3.0-Model Training By Virat Tiwari

December 8, 2023

1 Model Training of Algerian Forest Fire By Virat Tiwari

[31]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

[32]:
```
df=pd.read_csv('Algerian_forest_fire_dataset_cleaned.csv')
```

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

[33]:

	day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	Classes	Region
0	1	6	2012	29	57	18	0.0	65.7	3.4	7.6	1.3	3.4	0.5	not fire	0
1	2	6	2012	29	61	13	1.3	64.4	4.1	7.6	1.0	3.9	0.4	not fire	0
2	3	6	2012	26	82	22	13.1	47.1	2.5	7.1	0.3	2.7	0.1	not fire	0
3	4	6	2012	25	89	13	2.5	28.6	1.3	6.9	0.0	1.7	0.0	not fire	0
4	5	6	2012	27	77	16	0.0	64.8	3.0	14.2	1.2	3.9	0.5	not fire	0

[34]:
```
df.columns
```

[34]:
```
Index(['day', 'month', 'year', 'Temperature', 'RH', 'Ws', 'Rain', 'FFMC',
       'DMC', 'DC', 'ISI', 'BUI', 'FWI', 'Classes', 'Region'],
      dtype='object')
```

[35]:
```
##drop month,day and yyear
df.drop(['day','month','year'],axis=1,inplace=True)
```

[36]:
```
df.head()
```

[36]:
```
    Temperature  RH  Ws  Rain  FFMC  DMC    DC  ISI  BUI  FWI      Classes  \
0            29  57  18   0.0  65.7  3.4   7.6  1.3  3.4  0.5  not fire
1            29  61  13   1.3  64.4  4.1   7.6  1.0  3.9  0.4  not fire
2            26  82  22  13.1  47.1  2.5   7.1  0.3  2.7  0.1  not fire
3            25  89  13   2.5  28.6  1.3   6.9  0.0  1.7  0.0  not fire
4            27  77  16   0.0  64.8  3.0  14.2  1.2  3.9  0.5  not fire

   Region
0       0
1       0
2       0
3       0
4       0
```

[37]:
```
df['Classes'].value_counts()
```

[37]:
```
fire            131
not fire        101
fire              4
fire              2
not fire          2
not fire          1
not fire          1
not fire          1
Name: Classes, dtype: int64
```

[38]:
```
## Encoding
df['Classes']=np.where(df['Classes'].str.contains("not fire"),0,1)
```

[39]:
```
df.tail()
```

[39]:
```
     Temperature  RH  Ws  Rain  FFMC   DMC    DC  ISI   BUI  FWI  Classes  \
238           30  65  14   0.0  85.4  16.0  44.5  4.5  16.9  6.5        1
239           28  87  15   4.4  41.1   6.5   8.0  0.1   6.2  0.0        0
240           27  87  29   0.5  45.9   3.5   7.9  0.4   3.4  0.2        0
241           24  54  18   0.1  79.7   4.3  15.2  1.7   5.1  0.7        0
242           24  64  15   0.2  67.3   3.8  16.5  1.2   4.8  0.5        0

     Region
238       1
239       1
240       1
241       1
242       1
```

[40]:
```
df['Classes'].value_counts()
```

[40]:
```
1    137
0    106
Name: Classes, dtype: int64
```

[41]:
```
## Independent And dependent features
X=df.drop('FWI',axis=1)
y=df['FWI']
```

[42]:
```
X.head()
```

[42]:

	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	Classes	Region
0	29	57	18	0.0	65.7	3.4	7.6	1.3	3.4	0	0
1	29	61	13	1.3	64.4	4.1	7.6	1.0	3.9	0	0
2	26	82	22	13.1	47.1	2.5	7.1	0.3	2.7	0	0
3	25	89	13	2.5	28.6	1.3	6.9	0.0	1.7	0	0
4	27	77	16	0.0	64.8	3.0	14.2	1.2	3.9	0	0

