Import liabrary

In [35]:

```
import warnings
warnings.filterwarnings('ignore')
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from keras.models import Sequential
from keras.layers import Dense, Activation,Layer,Lambda
import seaborn as sns
from sklearn.model_selection import train_test_split
from matplotlib import pyplot as plt
from mlxtend.plotting import plot_decision_regions
```

Business Problem

Predict the Burn areas of Forest fires with Neural Networks.

Importing dataset

In [2]:

```
burns_data=pd.read_csv("forestfires (1).csv")
burns_data
```

Out[2]:

	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	...	monthfeb	monthjan
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 [3]:

```
pd.set_option('max_columns',None)
```

In [4]:

```
burns_data.head(10)
```

Out[4]:

	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area	dayfri	daymon	daysa
0	mar	fri	86.2	26.2	94.3	5.1	8.2	51	6.7	0.0	0.0	1	0	
1	oct	tue	90.6	35.4	669.1	6.7	18.0	33	0.9	0.0	0.0	0	0	
2	oct	sat	90.6	43.7	686.9	6.7	14.6	33	1.3	0.0	0.0	0	0	
3	mar	fri	91.7	33.3	77.5	9.0	8.3	97	4.0	0.2	0.0	1	0	
4	mar	sun	89.3	51.3	102.2	9.6	11.4	99	1.8	0.0	0.0	0	0	
5	aug	sun	92.3	85.3	488.0	14.7	22.2	29	5.4	0.0	0.0	0	0	
6	aug	mon	92.3	88.9	495.6	8.5	24.1	27	3.1	0.0	0.0	0	1	
7	aug	mon	91.5	145.4	608.2	10.7	8.0	86	2.2	0.0	0.0	0	1	
8	sep	tue	91.0	129.5	692.6	7.0	13.1	63	5.4	0.0	0.0	0	0	
9	sep	sat	92.5	88.0	698.6	7.1	22.8	40	4.0	0.0	0.0	0	0	

Data preprocessing

In [5]:

```
#Hence we have already dummy columns of month and day, so we can drop it.
burns_data=burns_data.drop(['month','day'],axis=1)
```

In [6]:

```
burns_data.loc[burns_data["size_category"]=='small','size_category']=0
burns_data.loc[burns_data["size_category"]=='large','size_category']=1
burns_data["size_category"].value_counts()
```

Out[6]:

```
0    378
1    139
Name: size_category, dtype: int64
```

In [7]:

```
burns_data
```

Out[7]:

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

517 rows × 29 columns

In [8]:

```
burns_data.isna().sum()
```

Out[8]:

```
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 [9]:

```
burns_data.dtypes
```

Out[9]:

```
FFMC             float64
DMC              float64
DC               float64
ISI              float64
temp             float64
RH                 int64
wind             float64
rain             float64
area             float64
dayfri             int64
daymon             int64
daysat             int64
daysun             int64
daythu             int64
daytue             int64
daywed             int64
monthapr           int64
monthaug           int64
monthdec           int64
monthfeb           int64
monthjan           int64
monthjul           int64
monthjun           int64
monthmar           int64
monthmay           int64
monthnov           int64
monthoct           int64
monthsep           int64
size_category      int64
dtype: object
```

In [10]:

```
burns_data.shape
```

Out[10]:

