In [2]:
```
!pip install imblearn
```

```
Collecting imblearn
  Using cached imblearn-0.0-py2.py3-none-any.whl (1.9 kB)
Collecting imbalanced-learn
  Using cached imbalanced_learn-0.9.0-py3-none-any.whl (199 kB)
Requirement already satisfied: numpy>=1.14.6 in c:\users\lenovo\anaconda3\lib\s
ite-packages (from imbalanced-learn->imblearn) (1.20.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\lenovo\anaconda
3\lib\site-packages (from imbalanced-learn->imblearn) (2.1.0)
Requirement already satisfied: scikit-learn>=1.0.1 in c:\users\lenovo\anaconda3
\lib\site-packages (from imbalanced-learn->imblearn) (1.0.2)
Requirement already satisfied: scipy>=1.1.0 in c:\users\lenovo\anaconda3\lib\si
te-packages (from imbalanced-learn->imblearn) (1.6.2)
Requirement already satisfied: joblib>=0.11 in c:\users\lenovo\anaconda3\lib\si
te-packages (from imbalanced-learn->imblearn) (1.0.1)
Installing collected packages: imbalanced-learn, imblearn
Successfully installed imbalanced-learn-0.9.0 imblearn-0.0
```

In [3]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import SMOTE
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.wrappers.scikit_learn import KerasClassifier
from tensorflow.keras.layers import Dense,Dropout
from sklearn.metrics import accuracy_score,classification_report,confusion_matri
from keras.models import Sequential
from tensorflow.keras.optimizers import Adam,RMSprop
from sklearn.model_selection import GridSearchCV,KFold
```

In [4]:
```
forestfire_data = pd.read_csv('forestfires.csv')
forestfire_data
```

Out[4]:

	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	...	monthfeb	monthjan	mont
0	mar	fri	86.2	26.2	94.3	5.1	8.2	51	6.7	0.0	...	0	0	
1	oct	tue	90.6	35.4	669.1	6.7	18.0	33	0.9	0.0	...	0	0	
2	oct	sat	90.6	43.7	686.9	6.7	14.6	33	1.3	0.0	...	0	0	
3	mar	fri	91.7	33.3	77.5	9.0	8.3	97	4.0	0.2	...	0	0	
4	mar	sun	89.3	51.3	102.2	9.6	11.4	99	1.8	0.0	...	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
512	aug	sun	81.6	56.7	665.6	1.9	27.8	32	2.7	0.0	...	0	0	
513	aug	sun	81.6	56.7	665.6	1.9	21.9	71	5.8	0.0	...	0	0	
514	aug	sun	81.6	56.7	665.6	1.9	21.2	70	6.7	0.0	...	0	0	
515	aug	sat	94.4	146.0	614.7	11.3	25.6	42	4.0	0.0	...	0	0	
516	nov	tue	79.5	3.0	106.7	1.1	11.8	31	4.5	0.0	...	0	0	

517 rows × 31 columns

In [5]:
```
forestfire_data.shape
```

Out[5]:
```
(517, 31)
```

In [6]:
```
forestfire_data.isna().sum()
```

Out[6]:
```
month           0
day             0
FFMC            0
DMC             0
DC              0
ISI             0
temp            0
RH              0
wind            0
rain            0
area            0
dayfri          0
daymon          0
daysat          0
daysun          0
daythu          0
daytue          0
daywed          0
monthapr        0
monthaug        0
monthdec        0
monthfeb        0
monthjan        0
monthjul        0
monthjun        0
monthmar        0
monthmay        0
monthnov        0
monthoct        0
monthsep        0
size_category   0
dtype: int64
```

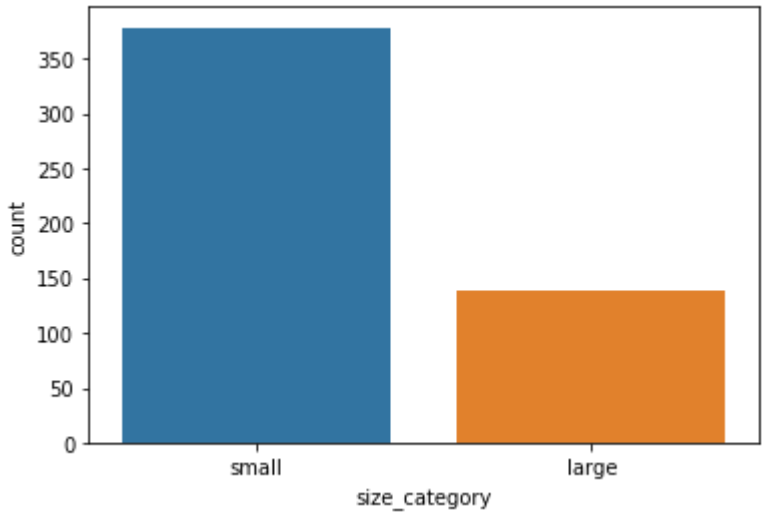
In [7]:
```
forestfire_data.describe(include='all')
```

Out[7]:

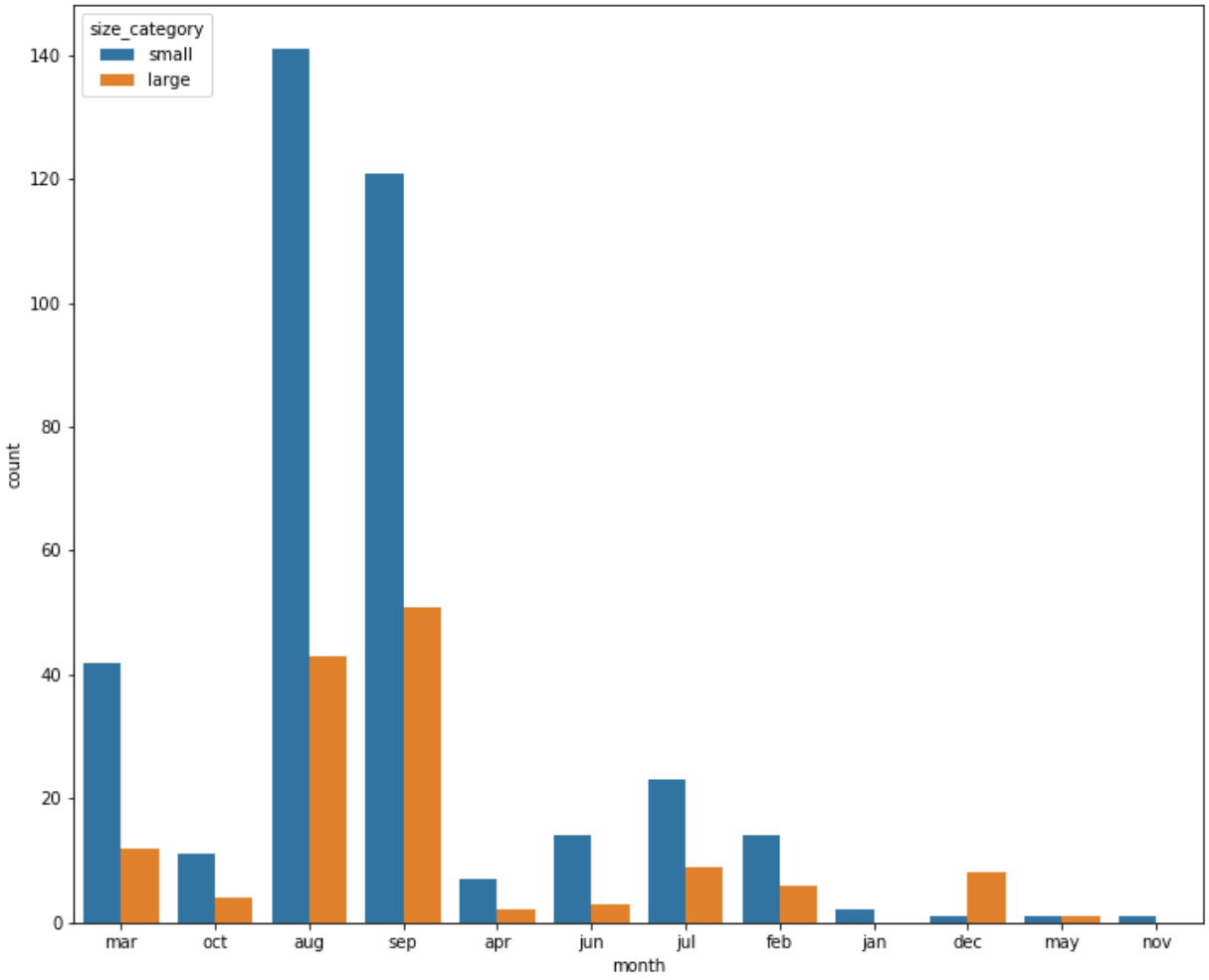
	month	day	FFMC	DMC	DC	ISI	temp	RH
count	517	517	517.000000	517.000000	517.000000	517.000000	517.000000	517.000000
unique	12	7	NaN	NaN	NaN	NaN	NaN	NaN
top	aug	sun	NaN	NaN	NaN	NaN	NaN	NaN
freq	184	95	NaN	NaN	NaN	NaN	NaN	NaN
mean	NaN	NaN	90.644681	110.872340	547.940039	9.021663	18.889168	44.288201
std	NaN	NaN	5.520111	64.046482	248.066192	4.559477	5.806625	16.317469
min	NaN	NaN	18.700000	1.100000	7.900000	0.000000	2.200000	15.000000
25%	NaN	NaN	90.200000	68.600000	437.700000	6.500000	15.500000	33.000000
50%	NaN	NaN	91.600000	108.300000	664.200000	8.400000	19.300000	42.000000
75%	NaN	NaN	92.900000	142.400000	713.900000	10.800000	22.800000	53.000000
max	NaN	NaN	96.200000	291.300000	860.600000	56.100000	33.300000	100.000000

11 rows × 31 columns

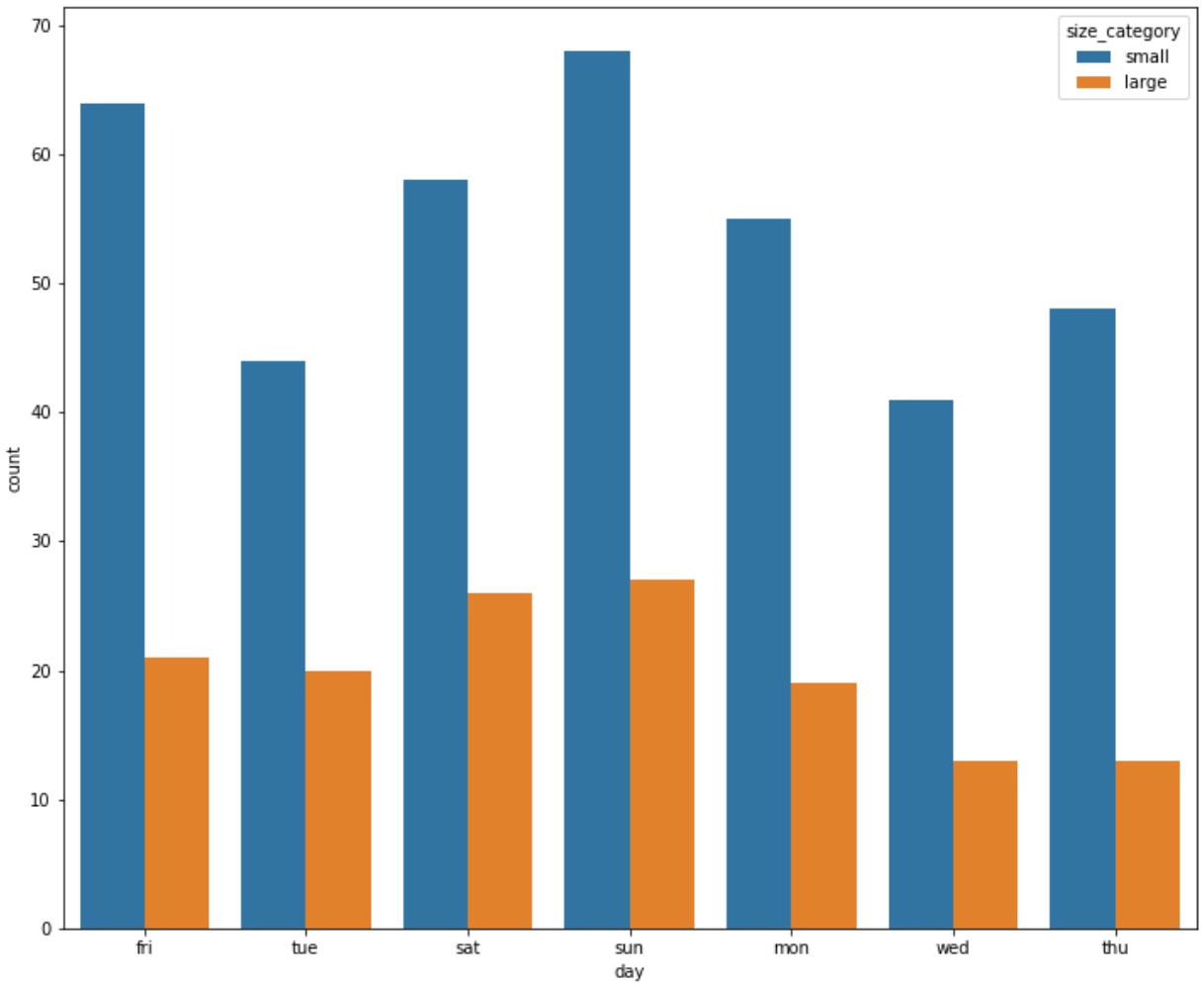
In [8]:
```
sns.countplot(forestfire_data['size_category'])
plt.show()
```



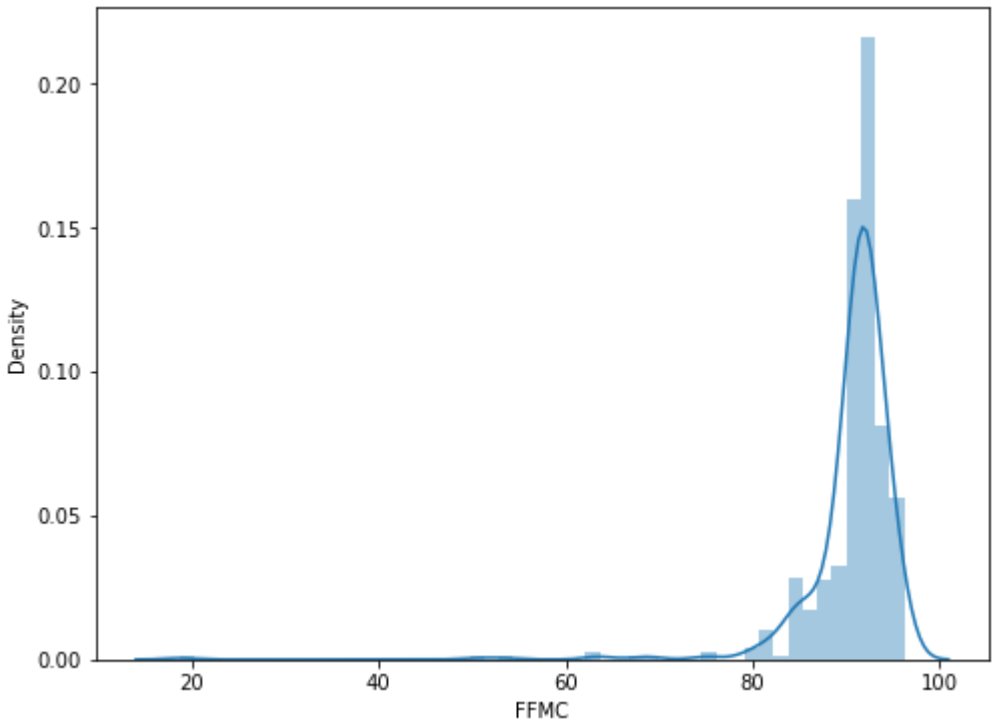
In [9]:
```
plt.figure(figsize=(12,10))
sns.countplot(x = forestfire_data['month'],hue=forestfire_data['size_category'])
plt.show()
```



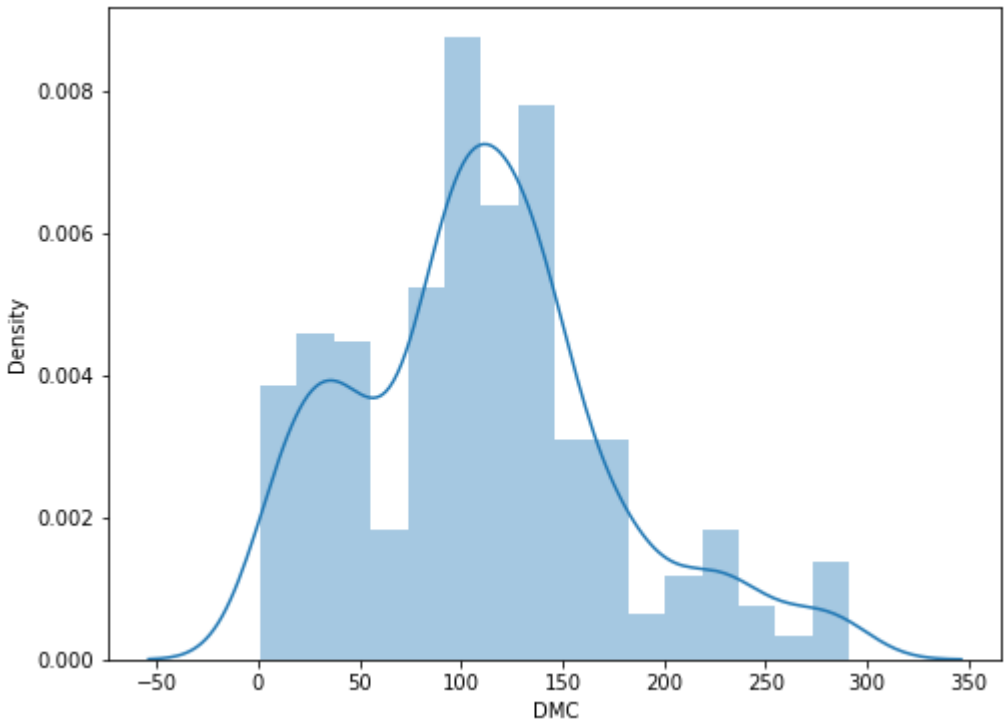
In [10]:
```
plt.figure(figsize=(12,10))
sns.countplot(x = forestfire_data['day'],hue=forestfire_data['size_category'])
plt.show()
```



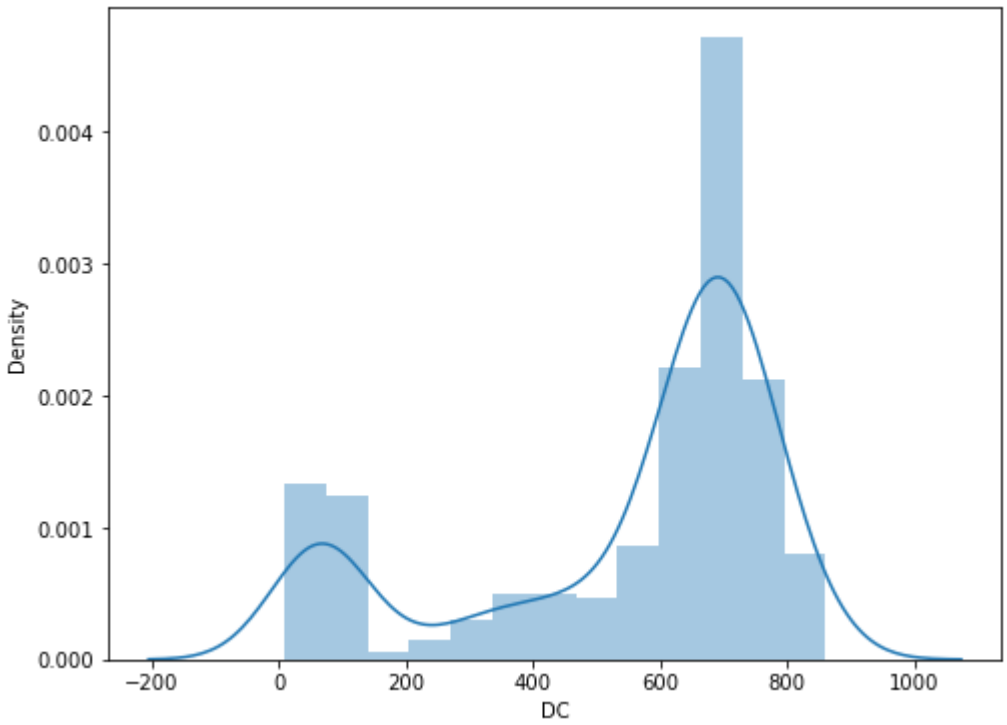
In [11]:
```
plt.figure(figsize=(8,6))
sns.distplot(forestfire_data['FFMC'],)
plt.show()
```



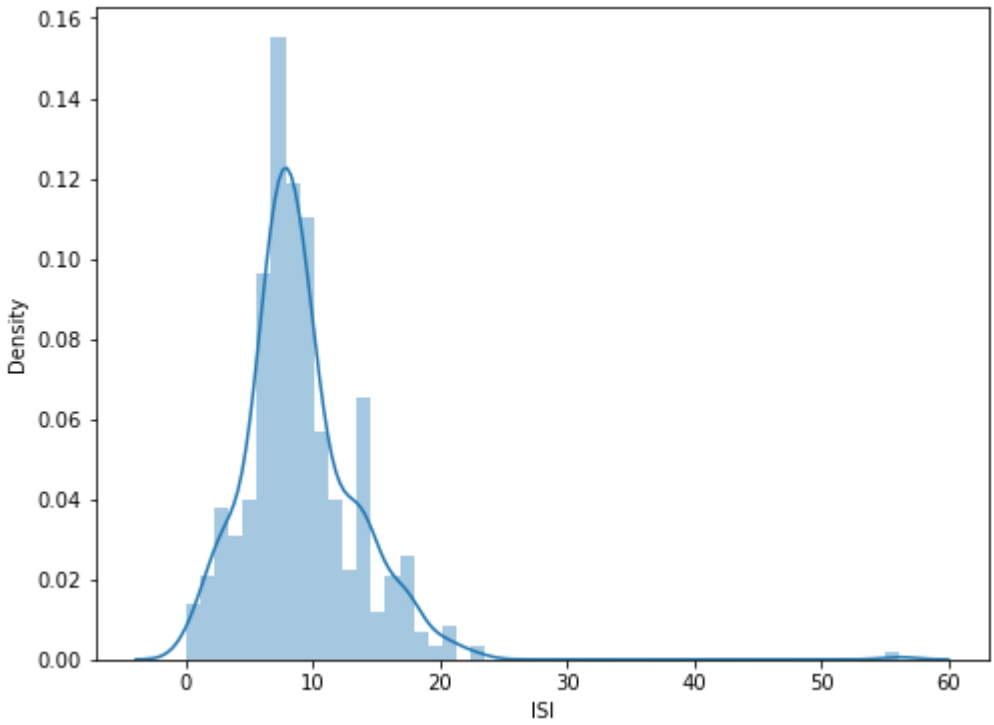
In [12]:
```
plt.figure(figsize=(8,6))
sns.distplot(forestfire_data['DMC'],)
plt.show()
```



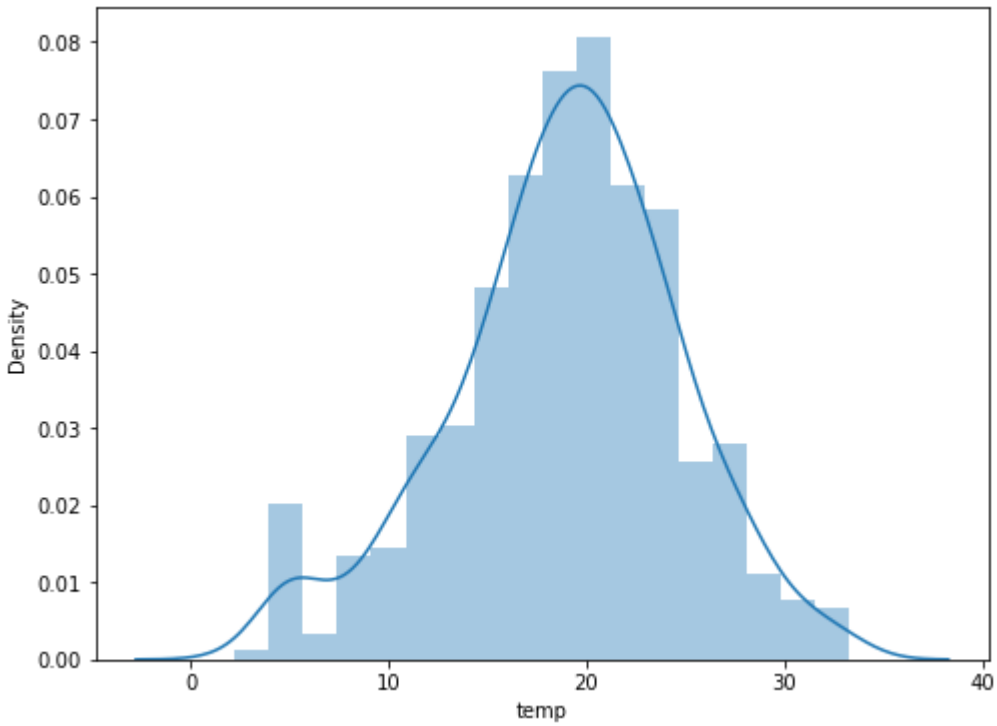
In [13]:
```
plt.figure(figsize=(8,6))
sns.distplot(forestfire_data['DC'],)
plt.show()
```



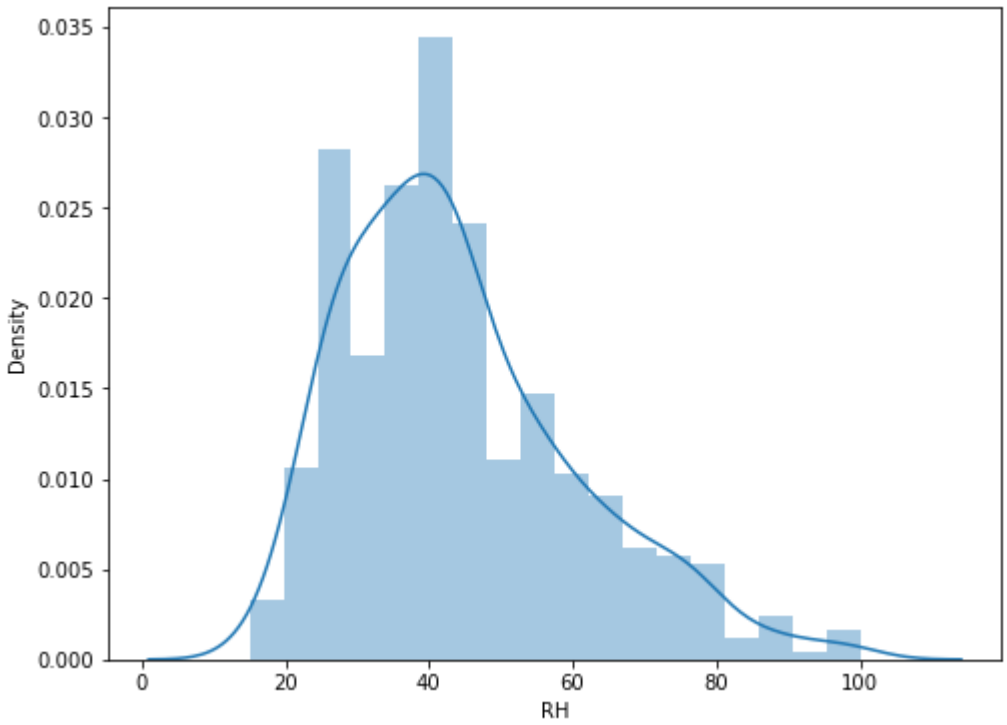
In [14]:
```
plt.figure(figsize=(8,6))
sns.distplot(forestfire_data['ISI'],)
plt.show()
```



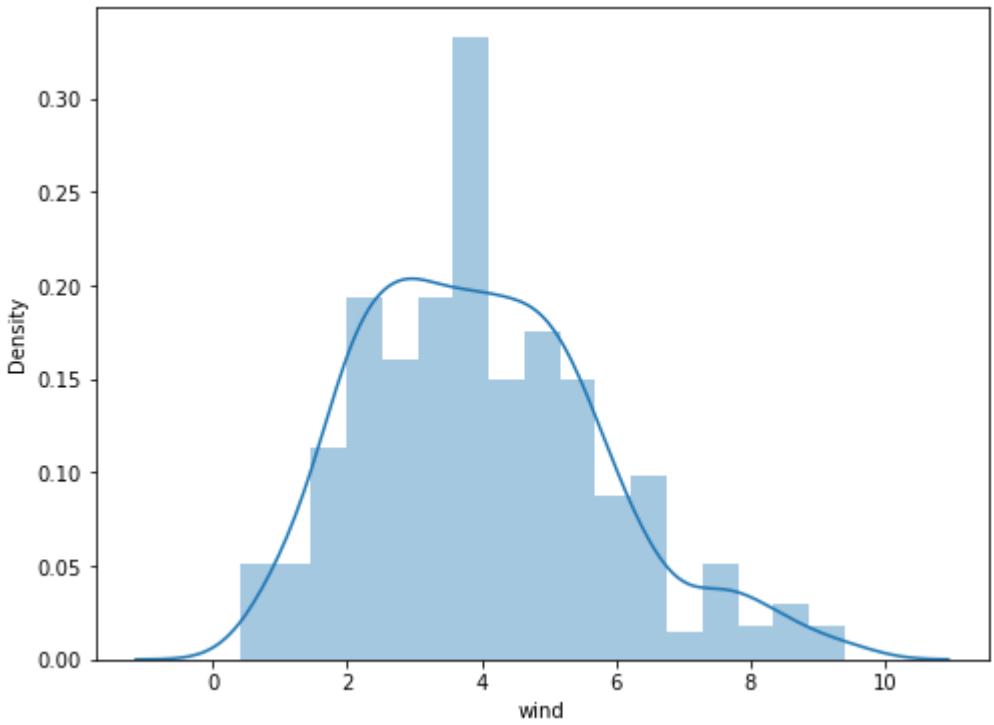
In [15]:
```
plt.figure(figsize=(8,6))
sns.distplot(forestfire_data['temp'],)
plt.show()
```



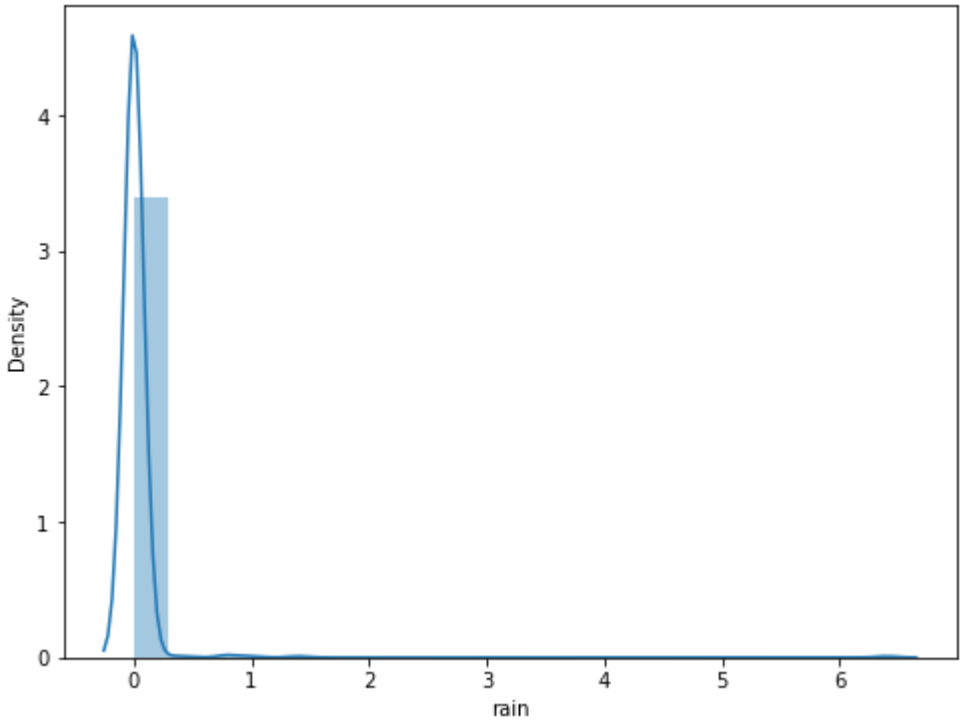
In [16]:
```
plt.figure(figsize=(8,6))
sns.distplot(forestfire_data['RH'],)
plt.show()
```



In [17]:
```
plt.figure(figsize=(8,6))
sns.distplot(forestfire_data['wind'],)
plt.show()
```



In [18]:
```
plt.figure(figsize=(8,6))
sns.distplot(forestfire_data['rain'],)
plt.show()
```



In [19]:
```
drop_data = forestfire_data.drop(labels=['month','day'],axis = 1)
drop_data
```

Out[19]:

	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area	dayfri	...	monthfeb	monthjan	mon
0	86.2	26.2	94.3	5.1	8.2	51	6.7	0.0	0.00	1	...	0	0	
1	90.6	35.4	669.1	6.7	18.0	33	0.9	0.0	0.00	0	...	0	0	
2	90.6	43.7	686.9	6.7	14.6	33	1.3	0.0	0.00	0	...	0	0	
3	91.7	33.3	77.5	9.0	8.3	97	4.0	0.2	0.00	1	...	0	0	
4	89.3	51.3	102.2	9.6	11.4	99	1.8	0.0	0.00	0	...	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
512	81.6	56.7	665.6	1.9	27.8	32	2.7	0.0	6.44	0	...	0	0	
513	81.6	56.7	665.6	1.9	21.9	71	5.8	0.0	54.29	0	...	0	0	
514	81.6	56.7	665.6	1.9	21.2	70	6.7	0.0	11.16	0	...	0	0	
515	94.4	146.0	614.7	11.3	25.6	42	4.0	0.0	0.00	0	...	0	0	
516	79.5	3.0	106.7	1.1	11.8	31	4.5	0.0	0.00	0	...	0	0	

517 rows × 29 columns

In [20]:
```
le = LabelEncoder()
drop_data['size_category'] = le.fit_transform(drop_data['size_category'])
drop_data
```

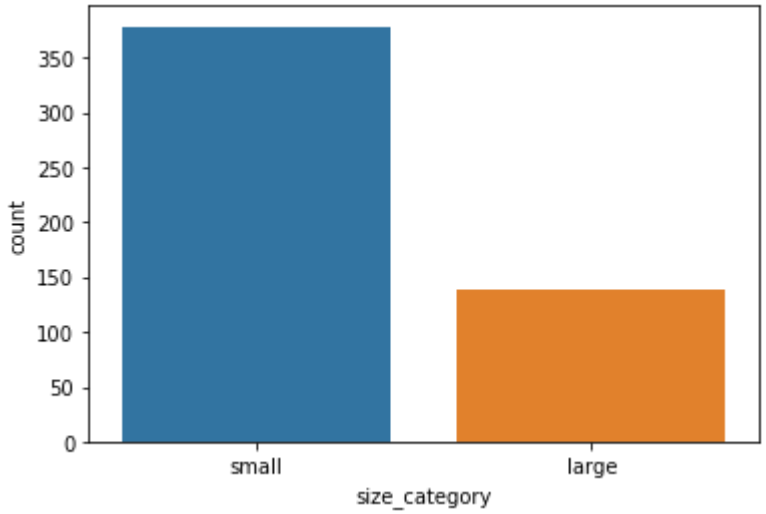
Out[20]:

	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area	dayfri	...	monthfeb	monthjan	mon
0	86.2	26.2	94.3	5.1	8.2	51	6.7	0.0	0.00	1	...	0	0	
1	90.6	35.4	669.1	6.7	18.0	33	0.9	0.0	0.00	0	...	0	0	
2	90.6	43.7	686.9	6.7	14.6	33	1.3	0.0	0.00	0	...	0	0	
3	91.7	33.3	77.5	9.0	8.3	97	4.0	0.2	0.00	1	...	0	0	
4	89.3	51.3	102.2	9.6	11.4	99	1.8	0.0	0.00	0	...	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
512	81.6	56.7	665.6	1.9	27.8	32	2.7	0.0	6.44	0	...	0	0	
513	81.6	56.7	665.6	1.9	21.9	71	5.8	0.0	54.29	0	...	0	0	
514	81.6	56.7	665.6	1.9	21.2	70	6.7	0.0	11.16	0	...	0	0	
515	94.4	146.0	614.7	11.3	25.6	42	4.0	0.0	0.00	0	...	0	0	
516	79.5	3.0	106.7	1.1	11.8	31	4.5	0.0	0.00	0	...	0	0	

517 rows × 29 columns

In [21]:
```
sns.countplot(forestfire_data['size_category'])
```

Out[21]:
```
<AxesSubplot:xlabel='size_category', ylabel='count'>
```



In [22]:
```
drop_data['size_category'].replace({'small':0,'large':0},inplace = True)
drop_data
```

Out[22]:

	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area	dayfri	...	monthfeb	monthjan	mon
0	86.2	26.2	94.3	5.1	8.2	51	6.7	0.0	0.00	1	...	0	0	
1	90.6	35.4	669.1	6.7	18.0	33	0.9	0.0	0.00	0	...	0	0	
2	90.6	43.7	686.9	6.7	14.6	33	1.3	0.0	0.00	0	...	0	0	
3	91.7	33.3	77.5	9.0	8.3	97	4.0	0.2	0.00	1	...	0	0	
4	89.3	51.3	102.2	9.6	11.4	99	1.8	0.0	0.00	0	...	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
512	81.6	56.7	665.6	1.9	27.8	32	2.7	0.0	6.44	0	...	0	0	
513	81.6	56.7	665.6	1.9	21.9	71	5.8	0.0	54.29	0	...	0	0	
514	81.6	56.7	665.6	1.9	21.2	70	6.7	0.0	11.16	0	...	0	0	
515	94.4	146.0	614.7	11.3	25.6	42	4.0	0.0	0.00	0	...	0	0	
516	79.5	3.0	106.7	1.1	11.8	31	4.5	0.0	0.00	0	...	0	0	

517 rows × 29 columns

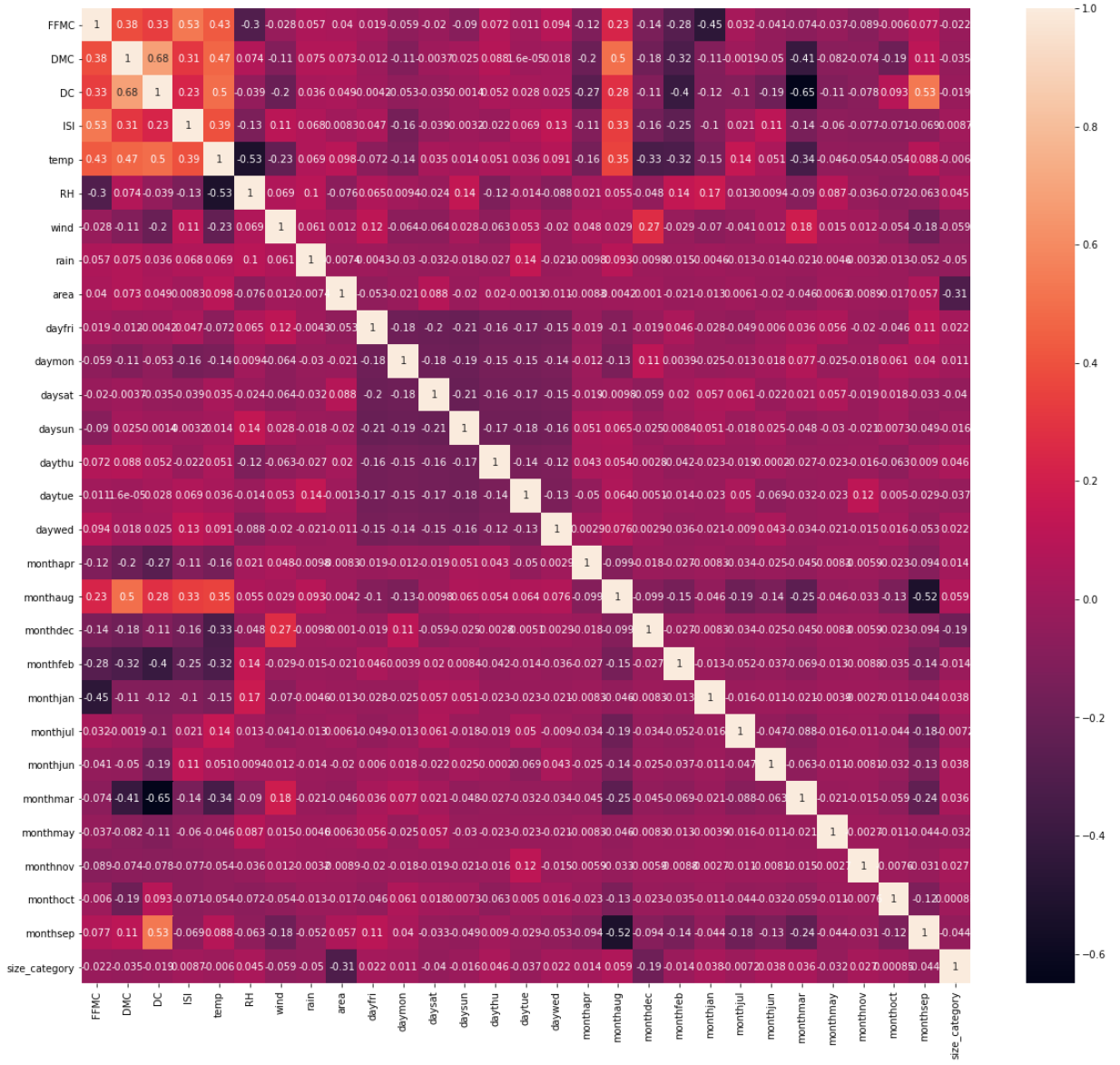
In [23]:
```
corr = drop_data.corr()
corr
```

Out[23]:

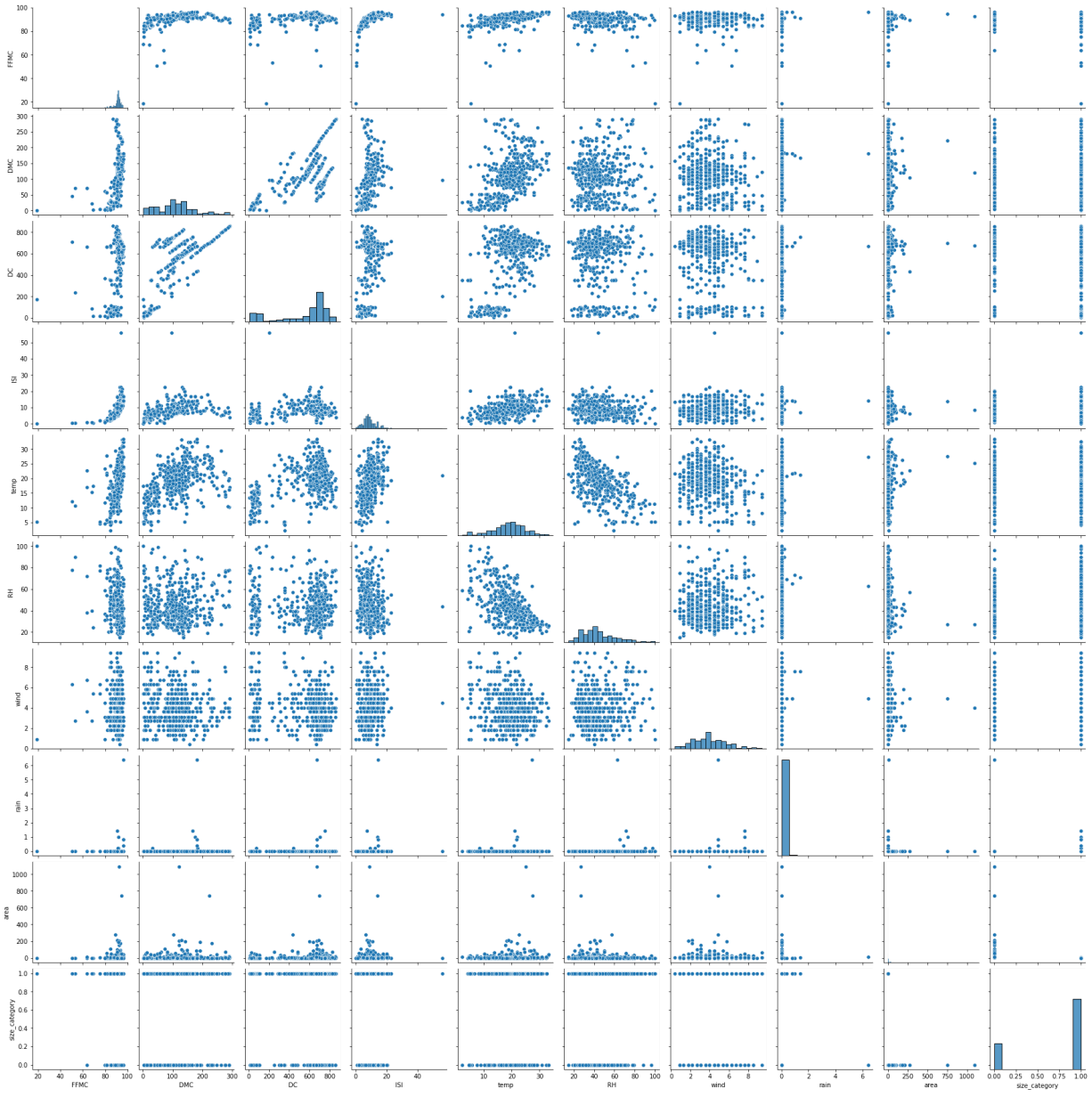
	FFMC	DMC	DC	ISI	temp	RH	wind	rain
