Neural Network

PREDICT THE BURNED AREA OF FOREST FIRES WITH NEURAL NETWORKS

Import Necessary Libraries

In [1]:
```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
import warnings
warnings.filterwarnings('ignore')
```

Import data

In [2]:
```
forest_data = pd.read_csv('forestfires.csv')
forest_data.head(7)
```

Out[2]:

	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	...	monthfeb	monthjan	monthjul	monthjun	monthmar	monthmay	monthnov	monthoct	m
0	mar	fri	86.2	26.2	94.3	5.1	8.2	51	6.7	0.0	...	0	0	0	0	1	0	0	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	0	1	
2	oct	sat	90.6	43.7	686.9	6.7	14.6	33	1.3	0.0	...	0	0	0	0	0	0	0	1	
3	mar	fri	91.7	33.3	77.5	9.0	8.3	97	4.0	0.2	...	0	0	0	0	1	0	0	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	0	0	
5	aug	sun	92.3	85.3	488.0	14.7	22.2	29	5.4	0.0	...	0	0	0	0	0	0	0	0	
6	aug	mon	92.3	88.9	495.6	8.5	24.1	27	3.1	0.0	...	0	0	0	0	0	0	0	0	

7 rows × 31 columns

Data Understanding

In [3]:
```
forest_data.shape
```

Out[3]: (517, 31)

In [4]:
```
forest_data.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
```

Data Preprocessing

Check for Duplicate Values

In [5]:
```
forest_data = forest_data.drop_duplicates()
forest_data.head()
```

Out[5]:

	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	...	monthfeb	monthjan	monthjul	monthjun	monthmar	monthmay	monthnov	monthoct	mo
0	mar	fri	86.2	26.2	94.3	5.1	8.2	51	6.7	0.0	...	0	0	0	0	1	0	0	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	0	1	
2	oct	sat	90.6	43.7	686.9	6.7	14.6	33	1.3	0.0	...	0	0	0	0	0	0	0	1	
3	mar	fri	91.7	33.3	77.5	9.0	8.3	97	4.0	0.2	...	0	0	0	0	1	0	0	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	0	0	

5 rows × 31 columns

Removing unnecessary Features

In [6]:
```
forest_data = forest_data.drop(forest_data.columns[10:30], axis = 1)
forest_data.head()
```

Out[6]:

	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	size_category
0	mar	fri	86.2	26.2	94.3	5.1	8.2	51	6.7	0.0	small
1	oct	tue	90.6	35.4	669.1	6.7	18.0	33	0.9	0.0	small
2	oct	sat	90.6	43.7	686.9	6.7	14.6	33	1.3	0.0	small
3	mar	fri	91.7	33.3	77.5	9.0	8.3	97	4.0	0.2	small
4	mar	sun	89.3	51.3	102.2	9.6	11.4	99	1.8	0.0	small

In [7]:
```
forest_data.shape

```
Out[7]: (509, 11)

Label Encoding

In [8]:
```
label_encoder = preprocessing.LabelEncoder()
label_encoder
```
Out[8]: LabelEncoder()

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [9]:
```
forest_data['size_category'] = label_encoder.fit_transform(forest_data['size_category'])
```

In [10]:
```
forest_data.head()
```
Out[10]:

	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	size_category
0	mar	fri	86.2	26.2	94.3	5.1	8.2	51	6.7	0.0	1
1	oct	tue	90.6	35.4	669.1	6.7	18.0	33	0.9	0.0	1
2	oct	sat	90.6	43.7	686.9	6.7	14.6	33	1.3	0.0	1
3	mar	fri	91.7	33.3	77.5	9.0	8.3	97	4.0	0.2	1
4	mar	sun	89.3	51.3	102.2	9.6	11.4	99	1.8	0.0	1

Converting catergorical values of days and months into intergers

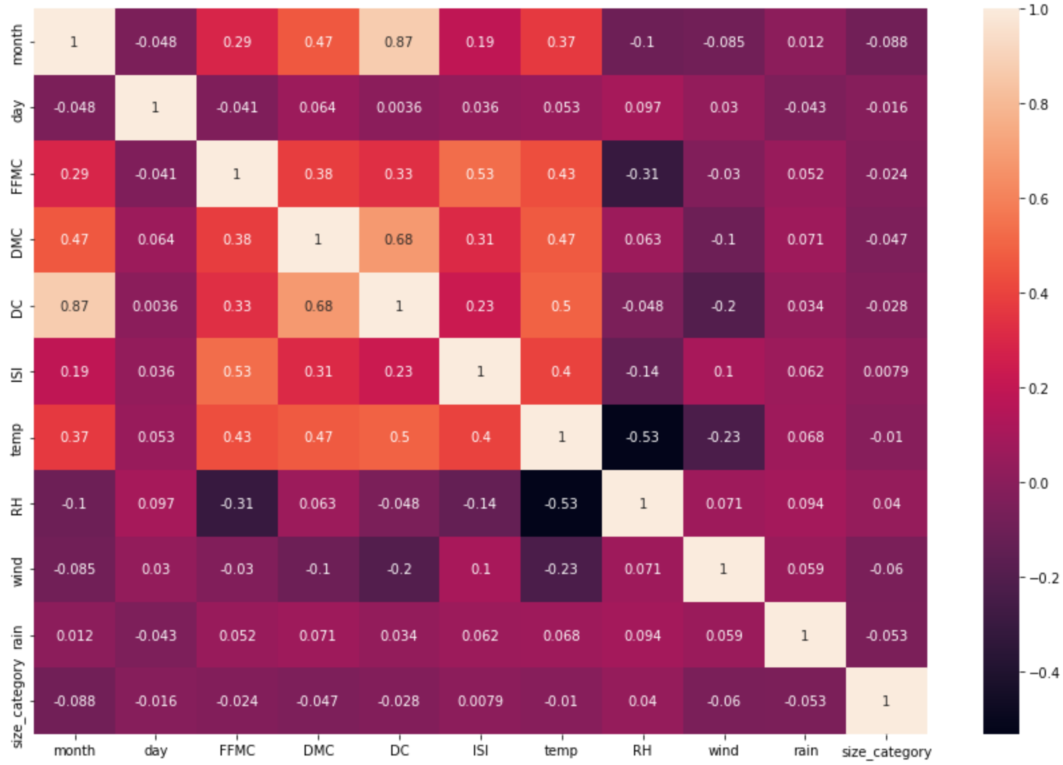
In [11]:
```
forest_data.month.replace(('jan','feb','mar','apr','may','jun','jul','aug','sep','oct','nov','dec'),
                          (1,2,3,4,5,6,7,8,9,10,11,12), inplace = True)
forest_data.day.replace(('mon','tue','wed','thu','fri','sat','sun'),(1,2,3,4,5,6,7),inplace = True)
```

