14  region       243 non-null    int64
dtypes: float64(7), int64(7), object(1)
memory usage: 30.4+ KB
```

1.13 Decode Classe's into Numericals

[24]:
```
df["Classes"]=np.where(df["Classes"].str.contains("not fire"),0,1)
```

[25]:
```
df
```

[25]:

	day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	\
0	1	6	2012	29	57	18	0.0	65.7	3.4	7.6	1.3	3.4	
1	2	6	2012	29	61	13	1.3	64.4	4.1	7.6	1.0	3.9	
2	3	6	2012	26	82	22	13.1	47.1	2.5	7.1	0.3	2.7	
3	4	6	2012	25	89	13	2.5	28.6	1.3	6.9	0.0	1.7	
4	5	6	2012	27	77	16	0.0	64.8	3.0	14.2	1.2	3.9	
..	…	…	…	…	..	..	…	…	…	…	…	…	
239	26	9	2012	30	65	14	0.0	85.4	16.0	44.5	4.5	16.9	
240	27	9	2012	28	87	15	4.4	41.1	6.5	8.0	0.1	6.2	
241	28	9	2012	27	87	29	0.5	45.9	3.5	7.9	0.4	3.4	
242	29	9	2012	24	54	18	0.1	79.7	4.3	15.2	1.7	5.1	
243	30	9	2012	24	64	15	0.2	67.3	3.8	16.5	1.2	4.8	

	FWI	Classes	region
0	0.5	0	0
1	0.4	0	0
2	0.1	0	0
3	0.0	0	0
4	0.5	0	0
..	…	…	…
239	6.5	1	1
240	0.0	0	1
241	0.2	0	1
242	0.7	0	1
243	0.5	0	1

[243 rows x 15 columns]

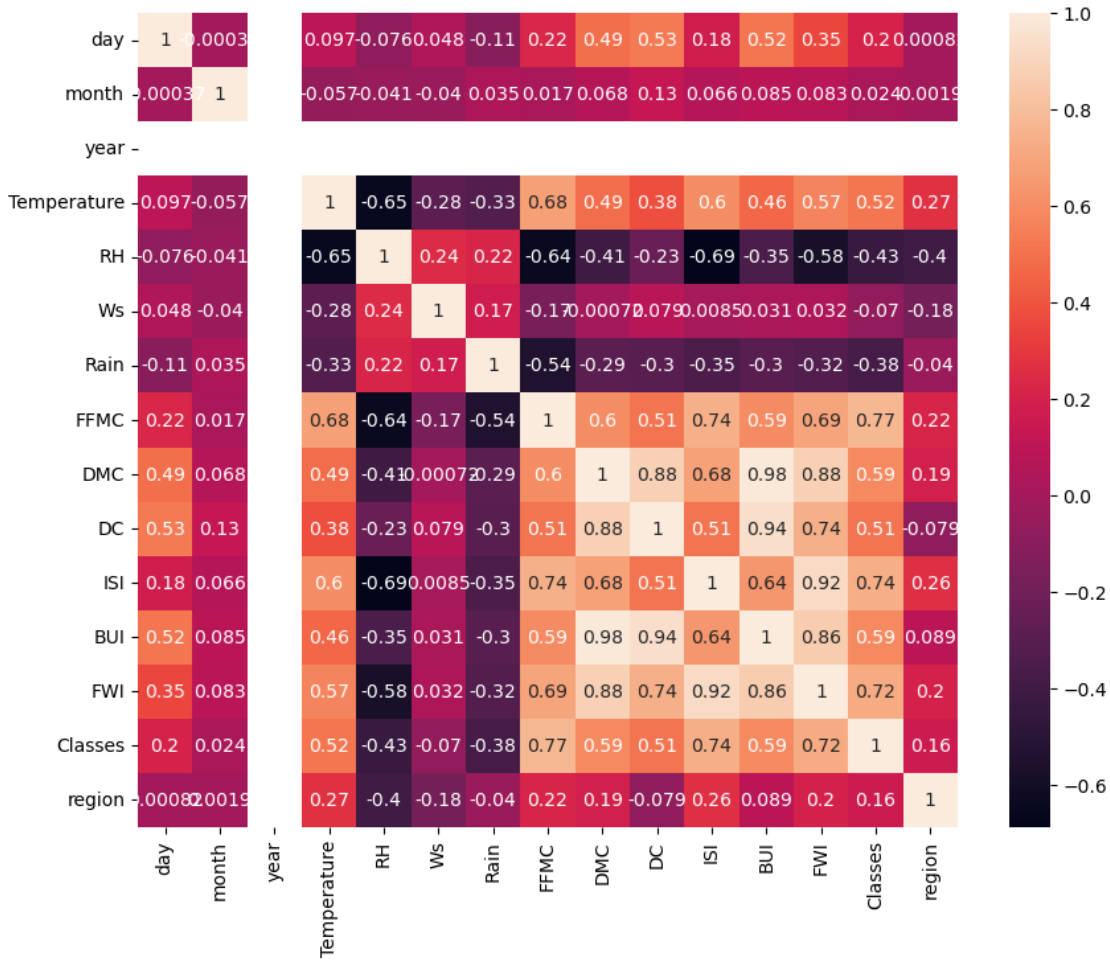
[26]:
```
df["Classes"].value_counts()
```

[26]:
```
1    137
0    106
Name: Classes, dtype: int64
```

1.14 Find the Coreleations between Data's

[27]:
```
plt.figure(figsize=(10,8))
sns.heatmap(df.corr(),annot=True)
```

[27]:
```
<AxesSubplot: >
```



1.14.1 Observations:

- Day, Year, Month and WS don't have much releation while determining FWI. so let's drop those columns

[28]:
```
df.drop(["day","month","year","Ws"],axis=1,inplace=True)
```

[29]:
```
df.head()
```

[29]:

	Temperature	RH	Rain	FFMC	DMC	DC	ISI	BUI	FWI	Classes	region
0	29	57	0.0	65.7	3.4	7.6	1.3	3.4	0.5	0	0

