In [2]:

```
from keras.models import Sequential
from keras.layers import Dense
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV, KFold
from keras.models import Sequential
from keras.layers import Dense
from keras.wrappers.scikit_learn import KerasClassifier
from tensorflow.keras.optimizers import Adam
from keras.layers import Dropout
```

executed in 298ms, finished 08:42:24 2021-11-27

In [3]:

```
df=pd.read_csv("gas_turbines.csv")
df=df.iloc[:,[7,0,1,2]]
df.shape
```

executed in 87ms, finished 08:42:51 2021-11-27

Out[3]:

```
(15039, 4)
```

In [4]:

```
df.describe()
```

executed in 86ms, finished 08:43:02 2021-11-27

Out[4]:

	TEY	AT	AP	AH
count	15039.000000	15039.000000	15039.00000	15039.000000
mean	134.188464	17.764381	1013.19924	79.124174
std	15.829717	7.574323	6.41076	13.793439
min	100.170000	0.522300	985.85000	30.344000
25%	127.985000	11.408000	1008.90000	69.750000
50%	133.780000	18.186000	1012.80000	82.266000
75%	140.895000	23.862500	1016.90000	90.043500
max	174.610000	34.929000	1034.20000	100.200000

In [5]:

```
df.isnull().sum()
```

executed in 26ms, finished 08:43:14 2021-11-27

Out[5]:

```
TEY    0
AT     0
AP     0
AH     0
dtype: int64
```

In [6]:

```
x=df.iloc[:,1:5]
y=df.iloc[:,0]
```

executed in 20ms, finished 08:43:24 2021-11-27

In [7]:

```
x
```

executed in 60ms, finished 08:43:37 2021-11-27

Out[7]:

	AT	AP	AH
0	6.8594	1007.9	96.799
1	6.7850	1008.4	97.118
2	6.8977	1008.8	95.939
3	7.0569	1009.2	95.249
4	7.3978	1009.7	95.150
...	...	...	...
15034	9.0301	1005.6	98.460
15035	7.8879	1005.9	99.093
15036	7.2647	1006.3	99.496
15037	7.0060	1006.8	99.008
15038	6.9279	1007.2	97.533

15039 rows × 3 columns

In [8]:

```
y
```

executed in 26ms, finished 08:43:49 2021-11-27

Out[8]:

```
0        114.70
1        114.72
2        114.71
3        114.72
4        114.72
          ...
15034    111.61
15035    111.78
15036    110.19
15037    110.74
15038    111.58
Name: TEY, Length: 15039, dtype: float64
```

In [9]:

```
seed = 7
np.random.seed(seed)
model = Sequential()
model.add(Dense(12, input_dim=3,  kernel_initializer='uniform', activation='relu'))
model.add(Dense(8,  kernel_initializer='uniform', activation='relu'))
model.add(Dense(1,  kernel_initializer='uniform', activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
history= model.fit(x, y,validation_split=0.33, epochs=150, batch_size=10)
```

executed in 6m 51s, finished 08:51:00 2021-11-27

```
Epoch 1/150
1008/1008 [==============================] - 4s 4ms/step - loss: -4588953.
5000 - accuracy: 0.0000e+00 - val_loss: -17759206.0000 - val_accuracy: 0.0
000e+00
Epoch 2/150
1008/1008 [==============================] - 4s 4ms/step - loss: -6137645
6.0000 - accuracy: 0.0000e+00 - val_loss: -120416464.0000 - val_accuracy:
0.0000e+00
Epoch 3/150
1008/1008 [==============================] - 3s 3ms/step - loss: -22713726
4.0000 - accuracy: 0.0000e+00 - val_loss: -345292928.0000 - val_accuracy:
0.0000e+00
Epoch 4/150
1008/1008 [==============================] - 3s 3ms/step - loss: -52835856
0.0000 - accuracy: 0.0000e+00 - val_loss: -710989696.0000 - val_accuracy:
0.0000e+00
Epoch 5/150
1008/1008 [==============================] - 3s 3ms/step - loss: -98124755
2.0000 - accuracy: 0.0000e+00 - val_loss: -1232011136.0000 - val_accuracy:
```

In [10]:

```
scores = model.evaluate(x, y)
print("%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
```

executed in 1.21s, finished 08:51:16 2021-11-27

```
470/470 [==============================] - 1s 2ms/step - loss: -970035875020
8.0000 - accuracy: 0.0000e+00
accuracy: 0.00%
```

