```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

```
df=pd.read_csv('/content/Algerian_forest_fires_cleaned_dataset.csv')
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 243 entries, 0 to 242
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: 28.6+ KB
```

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

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

```
X=df.drop('FWI',axis=1)
y=df['FWI']
```

```
#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)
```

```
X_train.corr()
```

	Temperature	RH	Ws	Rain	FFMC	DMC	DC	
Temperature	1.000000	-0.656095	-0.305977	-0.317512	0.694768	0.498173	0.390684	(
RH	-0.656095	1.000000	0.225736	0.241656	-0.653023	-0.414601	-0.236078	-(
Ws	-0.305977	0.225736	1.000000	0.251932	-0.190076	0.000379	0.096576	-(
Rain	-0.317512	0.241656	0.251932	1.000000	-0.545491	-0.289754	-0.302341	-(
FFMC	0.694768	-0.653023	-0.190076	-0.545491	1.000000	0.620807	0.524101	(
DMC	0.498173	-0.414601	0.000379	-0.289754	0.620807	1.000000	0.868647	(
DC	0.390684	-0.236078	0.096576	-0.302341	0.524101	0.868647	1.000000	(
ISI	0.629848	-0.717804	-0.023558	-0.345707	0.750799	0.685656	0.513701	'
BUI	0.473609	-0.362317	0.035633	-0.300964	0.607210	0.983175	0.942414	(
Classes	0.542141	-0.456876	-0.082570	-0.369357	0.781259	0.617273	0.543581	(
Region	0.254549	-0.394665	-0.199969	-0.059022	0.249514	0.212582	-0.060838	(

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

```
<Axes: >
```


```
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
```

```
## threshold--Domain expertise
corr_features=correlation(X_train,0.85)
```

```
## 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
```

```
((182, 9), (61, 9))
```

Feature Scaling or Standardization

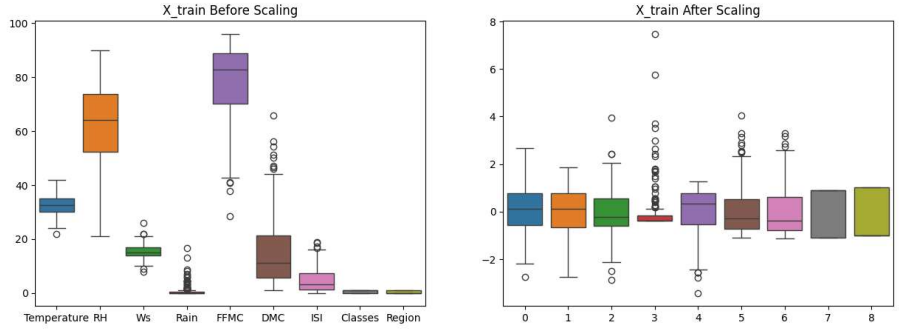
```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
X_train_scaled=scaler.fit_transform(X_train)
X_test_scaled=scaler.transform(X_test)
```

```
X_train_scaled
```

```
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]])
```

```
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')
```

```
<ipython-input-23-41fb1d7ced73>:2: MatplotlibDeprecationWarning: Auto-removal of overlap
  plt.subplot(1, 2, 1)
Text(0.5, 1.0, 'X_train After Scaling')
```



Start coding or generate with AI.

Start coding or generate with AI.

Linear Regression Model

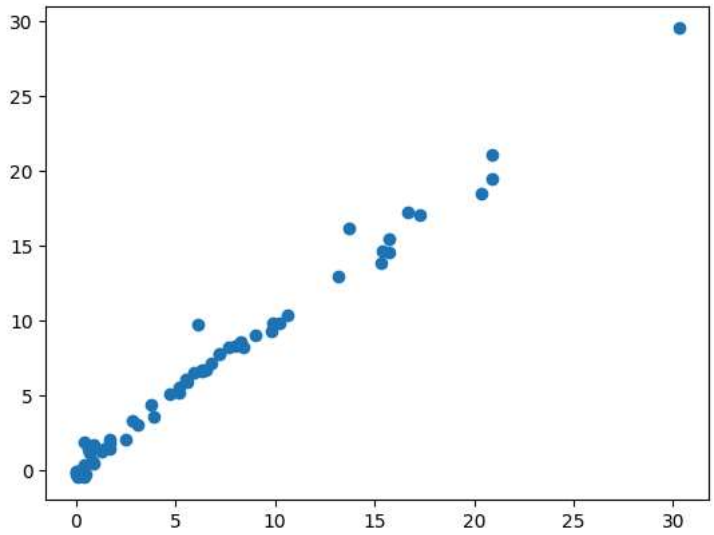
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score
```

```
linreg=LinearRegression()
linreg.fit(X_train_scaled, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
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)

plt.scatter(y_test, y_pred)
```

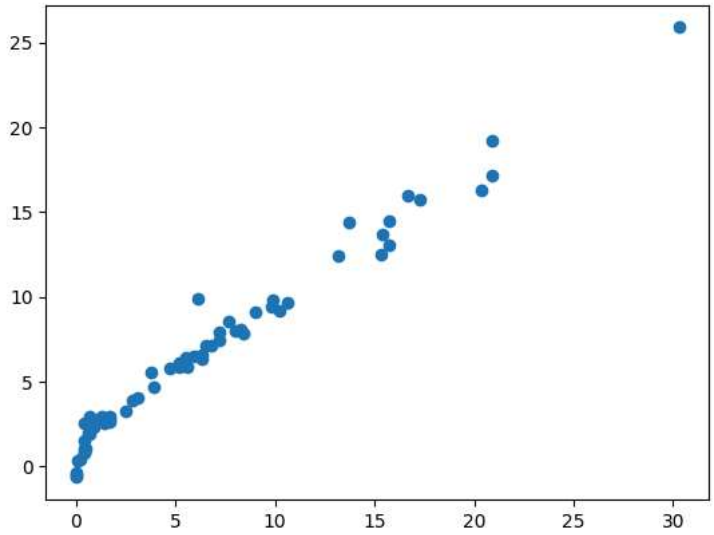
```
Mean absolute error 0.5468236465249986
R2 Score 0.9847657384266951
<matplotlib.collections.PathCollection at 0x7bc753a14940>
```



Lasso Regression

```
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)

plt.scatter(y_test,y_pred)
```

```
Mean absolute error 1.133175994914409
R2 Score 0.9492020263112388
<matplotlib.collections.PathCollection at 0x7bc757b2f0a0>
```



Cross Validation Lasso

```
from sklearn.linear_model import LassoCV
lassocv = LassoCV(cv=5)
lassocv.fit(X_train_scaled, y_train)
```

```
LassoCV
LassoCV(cv=5)
```

```
lassocv.alpha_
```

```
0.05725391318234411
```

```
lassocv.alphas_
```

