In [3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn.preprocessing import OrdinalEncoder
from tensorflow.keras.models import Sequential
from keras.layers import Dense
import tensorflow as tf
from sklearn.model_selection import train_test_split

import warnings
warnings.filterwarnings('ignore')
```

In [7]:
```
data = pd.read_csv("forestfires (1).csv")
data
```

Out[7]:

	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 [8]:
```
data.shape
```

Out[8]:
```
(517, 31)
```

In [9]:
```
data.dtypes
```

Out[9]:
```
month            object
day              object
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
```

In [10]:
```
data.isna().sum()
```

Out[10]:
```
month           0
day             0
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 [16]:
```
data.describe(include = 'all')
```

Out[16]:

	month	day	FFMC	DMC	DC	ISI	temp	RH
count	517	517	517.000000	517.000000	517.000000	517.000000	517.000000	517.000000
unique	12	7	NaN	NaN	NaN	NaN	NaN	NaN
top	aug	sun	NaN	NaN	NaN	NaN	NaN	NaN
freq	184	95	NaN	NaN	NaN	NaN	NaN	NaN
mean	NaN	NaN	90.644681	110.872340	547.940039	9.021663	18.889168	44.288201
std	NaN	NaN	5.520111	64.046482	248.066192	4.559477	5.806625	16.317469
min	NaN	NaN	18.700000	1.100000	7.900000	0.000000	2.200000	15.000000
25%	NaN	NaN	90.200000	68.600000	437.700000	6.500000	15.500000	33.000000
50%	NaN	NaN	91.600000	108.300000	664.200000	8.400000	19.300000	42.000000
75%	NaN	NaN	92.900000	142.400000	713.900000	10.800000	22.800000	53.000000
max	NaN	NaN	96.200000	291.300000	860.600000	56.100000	33.300000	100.000000