FFMC	1.000000	0.382619	0.330512	0.531805	0.431532	-0.300995	-0.028485	0.056702
DMC	0.382619	1.000000	0.682192	0.305128	0.469594	0.073795	-0.105342	0.074790
DC	0.330512	0.682192	1.000000	0.229154	0.496208	-0.039192	-0.203466	0.035861
ISI	0.531805	0.305128	0.229154	1.000000	0.394287	-0.132517	0.106826	0.067668
temp	0.431532	0.469594	0.496208	0.394287	1.000000	-0.527390	-0.227116	0.069491
RH	-0.300995	0.073795	-0.039192	-0.132517	-0.527390	1.000000	0.069410	0.099751
wind	-0.028485	-0.105342	-0.203466	0.106826	-0.227116	0.069410	1.000000	0.061119
rain	0.056702	0.074790	0.035861	0.067668	0.069491	0.099751	0.061119	1.000000
area	0.040122	0.072994	0.049383	0.008258	0.097844	-0.075519	0.012317	-0.007366
dayfri	0.019306	-0.012010	-0.004220	0.046695	-0.071949	0.064506	0.118090	-0.004261
daymon	-0.059396	-0.107921	-0.052993	-0.158601	-0.136529	0.009376	-0.063881	-0.029945
daysat	-0.019637	-0.003653	-0.035189	-0.038585	0.034899	-0.023869	-0.063799	-0.032271
daysun	-0.089517	0.025355	-0.001431	-0.003243	0.014403	0.136220	0.027981	-0.017872
daythu	0.071730	0.087672	0.051859	-0.022406	0.051432	-0.123061	-0.062553	-0.026798
daytue	0.011225	0.000016	0.028368	0.068610	0.035630	-0.014211	0.053396	0.139311
daywed	0.093908	0.017939	0.024803	0.125415	0.090580	-0.087508	-0.019965	-0.020744
monthapr	-0.117199	-0.197543	-0.268211	-0.106478	-0.157051	0.021235	0.048266	-0.009752
monthaug	0.228103	0.497928	0.279361	0.334639	0.351404	0.054761	0.028577	0.093101
monthdec	-0.137044	-0.176301	-0.105642	-0.162322	-0.329648	-0.047714	0.269702	-0.009752
monthfeb	-0.281535	-0.317899	-0.399277	-0.249777	-0.320015	0.140430	-0.029431	-0.014698
monthjan	-0.454771	-0.105647	-0.115064	-0.103588	-0.146520	0.170923	-0.070245	-0.004566
monthjul	0.031833	-0.001946	-0.100887	0.020982	0.142588	0.013185	-0.040645	-0.013390
monthjun	-0.040634	-0.050403	-0.186183	0.111516	0.051015	0.009382	0.012124	-0.013510
monthmar	-0.074327	-0.407404	-0.650427	-0.143520	-0.341797	-0.089836	0.181433	-0.020744
monthmay	-0.037230	-0.081980	-0.114209	-0.060493	-0.045540	0.086822	0.015054	-0.004566
monthnov	-0.088964	-0.074218	-0.078380	-0.076559	-0.053798	-0.035885	0.011864	-0.003225
monthoct	-0.005998	-0.187632	0.093279	-0.071154	-0.053513	-0.072334	-0.053850	-0.012665
monthsep	0.076609	0.110907	0.531857	-0.068877	0.088006	-0.062596	-0.181476	-0.051733
size_category	-0.022063	-0.034715	-0.019428	0.008726	-0.006021	0.045243	-0.059113	-0.050001

29 rows × 29 columns

In [24]:
```
plt.figure(figsize=(20,18))
sns.heatmap(corr,annot=True)
plt.show()
```



In [25]:
```
pair_data = pd.concat([drop_data.iloc[:,0:9],drop_data.iloc[:,-1]],axis = 1)
sns.pairplot(pair_data)
plt.show()
```



Model Building

In [26]:
```
x = drop_data.drop(labels='size_category',axis = 1)
y = drop_data[['size_category']]
```

In [27]:
```
x_train, x_test, y_train, y_test  = train_test_split(x,y,test_size=0.3,random_sta
```

In [28]:
```
x_train.shape,y_train.shape
```

Out[28]:
```
((361, 28), (361, 1))
```

Our Data Is Imbalance so i have to balance it , so i m using here smote operation of balancing technique

In [29]:
```
sm = SMOTE(random_state=12)
x_train_sm,y_train_sm = sm.fit_resample(x_train,np.array(y_train).ravel())
x_train_sm,y_train_sm
```

Out[29]:
```
(          FFMC         DMC          DC        ISI       temp  RH      wind
\
 0    93.700000  101.300000  458.800000  11.900000  19.300000  39  7.200000
 1    92.800000  119.000000  783.500000   7.500000  16.800000  28  4.000000
 2    92.100000  152.600000  658.200000  14.300000  20.200000  47  4.000000
 3    93.700000  101.300000  423.400000  14.700000  26.100000  45  4.000000
 4    90.800000   41.900000   89.400000   7.900000  13.300000  42  0.900000
 ..         ...         ...         ...        ...        ...  ..       ...
 533  91.372665  137.325971  680.945740   9.473142  16.000477  58  5.690184
 534  87.307221   10.323502   25.368854   7.059981   9.503587  39  6.480038
 535  93.839786   81.878499  686.854129  17.434048  22.477775  30  4.900000
 536  91.810487  169.218024  632.028060  10.420531  20.563220  53  3.168242
 537  93.914567  165.944421  706.169390  15.226172  20.152591  49  4.182222

