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

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

**import** matplotlib.pyplot **as** plt

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

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

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

In [234]:

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

In [235]:

df**.**head()

Out[235]:

|  | **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 |

In [238]:

df['area']**.**value\_counts()

Out[238]:

0.00 247

1.94 3

3.71 2

0.90 2

1.95 2

...

1.26 1

2.44 1

2.03 1

2.21 1

24.24 1

Name: area, Length: 251, dtype: int64

In [236]:

df**.**count()

Out[236]:

X 517

Y 517

month 517

day 517

FFMC 517

DMC 517

DC 517

ISI 517

temp 517

RH 517

wind 517

rain 517

area 517

dtype: int64

In [244]:

df**=**df**.**drop(['X','Y','month','day'],axis**=**1)

In [245]:

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

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

**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**=False**,random\_state**=**0)

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

In [246]:

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

In [247]:

X\_train**.**head()

Out[247]:

|  | **FFMC** | **DMC** | **DC** | **ISI** | **temp** | **RH** | **wind** | **rain** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | -0.851337 | -1.497066 | -1.741817 | -0.786052 | -1.853885 | 0.542042 | 1.488535 | -0.073692 |
| **1** | 0.045241 | -1.290273 | 0.604909 | -0.449092 | -0.028985 | -0.591916 | -1.703078 | -0.073692 |
| **2** | 0.045241 | -1.103710 | 0.677581 | -0.449092 | -0.662114 | -0.591916 | -1.482967 | -0.073692 |
| **3** | 0.269386 | -1.337476 | -1.810406 | 0.035288 | -1.835263 | 3.439935 | 0.002784 | 3.463542 |
| **4** | -0.219657 | -0.932881 | -1.709563 | 0.161648 | -1.257999 | 3.565930 | -1.207828 | -0.073692 |

In [248]:

**import** seaborn **as** sns

plt**.**figure(figsize**=**(12,10))

corr**=**X\_train**.**corr()

sns**.**heatmap(corr, annot**=True**, cmap**=**plt**.**cm**.**CMRmap\_r )

plt**.**show()

In [249]:

**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

In [250]:

corr\_features**=**correlation(X\_train,0.7)

len(set(corr\_features))

Out[250]:

0

In [251]:

corr\_features

Out[251]:

set()

In [252]:

X\_train**.**drop(corr\_features,axis**=**1)**.**head()

Out[252]:

|  | **FFMC** | **DMC** | **DC** | **ISI** | **temp** | **RH** | **wind** | **rain** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | -0.851337 | -1.497066 | -1.741817 | -0.786052 | -1.853885 | 0.542042 | 1.488535 | -0.073692 |
| **1** | 0.045241 | -1.290273 | 0.604909 | -0.449092 | -0.028985 | -0.591916 | -1.703078 | -0.073692 |
| **2** | 0.045241 | -1.103710 | 0.677581 | -0.449092 | -0.662114 | -0.591916 | -1.482967 | -0.073692 |
| **3** | 0.269386 | -1.337476 | -1.810406 | 0.035288 | -1.835263 | 3.439935 | 0.002784 | 3.463542 |
| **4** | -0.219657 | -0.932881 | -1.709563 | 0.161648 | -1.257999 | 3.565930 | -1.207828 | -0.073692 |

In [253]:

X\_test**.**drop(corr\_features,axis**=**1)**.**head()

Out[253]:

|  | **FFMC** | **DMC** | **DC** | **ISI** | **temp** | **RH** | **wind** | **rain** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 0.249009 | 0.350581 | 0.992355 | -0.554392 | 0.213093 | 0.101059 | -0.987716 | -0.073692 |
| **1** | 0.249009 | 0.350581 | 0.992355 | -0.554392 | 0.902086 | -0.528918 | -0.987716 | -0.073692 |
| **2** | 0.249009 | 0.350581 | 0.992355 | -0.554392 | 0.846222 | -0.465920 | -0.987716 | -0.073692 |
| **3** | 0.249009 | 0.350581 | 0.992355 | -0.554392 | 0.417929 | -0.087934 | -1.207828 | -0.073692 |
| **4** | 0.249009 | 0.350581 | 0.992355 | -0.554392 | 0.213093 | 0.101059 | -0.987716 | -0.073692 |

In [254]:

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)

/usr/local/lib/python3.7/dist-packages/sklearn/neural\_network/\_multilayer\_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the optimization hasn't converged yet.

% self.max\_iter, ConvergenceWarning)

Out[254]:

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 [255]:

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

MLP Classifier Accuracy, 50.82873%

In [256]:

nn\_classifier\_model**.**predict\_proba(X\_test[:10])

Out[256]:

array([[0.57724435, 0.42275565],

[0.67336303, 0.32663697],

[0.66327232, 0.33672768],

[0.58399674, 0.41600326],

[0.57724435, 0.42275565],

[0.52085341, 0.47914659],

[0.43364496, 0.56635504],

[0.63576475, 0.36423525],

[0.76551969, 0.23448031],

[0.42865212, 0.57134788]])

In [257]:

nn\_classifier\_model**.**predict(X\_test[:10])

Out[257]:

array([0, 0, 0, 0, 0, 0, 1, 0, 0, 1])

In [258]:

X\_test[:10]

Out[258]:

|  | **FFMC** | **DMC** | **DC** | **ISI** | **temp** | **RH** | **wind** | **rain** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 0.249009 | 0.350581 | 0.992355 | -0.554392 | 0.213093 | 0.101059 | -0.987716 | -0.073692 |
| **1** | 0.249009 | 0.350581 | 0.992355 | -0.554392 | 0.902086 | -0.528918 | -0.987716 | -0.073692 |
| **2** | 0.249009 | 0.350581 | 0.992355 | -0.554392 | 0.846222 | -0.465920 | -0.987716 | -0.073692 |
| **3** | 0.249009 | 0.350581 | 0.992355 | -0.554392 | 0.417929 | -0.087934 | -1.207828 | -0.073692 |
| **4** | 0.249009 | 0.350581 | 0.992355 | -0.554392 | 0.213093 | 0.101059 | -0.987716 | -0.073692 |
| **5** | 0.310139 | 0.424757 | 1.018076 | -0.491212 | -0.457278 | 0.542042 | -0.987716 | -0.073692 |
| **6** | 0.228632 | 0.838342 | 1.168319 | -0.280612 | 0.455172 | -0.339925 | -1.207828 | -0.073692 |
| **7** | 0.228632 | 0.838342 | 1.168319 | -0.280612 | -0.420035 | 0.542042 | 0.277923 | -0.073692 |
| **8** | 0.228632 | 0.838342 | 1.168319 | -0.280612 | -1.109028 | 1.487007 | 0.498034 | -0.073692 |
| **9** | 0.228632 | 0.838342 | 1.168319 | -0.280612 | -0.252442 | 0.038061 | -0.492466 | -0.073692 |

In [219]:

Y\_test[:10]

Out[219]:

336 0

337 1

338 1

339 1

340 1

341 0

342 0

343 1

344 1

345 1

Name: area, dtype: int64

In [224]:

df\_test['area']**.**value\_counts()

Out[224]:

0.00 247

1.94 3

3.71 2

0.90 2

1.95 2

...

1.26 1

2.44 1

2.03 1

2.21 1

24.24 1

Name: area, Length: 251, dtype: int64

**Test to check:**

In [261]:

nn\_classifier\_model**.**predict(X\_test)

Out[261]:

array([0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0,

1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0,

0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0,

1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0,

1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1,

0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,

1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1,

1, 0, 0, 1, 0])

In [262]:

np**.**array(Y\_test)

Out[262]:

array([0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0,

0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0,

1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1,

0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1,

1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0,

0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1,

1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0,

0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0,

1, 1, 1, 0, 0])

In [263]:

Y\_pred**=**nn\_classifier\_model**.**predict(X\_test)

print('MLP Classifier Accuracy, {:.5f}%'**.**format(np**.**mean(Y\_pred**==**Y\_test)**\***100))

MLP Classifier Accuracy, 50.82873%