```
1           29  61   1.3  64.4  4.1   7.6  1.0  3.9  0.4        0       0
2           26  82  13.1  47.1  2.5   7.1  0.3  2.7  0.1        0       0
3           25  89   2.5  28.6  1.3   6.9  0.0  1.7  0.0        0       0
4           27  77   0.0  64.8  3.0  14.2  1.2  3.9  0.5        0       0
```

1.15 Split the Data

[30]:
```
x=df.drop("FWI",axis=1)
```

[31]:
```
x.head()
```

[31]:
```
   Temperature  RH  Rain  FFMC  DMC    DC  ISI  BUI  Classes  region
0           29  57   0.0  65.7  3.4   7.6  1.3  3.4        0       0
1           29  61   1.3  64.4  4.1   7.6  1.0  3.9        0       0
2           26  82  13.1  47.1  2.5   7.1  0.3  2.7        0       0
3           25  89   2.5  28.6  1.3   6.9  0.0  1.7        0       0
4           27  77   0.0  64.8  3.0  14.2  1.2  3.9        0       0
```

[32]:
```
y=df["FWI"]
```

[33]:
```
y.head()
```

[33]:
```
0    0.5
1    0.4
2    0.1
3    0.0
4    0.5
Name: FWI, dtype: float64
```

1.16 Now, Let's Standardize data of x

[34]:
```
from sklearn.preprocessing import MinMaxScaler
```

[35]:
```
scaler=MinMaxScaler()
```

[36]:
```
x_scaled=scaler.fit_transform(x)
```

[37]:
```
x_scaled
```

[37]:
```
array([[0.35      , 0.52173913, 0.        , …, 0.03437967, 0.        ,
        0.        ],
       [0.35      , 0.57971014, 0.07738095, …, 0.04185351, 0.        ,
        0.        ],
       [0.2       , 0.88405797, 0.7797619 , …, 0.02391629, 0.        ,
        0.        ],
       …,
```

```
       [0.25      , 0.95652174, 0.0297619 , …, 0.03437967, 0.        ,
        1.        ],
       [0.1       , 0.47826087, 0.00595238, …, 0.05979073, 0.        ,
        1.        ],
       [0.1       , 0.62318841, 0.01190476, …, 0.05530643, 0.        ,
        1.        ]])
```

[38]:
```
x_new=pd.DataFrame(x_scaled,columns=[x.columns])
```

[39]:
```
x_new.head()
```

[39]:

	Temperature	RH	Rain	FFMC	DMC	DC	ISI	BUI	Classes	region
0	0.35	0.521739	0.000000	0.550445	0.041411	0.003279	0.068421	0.034380	0.0	0.0
1	0.35	0.579710	0.077381	0.531157	0.052147	0.003279	0.052632	0.041854	0.0	0.0
2	0.20	0.884058	0.779762	0.274481	0.027607	0.000937	0.015789	0.023916	0.0	0.0
3	0.15	0.985507	0.148810	0.000000	0.009202	0.000000	0.000000	0.008969	0.0	0.0
4	0.25	0.811594	0.000000	0.537092	0.035276	0.034192	0.063158	0.041854	0.0	0.0

[40]:
```
y
```

[40]:
```
0      0.5
1      0.4
2      0.1
3      0.0
4      0.5
      …
239    6.5
240    0.0
241    0.2
242    0.7
243    0.5
Name: FWI, Length: 243, dtype: float64
```

[41]:
```
import pickle
pickle.dump(scaler,open("scaler.pkl","wb"))
```

1.17 Split the Data and apply model

[42]:
```
from sklearn.linear_model import Ridge,LinearRegression,Lasso,ElasticNet
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
modelss=[LinearRegression,Ridge,Lasso,ElasticNet]
```

[43]:
```
x_train,x_test,y_train,y_test=train_test_split(x_new,y,test_size=0.25)
for j in modelss:
    model=j()
    model.fit(x_train,y_train)
    y_pred=model.predict(x_test)
    r2=r2_score(y_test,y_pred)
    print(str(j).split(".")[-1][0:-2]+" : "+str(r2))
    print()
```

```
LinearRegression : 0.9817378870890728

Ridge : 0.9491385560970339

Lasso : 0.36095693863486566

ElasticNet : 0.3420402414555741
```

1.17.1 Observation:

- LinearRegression and Ridge gave better predictions

[ ]:
```

```