In [12]:
```
forest_data.head()
```
Out[12]:

	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	size_category
0	3	5	86.2	26.2	94.3	5.1	8.2	51	6.7	0.0	1
1	10	2	90.6	35.4	669.1	6.7	18.0	33	0.9	0.0	1
2	10	6	90.6	43.7	686.9	6.7	14.6	33	1.3	0.0	1
3	3	5	91.7	33.3	77.5	9.0	8.3	97	4.0	0.2	1
4	3	7	89.3	51.3	102.2	9.6	11.4	99	1.8	0.0	1

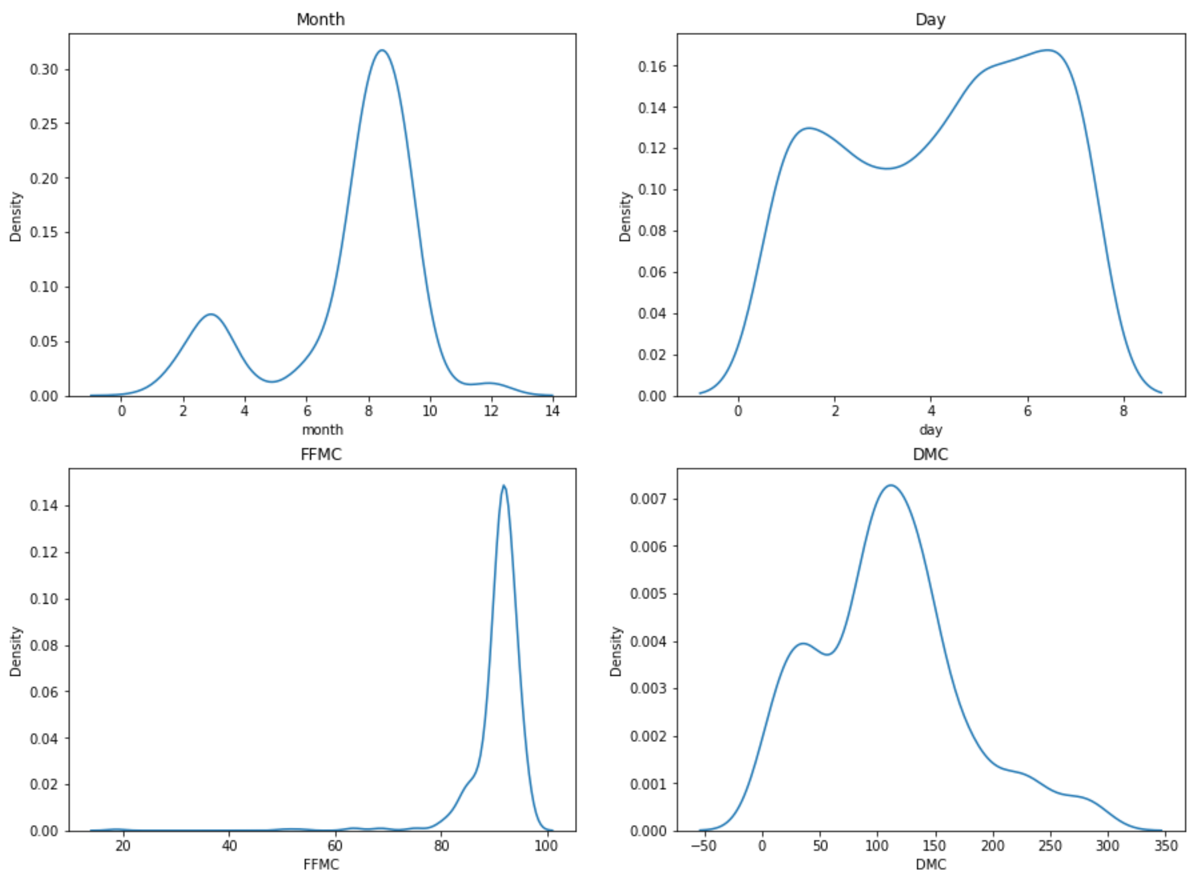
Correlation Analysis

In [13]:
```
plt.figure(figsize= (15,10))
sns.heatmap(forest_data.corr(),annot= True)
plt.show()
```