```
(517, 29)
```

In [11]:

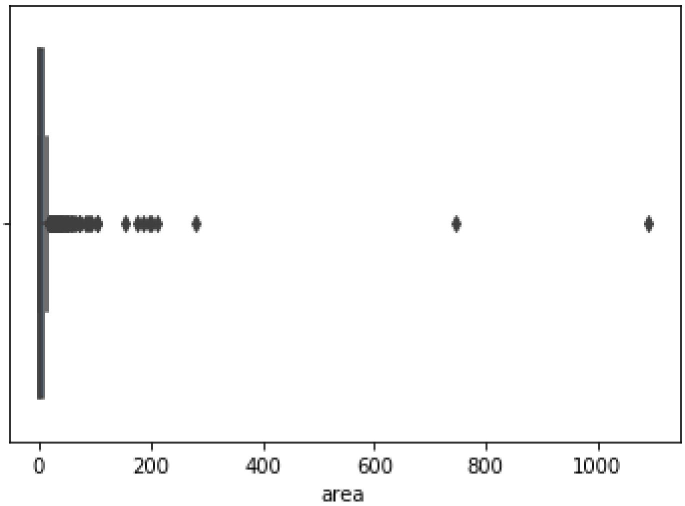
```
burns_data.describe(include='all')
```

Out[11]:

	FFMC	DMC	DC	ISI	temp	RH	wind	
count	517.000000	517.000000	517.000000	517.000000	517.000000	517.000000	517.000000	517
mean	90.644681	110.872340	547.940039	9.021663	18.889168	44.288201	4.017602	(
std	5.520111	64.046482	248.066192	4.559477	5.806625	16.317469	1.791653	(
min	18.700000	1.100000	7.900000	0.000000	2.200000	15.000000	0.400000	(
25%	90.200000	68.600000	437.700000	6.500000	15.500000	33.000000	2.700000	(
50%	91.600000	108.300000	664.200000	8.400000	19.300000	42.000000	4.000000	(
75%	92.900000	142.400000	713.900000	10.800000	22.800000	53.000000	4.900000	(
max	96.200000	291.300000	860.600000	56.100000	33.300000	100.000000	9.400000	6

In [12]:

```
ax = sns.boxplot(burns_data['area'])
```

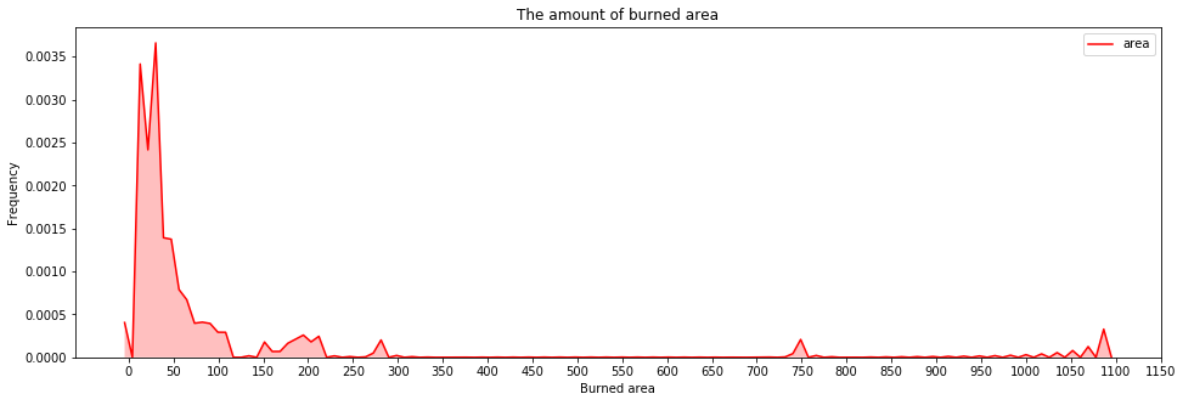


There are three outliers in the data

In [13]:

```
plt.figure(figsize=(16,5))
print("Skew: {}".format(burns_data['area'].skew()))
print("Kurtosis: {}".format(burns_data['area'].kurtosis()))
ax = sns.kdeplot(burns_data['area'],shade=True,color='r')
plt.xticks([i for i in range(0,1200,50)])
plt.xlabel("Burned area")
plt.ylabel("Frequency")
plt.title("The amount of burned area")
plt.show()
```

```
Skew: 12.846933533934868
Kurtosis: 194.1407210942299
```



The data is highly skewed and large Kurtosis value

Forest fires not covered the large area , highly damaged area undrer the 100 hectars of the land.

Majority of fires across in month Aug and Sept

In [14]:

```
#sns.pairplot(burns_data)
```

In [16]:

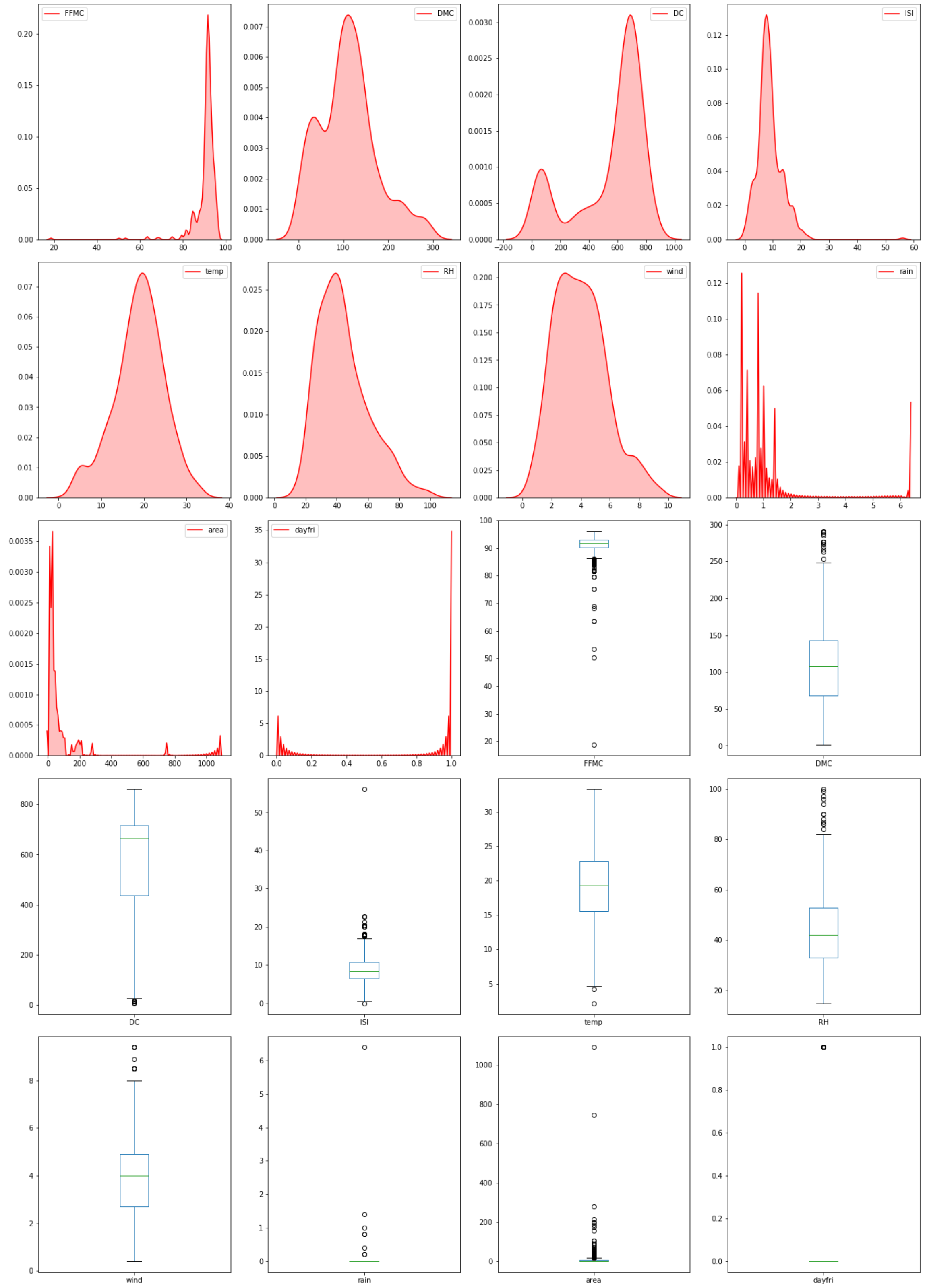
```
dfa = burns_data[burns_data.columns[0:10]]
month_colum = dfa.select_dtypes(include='object').columns.tolist()
```

In [17]:

```
num_columns = dfa.select_dtypes(exclude='object').columns.tolist()
```

In [18]:

