PREDICT THE BURNED AREA OF FOREST FIRES WITH NEURAL NETWORKS

```
import pandas as pd
```

```
data = pd.read_csv('forestfires (1).csv')
data
```

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

517 rows × 31 columns

```
data=data.drop(['month','day'],axis=1)
data
```

	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area	dayfri	daymon	daysat	da
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	1	
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	

```
data.isna().sum()
```

```
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
```

```
data.dtypes
```

```
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     object
dtype: object
```

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

	FFMC	DMC	DC	ISI	temp	RH	w
count	517.000000	517.000000	517.000000	517.000000	517.000000	517.000000	517.000
unique	NaN	NaN	NaN	NaN	NaN	NaN	N
top	NaN	NaN	NaN	NaN	NaN	NaN	N
freq	NaN	NaN	NaN	NaN	NaN	NaN	N
mean	90.644681	110.872340	547.940039	9.021663	18.889168	44.288201	4.017
std	5.520111	64.046482	248.066192	4.559477	5.806625	16.317469	1.791
min	18.700000	1.100000	7.900000	0.000000	2.200000	15.000000	0.400
25%	90.200000	68.600000	437.700000	6.500000	15.500000	33.000000	2.700
50%	91.600000	108.300000	664.200000	8.400000	19.300000	42.000000	4.000
75%	92.900000	142.400000	713.900000	10.800000	22.800000	53.000000	4.900
max	96.200000	291.300000	860.600000	56.100000	33.300000	100.000000	9.400

```
# label Encoder
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
data['size_category_encoded']=le.fit_transform(data['size_category'])
data
# data.loc[data['size_category']=='small','size_category']=0
# data['size_category']
```

	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area	dayfri	daymon	daysat	da
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	1	
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	1	
516	79.5	3.0	106.7	1.1	11.8	31	4.5	0.0	0.00	0	0	0	

517 rows × 30 columns

```
X = data.drop(['size_category','size_category_encoded'],axis=1)
y = data[['size_category_encoded']]


# nornalization
def norm_fun(i):
  x = (i-i.min())/(i.max()-i.min())
  return(x)
scaled_x = norm_fun(X)
scaled_x
```

	FFMC	DMC	DC	ISI	temp	RH	wind	rain	
0	0.870968	0.086492	0.101325	0.090909	0.192926	0.423529	0.700000	0.00000	0.0(
1	0.927742	0.118194	0.775419	0.119430	0.508039	0.211765	0.055556	0.00000	0.0(

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(scaled_x,y,test_size=0.2,random_state=12)
```

4	0.910968	0.172984	0.110590	0.171123	0.295820	0.988235	0.155556	0.00000	0.0(

```
import tensorflow as tf
from tensorflow import keras
```

512	0.811613	0.191592	0.771315	0.033868	0.823151	0.200000	0.255556	0.00000	0.0(

```
#Model Building
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Dense(42,input_dim=28, activation='relu'))   # Hidden Layer
model.add(tf.keras.layers.Dense(10,activation='linear'))               # Hidden Layer
model.add(tf.keras.layers.Dense(10,activation='relu'))                 # Hidden Layer
model.add(tf.keras.layers.Dense(1))                                    # Output Layer

model.summary()
```

```
Model: "sequential_17"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_48 (Dense)            (None, 42)                1218

 dense_49 (Dense)            (None, 10)                430

 dense_50 (Dense)            (None, 10)                110

 dense_51 (Dense)            (None, 1)                 11

=================================================================
Total params: 1,769
Trainable params: 1,769
Non-trainable params: 0
_________________________________________________________________
```

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


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

```
Epoch 1/10
13/13 [==============================] - 1s 2ms/step - loss: 3.4753 - accuracy: 0.25
Epoch 2/10
13/13 [==============================] - 0s 2ms/step - loss: 1.0607 - accuracy: 0.34
Epoch 3/10
13/13 [==============================] - 0s 2ms/step - loss: 0.7596 - accuracy: 0.46
Epoch 4/10
13/13 [==============================] - 0s 2ms/step - loss: 0.6884 - accuracy: 0.58
Epoch 5/10
13/13 [==============================] - 0s 2ms/step - loss: 0.6903 - accuracy: 0.67
```

```
Epoch 6/10
13/13 [==============================] - 0s 2ms/step - loss: 0.6756 - accuracy: 0.69
Epoch 7/10
13/13 [==============================] - 0s 2ms/step - loss: 0.6330 - accuracy: 0.70
Epoch 8/10
13/13 [==============================] - 0s 2ms/step - loss: 0.6194 - accuracy: 0.72
Epoch 9/10
13/13 [==============================] - 0s 2ms/step - loss: 0.6088 - accuracy: 0.72
Epoch 10/10
13/13 [==============================] - 0s 2ms/step - loss: 0.6012 - accuracy: 0.74
<keras.callbacks.History at 0x7f9b9cc8cf50>
```

```
result = model.evaluate(x_test,y_test)
result
```

```
4/4 [==============================] - 0s 3ms/step - loss: 0.6764 - accuracy: 0.6635
[0.676379919052124, 0.6634615659713745]
```

```
print('accuracy : ',round(result[1],4))
print('Loss     : ',round(result[0],4))
```

```
accuracy :  0.6635
Loss     :  0.6764
```