Problem statement:

PREDICT THE BURNED AREA OF FOREST FIRES WITH NEURAL NETWORKS

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from keras.wrappers.scikit_learn import KerasRegressor
from keras.layers import Dense,Dropout
from sklearn.metrics import accuracy_score,mean_absolute_error,mean_squared_error
from sklearn.model_selection import GridSearchCV,KFold
from keras.models import Sequential
from tensorflow.keras.optimizers import Adam,RMSprop
from sklearn import preprocessing
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
```

In [2]:

```
#load data
df = pd.read_csv("forestfires.csv")
df.head(20)
```

Out[2]:

	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	...	monthfeb	monthjan	m
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	
5	aug	sun	92.3	85.3	488.0	14.7	22.2	29	5.4	0.0	...	0	0	
6	aug	mon	92.3	88.9	495.6	8.5	24.1	27	3.1	0.0	...	0	0	
7	aug	mon	91.5	145.4	608.2	10.7	8.0	86	2.2	0.0	...	0	0	
8	sep	tue	91.0	129.5	692.6	7.0	13.1	63	5.4	0.0	...	0	0	
9	sep	sat	92.5	88.0	698.6	7.1	22.8	40	4.0	0.0	...	0	0	
10	sep	sat	92.5	88.0	698.6	7.1	17.8	51	7.2	0.0	...	0	0	
11	sep	sat	92.8	73.2	713.0	22.6	19.3	38	4.0	0.0	...	0	0	
12	aug	fri	63.5	70.8	665.3	0.8	17.0	72	6.7	0.0	...	0	0	
13	sep	mon	90.9	126.5	686.5	7.0	21.3	42	2.2	0.0	...	0	0	
14	sep	wed	92.9	133.3	699.6	9.2	26.4	21	4.5	0.0	...	0	0	
15	sep	fri	93.3	141.2	713.9	13.9	22.9	44	5.4	0.0	...	0	0	
16	mar	sat	91.7	35.8	80.8	7.8	15.1	27	5.4	0.0	...	0	0	
17	oct	mon	84.9	32.8	664.2	3.0	16.7	47	4.9	0.0	...	0	0	
18	mar	wed	89.2	27.9	70.8	6.3	15.9	35	4.0	0.0	...	0	0	
19	apr	sat	86.3	27.4	97.1	5.1	9.3	44	4.5	0.0	...	0	0	

20 rows × 31 columns

In [3]:

```
df.shape
```

Out[3]:

```
(517, 31)
```

In [4]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 517 entries, 0 to 516
Data columns (total 31 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   month          517 non-null    object 
 1   day            517 non-null    object 
 2   FFMC           517 non-null    float64
 3   DMC            517 non-null    float64
 4   DC             517 non-null    float64
 5   ISI            517 non-null    float64
 6   temp           517 non-null    float64
 7   RH             517 non-null    int64  
 8   wind           517 non-null    float64
 9   rain           517 non-null    float64
 10  area           517 non-null    float64
 11  dayfri         517 non-null    int64  
 12  daymon         517 non-null    int64  
 13  daysat         517 non-null    int64  
 14  daysun         517 non-null    int64  
 15  daythu         517 non-null    int64  
 16  daytue         517 non-null    int64  
 17  daywed         517 non-null    int64  
 18  monthapr       517 non-null    int64  
 19  monthaug       517 non-null    int64  
 20  monthdec       517 non-null    int64  
 21  monthfeb       517 non-null    int64  
 22  monthjan       517 non-null    int64  
 23  monthjul       517 non-null    int64  
 24  monthjun       517 non-null    int64  
 25  monthmar       517 non-null    int64  
 26  monthmay       517 non-null    int64  
 27  monthnov       517 non-null    int64  
 28  monthoct       517 non-null    int64  
 29  monthsep       517 non-null    int64  
 30  size_category  517 non-null    object 
dtypes: float64(8), int64(20), object(3)
memory usage: 125.3+ KB
```

In [5]:

```
df.describe()
```

Out[5]:

	FFMC	DMC	DC	ISI	temp	RH	wind	
count	517.000000	517.000000	517.000000	517.000000	517.000000	517.000000	517.000000	51
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	

8 rows × 28 columns

In [6]:

```
# find categorical variables in training data set

traincategorical = [var for var in df.columns if df[var].dtype=='O']

print('There are {} categorical variables\n'.format(len(traincategorical)))

print('The categorical variables are :\n\n', traincategorical)
```

```
There are 3 categorical variables

The categorical variables are :

 ['month', 'day', 'size_category']
```

In [7]:

```
label_encoder = preprocessing.LabelEncoder()
df['month']= label_encoder.fit_transform(df['month'])
df['day']= label_encoder.fit_transform(df['day'])
df['size_category']= label_encoder.fit_transform(df['size_category'])
```

In [8]:

```
df
```

Out[8]:

	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	...	monthfeb	monthjan	m
0	7	0	86.2	26.2	94.3	5.1	8.2	51	6.7	0.0	...	0	0	
1	10	5	90.6	35.4	669.1	6.7	18.0	33	0.9	0.0	...	0	0	
2	10	2	90.6	43.7	686.9	6.7	14.6	33	1.3	0.0	...	0	0	
3	7	0	91.7	33.3	77.5	9.0	8.3	97	4.0	0.2	...	0	0	
4	7	3	89.3	51.3	102.2	9.6	11.4	99	1.8	0.0	...	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
512	1	3	81.6	56.7	665.6	1.9	27.8	32	2.7	0.0	...	0	0	
513	1	3	81.6	56.7	665.6	1.9	21.9	71	5.8	0.0	...	0	0	
514	1	3	81.6	56.7	665.6	1.9	21.2	70	6.7	0.0	...	0	0	
515	1	2	94.4	146.0	614.7	11.3	25.6	42	4.0	0.0	...	0	0	
516	9	5	79.5	3.0	106.7	1.1	11.8	31	4.5	0.0	...	0	0	

517 rows × 31 columns

In [9]:

```
x = df.iloc[:,0:30]
y = df.iloc[:,-1]
```

In [10]:

```
y
```

Out[10]:

```
0      1
1      1
2      1
3      1
4      1
      ..
512    0
513    0
514    0
515    1
516    1
Name: size_category, Length: 517, dtype: int32
```

In [11]:

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

Out[11]:

	month	day	FFMC	DMC	DC	ISI	temp	RH	wind
0	0.636364	0.000000	0.870968	0.086492	0.101325	0.090909	0.192926	0.423529	0.700000
1	0.909091	0.833333	0.927742	0.118194	0.775419	0.119430	0.508039	0.211765	0.055556
2	0.909091	0.333333	0.927742	0.146795	0.796294	0.119430	0.398714	0.211765	0.100000
3	0.636364	0.000000	0.941935	0.110958	0.081623	0.160428	0.196141	0.964706	0.400000
4	0.636364	0.500000	0.910968	0.172984	0.110590	0.171123	0.295820	0.988235	0.155556
...	...	...	...	...	...	...	...	...	...
512	0.090909	0.500000	0.811613	0.191592	0.771315	0.033868	0.823151	0.200000	0.255556
513	0.090909	0.500000	0.811613	0.191592	0.771315	0.033868	0.633441	0.658824	0.600000
514	0.090909	0.500000	0.811613	0.191592	0.771315	0.033868	0.610932	0.647059	0.700000
515	0.090909	0.333333	0.976774	0.499311	0.711622	0.201426	0.752412	0.317647	0.400000
516	0.818182	0.833333	0.784516	0.006547	0.115867	0.019608	0.308682	0.188235	0.455556

517 rows × 30 columns

In [12]:

```
# Splitting data into training and testing data set
x_train, x_test,y_train,y_test = train_test_split(X,y, test_size=0.2,random_state=50)
```

In [13]:

```
model = Sequential()
model.add(layers.Dense(8, input_dim=30,  activation='relu'))
model.add(layers.Dense(4,  activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
```

In [14]:

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

In [15]:

```
# Fit the model
fit_model=model.fit(x_train, y_train, validation_split=0.3, epochs=120, batch_size=10)
```

```
Epoch 1/120
29/29 [==============================] - 1s 8ms/step - loss: 0.6888 - accu
racy: 0.6055 - val_loss: 0.6773 - val_accuracy: 0.7339
Epoch 2/120
29/29 [==============================] - 0s 2ms/step - loss: 0.6787 - accu
racy: 0.6782 - val_loss: 0.6626 - val_accuracy: 0.7903
Epoch 3/120
29/29 [==============================] - 0s 2ms/step - loss: 0.6697 - accu
racy: 0.6886 - val_loss: 0.6494 - val_accuracy: 0.7903
Epoch 4/120
29/29 [==============================] - 0s 2ms/step - loss: 0.6606 - accu
racy: 0.6886 - val_loss: 0.6375 - val_accuracy: 0.7903
Epoch 5/120
29/29 [==============================] - 0s 2ms/step - loss: 0.6514 - accu
racy: 0.6886 - val_loss: 0.6188 - val_accuracy: 0.7903
Epoch 6/120
29/29 [==============================] - 0s 2ms/step - loss: 0.6404 - accu
racy: 0.6886 - val_loss: 0.5996 - val_accuracy: 0.7903
Epoch 7/120
```

In [16]:

```
y_pred = model.predict(x_train)
```

```
13/13 [==============================] - 0s 834us/step
```

In [17]:

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 8)                 248

 dense_1 (Dense)             (None, 4)                 36

 dense_2 (Dense)             (None, 1)                 5

=================================================================
Total params: 289
Trainable params: 289
Non-trainable params: 0
_________________________________________________________________
```

In [18]:

```
mean_absolute_error(y_train,y_pred)
```

Out[18]:

```
0.30583577850666627
```

In [19]:

```
mean_squared_error(y_train,y_pred)
```

Out[19]:

```
0.14567592323676
```

In [20]:

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

```
4/4 [==============================] - 0s 1ms/step - loss: 0.5840 - accurac
y: 0.7596
```

In [21]:

```
y_test_pred = model.predict(x_test)
```

```
4/4 [==============================] - 0s 1ms/step
```

In [22]:

```
mean_absolute_error(y_test,y_test_pred)
```

Out[22]:

```
0.3560174982278393
```

In [23]:

```
mean_squared_error(y_test,y_test_pred)
```

Out[23]:

```
0.18699287239479043
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: