FOREST FIRES

In [133]:
```
# Import the libraries
import pandas as pd
import numpy as np

from keras.models import Sequential
from keras.layers import Dense, Activation,Layer,Lambda
```

In [134]:
```
# Import the dataset
ForestFires = pd.read_csv("C:\\Users\\nishi\\Desktop\\Assignments\\Neural_Network
```

In [135]:
```
ForestFires
```

Out[135]:

	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 [136]:
```
#As dummy variables are already created,remove the month and day columns
ForestFires.drop(["month","day"],axis=1,inplace = True)
```

In [137]:
```
ForestFires["size_category"].value_counts()
ForestFires.isnull().sum()
ForestFires.describe()
```

Out[137]:

monthfeb	monthjan	monthjul	monthjun	monthmar	monthmay	monthnov	monthoct	m
517.000000	517.000000	517.000000	517.000000	517.000000	517.000000	517.000000	517.000000	517
0.038685	0.003868	0.061896	0.032882	0.104449	0.003868	0.001934	0.029014	0
0.193029	0.062137	0.241199	0.178500	0.306138	0.062137	0.043980	0.168007	0
0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0
0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0
0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0
0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1
1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1

In [138]:
```
##Take small value small as 0 and large as 1
ForestFires.loc[ForestFires["size_category"]=='small','size_category']=0
ForestFires.loc[ForestFires["size_category"]=='large','size_category']=1
ForestFires["size_category"].value_counts()
```

Out[138]:
```
0    378
1    139
Name: size_category, dtype: int64
```

In [139]:
```
#Normalization being done.
def norm_func(i):
     x = (i-i.min())/(i.max()-i.min())
     return (x)

predictors = ForestFires.iloc[:,0:28]
target = ForestFires.iloc[:,28]

predictors1 = norm_func(predictors)
```

In [140]:
```
#data = pd.concat([predictors1,target],axis=1)
```

In [141]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test= train_test_split(predictors1,target, test_size=0.3
```

In [142]:
```
def prep_model(hidden_dim):
    model = Sequential()
    for i in range(1,len(hidden_dim)-1):
        if (i==1):
            model.add(Dense(hidden_dim[i],input_dim=hidden_dim[0],activation="rel
        else:
            model.add(Dense(hidden_dim[i],activation="relu"))
    model.add(Dense(hidden_dim[-1],kernel_initializer="normal",activation="sigmoi
    model.compile(loss="binary_crossentropy",optimizer = "rmsprop",metrics = ["ac
    return model
```

In [143]:
```
y_train = pd.DataFrame(y_train)
```

In [144]:
```
first_model = prep_model([28,50,40,20,1])
first_model.fit(np.array(x_train).astype(np.float32),np.array(y_train).astype(np.
pred_train = first_model.predict(np.array(x_train))
```

```
Epoch 1/500
12/12 [==============================] - 2s 3ms/step - loss: 0.6624 - accurac
y: 0.7119
Epoch 2/500
12/12 [==============================] - 0s 4ms/step - loss: 0.6080 - accurac
y: 0.7313
Epoch 3/500
12/12 [==============================] - 0s 4ms/step - loss: 0.5891 - accurac
y: 0.7313
Epoch 4/500
12/12 [==============================] - 0s 4ms/step - loss: 0.5843 - accurac
y: 0.7313
Epoch 5/500
12/12 [==============================] - 0s 4ms/step - loss: 0.5828 - accurac
y: 0.7313
Epoch 6/500
12/12 [==============================] - 0s 4ms/step - loss: 0.5812 - accurac
y: 0.7313
Epoch 7/500
12/12 [==============================] - 0s 4ms/step - loss: 0.5805 - accurac
```

In [145]:
```
#Converting the predicted values to series
pred_train = pd.Series([i[0] for i in pred_train])
size = ["small","large"]
pred_train_class = pd.Series(["small"]*361)
pred_train_class[[i>0.5 for i in pred_train]]= "large"
```

In [146]:
```
train = pd.concat([x_train,y_train],axis=1)
train["size_category"].value_counts()
```

Out[146]:
```
0    264
1     97
Name: size_category, dtype: int64
```

In [147]:

```
#For training data
from sklearn.metrics import confusion_matrix
train["original_class"] = "small"
train.loc[train["size_category"]==1,"original_class"] = "large"
train.original_class.value_counts()
confusion_matrix(pred_train_class,train["original_class"])
np.mean(pred_train_class==pd.Series(train["original_class"]).reset_index(drop=Tru
pd.crosstab(pred_train_class,pd.Series(train["original_class"]).reset_index(drop=
```

Out[147]:

original_class	large	small
row_0		
large	97	9
small	0	255

In [148]:

```
#For test data
pred_test = first_model.predict(np.array(x_test))
pred_test = pd.Series([i[0] for i in pred_test])
pred_test_class = pd.Series(["small"]*156)
pred_test_class[[i>0.5 for i in pred_test]] = "large"
test =pd.concat([x_test,y_test],axis=1)
test["original_class"]="small"
test.loc[test["size_category"]==1,"original_class"] = "large"

test["original_class"].value_counts()
np.mean(pred_test_class==pd.Series(test["original_class"]).reset_index(drop=True)
confusion_matrix(pred_test_class,test["original_class"])
pd.crosstab(pred_test_class,pd.Series(test["original_class"]).reset_index(drop=Tr
```

Out[148]:

original_class	large	small
row_0		
large	36	21
small	6	93