      rain       area  dayfri  ...  monthdec  monthfeb  monthjan  monthjul  \
 0     0.0   7.730000       0  ...         0         0         0         1
 1     0.0   7.210000       0  ...         0         0         0         0
 2     0.0   3.090000       0  ...         0         0         0         0
 3     0.0   7.360000       0  ...         0         0         0         1
 4     0.0   7.400000       0  ...         0         0         0         0
 ..    ...        ...     ...  ...       ...       ...       ...       ...
 533   0.0  10.572590       0  ...         0         0         0         0
 534   0.0  18.149787       0  ...         0         0         0         0
 535   0.0  24.043695       0  ...         0         0         0         0
 536   0.0  13.166840       0  ...         0         0         0         0
 537   0.0  26.354207       0  ...         0         0         0         0

      monthjun  monthmar  monthmay  monthnov  monthoct  monthsep
 0           0         0         0         0         0         0
 1           0         0         0         0         0         1
 2           0         0         0         0         0         0
 3           0         0         0         0         0         0
 4           0         1         0         0         0         0
 ..        ...       ...       ...       ...       ...       ...
 533         0         0         0         0         0         0
 534         0         0         0         0         0         0
 535         0         0         0         0         0         1
 536         0         0         0         0         0         0
 537         0         0         0         0         0         0

