In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
ff_data = pd.read_csv('forestfires.csv')
ff_data
```

Out[2]:

	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	...	monthfeb	monthjan	monthjul	monthjun	monthmar	monthmay
0	mar	fri	86.2	26.2	94.3	5.1	8.2	51	6.7	0.0	...	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	0	0
2	oct	sat	90.6	43.7	686.9	6.7	14.6	33	1.3	0.0	...	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	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	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
512	aug	sun	81.6	56.7	665.6	1.9	27.8	32	2.7	0.0	...	0	0	0	0	0	0
513	aug	sun	81.6	56.7	665.6	1.9	21.9	71	5.8	0.0	...	0	0	0	0	0	0
514	aug	sun	81.6	56.7	665.6	1.9	21.2	70	6.7	0.0	...	0	0	0	0	0	0
515	aug	sat	94.4	146.0	614.7	11.3	25.6	42	4.0	0.0	...	0	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	0	0	0	0

517 rows × 31 columns

In [3]:
```
ff_data_1 = ff_data.iloc[:,2:30]
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
ff_data_norm = scaler.fit_transform(ff_data_1)
ff_data_norm
```

Out[3]:
```
array([[-8.05959472e-01, -1.32332557e+00, -1.83047676e+00, ...,
        -4.40225453e-02, -1.72859706e-01, -7.06081245e-01],
       [-8.10203395e-03, -1.17954077e+00,  4.88890915e-01, ...,
        -4.40225453e-02,  5.78503817e+00, -7.06081245e-01],
       [-8.10203395e-03, -1.04982188e+00,  5.60715454e-01, ...,
        -4.40225453e-02,  5.78503817e+00, -7.06081245e-01],
       ...,
       [-1.64008316e+00, -8.46647711e-01,  4.74768113e-01, ...,
        -4.40225453e-02, -1.72859706e-01, -7.06081245e-01],
       [ 6.80956663e-01,  5.49002541e-01,  2.69382214e-01, ...,
        -4.40225453e-02, -1.72859706e-01, -7.06081245e-01],
       [-2.02087875e+00, -1.68591332e+00, -1.78044169e+00, ...,
         2.27156334e+01, -1.72859706e-01, -7.06081245e-01]])
```

In [4]:
```
#PCA