```
plt.figure(figsize=(18,40))
for i,col in enumerate(num_columns,1):
    plt.subplot(8,4,i)
    sns.kdeplot(burns_data[col],color='r',shade=True)
    plt.subplot(8,4,i+10)
    burns_data[col].plot.box()
plt.tight_layout()
plt.show()
num_data = burns_data[num_columns]
pd.DataFrame(data=[num_data.skew(),num_data.kurtosis()],index=['skewness','kurtosis'])
```



Out[18]:

	FFMC	DMC	DC	ISI	temp	RH	wind	rain
skewness	-6.575606	0.547498	-1.100445	2.536325	-0.331172	0.862904	0.571001	19.816344
kurtosis	67.066041	0.204822	-0.245244	21.458037	0.136166	0.438183	0.054324	421.295964

In [19]:

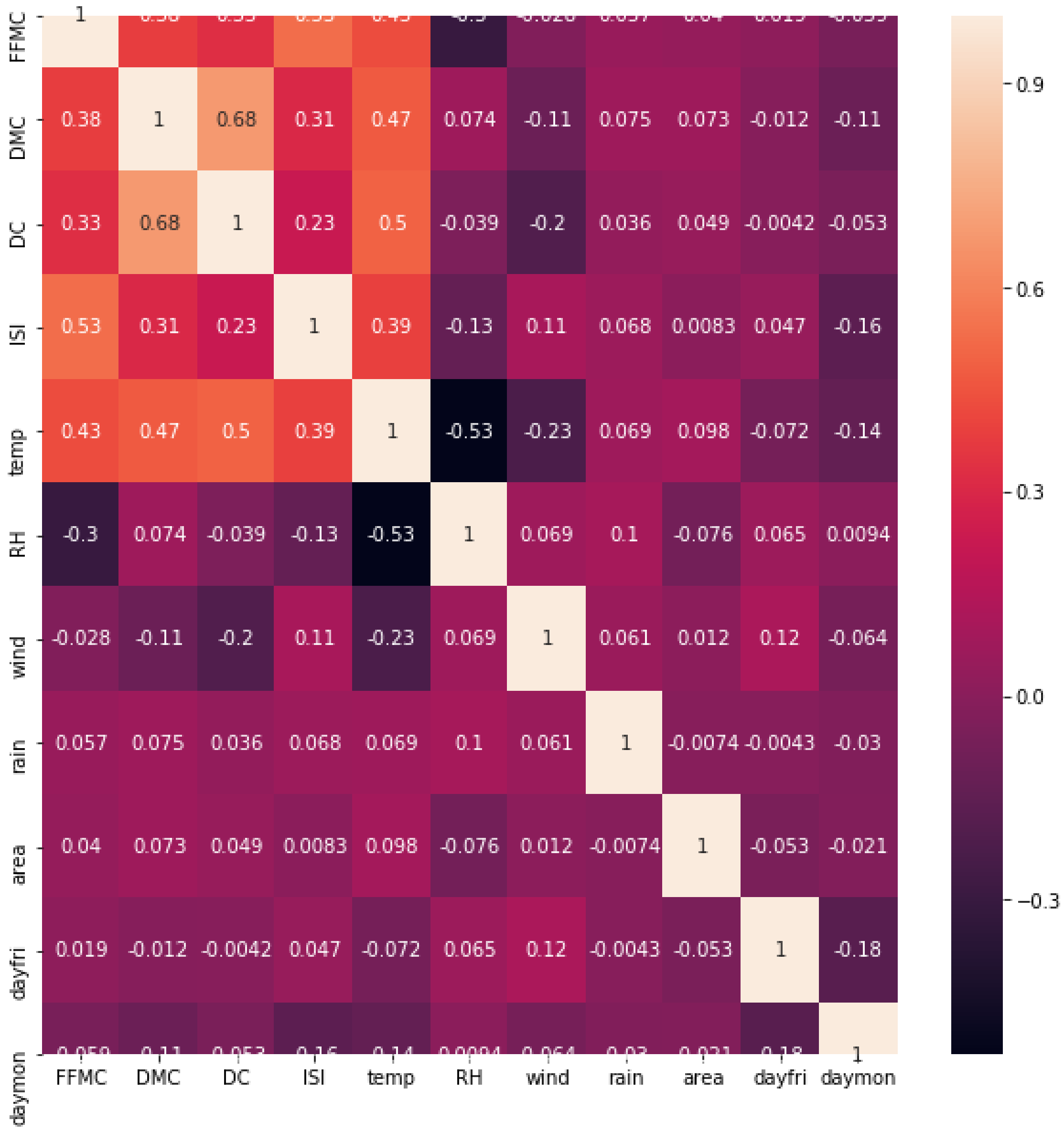
```
corr = burns_data[burns_data.columns[0:11]].corr()
```

In [20]:

```
plt.figure(figsize=(10,10))
sns.heatmap(corr,annot=True)
```

Out[20]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x20234aae048>
```



Neural Networks model

In [21]:

```
X = np.array(burns_data.iloc[:,0:28])
y = np.array(burns_data.iloc[:,28])
```

