Importing necessary liabrary

In [1]:

```
import warnings
warnings.filterwarnings("ignore")
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from keras.models import Sequential
from keras.layers import Dense, Activation,Layer,Lambda
import seaborn as sns
from sklearn.model_selection import train_test_split
from matplotlib import pyplot as plt
from sklearn.decomposition import PCA
from mlxtend.plotting import plot_decision_regions
from sklearn.preprocessing import StandardScaler
from keras.models import Sequential
from keras.layers import Dense
from keras.layers.normalization import BatchNormalization
from keras import backend
from keras.optimizers import Adam
from sklearn.metrics import r2_score
```

```
Using TensorFlow backend.
```

In [2]:

```
gasturbines_data=pd.read_csv("gas_turbines.csv")
gasturbines_data
```

Out[2]:

	AT	AP	AH	AFDP	GTEP	TIT	TAT	TEY	CDP	CO	NOX
0	6.8594	1007.9	96.799	3.5000	19.663	1059.2	550.00	114.70	10.605	3.1547	82.722
1	6.7850	1008.4	97.118	3.4998	19.728	1059.3	550.00	114.72	10.598	3.2363	82.776
2	6.8977	1008.8	95.939	3.4824	19.779	1059.4	549.87	114.71	10.601	3.2012	82.468
3	7.0569	1009.2	95.249	3.4805	19.792	1059.6	549.99	114.72	10.606	3.1923	82.670
4	7.3978	1009.7	95.150	3.4976	19.765	1059.7	549.98	114.72	10.612	3.2484	82.311
...	...	...	...	...	...	...	...	...	...	...	...
15034	9.0301	1005.6	98.460	3.5421	19.164	1049.7	546.21	111.61	10.400	4.5186	79.559
15035	7.8879	1005.9	99.093	3.5059	19.414	1046.3	543.22	111.78	10.433	4.8470	79.917
15036	7.2647	1006.3	99.496	3.4770	19.530	1037.7	537.32	110.19	10.483	7.9632	90.912
15037	7.0060	1006.8	99.008	3.4486	19.377	1043.2	541.24	110.74	10.533	6.2494	93.227
15038	6.9279	1007.2	97.533	3.4275	19.306	1049.9	545.85	111.58	10.583	4.9816	92.498

15039 rows × 11 columns

Initial analysis

In [3]:

```
gasturbines_data.shape
```

Out[3]:

```
(15039, 11)
```

In [4]:

```
gasturbines_data.dtypes
```

Out[4]:

```
AT      float64
AP      float64
AH      float64
AFDP    float64
GTEP    float64
TIT     float64
TAT     float64
TEY     float64
CDP     float64
CO      float64
NOX     float64
dtype: object
```

In [5]:

```
gasturbines_data.isna().sum()
```

Out[5]:

```
AT      0
AP      0
AH      0
AFDP    0
GTEP    0
TIT     0
TAT     0
TEY     0
CDP     0
CO      0
NOX     0
dtype: int64
```

In [6]:

```
gasturbines_data.describe().T
```

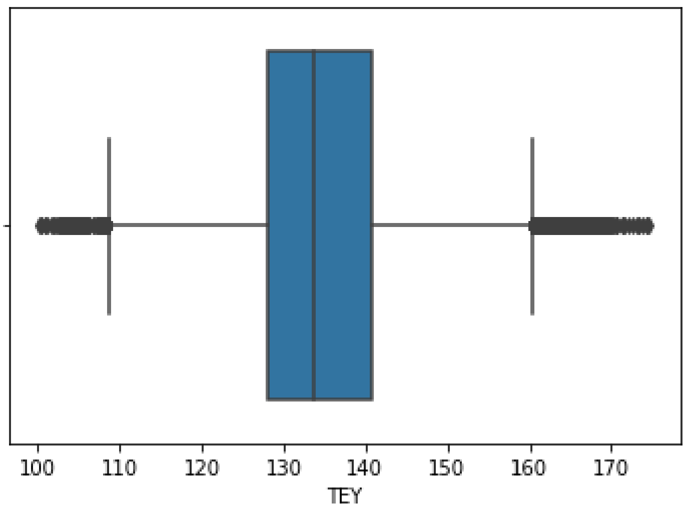
Out[6]:

	count	mean	std	min	25%	50%	75%	
AT	15039.0	17.764381	7.574323	0.522300	11.408000	18.1860	23.8625	34.9
AP	15039.0	1013.199240	6.410760	985.850000	1008.900000	1012.8000	1016.9000	1034.2
AH	15039.0	79.124174	13.793439	30.344000	69.750000	82.2660	90.0435	100.2
AFDP	15039.0	4.200294	0.760197	2.087400	3.723900	4.1862	4.5509	7.6
GTEP	15039.0	25.419061	4.173916	17.878000	23.294000	25.0820	27.1840	37.4
TIT	15039.0	1083.798770	16.527806	1000.800000	1079.600000	1088.7000	1096.0000	1100.8
TAT	15039.0	545.396183	7.866803	512.450000	542.170000	549.8900	550.0600	550.6
TEY	15039.0	134.188464	15.829717	100.170000	127.985000	133.7800	140.8950	174.6
CDP	15039.0	12.102353	1.103196	9.904400	11.622000	12.0250	12.5780	15.0
CO	15039.0	1.972499	2.222206	0.000388	0.858055	1.3902	2.1604	44.1
NOX	15039.0	68.190934	10.470586	27.765000	61.303500	66.6010	73.9355	119.8

Checking the outliers

In [7]:

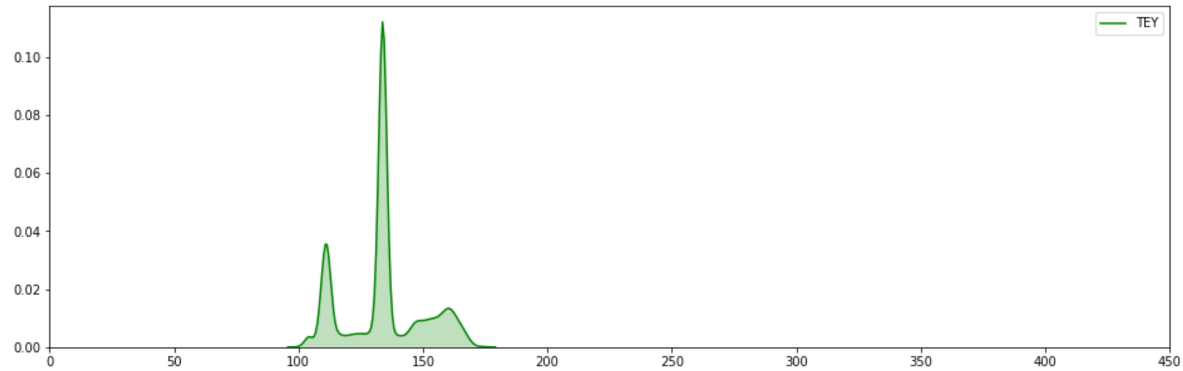
```
gas_outlier=sns.boxplot(gasturbines_data['TEY'])
```



In [8]:

```
plt.figure(figsize=(16,5))
print("Skew: {}".format(gasturbines_data['TEY'].skew()))
print("Kurtosis: {}".format(gasturbines_data['TEY'].kurtosis()))
ax = sns.kdeplot(gasturbines_data['TEY'],shade=True,color='g')
plt.xticks([i for i in range(0,500,50)])
plt.show()
```

```
Skew: 0.14596270190452942
Kurtosis: -0.4870582497451621
```



TEY lies on between 100 and 170

In [9]:

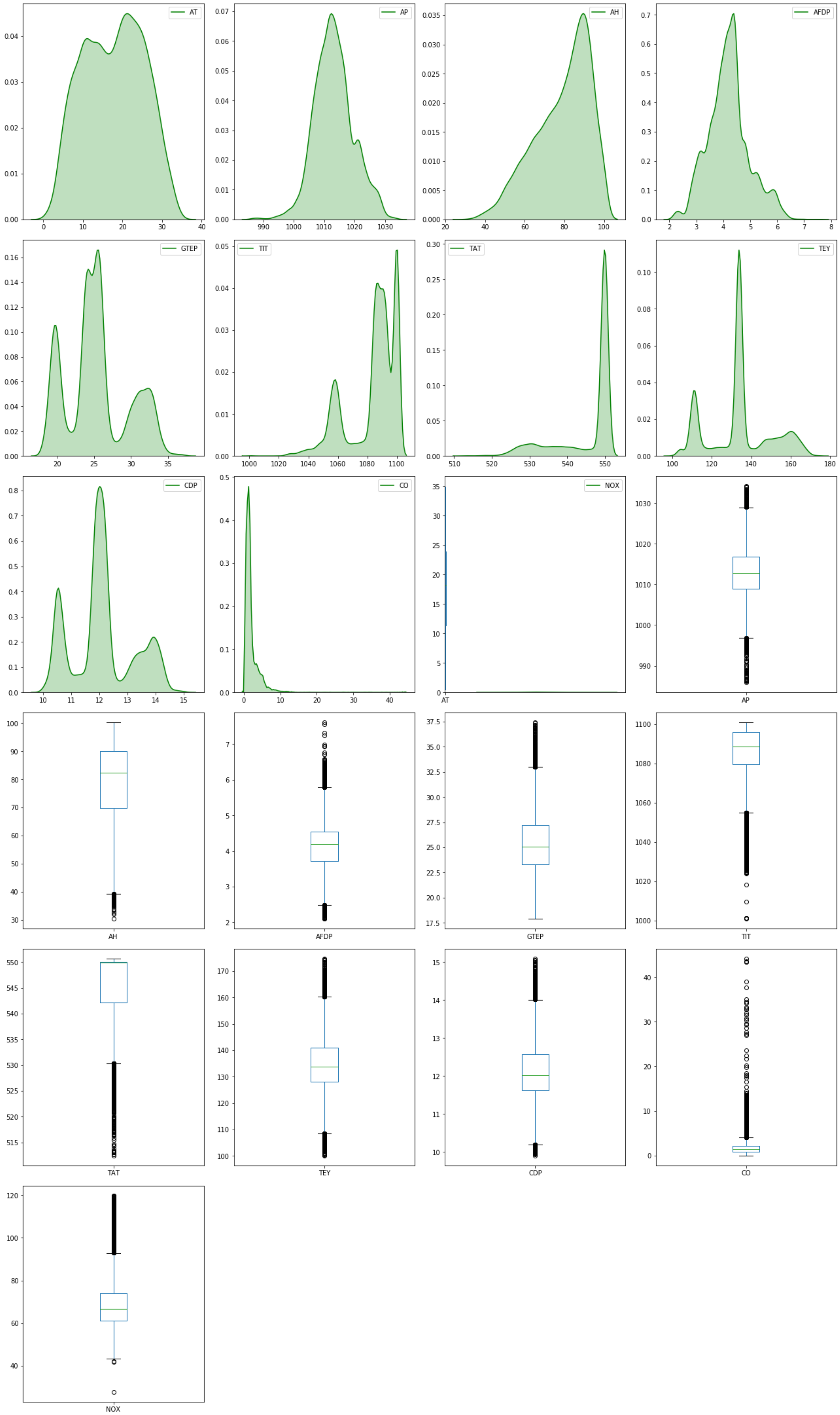
```
dfa = gasturbines_data[gasturbines_data.columns]
month_colum = dfa.select_dtypes(include='object').columns.tolist()
```

In [10]:

```
num_columns = dfa.select_dtypes(exclude='object').columns.tolist()
```

In [11]:

```
plt.figure(figsize=(18,40))
for i,col in enumerate(num_columns,1):
    plt.subplot(8,4,i)
    sns.kdeplot(gasturbines_data[col],color='g',shade=True)
    plt.subplot(8,4,i+10)
    gasturbines_data[col].plot.box()
plt.tight_layout()
plt.show()
num_data = gasturbines_data[num_columns]
pd.DataFrame(data=[num_data.skew(),num_data.kurtosis()],index=['skewness','kurtosis'])
```



Out[11]:

	AT	AP	AH	AFDP	GTEP	TIT	TAT	TEY

	AT	AP	AH	AFDP	GTEP	TIT	TAT	TEY
skewness	-0.030710	0.107601	-0.681224	0.315150	0.370987	-1.133744	-1.485524	0.145963

In [12]:

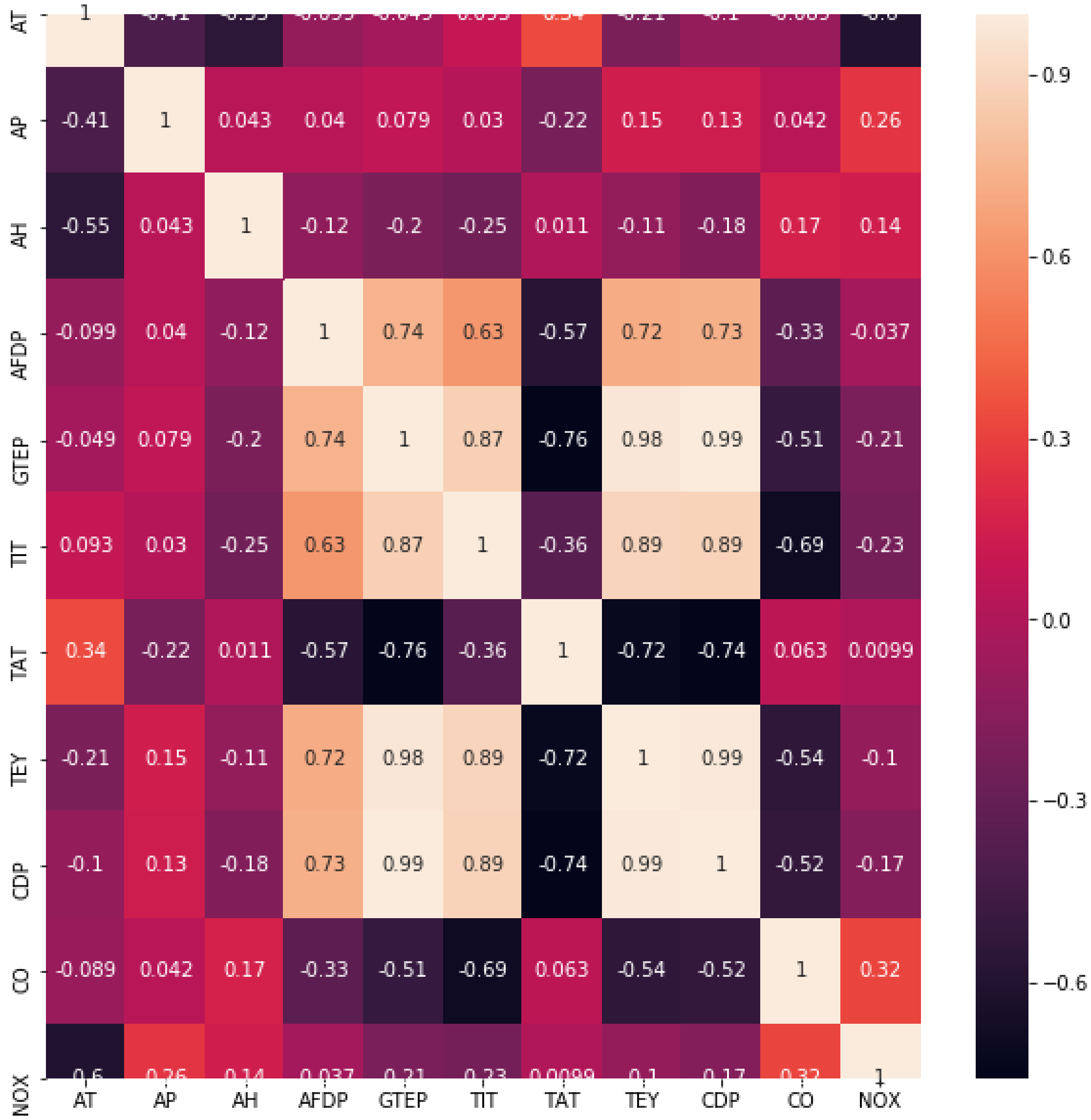
```
corr=gasturbines_data.corr()
```

In [13]:

```
plt.figure(figsize=(10,10))
sns.heatmap(corr,annot=True)
```

Out[13]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1e4a0178a08>
```



Neural Network model

In [14]:

```
gasturbines_data.columns
```

Out[14]:

```
Index(['AT', 'AP', 'AH', 'AFDP', 'GTEP', 'TIT', 'TAT', 'TEY', 'CDP', 'CO',
       'NOX'],
      dtype='object')
```

In [15]:

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

In [16]:

```
gasturbines_data = norm_func(gasturbines_data)
```

In [17]:

```
X = gasturbines_data.drop(columns=['TEY'])
y = gasturbines_data['TEY']
```

In [18]:

```
x_train,x_test,y_train,y_test= train_test_split(X,y, test_size=0.2)
```

In [19]:

```
model=Sequential()
model.add(Dense(64,input_dim=10,activation = 'relu'))
model.add(Dense(32,activation='relu'))
model.add(Dense(1,activation='linear'))
```

In [21]:

```
opt = Adam(lr=0.0015)
```

In [22]:

```
model.compile(optimizer=opt,loss='mean_squared_error',metrics=['mse'])
```

In [23]:

```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_1 (Dense)              (None, 64)                704       
_________________________________________________________________
dense_2 (Dense)              (None, 32)                2080      
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 33        
=================================================================
Total params: 2,817
Trainable params: 2,817
Non-trainable params: 0
_________________________________________________________________
```

In [24]:

```
history = model.fit(x_train,y_train,epochs = 70 ,batch_size=32,validation_split=0.1)
```

