**import** numpy **as** np

**import** pandas **as** pd

**from** sklearn.model\_selection **import** train\_test\_split

**from** sklearn.preprocessing **import** StandardScaler

**from** sklearn.linear\_model **import** LinearRegression,LogisticRegression

**from** sklearn.neural\_network **import** MLPClassifier,MLPRegressor

**from** sklearn.tree **import** DecisionTreeRegressor **as** dtr

**from** sklearn.ensemble **import** RandomForestRegressor

**from** sklearn.metrics **import** mean\_squared\_error **as** mse

**from** sklearn.metrics **import** mean\_absolute\_error **as** mae

**from** sklearn.metrics **import** r2\_score

**import** warnings

warnings**.**filterwarnings(action**=**'ignore')

In [2]:

df**=**pd**.**read\_csv('/content/forestfires.csv')

In [58]:

df

Out[58]:

|  | **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 | 0.00 |
| **1** | 7 | 4 | oct | tue | 90.6 | 35.4 | 669.1 | 6.7 | 18.0 | 33 | 0.9 | 0.0 | 0.00 |
| **2** | 7 | 4 | oct | sat | 90.6 | 43.7 | 686.9 | 6.7 | 14.6 | 33 | 1.3 | 0.0 | 0.00 |
| **3** | 8 | 6 | mar | fri | 91.7 | 33.3 | 77.5 | 9.0 | 8.3 | 97 | 4.0 | 0.2 | 0.00 |
| **4** | 8 | 6 | mar | sun | 89.3 | 51.3 | 102.2 | 9.6 | 11.4 | 99 | 1.8 | 0.0 | 0.00 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **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 | 0.00 |
| **516** | 6 | 3 | nov | tue | 79.5 | 3.0 | 106.7 | 1.1 | 11.8 | 31 | 4.5 | 0.0 | 0.00 |

517 rows × 13 columns

In [59]:

df**.**info()

RangeIndex: 517 entries, 0 to 516

Data columns (total 13 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 X 517 non-null int64

1 Y 517 non-null int64

2 month 517 non-null object

3 day 517 non-null object

4 FFMC 517 non-null float64

5 DMC 517 non-null float64

6 DC 517 non-null float64

7 ISI 517 non-null float64

8 temp 517 non-null float64

9 RH 517 non-null int64

10 wind 517 non-null float64

11 rain 517 non-null float64

12 area 517 non-null float64

dtypes: float64(8), int64(3), object(2)

memory usage: 52.6+ KB

In [60]:

df['month']**.**unique()

Out[60]:

array(['mar', 'oct', 'aug', 'sep', 'apr', 'jun', 'jul', 'feb', 'jan',

'dec', 'may', 'nov'], dtype=object)

In [61]:

df['day']**.**unique()

Out[61]:

array(['fri', 'tue', 'sat', 'sun', 'mon', 'wed', 'thu'], dtype=object)

In [62]:

df**.**head(10)

Out[62]:

|  | **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 |
| **5** | 8 | 6 | aug | sun | 92.3 | 85.3 | 488.0 | 14.7 | 22.2 | 29 | 5.4 | 0.0 | 0.0 |
| **6** | 8 | 6 | aug | mon | 92.3 | 88.9 | 495.6 | 8.5 | 24.1 | 27 | 3.1 | 0.0 | 0.0 |
| **7** | 8 | 6 | aug | mon | 91.5 | 145.4 | 608.2 | 10.7 | 8.0 | 86 | 2.2 | 0.0 | 0.0 |
| **8** | 8 | 6 | sep | tue | 91.0 | 129.5 | 692.6 | 7.0 | 13.1 | 63 | 5.4 | 0.0 | 0.0 |
| **9** | 7 | 5 | sep | sat | 92.5 | 88.0 | 698.6 | 7.1 | 22.8 | 40 | 4.0 | 0.0 | 0.0 |

**Preprocessing**

Text data included with the numeric data(Month & Days). So we need to encode that in some numeric form before splitting the train test data.