In [22]:

```
def norm_func(i):
    x = (i-i.min())/(i.max()-i.min())
    return (x)
```

In [23]:

```
X_norm = norm_func(X)
```

In [24]:

```
x_train,x_test,y_train,y_test= train_test_split(X_norm,y, test_size=0.2,stratify = y)
```

In [25]:

```
model = Sequential()
model.add(Dense(12, input_dim=28, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

In [26]:

```
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

In [27]:

```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_1 (Dense)              (None, 12)                348       
_________________________________________________________________
dense_2 (Dense)              (None, 8)                 104       
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 9         
=================================================================
Total params: 461
Trainable params: 461
Non-trainable params: 0
_________________________________________________________________
```

In [28]:

```
model.fit(x_train, y_train, epochs=150, batch_size=10)
```

```
Epoch 1/150
413/413 [==============================] - 0s 685us/step - loss: 0.6622 -
accuracy: 0.7312
Epoch 2/150
413/413 [==============================] - 0s 179us/step - loss: 0.6249 -
accuracy: 0.7312
Epoch 3/150
413/413 [==============================] - 0s 172us/step - loss: 0.6009 -
accuracy: 0.7312
Epoch 4/150
413/413 [==============================] - 0s 164us/step - loss: 0.5895 -
accuracy: 0.7312
Epoch 5/150
413/413 [==============================] - 0s 182us/step - loss: 0.5854 -
accuracy: 0.7312
Epoch 6/150
413/413 [==============================] - 0s 172us/step - loss: 0.5838 -
accuracy: 0.7312
Epoch 7/150
```

In [29]:

```
_, accuracy = model.evaluate(x_train, y_train)
print('Accuracy: %.2f' % (accuracy*100))
```

```
413/413 [==============================] - 0s 87us/step
Accuracy: 95.64
```

In [30]:

```
predictions = model.predict_classes(x_train)
predictions
```

Out[30]:

```
array([[1],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [1],
       [0],
       [1],
       [1],
       [0],
       [0],
       [1],
       [0],
       [0],
       [0],
```

In [31]:

```
predictions1 = model.predict_classes(x_test)
predictions1
```

Out[31]:

```
array([[0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [1],
       [0],
       [1],
       [0],
       [0],
       [0],
       [1],
       [0],
       [1],
       [0],
       [0],
       [1],
       [1],
       [0],
       [1],
       [0],
       [0],
       [0],
       [1],
       [0],
       [0],
       [0],
       [0],
       [0],
       [1],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [1],
       [1],
       [0],
       [0],
       [0],
       [1],
       [0],
       [0],
       [0],
       [0],
```

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

In [32]:

```
for i in range(5):
    print('%s => %d (expected %d)' % (X[i].tolist(), predictions[i], y[i]))
```

```
[86.2, 26.2, 94.3, 5.1, 8.2, 51.0, 6.7, 0.0, 0.0, 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, 1.0, 0.0, 0.0, 0.0, 0.0] => 1
(expected 0)
[90.6, 35.4, 669.1, 6.7, 18.0, 33.0, 0.9, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0] => 0
(expected 0)
[90.6, 43.7, 686.9, 6.7, 14.6, 33.0, 1.3, 0.0, 0.0, 0.0, 0.0, 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, 1.0, 0.0] => 0
(expected 0)
[91.7, 33.3, 77.5, 9.0, 8.3, 97.0, 4.0, 0.2, 0.0, 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, 1.0, 0.0, 0.0, 0.0, 0.0] => 0
(expected 0)
[89.3, 51.3, 102.2, 9.6, 11.4, 99.0, 1.8, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0] => 0
(expected 0)
```

In [33]:

```
from sklearn.metrics import confusion_matrix,classification_report,accuracy_score
print(confusion_matrix(y_test,predictions1))
```

```
[[76  0]
 [ 6 22]]
```

In [34]:

```
print(classification_report(y_test,predictions1))
print(accuracy_score(y_test,predictions1))
```

```
              precision    recall  f1-score   support

           0       0.93      1.00      0.96        76
           1       1.00      0.79      0.88        28

    accuracy                           0.94       104
   macro avg       0.96      0.89      0.92       104
weighted avg       0.95      0.94      0.94       104

0.9423076923076923
```

>>>The End!! <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<