```
Epoch 15/70
10827/10827 [==============================] - 1s 57us/step - loss: 9.3625
e-05 - mse: 9.3625e-05 - val_loss: 9.2855e-05 - val_mse: 9.2855e-05
Epoch 16/70
10827/10827 [==============================] - 1s 54us/step - loss: 9.9159
e-05 - mse: 9.9159e-05 - val_loss: 1.1454e-04 - val_mse: 1.1454e-04
Epoch 17/70
10827/10827 [==============================] - 1s 53us/step - loss: 9.0762
e-05 - mse: 9.0762e-05 - val_loss: 8.0840e-05 - val_mse: 8.0840e-05
Epoch 18/70
10827/10827 [==============================] - 1s 54us/step - loss: 9.1187
e-05 - mse: 9.1187e-05 - val_loss: 1.2386e-04 - val_mse: 1.2386e-04
Epoch 19/70
10827/10827 [==============================] - 1s 62us/step - loss: 1.0786
e-04 - mse: 1.0786e-04 - val_loss: 8.1758e-05 - val_mse: 8.1758e-05
Epoch 20/70
10827/10827 [==============================] - 1s 53us/step - loss: 9.4412
e-05 - mse: 9.4412e-05 - val_loss: 9.4431e-05 - val_mse: 9.4431e-05
Epoch 21/70
10827/10827 [==============================] - 1s 55us/step - loss: 8.9472
```

In [25]:

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

Out[25]:

```
array([[0.16611634],
       [0.44217357],
       [0.15625161],
       ...,
       [0.81341505],
       [0.44701335],
       [0.46238443]], dtype=float32)
```

In [26]:

```
r2_score(y_test,y_predict)
```

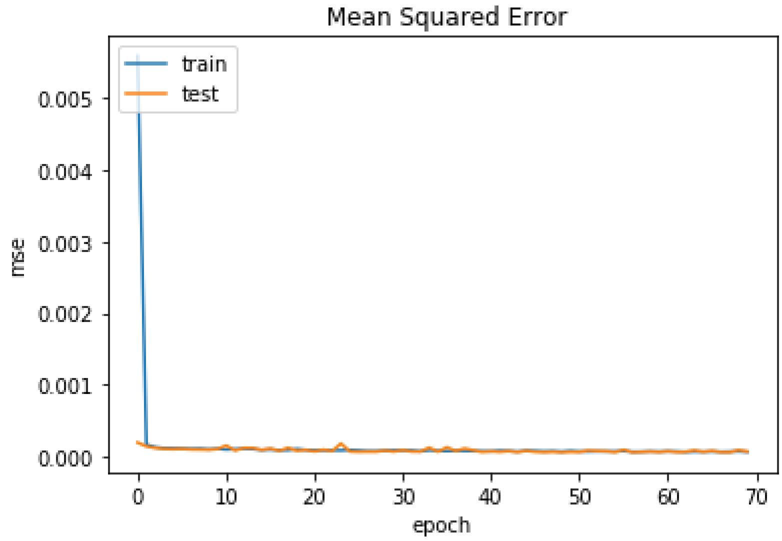
Out[26]:

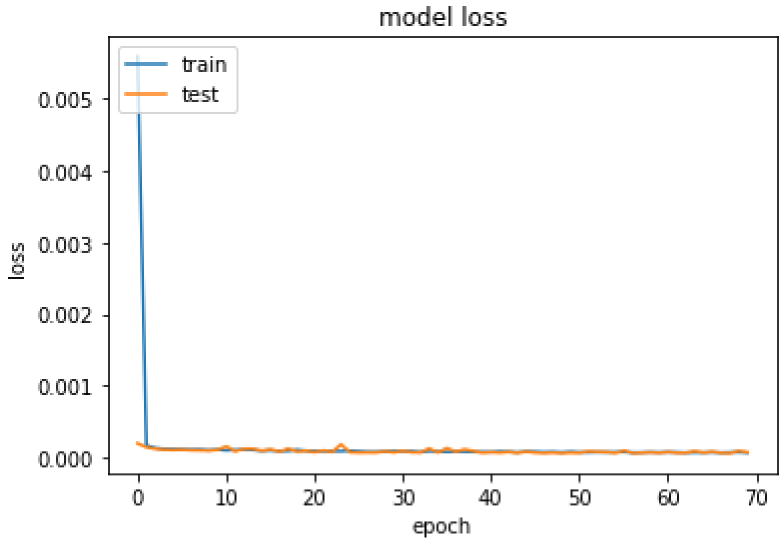
```
0.998366277280735
```

In [27]:

```
plt.plot(history.history['mse'])
plt.plot(history.history['val_mse'])
plt.title(' Mean Squared Error')
plt.ylabel('mse')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()

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.savefig('4.png')
plt.show()
```





>>>The End!!! <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<