In [11]:

```
history.history.keys()
```

executed in 22ms, finished 08:51:48 2021-11-27

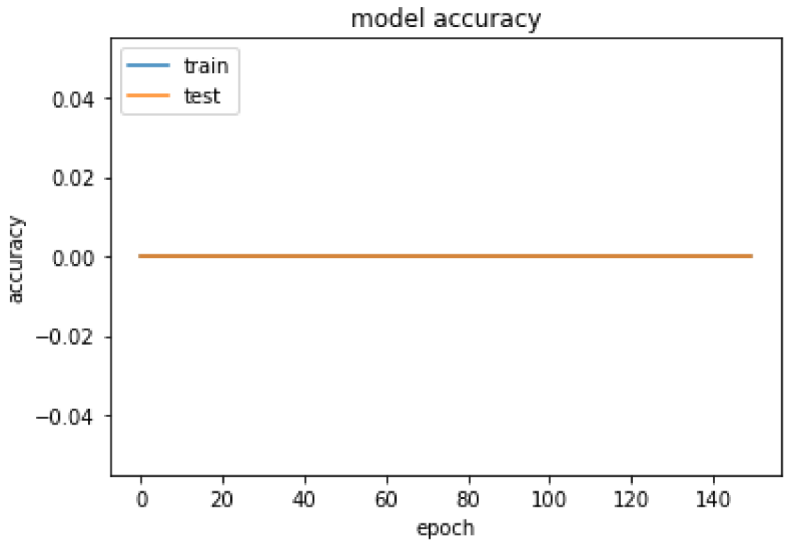
Out[11]:

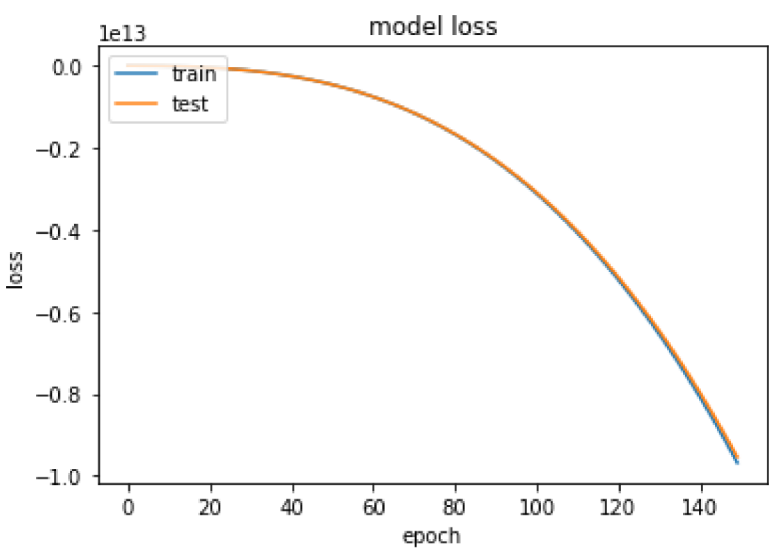
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

In [12]:

```
import matplotlib.pyplot as plt
# summarize history for accuracy
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()
# summarize history for loss
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()
```

executed in 1.26s, finished 08:52:04 2021-11-27





In [13]:

```
a = StandardScaler()
a.fit(x)
X_standardized = a.transform(x)
pd.DataFrame(X_standardized).describe()
```

executed in 83ms, finished 08:52:18 2021-11-27

Out[13]:

	0	1	2
count	1.503900e+04	1.503900e+04	1.503900e+04
mean	-2.320107e-16	-1.925280e-14	1.844983e-16
std	1.000033e+00	1.000033e+00	1.000033e+00
min	-2.276462e+00	-4.266288e+00	-3.536594e+00
25%	-8.392292e-01	-6.706510e-01	-6.796337e-01
50%	5.566605e-02	-6.227861e-02	2.277844e-01
75%	8.051309e-01	5.772924e-01	7.916582e-01
max	2.266234e+00	3.275970e+00	1.528011e+00

In [14]:

```
def create_model():
    model = Sequential()
    model.add(Dense(12, input_dim=3, kernel_initializer='uniform', activation='relu'))
    model.add(Dense(8, kernel_initializer='uniform', activation='relu'))
    model.add(Dense(1, kernel_initializer='uniform', activation='sigmoid'))\

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

executed in 21ms, finished 08:52:33 2021-11-27

In [15]:

```
model = KerasClassifier(build_fn = create_model,verbose = 0)
# Define the grid search parameters
batch_size = [10,20,40]
epochs = [10,50,100]
# Make a dictionary of the grid search parameters
param_grid = dict(batch_size = batch_size,epochs = epochs)
# Build and fit the GridSearchCV
grid = GridSearchCV(estimator = model,param_grid = param_grid,cv = KFold(),verbose = 10)
grid_result = grid.fit(X_standardized,y)
```

executed in 42m 23s, finished 09:35:09 2021-11-27

```
<ipython-input-15-05fa7a3e25fe>:1: DeprecationWarning: KerasClassifier is
deprecated, use Sci-Keras (https://github.com/adriangb/scikeras) instead.
  model = KerasClassifier(build_fn = create_model,verbose = 0)
C:\Users\win\anaconda3\lib\site-packages\keras\optimizer_v2\adam.py:105: U
serWarning: The `lr` argument is deprecated, use `learning_rate` instead.
  super(Adam, self).__init__(name, **kwargs)


Fitting 5 folds for each of 9 candidates, totalling 45 fits
[CV 1/5; 1/9] START batch_size=10, epochs=1
0.................................
[CV 1/5; 1/9] END ..................batch_size=10, epochs=10; total time=
20.4s
[CV 2/5; 1/9] START batch_size=10, epochs=1
0.................................

C:\Users\win\anaconda3\lib\site-packages\keras\optimizer_v2\adam.py:105: U
serWarning: The `lr` argument is deprecated, use `learning_rate` instead.
  super(Adam, self).__init__(name, **kwargs)
```

In [16]:

```
print('Best : {}, using {}'.format(grid_result.best_score_,grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
  print('{},{} with: {}'.format(mean, stdev, param))
```

executed in 16ms, finished 09:39:28 2021-11-27

```
Best : 0.0, using {'batch_size': 10, 'epochs': 10}
0.0,0.0 with: {'batch_size': 10, 'epochs': 10}
0.0,0.0 with: {'batch_size': 10, 'epochs': 50}
0.0,0.0 with: {'batch_size': 10, 'epochs': 100}
0.0,0.0 with: {'batch_size': 20, 'epochs': 10}
0.0,0.0 with: {'batch_size': 20, 'epochs': 50}
0.0,0.0 with: {'batch_size': 20, 'epochs': 100}
0.0,0.0 with: {'batch_size': 40, 'epochs': 10}
0.0,0.0 with: {'batch_size': 40, 'epochs': 50}
0.0,0.0 with: {'batch_size': 40, 'epochs': 100}
```

In [17]:

```
def create_model(learning_rate,dropout_rate):
    model = Sequential()
    model.add(Dense(8,input_dim = 3,kernel_initializer = 'normal',activation = 'relu'))
    model.add(Dropout(dropout_rate))
    model.add(Dense(4,input_dim = 3,kernel_initializer = 'normal',activation = 'relu'))
    model.add(Dropout(dropout_rate))
    model.add(Dense(1,activation = 'sigmoid'))

    adam = Adam(lr = learning_rate)
    model.compile(loss = 'binary_crossentropy',optimizer = adam,metrics = ['accuracy'])
    return model
```

executed in 21ms, finished 09:39:40 2021-11-27

In [18]:

```
model = KerasClassifier(build_fn = create_model,verbose = 0,batch_size = 40,epochs = 10)

# Define the grid search parameters

learning_rate = [0.001,0.01,0.1]
dropout_rate = [0.0,0.1,0.2]

# Make a dictionary of the grid search parameters

param_grids = dict(learning_rate = learning_rate,dropout_rate = dropout_rate)

# Build and fit the GridSearchCV

grid = GridSearchCV(estimator = model,param_grid = param_grids,cv = KFold(),verbose = 10)
grid_result = grid.fit(X_standardized,y)
```

executed in 4m 34s, finished 09:44:30 2021-11-27

```
<ipython-input-18-38fb56cb5200>:1: DeprecationWarning: KerasClassifier is
deprecated, use Sci-Keras (https://github.com/adriangb/scikeras) instead.
  model = KerasClassifier(build_fn = create_model,verbose = 0,batch_size =
40,epochs = 10)
C:\Users\win\anaconda3\lib\site-packages\keras\optimizer_v2\adam.py:105: U
serWarning: The `lr` argument is deprecated, use `learning_rate` instead.
  super(Adam, self).__init__(name, **kwargs)

Fitting 5 folds for each of 9 candidates, totalling 45 fits
[CV 1/5; 1/9] START dropout_rate=0.0, learning_rate=0.00
1.....................
[CV 1/5; 1/9] END .....dropout_rate=0.0, learning_rate=0.001; total time=
5.4s
[CV 2/5; 1/9] START dropout_rate=0.0, learning_rate=0.00
1.....................