 [538 rows x 28 columns],
 array([0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1,
        1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1,
        1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0,
        1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0,
        1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0,
        0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
```

```
        1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1,
        1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0,
        1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1,
        1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0]))
```

In [30]:

```
x_train_sm.shape,y_train_sm.shape
```

Out[30]:

```
((538, 28), (538,))
```

In [31]:

```
x_train = x_train_sm.copy()
y_train = y_train_sm.copy()
```

Convert Data into standard scale

In [32]:

```
scale = MinMaxScaler()
X_train = scale.fit_transform(x_train)
X_train
```

Out[32]:

```
array([[0.96774194, 0.34683281, 0.53209818, ..., 0.        , 0.        ,
        0.        ],
       [0.95612903, 0.40809969, 0.91527024, ..., 0.        , 0.        ,
        1.        ],
       [0.94709677, 0.52440291, 0.76740618, ..., 0.        , 0.        ,
        0.        ],
       ...,
       [0.96954562, 0.27960713, 0.80122036, ..., 0.        , 0.        ,
        1.        ],
       [0.94336113, 0.58192462, 0.73652119, ..., 0.        , 0.        ,
        0.        ],
       [0.97051055, 0.57059336, 0.82401391, ..., 0.        , 0.        ,
        0.        ]])
```

In [33]:
```
X_test = scale.fit_transform(x_test)
X_test
```

Out[33]:
```
array([[0.93251534, 0.57586327, 0.45927711, ..., 0.        , 0.        ,
        0.        ],
       [0.9202454 , 0.47017789, 0.67903614, ..., 0.        , 0.        ,
        0.        ],
       [0.84662577, 0.291594  , 0.8446988 , ..., 0.        , 0.        ,
        1.        ],
       ...,
       [0.82822086, 0.67038716, 0.74614458, ..., 0.        , 0.        ,
        0.        ],
       [0.88650307, 0.416812  , 0.78325301, ..., 0.        , 0.        ,
        1.        ],
       [0.9202454 , 0.47017789, 0.67903614, ..., 0.        , 0.        ,
        0.        ]])
```

Model Training

Tuning of Hyperparameter : Batch size and Epoch

In [34]:
```
def creat_model():
    model = Sequential()
    model.add(Dense(8, input_dim = 28,kernel_initializer='uniform', activation='r
    model.add(Dropout(0.2))
    model.add(Dense(4,kernel_initializer='uniform', activation='relu'))
    model.add(Dropout(0.2))
    model.add(Dense(1,kernel_initializer='uniform', activation='sigmoid'))
    adam = Adam(learning_rate=0.001)
    model.compile(loss='binary_crossentropy',optimizer = adam,metrics='accuracy')
    return model
```

In [35]:
```
model = KerasClassifier(build_fn=creat_model,verbose = 0)
batch_size = [10,30,50]
epochs = [10,20,30]
param_grid = dict(batch_size = batch_size,epochs = epochs)
gsv = GridSearchCV(estimator=model,param_grid=param_grid,cv = KFold(),verbose=5)
gsv_res = gsv.fit(X_train,y_train)
```

```
Fitting 5 folds for each of 9 candidates, totalling 45 fits
[CV 1/5] END .....batch_size=10, epochs=10;, score=0.583 total time=   4.6
s
[CV 2/5] END .....batch_size=10, epochs=10;, score=0.546 total time=   0.8
s
[CV 3/5] END .....batch_size=10, epochs=10;, score=0.472 total time=   0.6
s
[CV 4/5] END .....batch_size=10, epochs=10;, score=0.271 total time=   1.7
s
[CV 5/5] END .....batch_size=10, epochs=10;, score=0.000 total time=   0.8
s
[CV 1/5] END .....batch_size=10, epochs=20;, score=0.778 total time=   0.9
s
[CV 2/5] END .....batch_size=10, epochs=20;, score=0.694 total time=   0.9
s
[CV 3/5] END .....batch_size=10, epochs=20;, score=0.722 total time=   1.1
s
[CV 4/5] END .....batch_size=10, epochs=20;, score=0.841 total time=   1.0
s
[CV 5/5] END .....batch_size=10, epochs=20;, score=0.000 total time=   1.0
s
[CV 1/5] END .....batch_size=10, epochs=30;, score=0.769 total time=   1.2
s
[CV 2/5] END .....batch_size=10, epochs=30;, score=0.741 total time=   1.2
s
[CV 3/5] END .....batch_size=10, epochs=30;, score=0.815 total time=   1.2
s
[CV 4/5] END .....batch_size=10, epochs=30;, score=0.850 total time=   1.3
s
[CV 5/5] END .....batch_size=10, epochs=30;, score=0.935 total time=   1.3
s
[CV 1/5] END .....batch_size=30, epochs=10;, score=0.213 total time=   0.8
s
[CV 2/5] END .....batch_size=30, epochs=10;, score=0.491 total time=   0.5
s
[CV 3/5] END .....batch_size=30, epochs=10;, score=0.213 total time=   0.5
s
[CV 4/5] END .....batch_size=30, epochs=10;, score=0.271 total time=   0.5
s
[CV 5/5] END .....batch_size=30, epochs=10;, score=0.000 total time=   0.5
s
[CV 1/5] END .....batch_size=30, epochs=20;, score=0.620 total time=   0.6
s
[CV 2/5] END .....batch_size=30, epochs=20;, score=0.676 total time=   0.6
s
[CV 3/5] END .....batch_size=30, epochs=20;, score=0.213 total time=   0.6
s
[CV 4/5] END .....batch_size=30, epochs=20;, score=0.271 total time=   0.6
s
[CV 5/5] END .....batch_size=30, epochs=20;, score=0.000 total time=   0.8
```

```
s
[CV 1/5] END .........batch_size=30, epochs=30;, score=0.815 total time=   0.7
s
[CV 2/5] END .........batch_size=30, epochs=30;, score=0.685 total time=   0.7
s
[CV 3/5] END .........batch_size=30, epochs=30;, score=0.704 total time=   0.8
s
[CV 4/5] END .........batch_size=30, epochs=30;, score=0.738 total time=   0.8
s
[CV 5/5] END .........batch_size=30, epochs=30;, score=0.131 total time=   0.7
s
[CV 1/5] END .........batch_size=50, epochs=10;, score=0.213 total time=   0.6
s
[CV 2/5] END .........batch_size=50, epochs=10;, score=0.352 total time=   0.5
s
[CV 3/5] END .........batch_size=50, epochs=10;, score=0.213 total time=   0.7
s
WARNING:tensorflow:5 out of the last 14 calls to <function Model.make_test_func
tion.<locals>.test_function at 0x0000013D3ED19280> triggered tf.function retrac
ing. Tracing is expensive and the excessive number of tracings could be due to
(1) creating @tf.function repeatedly in a loop, (2) passing tensors with differ
ent shapes, (3) passing Python objects instead of tensors. For (1), please defi
ne your @tf.function outside of the loop. For (2), @tf.function has experimenta
l_relax_shapes=True option that relaxes argument shapes that can avoid unnecess
ary retracing. For (3), please refer to https://www.tensorflow.org/guide/functi
on#controlling_retracing (https://www.tensorflow.org/guide/function#controlling
_retracing) and https://www.tensorflow.org/api_docs/python/tf/function (http
s://www.tensorflow.org/api_docs/python/tf/function) for  more details.
[CV 4/5] END .........batch_size=50, epochs=10;, score=0.271 total time=   0.4
s
WARNING:tensorflow:5 out of the last 13 calls to <function Model.make_test_func
tion.<locals>.test_function at 0x0000013D3EF50790> triggered tf.function retrac
ing. Tracing is expensive and the excessive number of tracings could be due to
 (1) creating @tf.function repeatedly in a loop, (2) passing tensors with diffe
rent shapes, (3) passing Python objects instead of tensors. For (1), please def
ine your @tf.function outside of the loop. For (2), @tf.function has experiment
al_relax_shapes=True option that relaxes argument shapes that can avoid unneces
sary retracing. For (3), please refer to https://www.tensorflow.org/guide/funct
ion#controlling_retracing (https://www.tensorflow.org/guide/function#controllin
g_retracing) and https://www.tensorflow.org/api_docs/python/tf/function (http
s://www.tensorflow.org/api_docs/python/tf/function) for  more details.
[CV 5/5] END .........batch_size=50, epochs=10;, score=0.000 total time=   0.5
s
[CV 1/5] END .........batch_size=50, epochs=20;, score=0.213 total time=   0.6
s
[CV 2/5] END .........batch_size=50, epochs=20;, score=0.519 total time=   0.6
s
[CV 3/5] END .........batch_size=50, epochs=20;, score=0.213 total time=   0.5
s
[CV 4/5] END .........batch_size=50, epochs=20;, score=0.271 total time=   0.5
s
[CV 5/5] END .........batch_size=50, epochs=20;, score=0.000 total time=   0.5
s
[CV 1/5] END .........batch_size=50, epochs=30;, score=0.667 total time=   0.7
s
[CV 2/5] END .........batch_size=50, epochs=30;, score=0.685 total time=   0.8
s
```

```
[CV 3/5] END .........batch_size=50, epochs=30;, score=0.213 total time=   0.6
s
[CV 4/5] END .........batch_size=50, epochs=30;, score=0.271 total time=   0.6
s
[CV 5/5] END .........batch_size=50, epochs=30;, score=0.000 total time=   0.6
s
```

In [36]:
```
print(gsv_res.best_params_,gsv_res.best_score_)
```

```
{'batch_size': 10, 'epochs': 30} 0.8218241453170776
```

Turning Hyperparameter: Learning rate and Dropout rate

In [37]:
```
def creat_model(learning_rate,dropout_rate):
    model = Sequential()
    model.add(Dense(8, input_dim = 28,kernel_initializer='uniform', activation='r
    model.add(Dropout(0.2))
    model.add(Dense(4,kernel_initializer='uniform', activation='relu'))
    model.add(Dropout(0.2))
    model.add(Dense(1,kernel_initializer='uniform', activation='sigmoid'))
    adam = Adam(learning_rate=learning_rate)
    model.compile(loss='binary_crossentropy',optimizer = adam,metrics='accuracy')
    return model
```

In [38]:
```
model = KerasClassifier(build_fn=creat_model,batch_size = 10,epochs = 30,verbose
learning_rate = [0.1,0.01,0.001]
dropout_rate = [0.0,0.1,0.2]
param_grid = dict(learning_rate = learning_rate,dropout_rate = dropout_rate)
gsv = GridSearchCV(estimator=model,param_grid=param_grid,cv= KFold(),verbose=5)
gsv_r = gsv.fit(X_train,y_train)
```

```
Fitting 5 folds for each of 9 candidates, totalling 45 fits
[CV 1/5] END dropout_rate=0.0, learning_rate=0.1;, score=0.213 total time=   1.
1s
[CV 2/5] END dropout_rate=0.0, learning_rate=0.1;, score=0.694 total time=   1.
2s
[CV 3/5] END dropout_rate=0.0, learning_rate=0.1;, score=0.213 total time=   1.
2s
[CV 4/5] END dropout_rate=0.0, learning_rate=0.1;, score=0.271 total time=   1.
4s
[CV 5/5] END dropout_rate=0.0, learning_rate=0.1;, score=0.000 total time=   1.
4s
[CV 1/5] END dropout_rate=0.0, learning_rate=0.01;, score=0.843 total time=
1.4s
[CV 2/5] END dropout_rate=0.0, learning_rate=0.01;, score=0.352 total time=
1.2s
[CV 3/5] END dropout_rate=0.0, learning_rate=0.01;, score=0.824 total time=
1.1s
[CV 4/5] END dropout_rate=0.0, learning_rate=0.01;, score=0.271 total time=
1.4s
[CV 5/5] END dropout_rate=0.0, learning_rate=0.01;, score=0.000 total time=
1.3s
[CV 1/5] END dropout_rate=0.0, learning_rate=0.001;, score=0.787 total time=
1.2s
[CV 2/5] END dropout_rate=0.0, learning_rate=0.001;, score=0.731 total time=
1.3s
[CV 3/5] END dropout_rate=0.0, learning_rate=0.001;, score=0.833 total time=
1.1s
[CV 4/5] END dropout_rate=0.0, learning_rate=0.001;, score=0.841 total time=
1.2s
[CV 5/5] END dropout_rate=0.0, learning_rate=0.001;, score=0.000 total time=
1.4s
[CV 1/5] END dropout_rate=0.1, learning_rate=0.1;, score=0.213 total time=   1.
4s
[CV 2/5] END dropout_rate=0.1, learning_rate=0.1;, score=0.676 total time=   1.
2s
[CV 3/5] END dropout_rate=0.1, learning_rate=0.1;, score=0.824 total time=   1.
1s
[CV 4/5] END dropout_rate=0.1, learning_rate=0.1;, score=0.271 total time=   1.
2s
[CV 5/5] END dropout_rate=0.1, learning_rate=0.1;, score=0.991 total time=   1.
2s
[CV 1/5] END dropout_rate=0.1, learning_rate=0.01;, score=0.870 total time=
1.0s
[CV 2/5] END dropout_rate=0.1, learning_rate=0.01;, score=0.352 total time=
1.2s
[CV 3/5] END dropout_rate=0.1, learning_rate=0.01;, score=0.852 total time=
1.2s
[CV 4/5] END dropout_rate=0.1, learning_rate=0.01;, score=0.879 total time=
1.3s
[CV 5/5] END dropout_rate=0.1, learning_rate=0.01;, score=0.981 total time=
```

```
1.5s
[CV 1/5] END dropout_rate=0.1, learning_rate=0.001;, score=0.778 total time=
1.1s
[CV 2/5] END dropout_rate=0.1, learning_rate=0.001;, score=0.713 total time=
1.1s
[CV 3/5] END dropout_rate=0.1, learning_rate=0.001;, score=0.833 total time=
1.2s
[CV 4/5] END dropout_rate=0.1, learning_rate=0.001;, score=0.850 total time=
1.3s
[CV 5/5] END dropout_rate=0.1, learning_rate=0.001;, score=0.645 total time=
1.2s
[CV 1/5] END dropout_rate=0.2, learning_rate=0.1;, score=0.583 total time=   1.
1s
[CV 2/5] END dropout_rate=0.2, learning_rate=0.1;, score=0.796 total time=   1.
1s
[CV 3/5] END dropout_rate=0.2, learning_rate=0.1;, score=0.787 total time=   1.
1s
[CV 4/5] END dropout_rate=0.2, learning_rate=0.1;, score=0.271 total time=   1.
2s
[CV 5/5] END dropout_rate=0.2, learning_rate=0.1;, score=0.000 total time=   1.
4s
[CV 1/5] END dropout_rate=0.2, learning_rate=0.01;, score=0.796 total time=
1.1s
[CV 2/5] END dropout_rate=0.2, learning_rate=0.01;, score=0.796 total time=
1.1s
[CV 3/5] END dropout_rate=0.2, learning_rate=0.01;, score=0.815 total time=
1.2s
[CV 4/5] END dropout_rate=0.2, learning_rate=0.01;, score=0.897 total time=
1.4s
[CV 5/5] END dropout_rate=0.2, learning_rate=0.01;, score=0.972 total time=
1.3s
[CV 1/5] END dropout_rate=0.2, learning_rate=0.001;, score=0.796 total time=
1.1s
[CV 2/5] END dropout_rate=0.2, learning_rate=0.001;, score=0.741 total time=
1.1s
[CV 3/5] END dropout_rate=0.2, learning_rate=0.001;, score=0.824 total time=
1.2s
[CV 4/5] END dropout_rate=0.2, learning_rate=0.001;, score=0.860 total time=
1.2s
[CV 5/5] END dropout_rate=0.2, learning_rate=0.001;, score=0.654 total time=
1.4s
```

In [39]:
```
print(gsv_r.best_params_,gsv_r.best_score_)
```

```
{'dropout_rate': 0.2, 'learning_rate': 0.01} 0.8553132534027099
```

Tuning of Hyperparameter :Activation Function & Kernel Initializer

In [40]:

```
def creat_model(Activation_Function,init):
    model = Sequential()
    model.add(Dense(8, input_dim = 28,kernel_initializer='uniform', activation='r
    model.add(Dropout(0.1))
    model.add(Dense(4,kernel_initializer='uniform', activation='relu'))
    model.add(Dropout(0.1))
    model.add(Dense(1,kernel_initializer='uniform', activation='sigmoid'))
    adam = Adam(learning_rate=0.01)
    model.compile(loss='binary_crossentropy',optimizer = adam,metrics='accuracy')
    return model
```

In [41]:
```
model = KerasClassifier(build_fn=creat_model,batch_size = 10,epochs = 30,verbose
Activation_Function = ['relu','tanh','softmax','linear']
init = ['zero','uniform','normal']
param_grid = dict(Activation_Function = Activation_Function,init = init)
gsv = GridSearchCV(estimator=model,param_grid=param_grid,cv= KFold(),verbose=5)
gsv_result = gsv.fit(X_train,y_train)
```

```
Fitting 5 folds for each of 12 candidates, totalling 60 fits
[CV 1/5] END Activation_Function=relu, init=zero;, score=0.852 total time=   1.
1s
[CV 2/5] END Activation_Function=relu, init=zero;, score=0.796 total time=   1.
3s
[CV 3/5] END Activation_Function=relu, init=zero;, score=0.833 total time=   1.
2s
[CV 4/5] END Activation_Function=relu, init=zero;, score=0.888 total time=   1.
5s
[CV 5/5] END Activation_Function=relu, init=zero;, score=0.972 total time=   1.
4s
[CV 1/5] END Activation_Function=relu, init=uniform;, score=0.880 total time=
1.2s
[CV 2/5] END Activation_Function=relu, init=uniform;, score=0.824 total time=
1.1s
[CV 3/5] END Activation_Function=relu, init=uniform;, score=0.870 total time=
1.4s
[CV 4/5] END Activation_Function=relu, init=uniform;, score=0.907 total time=
1.2s
[CV 5/5] END Activation_Function=relu, init=uniform;, score=0.981 total time=
1.2s
[CV 1/5] END Activation_Function=relu, init=normal;, score=0.852 total time=
1.1s
[CV 2/5] END Activation_Function=relu, init=normal;, score=0.769 total time=
1.2s
[CV 3/5] END Activation_Function=relu, init=normal;, score=0.833 total time=
1.1s
[CV 4/5] END Activation_Function=relu, init=normal;, score=0.879 total time=
1.2s
[CV 5/5] END Activation_Function=relu, init=normal;, score=0.972 total time=
1.2s
[CV 1/5] END Activation_Function=tanh, init=zero;, score=0.806 total time=   1.
1s
[CV 2/5] END Activation_Function=tanh, init=zero;, score=0.352 total time=   1.
1s
[CV 3/5] END Activation_Function=tanh, init=zero;, score=0.213 total time=   1.
3s
[CV 4/5] END Activation_Function=tanh, init=zero;, score=0.888 total time=   1.
2s
[CV 5/5] END Activation_Function=tanh, init=zero;, score=0.972 total time=   1.
2s
[CV 1/5] END Activation_Function=tanh, init=uniform;, score=0.833 total time=
1.0s
[CV 2/5] END Activation_Function=tanh, init=uniform;, score=0.796 total time=
1.1s
[CV 3/5] END Activation_Function=tanh, init=uniform;, score=0.852 total time=
1.2s
[CV 4/5] END Activation_Function=tanh, init=uniform;, score=0.897 total time=
1.2s
[CV 5/5] END Activation_Function=tanh, init=uniform;, score=0.991 total time=
```

```
1.2s
[CV 1/5] END Activation_Function=tanh, init=normal;, score=0.870 total time=
1.1s
[CV 2/5] END Activation_Function=tanh, init=normal;, score=0.352 total time=
1.1s
[CV 3/5] END Activation_Function=tanh, init=normal;, score=0.213 total time=
1.4s
[CV 4/5] END Activation_Function=tanh, init=normal;, score=0.271 total time=
1.3s
[CV 5/5] END Activation_Function=tanh, init=normal;, score=0.991 total time=
1.2s
[CV 1/5] END Activation_Function=softmax, init=zero;, score=0.213 total time=
1.2s
[CV 2/5] END Activation_Function=softmax, init=zero;, score=0.352 total time=
1.1s
[CV 3/5] END Activation_Function=softmax, init=zero;, score=0.213 total time=
1.2s
[CV 4/5] END Activation_Function=softmax, init=zero;, score=0.907 total time=
1.2s
[CV 5/5] END Activation_Function=softmax, init=zero;, score=0.000 total time=
1.3s
[CV 1/5] END Activation_Function=softmax, init=uniform;, score=0.824 total time
=   1.1s
[CV 2/5] END Activation_Function=softmax, init=uniform;, score=0.815 total time
=   1.3s
[CV 3/5] END Activation_Function=softmax, init=uniform;, score=0.833 total time
=   1.2s
[CV 4/5] END Activation_Function=softmax, init=uniform;, score=0.271 total time
=   1.3s
[CV 5/5] END Activation_Function=softmax, init=uniform;, score=0.981 total time
=   1.2s
[CV 1/5] END Activation_Function=softmax, init=normal;, score=0.861 total time=
1.1s
[CV 2/5] END Activation_Function=softmax, init=normal;, score=0.722 total time=
1.1s
[CV 3/5] END Activation_Function=softmax, init=normal;, score=0.694 total time=
1.2s
[CV 4/5] END Activation_Function=softmax, init=normal;, score=0.916 total time=
1.2s
[CV 5/5] END Activation_Function=softmax, init=normal;, score=0.000 total time=
1.4s
[CV 1/5] END Activation_Function=linear, init=zero;, score=0.852 total time=
1.1s
[CV 2/5] END Activation_Function=linear, init=zero;, score=0.787 total time=
1.4s
[CV 3/5] END Activation_Function=linear, init=zero;, score=0.824 total time=
1.1s
[CV 4/5] END Activation_Function=linear, init=zero;, score=0.916 total time=
1.3s
[CV 5/5] END Activation_Function=linear, init=zero;, score=0.981 total time=
1.2s
[CV 1/5] END Activation_Function=linear, init=uniform;, score=0.880 total time=
1.1s
[CV 2/5] END Activation_Function=linear, init=uniform;, score=0.796 total time=
1.1s
[CV 3/5] END Activation_Function=linear, init=uniform;, score=0.213 total time=
1.1s
```

```
[CV 4/5] END Activation_Function=linear, init=uniform;, score=0.879 total time=
1.5s
[CV 5/5] END Activation_Function=linear, init=uniform;, score=0.981 total time=
1.2s
[CV 1/5] END Activation_Function=linear, init=normal;, score=0.852 total time=
1.3s
[CV 2/5] END Activation_Function=linear, init=normal;, score=0.796 total time=
1.1s
[CV 3/5] END Activation_Function=linear, init=normal;, score=0.833 total time=
1.1s
[CV 4/5] END Activation_Function=linear, init=normal;, score=0.935 total time=
1.2s
[CV 5/5] END Activation_Function=linear, init=normal;, score=1.000 total time=
1.2s
```

In [42]:
```
print(gsv_result.best_score_,gsv_result.best_params_)
```

```
0.8923849105834961 {'Activation_Function': 'relu', 'init': 'uniform'}
```

Tuning of Hyperparameter :Number of Neurons in hidden layer

In [43]:
```
def creat_model(neuron1,neuron2):
    model = Sequential()
    model.add(Dense(8,input_dim=28,kernel_initializer='uniform', activation='tanh
    model.add(Dropout(0.1))
    model.add(Dense(4,kernel_initializer='uniform',activation='tanh'))
    model.add(Dropout(0.1))
    model.add(Dense(1,kernel_initializer='uniform',activation='sigmoid'))
    adam = Adam(learning_rate=0.01)
    model.compile(loss='binary_crossentropy', optimizer=adam, metrics=['accuracy'
    return model
```

In [44]:
```
model = KerasClassifier(build_fn=creat_model,batch_size = 10,epochs = 30,verbose
neuron1 = [24,16,8]
neuron2 = [12,8,4]
param_grid = dict(neuron1 = neuron1,neuron2=neuron2)
gsv = GridSearchCV(estimator=model,param_grid=param_grid,cv=KFold(),verbose=5)
gsv_n = gsv.fit(X_train,y_train)
```