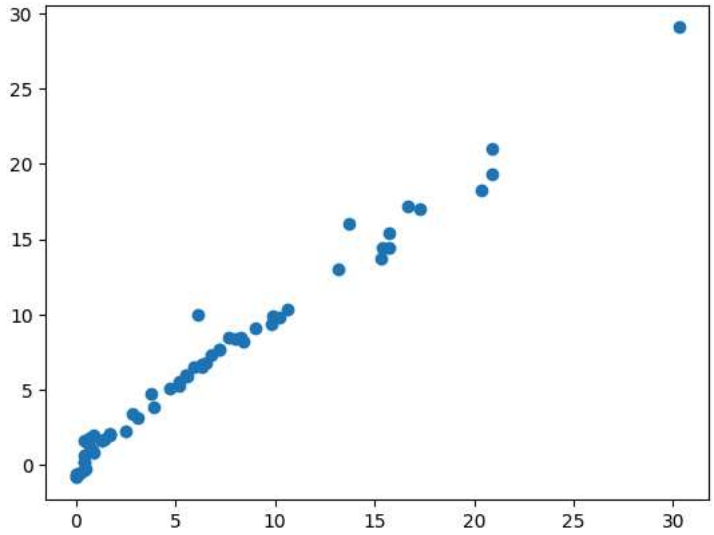
```
array([7.05853002, 6.58280872, 6.13914944, 5.72539132, 5.33951911,
       4.97965339, 4.64404142, 4.33104857, 4.03915039, 3.76692517,
       3.51304702, 3.27627941, 3.05546914, 2.84954075, 2.65749124,
       2.47838523, 2.31135036, 2.15557308, 2.01029467, 1.87480753,
       1.74845178, 1.63061198, 1.52071419, 1.41822315, 1.32263965,
       1.23349817, 1.15036452, 1.0728338 , 1.00052839, 0.93309613,
       0.87020857, 0.81155943, 0.75686304, 0.705853  , 0.65828087,
       0.61391494, 0.57253913, 0.53395191, 0.49796534, 0.46440414,
       0.43310486, 0.40391504, 0.37669252, 0.3513047 , 0.32762794,
       0.30554691, 0.28495408, 0.26574912, 0.24783852, 0.23113504,
       0.21555731, 0.20102947, 0.18748075, 0.17484518, 0.1630612 ,
       0.15207142, 0.14182231, 0.13226397, 0.12334982, 0.11503645,
       0.10728338, 0.10005284, 0.09330961, 0.08702086, 0.08115594,
       0.0756863 , 0.0705853 , 0.06582809, 0.06139149, 0.05725391,
       0.05339519, 0.04979653, 0.04644041, 0.04331049, 0.0403915 ,
       0.03766925, 0.03513047, 0.03276279, 0.03055469, 0.02849541,
       0.02657491, 0.02478385, 0.0231135 , 0.02155573, 0.02010295,
       0.01874808, 0.01748452, 0.01630612, 0.01520714, 0.01418223,
       0.0132264 , 0.01233498, 0.01150365, 0.01072834, 0.01000528,
       0.00933096, 0.00870209, 0.00811559, 0.00756863, 0.00705853])
```

```
lassocv.mse_path_
```

```
[ 1.06616655,  1.93941688,  6.60120289,  2.58826543,  0.91600498],
[ 1.05471212,  1.91540122,  6.66074506,  2.53939631,  0.91492536],
[ 1.04483316,  1.89395167,  6.72040081,  2.49354558,  0.91475751],
[ 1.03631885,  1.87477186,  6.77985049,  2.45183158,  0.91533073],
[ 1.02898619,  1.85760147,  6.8386118 ,  2.41402473,  0.91650002],
[ 1.02267637,  1.84221172,  6.89546904,  2.37952566,  0.91817465],
[ 1.0172516 ,  1.81986019,  6.95182997,  2.34943959,  0.92100746],
[ 1.01259234,  1.7874912 ,  7.00657253,  2.30905785,  0.91090128],
[ 0.99291676,  1.75813753,  7.05952508,  2.26689771,  0.88812743],
[ 0.96711245,  1.73133215,  7.11055395,  2.22965179,  0.86893338],
[ 0.94404465,  1.70754321,  7.15957739,  2.19646   ,  0.85251259],
[ 0.91746069,  1.68586828,  7.21115863,  2.16644165,  0.83841802],
[ 0.89121876,  1.66666838,  7.26823916,  2.14003416,  0.82646203],
[ 0.86783937,  1.64937312,  7.32193772,  2.11642121,  0.81629395],
[ 0.84703112,  1.6337788 ,  7.37194387,  2.09528441,  0.80766048],
[ 0.82845196,  1.619701  ,  7.42070575,  2.07634166,  0.80034774],
[ 0.81184328,  1.6069769 ,  7.46783924,  2.05934486,  0.79417047],
[ 0.79697877,  1.59523036,  7.51171241,  2.04379341,  0.78898574],
[ 0.78366252,  1.58481658,  7.5533042 ,  2.03007893,  0.78514158],
[ 0.77340653,  1.57536934,  7.59178479,  2.01773193,  0.78410497],
[ 0.76437368,  1.56730639,  7.62890427,  2.00633629,  0.78327866],
[ 0.75641103,  1.56014926,  7.66385201,  1.99569195,  0.78309295],
[ 0.74929762,  1.55377904,  7.69675973,  1.98581272,  0.78325254],
[ 0.7431075 ,  1.54808751,  7.72772336,  1.97708583,  0.78348718],
[ 0.73764056,  1.5428574 ,  7.75701245,  1.9690422 ,  0.78415382],
[ 0.73271889,  1.5383076 ,  7.78098988,  1.96195515,  0.78479522],
[ 0.72844826,  1.53422868,  7.80009362,  1.95555728,  0.78577592],
[ 0.72457927,  1.53042136,  7.81782859,  1.94960372,  0.78686385],
[ 0.72121402,  1.5271394 ,  7.83584096,  1.94420011,  0.78783843],
[ 0.71854269,  1.52403047,  7.8521645 ,  1.93945512,  0.78886011],
[ 0.71624922,  1.52137747,  7.86797141,  1.93532188,  0.79008917],
[ 0.71419505,  1.51882628,  7.8824946 ,  1.93156393,  0.7910736 ],
[ 0.71283686,  1.51649634,  7.89597341,  1.92813104,  0.79328236],
[ 0.7117556 ,  1.51454548,  7.90862683,  1.92492966,  0.7959553 ],
[ 0.71078691,  1.5128162 ,  7.92077339,  1.92207644,  0.79869912],
[ 0.71003406,  1.51137977,  7.93211766,  1.9195065 ,  0.80158876],
[ 0.7094272 ,  1.51017923,  7.94254787,  1.9171673 ,  0.80451499],
[ 0.70893209,  1.50910355,  7.95231005,  1.91555613,  0.80717091],
[ 0.70847636,  1.50819995,  7.96151575,  1.914521  ,  0.8098638 ],
[ 0.70814046,  1.50740984,  7.97034636,  1.91358558,  0.81227152],
[ 0.70789298,  1.5065737 ,  7.97838619,  1.91277526,  0.81468439],
[ 0.70770357,  1.50591279,  7.98587605,  1.9120262 ,  0.8170304 ],
[ 0.70752166,  1.50536216,  7.99241057,  1.91138883,  0.81925406],
[ 0.70734296,  1.50487616,  7.99849196,  1.91084915,  0.82119901],
[ 0.70724307,  1.50444309,  8.00451482,  1.91033293,  0.82327046],
[ 0.70719344,  1.50391791,  8.01011355,  1.9098903 ,  0.8250587 ],
[ 0.70714379,  1.50342997,  8.01481494,  1.90951275,  0.826765  ],
[ 0.70711086,  1.50300182,  8.01992921,  1.90919915,  0.82842365]])
```

```
y_pred=lassocv.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)

plt.scatter(y_test,y_pred)
```

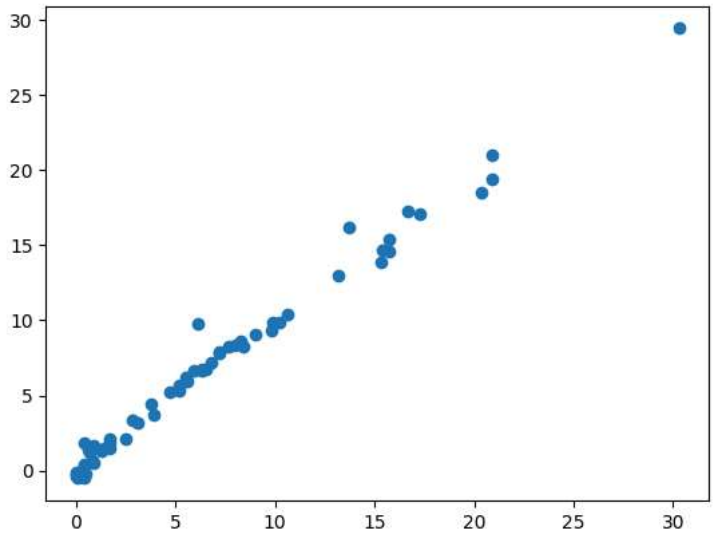
```
Mean absolute error 0.6199701158263433
R2 Score 0.9820946715928275
<matplotlib.collections.PathCollection at 0x7bc7532edde0>
```



Ridge Regression model

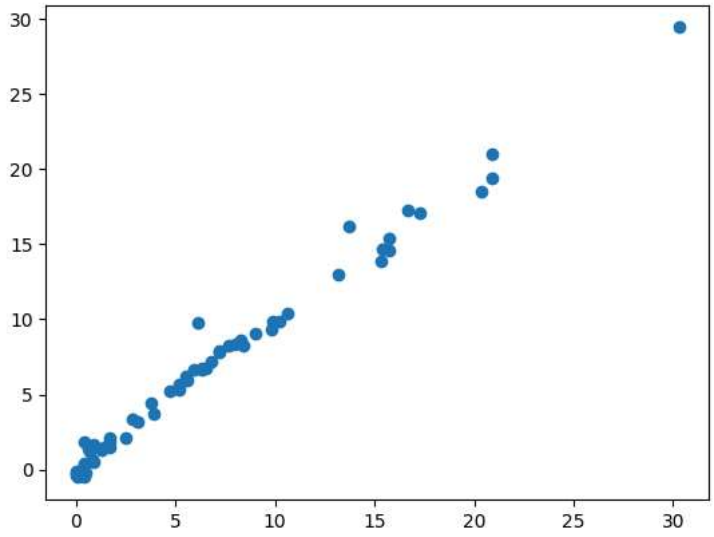
```
from sklearn.linear_model import Ridge
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score

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)

plt.scatter(y_test,y_pred)
```

```
Mean absolute error 0.5642305340105692
R2 Score 0.9842993364555513
<matplotlib.collections.PathCollection at 0x7bc7536155a0>
```



```
from sklearn.linear_model import RidgeCV
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score

ridgecv=RidgeCV(cv=5)
ridgecv.fit(X_train_scaled,y_train)

y_pred=ridgecv.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)

plt.scatter(y_test,y_pred)
```

```
Mean absolute error 0.5642305340105692
R2 Score 0.9842993364555513
<matplotlib.collections.PathCollection at 0x7bc757a769e0>
```



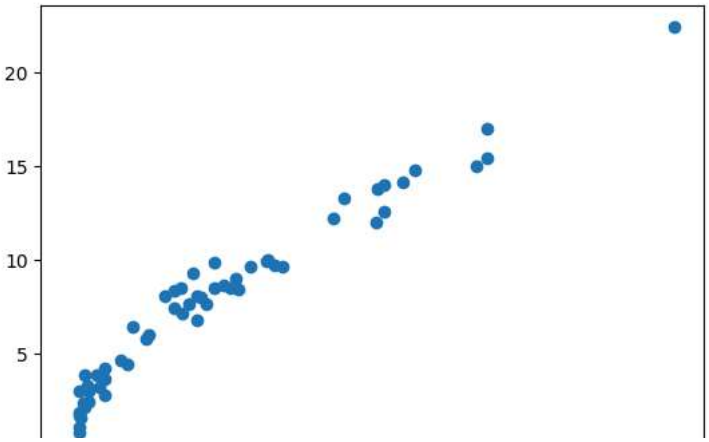

```
ridgecv.get_params()
```

```
{'alpha_per_target': False,
 'alphas': (0.1, 1.0, 10.0),
 'cv': 5,
 'fit_intercept': True,
 'gcv_mode': None,
 'scoring': None,
 'store_cv_values': False}
```

Elasticnet Regression

```
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)

plt.scatter(y_test,y_pred)
```

```
Mean absolute error 1.8822353634896005
R2 Score 0.8753460589519703
<matplotlib.collections.PathCollection at 0x7bc755aa3430>
```




```
from sklearn.linear_model import ElasticNetCV
elasticcv=ElasticNetCV(cv=5)
elasticcv.fit(X_train_scaled,y_train)
y_pred=elasticcv.predict(X_test_scaled)
plt.scatter(y_test,y_pred)
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 0.6575946731430898
R2 Score 0.9814217587854941
```