C:\Users\win\anaconda3\lib\site-packages\keras\optimizer_v2\adam.py:105: U
serWarning: The `lr` argument is deprecated, use `learning_rate` instead.
  super(Adam, self).__init__(name, **kwargs)
```

In [19]:

```
print('Best : {}, using {}'.format(grid_result.best_score_,grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
  print('{},{} with: {}'.format(mean, stdev, param))
```

executed in 14ms, finished 09:44:48 2021-11-27

```
Best : 0.0, using {'dropout_rate': 0.0, 'learning_rate': 0.001}
0.0,0.0 with: {'dropout_rate': 0.0, 'learning_rate': 0.001}
0.0,0.0 with: {'dropout_rate': 0.0, 'learning_rate': 0.01}
0.0,0.0 with: {'dropout_rate': 0.0, 'learning_rate': 0.1}
0.0,0.0 with: {'dropout_rate': 0.1, 'learning_rate': 0.001}
0.0,0.0 with: {'dropout_rate': 0.1, 'learning_rate': 0.01}
0.0,0.0 with: {'dropout_rate': 0.1, 'learning_rate': 0.1}
0.0,0.0 with: {'dropout_rate': 0.2, 'learning_rate': 0.001}
0.0,0.0 with: {'dropout_rate': 0.2, 'learning_rate': 0.01}
0.0,0.0 with: {'dropout_rate': 0.2, 'learning_rate': 0.1}
```

In [20]:

```
def create_model(activation_function,init):
    model = Sequential()
    model.add(Dense(8,input_dim = 3,kernel_initializer = init,activation = activation_funct
    model.add(Dropout(0.1))
    model.add(Dense(4,input_dim = 3,kernel_initializer = init,activation = activation_funct
    model.add(Dropout(0.1))
    model.add(Dense(1,activation = 'sigmoid'))

    adam = Adam(lr = 0.001)
    model.compile(loss = 'binary_crossentropy',optimizer = adam,metrics = ['accuracy'])
    return model
```

executed in 16ms, finished 09:45:01 2021-11-27

In [21]:

```
model = KerasClassifier(build_fn = create_model,verbose = 0,batch_size = 40,epochs = 10)

# Define the grid search parameters
activation_function = ['softmax','relu','tanh','linear']
init = ['uniform','normal','zero']

# Make a dictionary of the grid search parameters
param_grids = dict(activation_function = activation_function,init = init)

# Build and fit the GridSearchCV

grid = GridSearchCV(estimator = model,param_grid = param_grids,cv = KFold(),verbose = 10)
grid_result = grid.fit(X_standardized,y)
```

executed in 6m 15s, finished 09:51:30 2021-11-27

```
<ipython-input-21-71e17feed8ca>:1: DeprecationWarning: KerasClassifier is
deprecated, use Sci-Keras (https://github.com/adriangb/scikeras) instead.
  model = KerasClassifier(build_fn = create_model,verbose = 0,batch_size =
40,epochs = 10)
C:\Users\win\anaconda3\lib\site-packages\keras\optimizer_v2\adam.py:105: U
serWarning: The `lr` argument is deprecated, use `learning_rate` instead.
  super(Adam, self).__init__(name, **kwargs)

Fitting 5 folds for each of 12 candidates, totalling 60 fits
[CV 1/5; 1/12] START activation_function=softmax, init=unifor
m.................
[CV 1/5; 1/12] END activation_function=softmax, init=uniform; total time=
6.2s
[CV 2/5; 1/12] START activation_function=softmax, init=unifor
m.................

C:\Users\win\anaconda3\lib\site-packages\keras\optimizer_v2\adam.py:105: U
serWarning: The `lr` argument is deprecated, use `learning_rate` instead.
  super(Adam, self).__init__(name, **kwargs)
```

In [22]:

```
def create_model(neuron1,neuron2):
    model = Sequential()
    model.add(Dense(neuron1,input_dim = 3,kernel_initializer = 'uniform',activation = 'tanh
    model.add(Dropout(0.2))
    model.add(Dense(neuron2,input_dim = neuron1,kernel_initializer = 'uniform',activation =
    model.add(Dropout(0.1))
    model.add(Dense(1,activation = 'sigmoid'))

    adam = Adam(lr = 0.001)
    model.compile(loss = 'binary_crossentropy',optimizer = adam,metrics = ['accuracy'])
    return model
```

executed in 23ms, finished 09:51:49 2021-11-27

In [23]:

```
model = KerasClassifier(build_fn = create_model,verbose = 0,batch_size = 40,epochs = 10)

# Define the grid search parameters

neuron1 = [4,8,16]
neuron2 = [2,4,8]

# Make a dictionary of the grid search parameters

param_grids = dict(neuron1 = neuron1,neuron2 = neuron2)

# Build and fit the GridSearchCV

grid = GridSearchCV(estimator = model,param_grid = param_grids,cv = KFold(),verbose = 10)
grid_result = grid.fit(X_standardized,y)
```

executed in 4m 43s, finished 09:56:45 2021-11-27

```
<ipython-input-23-67fe63bd7fd4>:1: DeprecationWarning: KerasClassifier is
deprecated, use Sci-Keras (https://github.com/adriangb/scikeras) instead.
  model = KerasClassifier(build_fn = create_model,verbose = 0,batch_size =
40,epochs = 10)
C:\Users\win\anaconda3\lib\site-packages\keras\optimizer_v2\adam.py:105: U
serWarning: The `lr` argument is deprecated, use `learning_rate` instead.
  super(Adam, self).__init__(name, **kwargs)


Fitting 5 folds for each of 9 candidates, totalling 45 fits
[CV 1/5; 1/9] START neuron1=4, neuron2=
2.....................................
[CV 1/5; 1/9] END .....................neuron1=4, neuron2=2; total time=
6.2s
[CV 2/5; 1/9] START neuron1=4, neuron2=
2.....................................

C:\Users\win\anaconda3\lib\site-packages\keras\optimizer_v2\adam.py:105: U
serWarning: The `lr` argument is deprecated, use `learning_rate` instead.
```

In [24]:

```
print('Best : {}, using {}'.format(grid_result.best_score_,grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
  print('{},{} with: {}'.format(mean, stdev, param))
```

executed in 21ms, finished 09:58:06 2021-11-27

```
Best : 0.0, using {'neuron1': 4, 'neuron2': 2}
0.0,0.0 with: {'neuron1': 4, 'neuron2': 2}
0.0,0.0 with: {'neuron1': 4, 'neuron2': 4}
0.0,0.0 with: {'neuron1': 4, 'neuron2': 8}
0.0,0.0 with: {'neuron1': 8, 'neuron2': 2}
0.0,0.0 with: {'neuron1': 8, 'neuron2': 4}
0.0,0.0 with: {'neuron1': 8, 'neuron2': 8}
0.0,0.0 with: {'neuron1': 16, 'neuron2': 2}
0.0,0.0 with: {'neuron1': 16, 'neuron2': 4}
0.0,0.0 with: {'neuron1': 16, 'neuron2': 8}
```

In [25]:

```
from sklearn.metrics import classification_report, accuracy_score
def create_model():
    model = Sequential()
    model.add(Dense(16,input_dim = 3,kernel_initializer = 'normal',activation = 'linear'))
    model.add(Dropout(0.1))
    model.add(Dense(8,input_dim = 3,kernel_initializer = 'normal',activation = 'linear'))
    model.add(Dropout(0.1))
    model.add(Dense(1,activation = 'linear'))

    adam = Adam(lr = 0.01) #sgd = SGD(lr=learning_rate, momentum=momentum, decay=decay_rate
    model.compile(loss = 'binary_crossentropy',optimizer = adam,metrics = ['accuracy'])
    return model
```

executed in 22ms, finished 09:58:18 2021-11-27

In [26]:

```
model = KerasClassifier(build_fn = create_model,verbose = 0,batch_size = 40,epochs = 100)

# Fitting the model

model.fit(X_standardized,y)

# Predicting using trained model

y_predict = model.predict(X_standardized)

# Printing the metrics
print(accuracy_score(y_predict.round(),y.round()))
```

executed in 1m 4.25s, finished 09:59:45 2021-11-27

```
<ipython-input-26-8ec86164b382>:1: DeprecationWarning: KerasClassifier is de
precated, use Sci-Keras (https://github.com/adriangb/scikeras) instead.
  model = KerasClassifier(build_fn = create_model,verbose = 0,batch_size = 4
0,epochs = 100)
C:\Users\win\anaconda3\lib\site-packages\keras\optimizer_v2\adam.py:105: Use
rWarning: The `lr` argument is deprecated, use `learning_rate` instead.
  super(Adam, self).__init__(name, **kwargs)

0.00013298756566261055
```