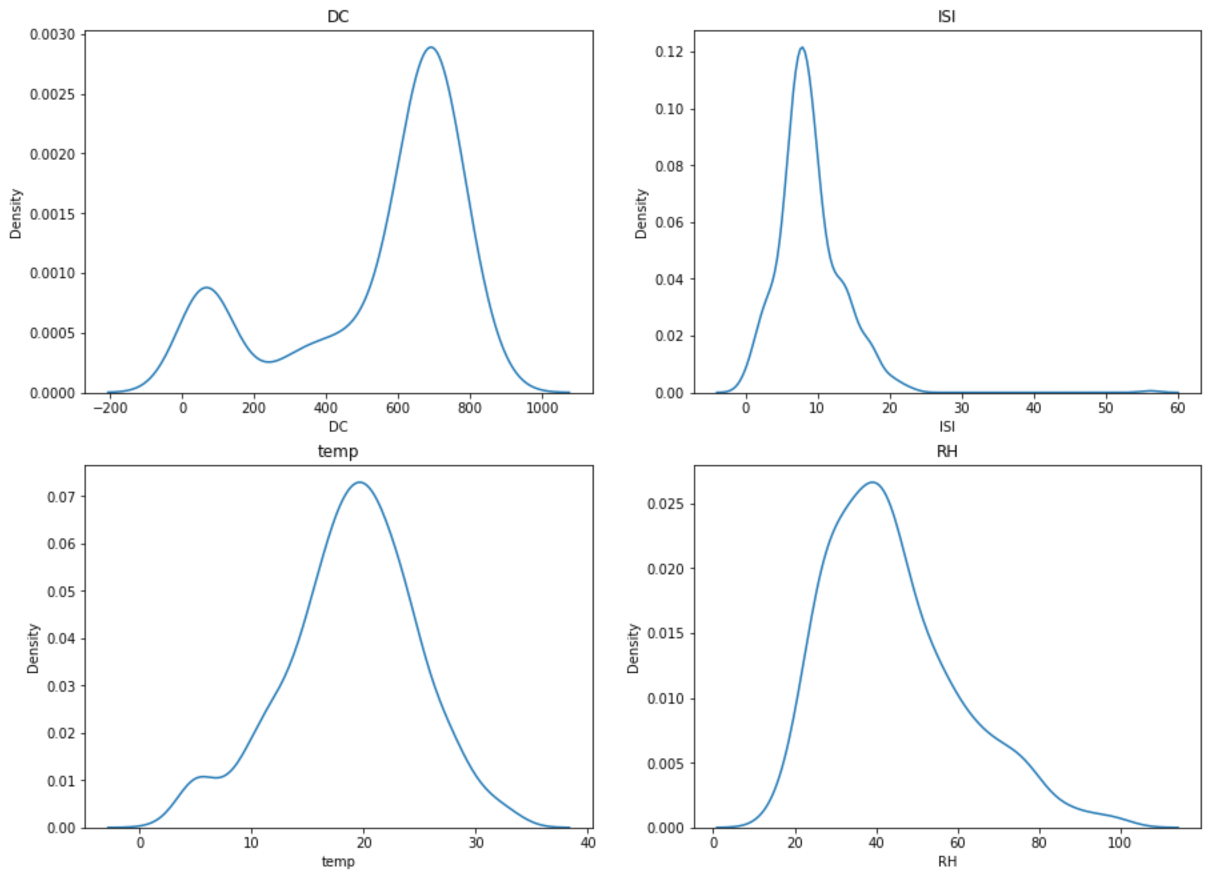
Visualization of Independent varibales

In [14]:
```python
Normality_fig,Axes2=plt.subplots(2,2)

Normality_fig.set_figheight(11)
Normality_fig.set_figwidth(15)

sns.kdeplot(x="month",data=forest_data,ax=Axes2[0,0])
Axes2[0,0].set_title("Month")

sns.kdeplot(x="day",data=forest_data,ax=Axes2[0,1])
Axes2[0,1].set_title("Day")

sns.kdeplot(x="FFMC",data=forest_data,ax=Axes2[1,0])
Axes2[1,0].set_title("FFMC")

sns.kdeplot(x="DMC",data=forest_data,ax=Axes2[1,1])
Axes2[1,1].set_title("DMC")

plt.show()
```



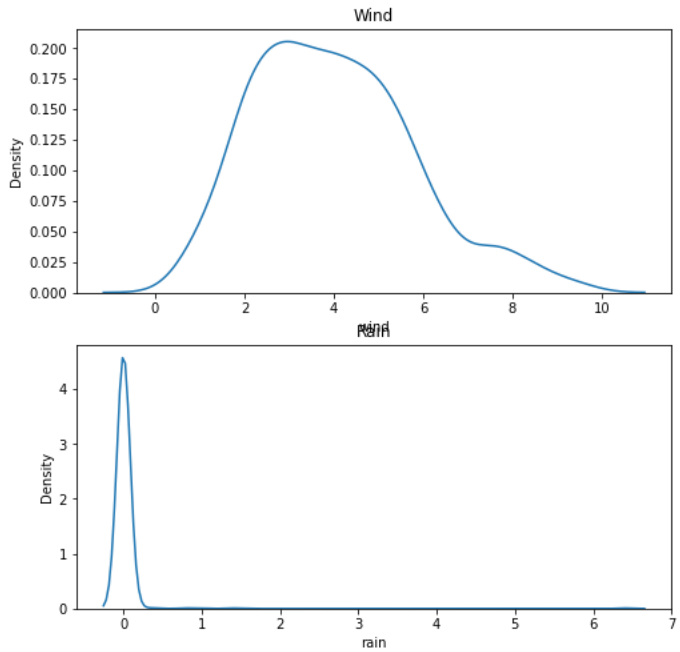
In [15]:

```
Normality_fig,Axes2=plt.subplots(2,2)

Normality_fig.set_figheight(11)
Normality_fig.set_figwidth(15)

sns.kdeplot(x="DC",data=forest_data,ax=Axes2[0,0])
Axes2[0,0].set_title("DC")

sns.kdeplot(x="ISI",data=forest_data,ax=Axes2[0,1])
Axes2[0,1].set_title("ISI")

sns.kdeplot(x="temp",data=forest_data,ax=Axes2[1,0])
Axes2[1,0].set_title("temp")

sns.kdeplot(x="RH",data=forest_data,ax=Axes2[1,1])
Axes2[1,1].set_title("RH")

plt.show()
```



In [16]:
```
Normality_fig,Axes2=plt.subplots(2)
Normality_fig.set_figheight(8)
Normality_fig.set_figwidth(8)
sns.kdeplot(x="wind",data=forest_data,ax=Axes2[0])
Axes2[0].set_title("Wind")
sns.kdeplot(x="rain",data=forest_data,ax=Axes2[1])
Axes2[1].set_title("Rain")
```