from sklearn.decomposition import PCA
pca = PCA(n_components=28)
pca_values=pca.fit_transform(ff_data_norm)
pca_values
```

Out[4]:
```
array([[ 3.76670947e+00, -1.32025451e+00, -8.43971398e-01, ...,
        -6.53345819e-02,  4.98037274e-16, -2.73530281e-16],
       [ 3.90786263e-01,  8.31061522e-01, -1.10136513e+00, ...,
         3.42618601e-02, -9.55928328e-15,  1.15055466e-15],
       [ 6.90415596e-01,  1.17774562e+00, -1.22199841e+00, ...,
         2.63235187e-02,  2.58690766e-15, -5.66797432e-17],
       ...,
       [ 9.21634000e-01, -2.64543072e-01,  2.71921606e+00, ...,
        -2.97865814e-01, -1.84247930e-16,  2.36645381e-16],
       [-1.62054896e+00, -9.78838231e-01,  3.31987355e-01, ...,
         3.91949863e-02, -2.30354869e-16,  2.72058887e-16],
       [ 4.07590654e+00, -3.67440726e-01, -2.47151775e-01, ...,
        -2.50420726e-02,  5.70142521e-17,  8.50237385e-17]])
```

In [5]:
```
var = pca.explained_variance_ratio_
var
```

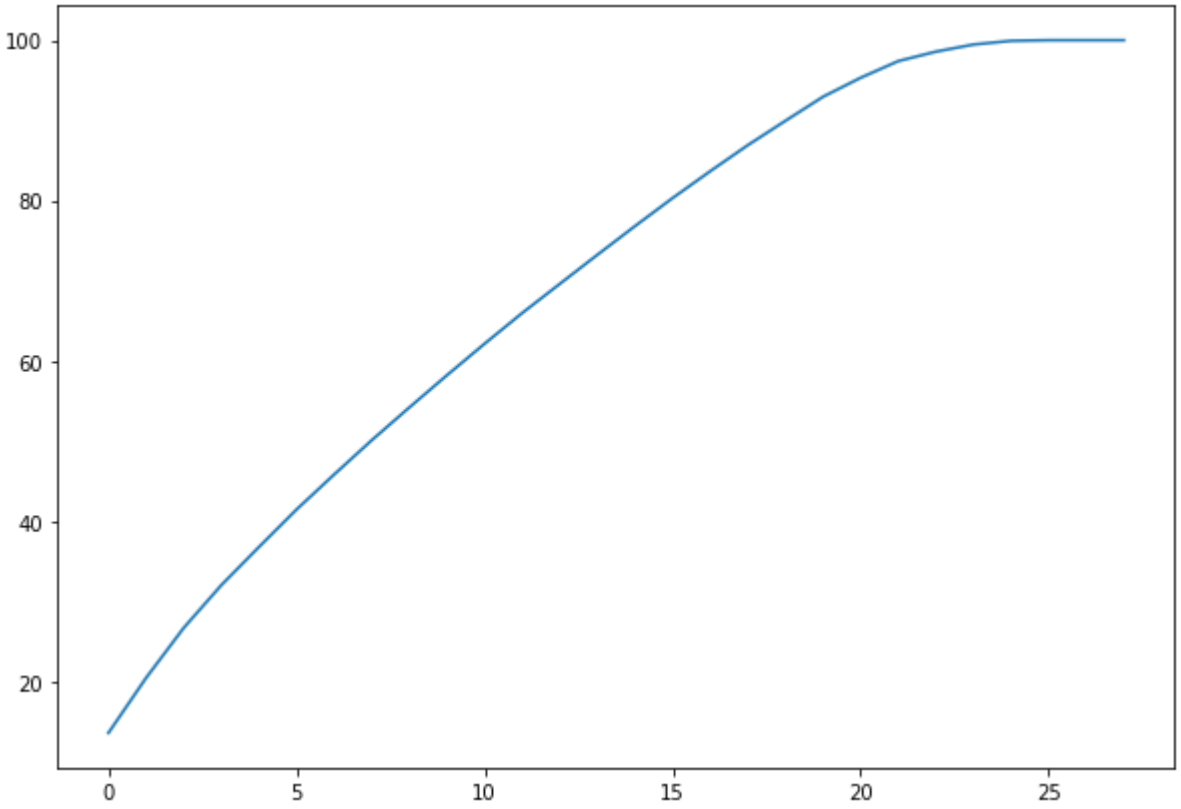
Out[5]:
```
array([1.35522746e-01, 6.85788793e-02, 6.23572652e-02, 5.32713255e-02,
       4.75942360e-02, 4.68009902e-02, 4.37490015e-02, 4.28025164e-02,
       4.08875728e-02, 4.01633268e-02, 3.92926854e-02, 3.83232321e-02,
       3.64221503e-02, 3.63217289e-02, 3.57856782e-02, 3.50087806e-02,
       3.35447704e-02, 3.24777366e-02, 3.04490902e-02, 3.00246758e-02,
       2.37167400e-02, 2.08329788e-02, 1.18357869e-02, 8.88449559e-03,
       4.55347471e-03, 7.98135931e-04, 2.67271490e-32, 1.95971390e-33])
```

In [7]:
```
var1 = np.cumsum(np.round(var,decimals=4)*100)
var1
```

Out[7]:
```
array([13.55, 20.41, 26.65, 31.98, 36.74, 41.42, 45.79, 50.07, 54.16,
       58.18, 62.11, 65.94, 69.58, 73.21, 76.79, 80.29, 83.64, 86.89,
       89.93, 92.93, 95.3 , 97.38, 98.56, 99.45, 99.91, 99.99, 99.99,
       99.99])
```

In [8]:
```
plt.figure(figsize=(10,7))
plt.plot(var1)
```

Out[8]: [<matplotlib.lines.Line2D at 0x1d88a921fd0>]



In [9]:
```
1 #hence here we will choose 24 pcs outoff 28 for further procedure
```

In [10]:
```
finaldf = pd.concat([pd.DataFrame(pca_values[:,0:24],columns=['pc1','pc2','pc3','pc4','pc5','pc6','pc7',
                                                              'pc8','pc9','pc10','pc11','pc12','pc13','pc14'
                                                              'pc15','pc16','pc17','pc18','pc19','pc20','pc2
                                                              'pc22','pc23','pc24']), ff_data[['size_categor
finaldf.size_category.replace(('large','small'),(1,0),inplace=True)
finaldf
```

Out[10]:

	pc1	pc2	pc3	pc4	pc5	pc6	pc7	pc8	pc9	pc10	...	pc16	pc17
0	3.766709	-1.320255	-0.843971	-1.994738	-1.453359	0.693985	0.308104	-0.019764	0.010161	-0.437314	...	-0.197543	-0.021839
1	0.390786	0.831062	-1.101365	1.400671	2.869388	0.965898	-2.795574	0.041095	-0.548879	0.104500	...	-2.503167	0.499649
2	0.690416	1.177746	-1.221998	2.442038	1.090630	0.390801	-1.586675	-2.159336	-0.090580	0.260888	...	-2.545144	-0.658411
3	3.359951	-1.161443	0.385728	-2.118328	-1.949601	1.027664	-0.179422	-0.250227	-0.620329	-1.343189	...	-0.040887	0.017843
4	2.974329	-0.842626	1.327788	0.038086	-1.124763	-0.574676	-0.777155	0.303635	0.861126	-2.024719	...	0.844431	1.014944
...	...	...	...	...	...	...	...	...	...	...	...	...	...
512	-0.087560	0.153964	1.241810	1.536581	0.372425	-1.133422	-0.362287	0.766946	0.818745	-0.289632	...	0.300522	0.513876
513	0.794366	-0.083966	2.670485	0.284995	0.223323	-0.904232	-0.014849	0.107226	1.340049	-0.147246	...	0.342367	0.485571
514	0.921634	-0.264543	2.719216	-0.019643	0.242195	-0.966939	-0.118080	0.123010	1.290364	-0.177553	...	0.332816	0.344047
515	-1.620549	-0.978838	0.331987	1.256638	-0.408164	0.735698	0.815510	-1.398344	0.076379	-0.005814	...	-0.011739	-1.035533
516	4.075907	-0.367441	-0.247152	0.979966	6.792273	5.943666	-1.639583	8.121827	-0.627980	4.953722	...	10.467443	-7.333036

517 rows × 25 columns

In [12]:
```
#splitting data into x and y