[43]:
```
y
```

[43]:
```
0      0.5
1      0.4
2      0.1
3      0.0
4      0.5
      …
238    6.5
239    0.0
240    0.2
241    0.7
242    0.5
Name: FWI, Length: 243, dtype: float64
```

[44]:
```
#Train Test Split
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.
 ↪25,random_state=42)
```

[45]:
```
X_train.shape,X_test.shape
```

[45]:
```
((182, 11), (61, 11))
```

[46]:
```
## Feature Selection based on correlaltion
X_train.corr()
```

[46]:

	Temperature	RH	Ws	Rain	FFMC	DMC	\
Temperature	1.000000	-0.656095	-0.305977	-0.317512	0.694768	0.498173	
RH	-0.656095	1.000000	0.225736	0.241656	-0.653023	-0.414601	

```
Ws             -0.305977  0.225736  1.000000  0.251932 -0.190076  0.000379
Rain           -0.317512  0.241656  0.251932  1.000000 -0.545491 -0.289754
FFMC            0.694768 -0.653023 -0.190076 -0.545491  1.000000  0.620807
DMC             0.498173 -0.414601  0.000379 -0.289754  0.620807  1.000000
DC              0.390684 -0.236078  0.096576 -0.302341  0.524101  0.868647
ISI             0.629848 -0.717804 -0.023558 -0.345707  0.750799  0.685656
BUI             0.473609 -0.362317  0.035633 -0.300964  0.607210  0.983175
Classes         0.542141 -0.456876 -0.082570 -0.369357  0.781259  0.617273
Region          0.254549 -0.394665 -0.199969 -0.059022  0.249514  0.212582

                   DC       ISI       BUI   Classes    Region
Temperature  0.390684  0.629848  0.473609  0.542141  0.254549
RH          -0.236078 -0.717804 -0.362317 -0.456876 -0.394665
Ws           0.096576 -0.023558  0.035633 -0.082570 -0.199969
Rain        -0.302341 -0.345707 -0.300964 -0.369357 -0.059022
FFMC         0.524101  0.750799  0.607210  0.781259  0.249514
DMC          0.868647  0.685656  0.983175  0.617273  0.212582
DC           1.000000  0.513701  0.942414  0.543581 -0.060838
ISI          0.513701  1.000000  0.643818  0.742977  0.296441
BUI          0.942414  0.643818  1.000000  0.612239  0.114897
Classes      0.543581  0.742977  0.612239  1.000000  0.188837
Region      -0.060838  0.296441  0.114897  0.188837  1.000000
```

1.1 Feature Selection

[47]:
```
## Check for multicollinearity
plt.figure(figsize=(12,10))
corr=X_train.corr()
sns.heatmap(corr,annot=True)
```

[47]:
```
<AxesSubplot: >
```

[48]:
```
X_train.corr()
```

[48]:

	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	Classes	Region
Temperature	1.000000	-0.656095	-0.305977	-0.317512	0.694768	0.498173	0.390684	0.629848	0.473609	0.542141	0.254549
RH	-0.656095	1.000000	0.225736	0.241656	-0.653023	-0.414601	-0.236078	-0.717804	-0.362317	-0.456876	-0.394665
Ws	-0.305977	0.225736	1.000000	0.251932	-0.190076	0.000379	0.096576	-0.023558	0.035633	-0.082570	-0.199969
Rain	-0.317512	0.241656	0.251932	1.000000	-0.545491	-0.289754	-0.302341	-0.345707	-0.300964	-0.369357	-0.059022
FFMC	0.694768	-0.653023	-0.190076	-0.545491	1.000000	0.620807	0.524101	0.750799	0.607210	0.781259	0.249514
DMC	0.498173	-0.414601	0.000379	-0.289754	0.620807	1.000000	0.868647	0.685656	0.983175	0.617273	0.212582
DC	0.390684	-0.236078	0.096576	-0.302341	0.524101	0.868647	1.000000	0.512368	0.942414	0.543581	-0.060838
ISI	0.629848	-0.717804	-0.023558	-0.345707	0.750799	0.685656	0.512368	1.000000	0.643818	0.742977	0.296441
BUI	0.473609	-0.362317	0.035633	-0.300964	0.607210	0.983175	0.942414	0.643818	1.000000	0.612239	0.114897
Classes	0.542141	-0.456876	-0.082570	-0.369357	0.781259	0.617273	0.543581	0.742977	0.612239	1.000000	0.188837
Region	0.254549	-0.394665	-0.199969	-0.059022	0.249514	0.212582	-0.060838	0.296441	0.114897	0.188837	1.000000

```
Temperature  0.390684  0.629848  0.473609  0.542141  0.254549
RH          -0.236078 -0.717804 -0.362317 -0.456876 -0.394665
Ws           0.096576 -0.023558  0.035633 -0.082570 -0.199969
Rain        -0.302341 -0.345707 -0.300964 -0.369357 -0.059022
FFMC         0.524101  0.750799  0.607210  0.781259  0.249514
DMC          0.868647  0.685656  0.983175  0.617273  0.212582
DC           1.000000  0.513701  0.942414  0.543581 -0.060838
ISI          0.513701  1.000000  0.643818  0.742977  0.296441
BUI          0.942414  0.643818  1.000000  0.612239  0.114897
Classes      0.543581  0.742977  0.612239  1.000000  0.188837
Region      -0.060838  0.296441  0.114897  0.188837  1.000000
```

[49]:
```
def correlation(dataset, threshold):
    col_corr = set()
    corr_matrix = dataset.corr()
    for i in range(len(corr_matrix.columns)):
        for j in range(i):
            if abs(corr_matrix.iloc[i, j]) > threshold:
                colname = corr_matrix.columns[i]
                col_corr.add(colname)
    return col_corr
```

[50]:
```
## threshold--Domain expertise
corr_features=correlation(X_train,0.85)
```

[51]:
```
corr_features
```

[51]:
```
{'BUI', 'DC'}
```

[52]:
```
## drop features when correlation is more than 0.85
X_train.drop(corr_features,axis=1,inplace=True)
X_test.drop(corr_features,axis=1,inplace=True)
X_train.shape,X_test.shape
```

[52]:
```
((182, 9), (61, 9))
```

1.2 Feature Scaling Or Standardization

[53]:
```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
X_train_scaled=scaler.fit_transform(X_train)
X_test_scaled=scaler.transform(X_test)
```

[54]:
```
X_train_scaled
```

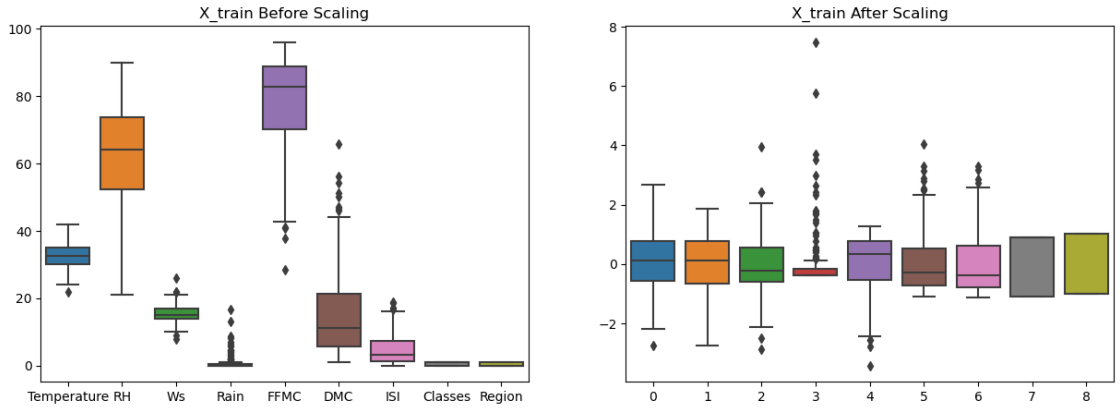
[54]:
```
array([[-0.84284248,  0.78307967,  1.29972026, …, -0.62963326,
        -1.10431526, -0.98907071],
       [-0.30175842,  0.64950844, -0.59874754, …, -0.93058524,
        -1.10431526,  1.01105006],
       [ 2.13311985, -2.08870172, -0.21905398, …,  2.7271388 ,
         0.90553851,  1.01105006],
       …,
       [-1.9250106 ,  0.9166509 ,  0.54033314, …, -1.06948615,
        -1.10431526, -0.98907071],
       [ 0.50986767, -0.21870454,  0.16063958, …,  0.5973248 ,
         0.90553851,  1.01105006],
       [-0.57230045,  0.98343651,  2.05910739, …, -0.86113478,
        -1.10431526, -0.98907071]])
```