Out[16]:
```
Text(0.5, 1.0, 'Rain')
```



Extracting the independent and dependent variables

In [17]:
```
X = forest_data.iloc[:,0:10].values
X
```

Out[17]:
```
array([[ 3. ,  5. , 86.2, ..., 51. ,  6.7,  0. ],
       [10. ,  2. , 90.6, ..., 33. ,  0.9,  0. ],
       [10. ,  6. , 90.6, ..., 33. ,  1.3,  0. ],
       ...,
       [ 8. ,  7. , 81.6, ..., 70. ,  6.7,  0. ],
       [ 8. ,  6. , 94.4, ..., 42. ,  4. ,  0. ],
       [11. ,  2. , 79.5, ..., 31. ,  4.5,  0. ]])
```

In [18]:
```
y = forest_data.iloc[:,-1].values
```

Normalizing data

In [19]:
```
def norm_func(i):
    X = (i-i.min())/(i.max()-i.min())
    return (X)
```

In [20]:
```
X_norm = norm_func(X)
```

Train Test Split

In [21]:
```
X_train,X_test,y_train,y_test = train_test_split(X_norm,y, test_size=0.2,stratify=y)
```

Applying Neural Network

In [22]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [23]:
```
pip install keras
```

```
Requirement already satisfied: keras in c:\users\admin\anaconda3\lib\site-packages (2.9.0)
Note: you may need to restart the kernel to use updated packages.

WARNING: Ignoring invalid distribution -tatsmodels (c:\users\admin\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -tatsmodels (c:\users\admin\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -tatsmodels (c:\users\admin\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -tatsmodels (c:\users\admin\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -tatsmodels (c:\users\admin\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -tatsmodels (c:\users\admin\anaconda3\lib\site-packages)
```

In [24]:
```
pip install tensorflow
```

```
Requirement already satisfied: tensorflow in c:\users\admin\anaconda3\lib\site-packages (2.9.1)
Requirement already satisfied: termcolor>=1.1.0 in c:\users\admin\anaconda3\lib\site-packages (from tensorflow) (1.1.0)
Requirement already satisfied: keras-preprocessing>=1.1.1 in c:\users\admin\anaconda3\lib\site-packages (from tensorflow) (1.
1.2)
Requirement already satisfied: protobuf<3.20,>=3.9.2 in c:\users\admin\anaconda3\lib\site-packages (from tensorflow) (3.19.4)
Requirement already satisfied: tensorflow-estimator<2.10.0,>=2.9.0rc0 in c:\users\admin\anaconda3\lib\site-packages (from ten
sorflow) (2.9.0)
Requirement already satisfied: flatbuffers<2,>=1.12 in c:\users\admin\anaconda3\lib\site-packages (from tensorflow) (1.12)
Requirement already satisfied: keras<2.10.0,>=2.9.0rc0 in c:\users\admin\anaconda3\lib\site-packages (from tensorflow) (2.9.
0)
Requirement already satisfied: libclang>=13.0.0 in c:\users\admin\anaconda3\lib\site-packages (from tensorflow) (14.0.6)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in c:\users\admin\anaconda3\lib\site-packages (from tensorflow) (1.47.0)
Requirement already satisfied: wrapt>=1.11.0 in c:\users\admin\anaconda3\lib\site-packages (from tensorflow) (1.12.1)
Requirement already satisfied: absl-py>=1.0.0 in c:\users\admin\anaconda3\lib\site-packages (from tensorflow) (1.2.0)
Requirement already satisfied: tensorboard<2.10,>=2.9 in c:\users\admin\anaconda3\lib\site-packages (from tensorflow) (2.9.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in c:\users\admin\anaconda3\lib\site-packages (from tenso
rflow) (0.26.0)
Requirement already satisfied: setuptools in c:\users\admin\anaconda3\lib\site-packages (from tensorflow) (58.0.4)
Requirement already satisfied: gast<=0.4.0,>=0.2.1 in c:\users\admin\anaconda3\lib\site-packages (from tensorflow) (0.4.0)
```

In [25]:
```
import keras.models
import tensorflow
from tensorflow.keras.models import Sequential
from keras.layers import Dense
```

In [26]:
```
model = Sequential()
```

In [27]:
```
# fix random seed for reproducibility
seed =123
np.random.seed(seed)
```

In [28]:
```
model.add(Dense(12, input_dim = 10, kernel_initializer='uniform', activation = 'relu'))
model.add(Dense(8, kernel_initializer='uniform', activation = 'relu'))
model.add(Dense(1, kernel_initializer='uniform', activation = 'linear'))
```

In [29]:
```
model.compile(loss='mse', optimizer = 'adam', metrics=['accuracy'])
```

In [30]:
```
model.fit(X_train,y_train, validation_split=0.33,epochs=100,batch_size =10)
```

```
Epoch 1/100
28/28 [==============================] - 2s 19ms/step - loss: 0.7048 - accuracy: 0.2721 - val_loss: 0.6822 - val_accuracy: 0.
2667
Epoch 2/100
28/28 [==============================] - 0s 6ms/step - loss: 0.6465 - accuracy: 0.2721 - val_loss: 0.6117 - val_accuracy: 0.2
667
Epoch 3/100
28/28 [==============================] - 0s 6ms/step - loss: 0.5646 - accuracy: 0.2721 - val_loss: 0.5081 - val_accuracy: 0.2
667
Epoch 4/100
28/28 [==============================] - 0s 6ms/step - loss: 0.4476 - accuracy: 0.2721 - val_loss: 0.3796 - val_accuracy: 0.2
667
Epoch 5/100
28/28 [==============================] - 0s 5ms/step - loss: 0.3207 - accuracy: 0.2831 - val_loss: 0.2656 - val_accuracy: 0.3
852
Epoch 6/100
28/28 [==============================] - 0s 5ms/step - loss: 0.2335 - accuracy: 0.6140 - val_loss: 0.2121 - val_accuracy: 0.6
667
Epoch 7/100
```

In [31]:
```
model.evaluate(X_test,y_test)
```

```
4/4 [==============================] - 0s 5ms/step - loss: 0.2021 - accuracy: 0.7255
```