In [63]:

**def** ordinal\_encoding(df,column,ordering):

df**=**df**.**copy()

df[column]**=**df[column]**.**apply(**lambda** x: ordering**.**index(x))

**return** df

In [64]:

**def** preprocessing(df,task):

df**=**df**.**copy()

df**=**ordinal\_encoding(

df,

column**=**'month',

ordering**=**[

'jan', 'feb', 'mar', 'apr', 'may', 'jun', 'jul', 'aug', 'sep',

'oct', 'nov', 'dec'

]

)

df**=**ordinal\_encoding(

df,

column**=**'day',

ordering**=**['mon', 'tue', 'wed', 'thu', 'fri', 'sat', 'sun']

)

**if** task**==**'Regression':

Y**=**df['area']

**elif** task**==**'Classification':

Y**=**df['area']**.**apply(**lambda** x: 1 **if** x**>**0 **else** 0)

X**=**df**.**drop('area',axis**=**1)

X\_train,X\_test,Y\_train,Y\_test**=**train\_test\_split(X,Y,train\_size**=**0.65,shuffle**=True**,random\_state**=**1)

scaler**=**StandardScaler()

scaler**.**fit(X\_train)

X\_train**=**pd**.**DataFrame(scaler**.**transform(X\_train),columns**=**X**.**columns)

X\_test**=**pd**.**DataFrame(scaler**.**transform(X\_test),columns**=**X**.**columns)

**return** X\_train,X\_test,Y\_train,Y\_test

**Splitting & Testing Models:**

In [65]:

X\_train,X\_test,Y\_train,Y\_test**=**preprocessing(df,task**=**'Regression')

In [66]:

X\_train**.**head()

Out[66]:

|  | **X** | **Y** | **month** | **day** | **FFMC** | **DMC** | **DC** | **ISI** | **temp** | **RH** | **wind** | **rain** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 1.460144 | 1.330887 | 0.226770 | -0.599195 | -0.750425 | -0.262938 | 0.268405 | -0.471881 | -0.247391 | 0.426174 | 0.058891 | -0.079936 |
| **1** | 0.603934 | -0.246461 | 0.226770 | -0.109342 | 0.748288 | 0.383050 | 0.145783 | 0.306640 | 0.240778 | -0.140559 | 0.058891 | -0.079936 |
| **2** | -0.252276 | 0.542213 | -1.913533 | 0.380511 | 0.201882 | -1.156059 | -1.850008 | 0.019817 | -0.281057 | -1.085114 | 0.348181 | -0.079936 |
| **3** | -1.108486 | 0.542213 | 0.226770 | -1.089047 | 0.264329 | 0.709954 | 0.461893 | 1.105648 | 0.223944 | 0.237263 | 0.058891 | -0.079936 |
| **4** | 1.460144 | 1.330887 | -1.913533 | 0.380511 | 0.201882 | -1.116955 | -1.836870 | -0.226032 | -0.247391 | -1.211054 | 0.868904 | -0.079936 |

In [15]:

X\_test**.**head()

Out[15]:

|  | **X** | **Y** | **month** | **day** | **FFMC** | **DMC** | **DC** | **ISI** | **temp** | **RH** | **wind** | **rain** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | -1.108486 | -1.823808 | 0.226770 | -1.089047 | 0.264329 | 0.709954 | 0.461893 | 1.105648 | 0.493278 | 0.803996 | -0.461831 | -0.079936 |
| **1** | 0.603934 | 0.542213 | 0.226770 | 0.870363 | -0.032292 | -0.161269 | 0.326531 | -0.000671 | -0.701892 | 0.992907 | 1.100336 | -0.079936 |
| **2** | -0.252276 | 1.330887 | 0.654831 | -0.109342 | 0.514114 | -0.411530 | 0.569386 | 1.843194 | -0.213724 | -0.077588 | -0.461831 | -0.079936 |
| **3** | -0.680381 | -0.246461 | 0.654831 | 0.380511 | 0.451668 | 0.531643 | 0.683647 | 1.023699 | -0.045390 | 0.363204 | -0.172541 | -0.079936 |
| **4** | 1.032039 | -0.246461 | 0.654831 | -0.599195 | -0.047903 | -0.380248 | 0.770438 | -0.553830 | -0.584058 | 0.866966 | 0.348181 | -0.079936 |

In [16]:

Y\_train**.**head()

Out[16]:

171 2.69

161 1.90

69 0.00

272 3.09

91 0.00

Name: area, dtype: float64

In [17]:

Y\_test**.**head()

Out[17]:

270 0.52

90 0.00

133 0.00

221 35.88

224 37.71

Name: area, dtype: float64

**Linear Regression**

In [67]:

*#The maximum val. of R^2 can be 1.0 that signifies that Linear Regression is working really well.*

*#Here R^2 score is 0.02051 which is really low signifying that LinearRegression is not doing well.*

linear\_reg\_model**=**LinearRegression()

linear\_reg\_model**.**fit(X\_train,Y\_train)

print("Performance of Linear Regression R^2 metric {:.5f}"**.**format(linear\_reg\_model**.**score(X\_train,Y\_train)))

Performance of Linear Regression R^2 metric 0.02051

**MLP Regressor Model**

In [68]:

*# We can see the MLP Regressor Model Works better than Linear Regression,however its works horrible as well.*

mlp\_reg\_model**=**MLPRegressor()**.**fit(X\_train,Y\_train)

print("Performance of MLP Regressor Model R^2 metric {:.5f}"**.**format(mlp\_reg\_model**.**score(X\_train,Y\_train)))

Performance of MLP Regressor Model R^2 metric 0.06076

**Decision Tree Regressor**

In [79]:

*#R2 score negative not a good fit!*

reg **=** dtr(random\_state **=** 42)

reg**.**fit(X\_train, Y\_train)

Y\_pred **=** reg**.**predict(X\_test)

print("MSE =", mse(Y\_pred, Y\_test))

print("MAE =", mae(Y\_pred, Y\_test))

print("R2 Score =", r2\_score(Y\_pred, Y\_test))

MSE = 11084.975803867403

MAE = 25.682707182320442

R2 Score = -11.230687894960365

**Random Forest Regressor**

In [81]:

*#This works even worse than Decision Tree Regressor*

regr **=** RandomForestRegressor(max\_depth**=**2, random\_state**=**0, n\_estimators**=**100)

regr**.**fit(X\_train, Y\_train)

Y\_pred **=** regr**.**predict(X\_test)

print("MSE =", mse(Y\_pred, Y\_test))

print("MAE =", mae(Y\_pred, Y\_test))

print("R2 Score =", r2\_score(Y\_pred, Y\_test))

MSE = 9900.080101011921

MAE = 22.488793674759876

R2 Score = -588.768145917561

**Reaction after seeing the R^2 Score everytime getting less and less!!!!!!**

**Logistic Regression**

In [19]:

X\_train,X\_test,Y\_train,Y\_test**=**preprocessing(df,task**=**'Classification')

In [20]:

Y\_train

Out[20]:

171 1

161 1

69 0

272 1

91 0

..

129 0

144 1

72 0

235 1

37 0

Name: area, Length: 336, dtype: int64

In [21]:

log\_reg\_model**=**LogisticRegression()

log\_reg\_model**.**fit(X\_train,Y\_train)

print('Logistic Regression Accuracy, {:.5f}%'**.**format(log\_reg\_model**.**score(X\_test,Y\_test)**\***100))

Logistic Regression Accuracy, 53.59116%

**Neural Network Classifier**

In [54]:

nn\_classifier\_model**=**MLPClassifier(activation**=**'relu',hidden\_layer\_sizes**=**(16,16),n\_iter\_no\_change**=**100,solver**=**'adam')

nn\_classifier\_model**.**fit(X\_train,Y\_train)

Out[54]:

MLPClassifier(activation='relu', alpha=0.0001, batch\_size='auto', beta\_1=0.9,

beta\_2=0.999, early\_stopping=False, epsilon=1e-08,

hidden\_layer\_sizes=(16, 16), learning\_rate='constant',

learning\_rate\_init=0.001, max\_fun=15000, max\_iter=200,

momentum=0.9, n\_iter\_no\_change=100, nesterovs\_momentum=True,

power\_t=0.5, random\_state=None, shuffle=True, solver='adam',

tol=0.0001, validation\_fraction=0.1, verbose=False,

warm\_start=False)

In [55]:

print('MLP Classifier Accuracy, {:.5f}%'**.**format(nn\_classifier\_model**.**score(X\_test,Y\_test)**\***100))

MLP Classifier Accuracy, 57.45856%

**None of the models are working good, which is pretty frustrating lmao!After running all these various models we see that MLP Classifier has the best accuracy out of the all.**