array=finaldf.values
x=array[:,0:24]
y=array[:,24]
```

In [16]:
```
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation
from tensorflow.keras.optimizers import SGD
```

In [18]:
```
# 1st Iteration
model=Sequential()
model.add(Dense(12,input_dim=24,activation='relu'))
model.add(Dense(8,activation='relu'))
model.add(Dense(1,activation='sigmoid'))
model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
model.fit(x,y, validation_split=0.3,epochs=50,batch_size=10)
```

```
Epoch 45/50
37/37 [==============================] - 0s 4ms/step - loss: 0.2856 - accuracy: 0.8781 - val_loss: 0.6988 -
val_accuracy: 0.7244
Epoch 46/50
37/37 [==============================] - 0s 4ms/step - loss: 0.2784 - accuracy: 0.8864 - val_loss: 0.6981 -
val_accuracy: 0.7308
Epoch 47/50
37/37 [==============================] - 0s 4ms/step - loss: 0.2748 - accuracy: 0.8837 - val_loss: 0.7082 -
val_accuracy: 0.7372
Epoch 48/50
37/37 [==============================] - 0s 4ms/step - loss: 0.2706 - accuracy: 0.8920 - val_loss: 0.7032 -
val_accuracy: 0.7372
Epoch 49/50
37/37 [==============================] - 0s 6ms/step - loss: 0.2648 - accuracy: 0.8920 - val_loss: 0.7067 -
val_accuracy: 0.7372
Epoch 50/50
37/37 [==============================] - 0s 6ms/step - loss: 0.2598 - accuracy: 0.8975 - val_loss: 0.7089 -
val_accuracy: 0.7372
```

Out[18]:
```
<keras.callbacks.History at 0x1d89585caf0>
```

In [19]:
```
#accuracy of model
scores=model.evaluate(x,y)
```