Out[31]:
```
[0.20208163559436798, 0.7254902124404907]
```

In [32]:
```
scores = model.evaluate(X_train, y_train)
print("%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
```

```
13/13 [==============================] - 0s 2ms/step - loss: 0.1965 - accuracy: 0.7297
accuracy: 72.97%
```

In [33]:
```
scores = model.evaluate(X_test, y_test)
print("%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
```

```
4/4 [==============================] - 0s 3ms/step - loss: 0.2021 - accuracy: 0.7255
accuracy: 72.55%
```

Validation accuracy is less

Tuning the number of layers in the model

In [34]:
```
model=Sequential()
```

In [35]:
```
model.add(Dense(28, input_dim=10,kernel_initializer='uniform', activation='relu'))
model.add(Dense(20, kernel_initializer='uniform' ,activation='relu'))
model.add(Dense(5, kernel_initializer='uniform',activation='linear'))
```

In [36]:
```
model.compile(loss='mse', optimizer = 'adam', metrics=['accuracy'])
```

In [37]:
```
model.fit(X_train,y_train, validation_split=0.33,epochs=100,batch_size=50)
```

```
Epoch 1/100
6/6 [==============================] - 1s 74ms/step - loss: 0.7239 - accuracy: 0.7243 - val_loss: 0.7221 - val_accuracy: 0.73
33
Epoch 2/100
6/6 [==============================] - 0s 14ms/step - loss: 0.7122 - accuracy: 0.7279 - val_loss: 0.7096 - val_accuracy: 0.73
33
Epoch 3/100
6/6 [==============================] - 0s 13ms/step - loss: 0.6993 - accuracy: 0.7279 - val_loss: 0.6947 - val_accuracy: 0.73
33
Epoch 4/100
6/6 [==============================] - 0s 16ms/step - loss: 0.6834 - accuracy: 0.7279 - val_loss: 0.6763 - val_accuracy: 0.73
33
Epoch 5/100
6/6 [==============================] - 0s 14ms/step - loss: 0.6635 - accuracy: 0.7279 - val_loss: 0.6532 - val_accuracy: 0.73
33
Epoch 6/100
6/6 [==============================] - 0s 14ms/step - loss: 0.6393 - accuracy: 0.7279 - val_loss: 0.6239 - val_accuracy: 0.73
33
Epoch 7/100
```

In [38]:
```
model.evaluate(X_train,y_train)
```

```
13/13 [==============================] - 0s 3ms/step - loss: 0.1970 - accuracy: 0.0958
```

Out[38]: [0.19699284434318542, 0.09582309424877167]

In [39]:
```
model.evaluate(X_test,y_test)
```

```
4/4 [==============================] - 0s 3ms/step - loss: 0.2035 - accuracy: 0.0882
```

Out[39]: [0.20350773632526398, 0.08823529630899434]

Trying different values

In [40]:
```
model=Sequential()
```

In [41]:
```
model.add(Dense(20, input_dim = 10, kernel_initializer = 'uniform', activation = 'relu'))
model.add(Dense(10,  kernel_initializer = 'uniform', activation = 'relu'))
model.add(Dense(2,  kernel_initializer = 'uniform', activation = 'linear'))
```

In [42]:
```
model.compile(loss = 'mse', optimizer = 'adam', metrics = ['accuracy'])
```

In [43]:
```
model.fit(X_train,y_train, validation_split = 0.33, epochs = 250, batch_size = 10)
```

```
Epoch 1/250
28/28 [==============================] - 1s 20ms/step - loss: 0.7056 - accuracy: 0.4118 - val_loss: 0.6796 - val_accuracy: 0.
2667
Epoch 2/250
28/28 [==============================] - 0s 5ms/step - loss: 0.6332 - accuracy: 0.2721 - val_loss: 0.5784 - val_accuracy: 0.2
667
Epoch 3/250
28/28 [==============================] - 0s 5ms/step - loss: 0.4987 - accuracy: 0.2721 - val_loss: 0.3975 - val_accuracy: 0.2
667
Epoch 4/250
28/28 [==============================] - 0s 6ms/step - loss: 0.3160 - accuracy: 0.2721 - val_loss: 0.2309 - val_accuracy: 0.2
667
Epoch 5/250
28/28 [==============================] - 0s 6ms/step - loss: 0.2120 - accuracy: 0.2721 - val_loss: 0.2053 - val_accuracy: 0.2
667
Epoch 6/250
28/28 [==============================] - 0s 6ms/step - loss: 0.2078 - accuracy: 0.2721 - val_loss: 0.2052 - val_accuracy: 0.2
667
Epoch 7/250
```

In [44]:
```
model.evaluate(X_train,y_train)
```

```
13/13 [==============================] - 0s 4ms/step - loss: 0.1945 - accuracy: 0.4988
```

Out[44]: [0.19451133906841278, 0.4987714886665344]

the accuracy has not improved with variations in layers

Applying Varried activation methods

In [45]:
```
model = Sequential()
```

In [46]:
```
model.add(Dense(20, input_dim = 10, kernel_initializer = 'uniform', activation = 'relu'))
model.add(Dense(10,  kernel_initializer = 'uniform', activation = 'linear'))
model.add(Dense(2,  kernel_initializer = 'uniform', activation = 'sigmoid'))
```

In [47]:
```
model.compile(loss='mse', optimizer='adam', metrics=['accuracy'])
```

In [48]:
```
model.fit(X_train,y_train, validation_split = 0.33, epochs = 250, batch_size = 10)
```

```
Epoch 1/250
28/28 [==============================] - 2s 23ms/step - loss: 0.2481 - accuracy: 0.3566 - val_loss: 0.2453 - val_accuracy: 0.
6000
Epoch 2/250
28/28 [==============================] - 0s 7ms/step - loss: 0.2409 - accuracy: 0.7022 - val_loss: 0.2344 - val_accuracy: 0.7
333
Epoch 3/250
28/28 [==============================] - 0s 7ms/step - loss: 0.2249 - accuracy: 0.7279 - val_loss: 0.2149 - val_accuracy: 0.7
333
Epoch 4/250
28/28 [==============================] - 0s 6ms/step - loss: 0.2072 - accuracy: 0.7279 - val_loss: 0.1995 - val_accuracy: 0.7
333
Epoch 5/250
28/28 [==============================] - 0s 8ms/step - loss: 0.1997 - accuracy: 0.7279 - val_loss: 0.1971 - val_accuracy: 0.7
333
Epoch 6/250
28/28 [==============================] - 0s 6ms/step - loss: 0.1992 - accuracy: 0.7279 - val_loss: 0.1970 - val_accuracy: 0.7
333
Epoch 7/250
```

In [49]:
```
model.evaluate(X_train,y_train)
```

```
13/13 [==============================] - 0s 3ms/step - loss: 0.1954 - accuracy: 0.4054
```

Out[49]: [0.19535548985004425, 0.4054054021835327]

no improvement is seen

In [50]:
```
model=Sequential()
```

In [51]:
```
model.add(Dense(20, input_dim = 10, kernel_initializer = 'uniform', activation = 'linear'))
model.add(Dense(10,  kernel_initializer = 'uniform', activation = 'relu'))
model.add(Dense(2,  kernel_initializer = 'uniform', activation = 'sigmoid'))
```

In [52]:
```
model.compile(loss = 'mse', optimizer = 'adam', metrics = ['accuracy'])
```

In [53]:
```
model.fit(X_train,y_train, validation_split = 0.33, epochs = 250, batch_size = 10)
```

```
Epoch 1/250
28/28 [==============================] - 1s 14ms/step - loss: 0.2480 - accuracy: 0.7169 - val_loss: 0.2453 - val_accuracy: 0.
7333
Epoch 2/250
28/28 [==============================] - 0s 4ms/step - loss: 0.2410 - accuracy: 0.7279 - val_loss: 0.2351 - val_accuracy: 0.7
333
Epoch 3/250
28/28 [==============================] - 0s 4ms/step - loss: 0.2270 - accuracy: 0.7279 - val_loss: 0.2164 - val_accuracy: 0.7
333
Epoch 4/250
28/28 [==============================] - 0s 4ms/step - loss: 0.2088 - accuracy: 0.7279 - val_loss: 0.2011 - val_accuracy: 0.7
333
Epoch 5/250
28/28 [==============================] - 0s 4ms/step - loss: 0.2001 - accuracy: 0.7279 - val_loss: 0.1978 - val_accuracy: 0.7
333
Epoch 6/250
28/28 [==============================] - 0s 5ms/step - loss: 0.1998 - accuracy: 0.7279 - val_loss: 0.1973 - val_accuracy: 0.7
333
Epoch 7/250
```

In [54]:
```
model.evaluate(X_train,y_train)
```

```
13/13 [==============================] - 0s 3ms/step - loss: 0.1954 - accuracy: 0.6314
```

Out[54]: [0.19544364511966705, 0.6314496397972107]

No improvement is seen.

Data Optimization

In [55]:
```
forest = pd.read_csv('forestfires.csv')
```

Label encoding the "Size category"feature.

In [56]:
```
forest.loc[forest.size_category=='small','size_category']=0
forest.loc[forest.size_category=='large','size_category']=1
```

Taking dummy columns of day and month and dropping the "day" and "month" feature

In [57]:
```
forest.drop(['month','day'],axis=1,inplace=True)
```

In [58]:
```
forest.head()
```

Out[58]:

	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area	dayfri	...	monthfeb	monthjan	monthjul	monthjun	monthmar	monthmay	monthnov	monthoct	mor
0	86.2	26.2	94.3	5.1	8.2	51	6.7	0.0	0.0	1	...	0	0	0	0	1	0	0	0	
1	90.6	35.4	669.1	6.7	18.0	33	0.9	0.0	0.0	0	...	0	0	0	0	0	0	0	1	
2	90.6	43.7	686.9	6.7	14.6	33	1.3	0.0	0.0	0	...	0	0	0	0	0	0	0	1	
3	91.7	33.3	77.5	9.0	8.3	97	4.0	0.2	0.0	1	...	0	0	0	0	1	0	0	0	
4	89.3	51.3	102.2	9.6	11.4	99	1.8	0.0	0.0	0	...	0	0	0	0	1	0	0	0	

5 rows × 29 columns

Dependant and Independant variables

In [59]:
```
x=forest.iloc[:,:-1]
y=forest.iloc[:,-1]

y=y.astype('float')
```

Scaling the Data

In [60]:
```
ss=preprocessing.StandardScaler()
x=ss.fit_transform(x)
```

Train Test Split

In [61]:
```
X_train,X_test,y_train,y_test = train_test_split(x,y, test_size=0.2,stratify=y)
```

In [62]:
```
X_train.shape
```

Out[62]:
```
(413, 28)
```

In [63]:
```
y_train.shape
```

Out[63]:
```
(413,)
```

In [64]:
```
X_test.shape
```

Out[64]:
```
(104, 28)
```

In [65]:
```
y_test.shape
```

Out[65]:
```
(104,)
```

Building New Model with loss function binary_crossentropy

In [66]:
```
model=Sequential()
model.add(Dense(28,  activation='relu'))
model.add(Dense(28,  activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

In [67]:
```
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

In [68]:
```
history=model.fit(X_train, y_train, validation_split=0.33, epochs=250, batch_size=50)
```

```
Epoch 1/250
6/6 [==============================] - 2s 107ms/step - loss: 0.7537 - accuracy: 0.3986 - val_loss: 0.7092 - val_accuracy: 0.4
453
Epoch 2/250
6/6 [==============================] - 0s 14ms/step - loss: 0.6788 - accuracy: 0.4819 - val_loss: 0.6581 - val_accuracy: 0.59
12
Epoch 3/250
6/6 [==============================] - 0s 52ms/step - loss: 0.6258 - accuracy: 0.6413 - val_loss: 0.6244 - val_accuracy: 0.72
26
Epoch 4/250
6/6 [==============================] - 0s 25ms/step - loss: 0.5897 - accuracy: 0.7572 - val_loss: 0.6019 - val_accuracy: 0.70
80
Epoch 5/250
6/6 [==============================] - 0s 21ms/step - loss: 0.5638 - accuracy: 0.7717 - val_loss: 0.5872 - val_accuracy: 0.72
99
Epoch 6/250
6/6 [==============================] - 0s 20ms/step - loss: 0.5470 - accuracy: 0.7717 - val_loss: 0.5770 - val_accuracy: 0.72
99
Epoch 7/250
```

In [69]:
```
model.evaluate(X_train,y_train)
```

```
13/13 [==============================] - 0s 4ms/step - loss: 0.2452 - accuracy: 0.9516
```

Out[69]:
```
[0.2451915442943573, 0.9515738487243652]
```

In [70]:
```
model.evaluate(X_test,y_test)
```

```
4/4 [==============================] - 0s 4ms/step - loss: 1.0461 - accuracy: 0.8846
```

Out[70]: [1.0460717678070068, 0.8846153616905212]

accuracy has improved

Visualize training history

In [71]:
```
history = model.fit(X_train,y_train, validation_split = 0.33, epochs = 100, batch_size = 10)
```

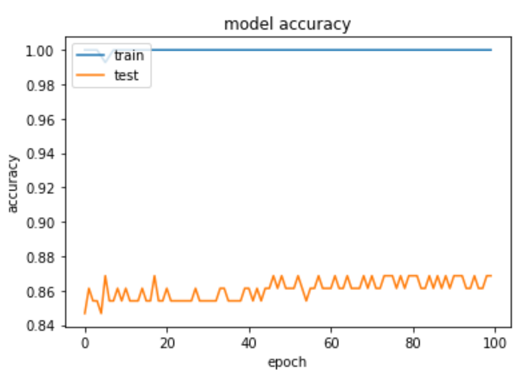
```
Epoch 1/100
28/28 [==============================] - 0s 12ms/step - loss: 0.0080 - accuracy: 1.0000 - val_loss: 0.7431 - val_accuracy: 0.
8467
Epoch 2/100
28/28 [==============================] - 0s 7ms/step - loss: 0.0078 - accuracy: 1.0000 - val_loss: 0.7365 - val_accuracy: 0.8
613
Epoch 3/100
28/28 [==============================] - 0s 7ms/step - loss: 0.0087 - accuracy: 1.0000 - val_loss: 0.8001 - val_accuracy: 0.8
540
Epoch 4/100
28/28 [==============================] - 0s 12ms/step - loss: 0.0092 - accuracy: 1.0000 - val_loss: 0.7557 - val_accuracy: 0.
8540
Epoch 5/100
28/28 [==============================] - 0s 9ms/step - loss: 0.0164 - accuracy: 0.9964 - val_loss: 0.6941 - val_accuracy: 0.8
467
Epoch 6/100
28/28 [==============================] - 0s 9ms/step - loss: 0.0194 - accuracy: 0.9928 - val_loss: 0.7655 - val_accuracy: 0.8
686
Epoch 7/100
```

In [72]:
```
# list all data in history
model.history.history.keys()
```

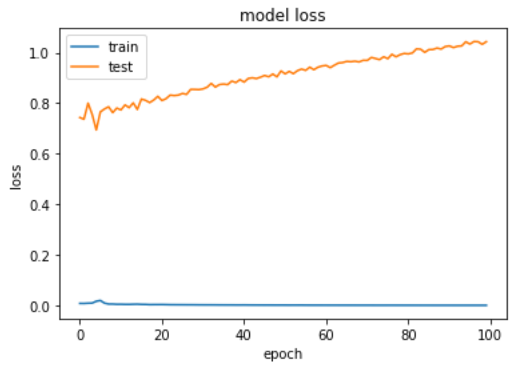
Out[72]: dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])

summarize history for accuracy and Loss

In [73]:
```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train','test'], loc = 'upper left')
plt.show()
```



In [74]:
```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train','test'], loc = 'upper left')
plt.show()
```



Tuning of Hyperparameters

In [75]:
```
from sklearn.model_selection import GridSearchCV, KFold
from keras.wrappers.scikit_learn import KerasClassifier
from keras.optimizers import Adam
from tensorflow.keras.models import load_model
```

Defining a Function for the Model

In [76]:
```
def create_model():
    model_2 = Sequential()
    model_2.add(Dense(10,input_dim = 10, kernel_initializer = 'uniform', activation = 'relu'))
    model_2.add(Dense(11,kernel_initializer = 'uniform', activation = 'relu'))
    model_2.add(Dense(1,kernel_initializer = 'uniform', activation = 'linear'))

    adam = Adam(lr = 0.01)
    model_2.compile(loss = 'mse', optimizer = adam, metrics = ['accuracy'])
    return model
```

In [77]:
```
model_1 = KerasClassifier(build_fn = create_model,verbose = 0)
```

In [78]:
```
validation_split = [0.33,0.2,0.4,0.37]
batch_size = [10,20,40]
epochs = [10,50,100]
```

Gridsearch CV

In [79]:
```
param_grid = dict(batch_size = batch_size,epochs = epochs,validation_split=validation_split)
```