1.3 Box Plots To understand Effect Of Standard Scaler

[55]:
```
plt.subplots(figsize=(15, 5))
plt.subplot(1, 2, 1)
sns.boxplot(data=X_train)
plt.title('X_train Before Scaling')
plt.subplot(1, 2, 2)
sns.boxplot(data=X_train_scaled)
plt.title('X_train After Scaling')
```

```
/tmp/ipykernel_97/160744393.py:2: MatplotlibDeprecationWarning: Auto-removal of
overlapping axes is deprecated since 3.6 and will be removed two minor releases
later; explicitly call ax.remove() as needed.
  plt.subplot(1, 2, 1)
```

[55]:
```
Text(0.5, 1.0, 'X_train After Scaling')
```



1.4 Linear Regression Model

[56]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score
linreg=LinearRegression()
linreg.fit(X_train_scaled,y_train)
y_pred=linreg.predict(X_test_scaled)
mae=mean_absolute_error(y_test,y_pred)
score=r2_score(y_test,y_pred)
print("Mean absolute error", mae)
print("R2 Score", score)
```

```
Mean absolute error 0.5468236465249985
R2 Score 0.9847657384266951
```

1.5 Lasso Regression

[57]:
```
from sklearn.linear_model import Lasso
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score
lasso=Lasso()
lasso.fit(X_train_scaled,y_train)
y_pred=lasso.predict(X_test_scaled)
mae=mean_absolute_error(y_test,y_pred)
score=r2_score(y_test,y_pred)
print("Mean absolute error", mae)
print("R2 Score", score)
```

```
Mean absolute error 1.133175994914409
R2 Score 0.9492020263112388
```

1.6 Ridge Regression model

[58]:
```
from sklearn.linear_model import Ridge
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score
ridge=Ridge()
ridge.fit(X_train_scaled,y_train)
y_pred=ridge.predict(X_test_scaled)
mae=mean_absolute_error(y_test,y_pred)
score=r2_score(y_test,y_pred)
print("Mean absolute error", mae)
print("R2 Score", score)
```

```
Mean absolute error 0.5642305340105692
R2 Score 0.9842993364555513
```

1.7 Elasticnet Regression

[59]:
```
from sklearn.linear_model import ElasticNet
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score
elastic=ElasticNet()
elastic.fit(X_train_scaled,y_train)
y_pred=elastic.predict(X_test_scaled)
mae=mean_absolute_error(y_test,y_pred)
score=r2_score(y_test,y_pred)
print("Mean absolute error", mae)
print("R2 Score", score)
```

```
Mean absolute error 1.8822353634896005
R2 Score 0.8753460589519703
```

[60]:
```
import pickle
pickle.dump(scaler,open('scaler.pkl','wb'))
pickle.dump(ridge,open('ridge.pkl','wb'))
```

THANK YOU SO MUCH !!

YOURS VIRAT TIWARI :)