```
17/17 [==============================] - 0s 3ms/step - loss: 0.3906 - accuracy: 0.8530
```

In [20]:
```
print("%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
```

```
accuracy: 85.30%
```

In [21]:
```
# 2nd Iteration
model1=Sequential()
model1.add(Dense(12,input_dim=24,activation='sigmoid'))
model1.add(Dense(8,activation='sigmoid'))
model1.add(Dense(1,activation='relu'))
model1.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
model1.fit(x, y, validation_split=0.3, epochs=40, batch_size=15)
```

```
Epoch 35/40
25/25 [==============================] - 0s 7ms/step - loss: 0.4378 - accuracy: 0.7950 - val_loss: 0.7715 -
val_accuracy: 0.6987
Epoch 36/40
25/25 [==============================] - 0s 6ms/step - loss: 0.4344 - accuracy: 0.7978 - val_loss: 0.7751 -
val_accuracy: 0.6987
Epoch 37/40
25/25 [==============================] - 0s 7ms/step - loss: 0.4315 - accuracy: 0.7950 - val_loss: 0.7791 -
val_accuracy: 0.7051
Epoch 38/40
25/25 [==============================] - 0s 6ms/step - loss: 0.4344 - accuracy: 0.8006 - val_loss: 0.6938 -
val_accuracy: 0.7051
Epoch 39/40
25/25 [==============================] - 0s 7ms/step - loss: 0.4251 - accuracy: 0.8006 - val_loss: 0.8643 -
val_accuracy: 0.7051
Epoch 40/40

25/25 [==============================] - 0s 7ms/step - loss: 0.4233 - accuracy: 0.7978 - val_loss: 0.9330 -
val_accuracy: 0.7051
```

In [22]:
```
#model accuracy
scores1=model1.evaluate(x,y)
print("%s: %.2f%%" % (model1.metrics_names[1], scores1[1]*100))
```

```
17/17 [==============================] - 0s 4ms/step - loss: 0.5735 - accuracy: 0.7698
accuracy: 76.98%
```

In [23]:
```
# 3rd Iteration
model2=Sequential()
model2.add(Dense(12,input_dim=24,activation='relu'))
model2.add(Dense(8,activation='relu'))
model2.add(Dense(1,activation='relu'))
model2.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
model2.fit(x,y,epochs=40, validation_split=0.3,batch_size=15)
```

```
Epoch 1/40
25/25 [==============================] - 1s 18ms/step - loss: 2.2775 - accuracy: 0.7202 - val_loss: 2.9356
- val_accuracy: 0.5962
Epoch 2/40
25/25 [==============================] - 0s 6ms/step - loss: 2.1896 - accuracy: 0.7285 - val_loss: 2.7511 -
val_accuracy: 0.5897
Epoch 3/40
25/25 [==============================] - 0s 8ms/step - loss: 2.1700 - accuracy: 0.7341 - val_loss: 2.8751 -
val_accuracy: 0.5962
Epoch 4/40
25/25 [==============================] - 0s 6ms/step - loss: 2.0251 - accuracy: 0.7396 - val_loss: 2.3615 -
val_accuracy: 0.5962
Epoch 5/40
25/25 [==============================] - 0s 8ms/step - loss: 1.5700 - accuracy: 0.7424 - val_loss: 2.0490 -
val_accuracy: 0.5897
Epoch 6/40
25/25 [==============================] - 0s 6ms/step - loss: 1.2480 - accuracy: 0.7452 - val_loss: 2.0093 -
val_accuracy: 0.5833
Epoch 7/40
```

In [24]:
```
#model accuracy
scores2=model2.evaluate(x,y)
print("%s: %.2f%%" % (model2.metrics_names[1], scores2[1]*100))
```

```
17/17 [==============================] - 0s 5ms/step - loss: 1.0419 - accuracy: 0.7756
accuracy: 77.56%
```

best accuracy is 85.30%