In [80]:
```
grid = GridSearchCV(estimator = model_1, param_grid = param_grid,cv = KFold(),verbose = 10)
grid_result = grid.fit(X_train,y_train)
```

```
Fitting 5 folds for each of 36 candidates, totalling 180 fits
[CV 1/5; 1/36] START batch_size=10, epochs=10, validation_split=0.33...........
[CV 1/5; 1/36] END batch_size=10, epochs=10, validation_split=0.33;, score=0.988 total time=   3.5s
[CV 2/5; 1/36] START batch_size=10, epochs=10, validation_split=0.33...........
[CV 2/5; 1/36] END batch_size=10, epochs=10, validation_split=0.33;, score=1.000 total time=   1.6s
[CV 3/5; 1/36] START batch_size=10, epochs=10, validation_split=0.33...........
[CV 3/5; 1/36] END batch_size=10, epochs=10, validation_split=0.33;, score=1.000 total time=   2.0s
[CV 4/5; 1/36] START batch_size=10, epochs=10, validation_split=0.33...........
[CV 4/5; 1/36] END batch_size=10, epochs=10, validation_split=0.33;, score=0.963 total time=   1.9s
[CV 5/5; 1/36] START batch_size=10, epochs=10, validation_split=0.33...........
[CV 5/5; 1/36] END batch_size=10, epochs=10, validation_split=0.33;, score=0.890 total time=   2.3s
[CV 1/5; 2/36] START batch_size=10, epochs=10, validation_split=0.2............
[CV 1/5; 2/36] END batch_size=10, epochs=10, validation_split=0.2;, score=0.988 total time=   1.9s
[CV 2/5; 2/36] START batch_size=10, epochs=10, validation_split=0.2............
[CV 2/5; 2/36] END batch_size=10, epochs=10, validation_split=0.2;, score=1.000 total time=   1.7s
[CV 3/5; 2/36] START batch_size=10, epochs=10, validation_split=0.2............
[CV 3/5; 2/36] END batch_size=10, epochs=10, validation_split=0.2;, score=1.000 total time=   1.8s
[CV 4/5; 2/36] START batch_size=10, epochs=10, validation_split=0.2............
[CV 4/5; 2/36] END batch_size=10, epochs=10, validation_split=0.2;, score=1.000 total time=   3.2s
```

In [81]:
```
grid_result.best_params_
```

Out[81]:
```
{'batch_size': 10, 'epochs': 10, 'validation_split': 0.4}
```

Creating the Final Model with Best Parameters

In [82]:
```
model_2 = Sequential()
model_2.add(Dense(10,input_dim = 10, kernel_initializer = 'uniform', activation = 'relu'))
model_2.add(Dense(11,kernel_initializer = 'uniform', activation = 'relu'))
model_2.add(Dense(1,kernel_initializer = 'uniform', activation = 'linear'))
```

In [83]:
```
history = model.fit(X_train,y_train, validation_split = 0.33, epochs = 10, batch_size = 10)
```

```
Epoch 1/10
28/28 [==============================] - 0s 10ms/step - loss: 1.4118e-09 - accuracy: 1.0000 - val_loss: 1.8696 - val_accuracy:
0.9197
Epoch 2/10
28/28 [==============================] - 0s 7ms/step - loss: 1.3692e-09 - accuracy: 1.0000 - val_loss: 1.8715 - val_accuracy:
0.9197
Epoch 3/10
28/28 [==============================] - 0s 6ms/step - loss: 1.3791e-09 - accuracy: 1.0000 - val_loss: 1.8692 - val_accuracy:
0.9197
Epoch 4/10
28/28 [==============================] - 0s 6ms/step - loss: 1.3667e-09 - accuracy: 1.0000 - val_loss: 1.8708 - val_accuracy:
0.9197
Epoch 5/10
28/28 [==============================] - 0s 7ms/step - loss: 1.3977e-09 - accuracy: 1.0000 - val_loss: 1.8726 - val_accuracy:
0.9197
Epoch 6/10
28/28 [==============================] - 0s 6ms/step - loss: 1.4252e-09 - accuracy: 1.0000 - val_loss: 1.8719 - val_accuracy:
0.9197
Epoch 7/10
28/28 [==============================] - 0s 6ms/step - loss: 1.3793e-09 - accuracy: 1.0000 - val_loss: 1.8712 - val_accuracy:
0.9197
Epoch 8/10
28/28 [==============================] - 0s 5ms/step - loss: 1.3962e-09 - accuracy: 1.0000 - val_loss: 1.8731 - val_accuracy:
0.9197
Epoch 9/10
28/28 [==============================] - 0s 5ms/step - loss: 1.4249e-09 - accuracy: 1.0000 - val_loss: 1.8748 - val_accuracy:
0.9197
Epoch 10/10
28/28 [==============================] - 0s 6ms/step - loss: 1.4518e-09 - accuracy: 1.0000 - val_loss: 1.8706 - val_accuracy:
0.9197
```

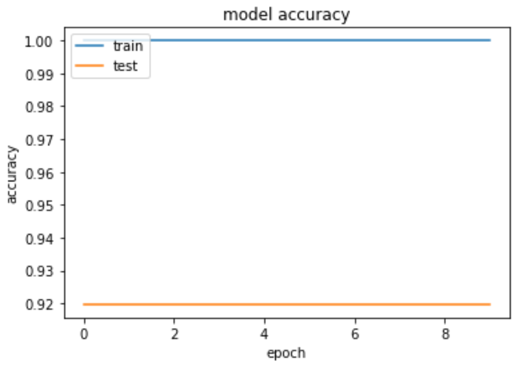
In [84]:
```
model.evaluate(X_train,y_train)
```

```
13/13 [==============================] - 0s 3ms/step - loss: 0.6205 - accuracy: 0.9734
```

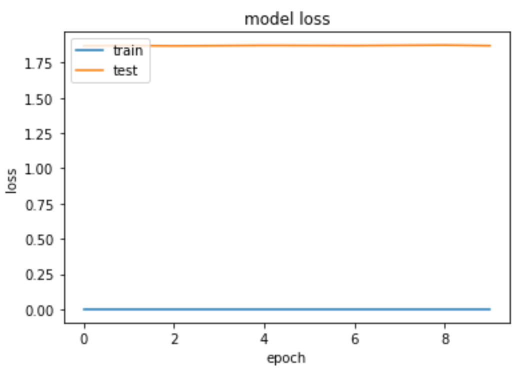
Out[84]:
```
[0.6205052733421326, 0.9733656048774719]
```

Visualizing the Final Model¶

In [85]:
```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train','test'], loc = 'upper left')
plt.show()
```



In [86]:
```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train','test'], loc = 'upper left')
plt.show()
```



The Model is then Deployed.

In [ ]:
```

```