```
Fitting 5 folds for each of 9 candidates, totalling 45 fits
[CV 1/5] END .....neuron1=24, neuron2=12;, score=0.880 total time=   1.2
s
[CV 2/5] END .....neuron1=24, neuron2=12;, score=0.787 total time=   1.2
s
[CV 3/5] END .....neuron1=24, neuron2=12;, score=0.778 total time=   1.3
s
[CV 4/5] END .....neuron1=24, neuron2=12;, score=0.897 total time=   1.3
s
[CV 5/5] END .....neuron1=24, neuron2=12;, score=1.000 total time=   1.4
s
[CV 1/5] END .....neuron1=24, neuron2=8;, score=0.898 total time=   1.2
s
[CV 2/5] END .....neuron1=24, neuron2=8;, score=0.824 total time=   1.1
s
[CV 3/5] END .....neuron1=24, neuron2=8;, score=0.889 total time=   1.2
s
[CV 4/5] END .....neuron1=24, neuron2=8;, score=0.888 total time=   1.2
s
[CV 5/5] END .....neuron1=24, neuron2=8;, score=0.981 total time=   1.2
s
[CV 1/5] END .....neuron1=24, neuron2=4;, score=0.907 total time=   1.2
s
[CV 2/5] END .....neuron1=24, neuron2=4;, score=0.843 total time=   1.1
s
[CV 3/5] END .....neuron1=24, neuron2=4;, score=0.889 total time=   1.2
s
[CV 4/5] END .....neuron1=24, neuron2=4;, score=0.916 total time=   1.3
s
[CV 5/5] END .....neuron1=24, neuron2=4;, score=0.991 total time=   1.4
s
[CV 1/5] END .....neuron1=16, neuron2=12;, score=0.889 total time=   1.1
s
[CV 2/5] END .....neuron1=16, neuron2=12;, score=0.861 total time=   1.1
s
[CV 3/5] END .....neuron1=16, neuron2=12;, score=0.889 total time=   1.1
s
[CV 4/5] END .....neuron1=16, neuron2=12;, score=0.944 total time=   1.2
s
[CV 5/5] END .....neuron1=16, neuron2=12;, score=0.981 total time=   1.2
s
[CV 1/5] END .....neuron1=16, neuron2=8;, score=0.898 total time=   1.1
s
[CV 2/5] END .....neuron1=16, neuron2=8;, score=0.889 total time=   1.1
s
[CV 3/5] END .....neuron1=16, neuron2=8;, score=0.833 total time=   1.2
s
[CV 4/5] END .....neuron1=16, neuron2=8;, score=0.972 total time=   1.4
s
[CV 5/5] END .....neuron1=16, neuron2=8;, score=1.000 total time=   1.2
```

```
s
[CV 1/5] END .............neuron1=16, neuron2=4;, score=0.917 total time=   1.4
s
[CV 2/5] END .............neuron1=16, neuron2=4;, score=0.880 total time=   1.3
s
[CV 3/5] END .............neuron1=16, neuron2=4;, score=0.787 total time=   1.2
s
[CV 4/5] END .............neuron1=16, neuron2=4;, score=0.953 total time=   1.3
s
[CV 5/5] END .............neuron1=16, neuron2=4;, score=0.991 total time=   1.2
s
[CV 1/5] END .............neuron1=8, neuron2=12;, score=0.880 total time=   1.0
s
[CV 2/5] END .............neuron1=8, neuron2=12;, score=0.852 total time=   1.1
s
[CV 3/5] END .............neuron1=8, neuron2=12;, score=0.898 total time=   1.1
s
[CV 4/5] END .............neuron1=8, neuron2=12;, score=0.888 total time=   1.2
s
[CV 5/5] END .............neuron1=8, neuron2=12;, score=0.981 total time=   1.4
s
[CV 1/5] END ..............neuron1=8, neuron2=8;, score=0.935 total time=   1.2
s
[CV 2/5] END ..............neuron1=8, neuron2=8;, score=0.917 total time=   1.3
s
[CV 3/5] END ..............neuron1=8, neuron2=8;, score=0.898 total time=   1.3
s
[CV 4/5] END ..............neuron1=8, neuron2=8;, score=0.907 total time=   1.2
s
[CV 5/5] END ..............neuron1=8, neuron2=8;, score=1.000 total time=   1.3
s
[CV 1/5] END .............neuron1=8, neuron2=4;, score=0.880 total time=   1.1
s
[CV 2/5] END .............neuron1=8, neuron2=4;, score=0.824 total time=   1.1
s
[CV 3/5] END .............neuron1=8, neuron2=4;, score=0.880 total time=   1.1
s
[CV 4/5] END .............neuron1=8, neuron2=4;, score=0.897 total time=   1.2
s
[CV 5/5] END .............neuron1=8, neuron2=4;, score=0.991 total time=   1.2
s
```

In [45]:
```
print(gsv_n.best_score_,gsv_n.best_params_)
```

```
0.9313084125518799 {'neuron1': 8, 'neuron2': 8}
```

Train a model with optimum values of hyperparameter

In [46]:
```
# best Parameters
# batch_size = 10
# epochs = 30
# dropout_rate = 0.1
# learning rate = 0.01
# activation_function = tanh
# kernel_initiative = uniform
# neuron1 = 16
# neuron2 = 4
```

In [47]:
```
def creat_model():
    model = Sequential()
    model.add(Dense(16,input_dim=28,kernel_initializer='uniform', activation='tar
    model.add(Dropout(0.1))
    model.add(Dense(4,kernel_initializer='uniform',activation='tanh'))
    model.add(Dropout(0.1))
    model.add(Dense(1,kernel_initializer='uniform',activation='sigmoid'))
    adam = Adam(learning_rate=0.01)
    model.compile(loss='binary_crossentropy', optimizer=adam, metrics=['accuracy'
    return model
```

In [48]:
```
model = KerasClassifier(build_fn=creat_model,batch_size = 10,epochs = 30)
model.fit(X_train,y_train)
```

```
Epoch 1/30
54/54 [==============================] - 0s 981us/step - loss: 0.6733 - accur
acy: 0.5688
Epoch 2/30
54/54 [==============================] - 0s 978us/step - loss: 0.4818 - accur
acy: 0.7937
Epoch 3/30
54/54 [==============================] - 0s 915us/step - loss: 0.4231 - accur
acy: 0.8197
Epoch 4/30
54/54 [==============================] - 0s 901us/step - loss: 0.3927 - accur
acy: 0.8364
Epoch 5/30
54/54 [==============================] - 0s 988us/step - loss: 0.3756 - accur
acy: 0.8364
Epoch 6/30
54/54 [==============================] - 0s 920us/step - loss: 0.3898 - accur
acy: 0.8420
Epoch 7/30
54/54 [==============================] - 0s 1ms/step - loss: 0.3559 - accurac
y: 0.8439
Epoch 8/30
54/54 [==============================] - 0s 1ms/step - loss: 0.3534 - accurac
y: 0.8587
Epoch 9/30
54/54 [==============================] - 0s 1ms/step - loss: 0.3437 - accurac
y: 0.8606
Epoch 10/30
54/54 [==============================] - 0s 1ms/step - loss: 0.3462 - accurac
y: 0.8587
Epoch 11/30
54/54 [==============================] - 0s 1ms/step - loss: 0.3144 - accurac
y: 0.8662
Epoch 12/30
54/54 [==============================] - 0s 1ms/step - loss: 0.2930 - accurac
y: 0.8755
Epoch 13/30
54/54 [==============================] - 0s 847us/step - loss: 0.3047 - accur
acy: 0.8680
Epoch 14/30
54/54 [==============================] - 0s 828us/step - loss: 0.2962 - accur
acy: 0.8680
Epoch 15/30
54/54 [==============================] - 0s 884us/step - loss: 0.2527 - accur
acy: 0.8941
Epoch 16/30
54/54 [==============================] - 0s 884us/step - loss: 0.2918 - accur
acy: 0.8885
Epoch 17/30
54/54 [==============================] - 0s 1ms/step - loss: 0.3033 - accurac
y: 0.8606
Epoch 18/30
54/54 [==============================] - 0s 1ms/step - loss: 0.2610 - accurac
```

```
y: 0.8903
Epoch 19/30
54/54 [==============================] - 0s 1ms/step - loss: 0.2851 - accurac
y: 0.8810
Epoch 20/30
54/54 [==============================] - 0s 1ms/step - loss: 0.2796 - accurac
y: 0.8885
Epoch 21/30
54/54 [==============================] - 0s 884us/step - loss: 0.2495 - accur
acy: 0.8959
Epoch 22/30
54/54 [==============================] - 0s 884us/step - loss: 0.2165 - accur
acy: 0.9164
Epoch 23/30
54/54 [==============================] - 0s 903us/step - loss: 0.2244 - accur
acy: 0.9108
Epoch 24/30
54/54 [==============================] - 0s 903us/step - loss: 0.3942 - accur
acy: 0.8401
Epoch 25/30
54/54 [==============================] - 0s 884us/step - loss: 0.2779 - accur
acy: 0.8959
Epoch 26/30
54/54 [==============================] - 0s 1ms/step - loss: 0.2307 - accurac
y: 0.9164
Epoch 27/30
54/54 [==============================] - 0s 922us/step - loss: 0.2600 - accur
acy: 0.8959
Epoch 28/30
54/54 [==============================] - 0s 866us/step - loss: 0.2267 - accur
acy: 0.9145
Epoch 29/30
54/54 [==============================] - 0s 922us/step - loss: 0.1867 - accur
acy: 0.9238
Epoch 30/30
54/54 [==============================] - 0s 884us/step - loss: 0.2133 - accur
acy: 0.9219
```

Out[48]:
```
<keras.callbacks.History at 0x13d3f1228e0>
```

In [49]:
```
y_predict = model.predict(X_train)
```

In [50]:
```
accuracy_score(y_train,y_predict)
```

Out[50]:
```
0.8996282527881041
```

In [51]:
```
confusion_matrix(y_train,y_predict)
```

Out[51]:
```
array([[215,  54],
       [  0, 269]], dtype=int64)
```

In [52]:
```
print(classification_report(y_train,y_predict))
```

```
              precision    recall  f1-score   support

           0       1.00      0.80      0.89       269
           1       0.83      1.00      0.91       269

    accuracy                           0.90       538
   macro avg       0.92      0.90      0.90       538
weighted avg       0.92      0.90      0.90       538
```

In [53]:
```
# testing data
y_test_pred = model.predict(X_test)
```

In [54]:
```
accuracy_score(y_test,y_test_pred)
```

Out[54]:
```
0.8974358974358975
```

In [55]:
```
confusion_matrix(y_test,y_test_pred)
```

Out[55]:
```
array([[46,  1],
       [15, 94]], dtype=int64)
```

In [56]:
```
print(classification_report(y_test,y_test_pred))
```

```
              precision    recall  f1-score   support

           0       0.75      0.98      0.85        47
           1       0.99      0.86      0.92       109

    accuracy                           0.90       156
   macro avg       0.87      0.92      0.89       156
weighted avg       0.92      0.90      0.90       156
```

Here we create model by using optimum value of hyperparameter it gives a approximately 90% accuracy score

In [ ]:
```
```