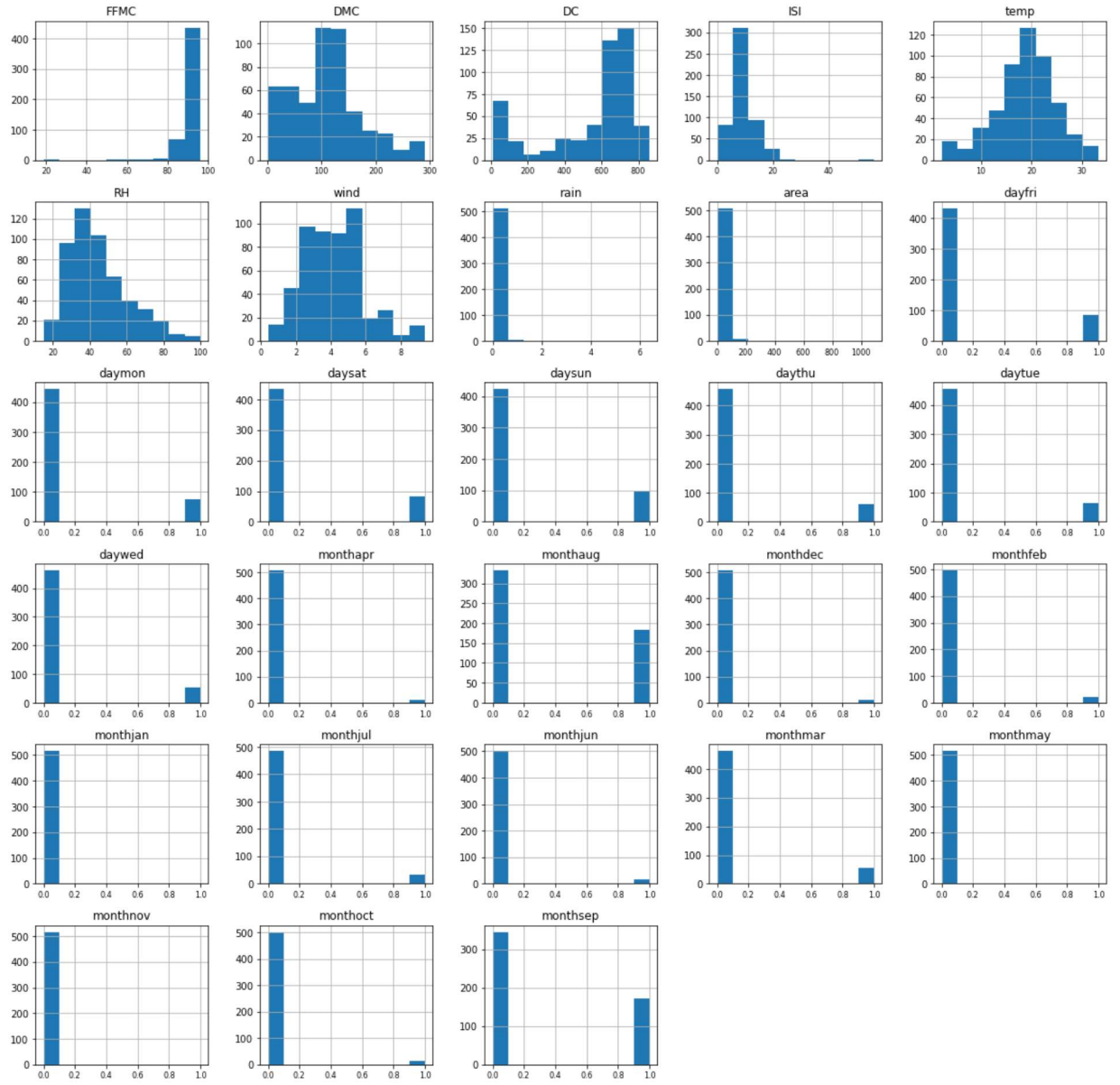
11 rows × 31 columns

In [11]:
```
data.describe(include='object')
```

Out[11]:

	month	day	size_category
count	517	517	517
unique	12	7	2
top	aug	sun	small
freq	184	95	378

In [18]:
```
data.hist(figsize = (20,20),xlabelsize = 8 , ylabelsize=10)
plt.show()
```



In [19]:
```
data = data.drop(['month','day'],axis =1 )
```

In [20]:
```
mapping = {'small':0 ,'large' :1}
```

In [21]:
```
data = data.replace(mapping)
```

In [22]:
```
data
```

Out[22]:

	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area	dayfri	...	monthfeb	monthjan	mon
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	0	
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	0	
516	79.5	3.0	106.7	1.1	11.8	31	4.5	0.0	0.00	0	...	0	0	

517 rows × 29 columns

In [23]:
```
x = np.array(data.iloc[:,0:28])
y = np.array(data.iloc[:,28])
```

In [26]:
```
def norm_func(i):
    x = (i-i.min())/(i.max()-i.min())
    return(x)
```

In [27]:
```
x_norm = norm_func(x)
```

In [31]:
```
in,x_test,y_train,y_test= train_test_split(x_norm,y, test_size=0.2,stratify = y)
```

In [32]:
```
model = Sequential()
model.add(Dense(8, input_dim=28, activation='linear'))
model.add(Dense(4, activation='tanh'))
model.add(Dense(1, activation='sigmoid'))
```

In [33]:
```
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

In [34]:
```
history = model.fit(x_train,y_train,validation_split=0.3,epochs = 120 ,batch_size
```

```
Epoch 2/120
29/29 [==============================] - 0s 4ms/step - loss: 0.6674 - accurac
y: 0.7336 - val_loss: 0.6514 - val_accuracy: 0.7258
Epoch 3/120
29/29 [==============================] - 0s 4ms/step - loss: 0.6346 - accurac
y: 0.7336 - val_loss: 0.6241 - val_accuracy: 0.7258
Epoch 4/120
29/29 [==============================] - 0s 5ms/step - loss: 0.6099 - accurac
y: 0.7336 - val_loss: 0.6067 - val_accuracy: 0.7258
Epoch 5/120
29/29 [==============================] - 0s 5ms/step - loss: 0.5946 - accurac
y: 0.7336 - val_loss: 0.5977 - val_accuracy: 0.7258
Epoch 6/120
29/29 [==============================] - 0s 5ms/step - loss: 0.5857 - accurac
y: 0.7336 - val_loss: 0.5940 - val_accuracy: 0.7258
Epoch 7/120
29/29 [==============================] - 0s 5ms/step - loss: 0.5817 - accurac
y: 0.7336 - val_loss: 0.5927 - val_accuracy: 0.7258
Epoch 8/120
29/29 [==============================] - 0s 5ms/step - loss: 0.5817 - accurac
```

In [36]:
```
scores = model.evaluate(x_train,y_train)
print("%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
```

```
13/13 [==============================] - 0s 2ms/step - loss: 0.2388 - accuracy:
0.8910
accuracy: 89.10%
```

In [37]:
```
scores = model.evaluate(x_test, y_test)
print("%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
```

```
4/4 [==============================] - 0s 3ms/step - loss: 0.2422 - accuracy:
0.9038
accuracy: 90.38%
```

In [ ]:
```

```