In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
gas_data = pd.read_csv('gas_turbines.csv')
gas_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

In [3]:
```
1 gas_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15039 entries, 0 to 15038
Data columns (total 11 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   AT      15039 non-null  float64
 1   AP      15039 non-null  float64
 2   AH      15039 non-null  float64
 3   AFDP    15039 non-null  float64
 4   GTEP    15039 non-null  float64
 5   TIT     15039 non-null  float64
 6   TAT     15039 non-null  float64
 7   TEY     15039 non-null  float64
 8   CDP     15039 non-null  float64
 9   CO      15039 non-null  float64
 10  NOX     15039 non-null  float64
dtypes: float64(11)
memory usage: 1.3 MB
```

In [4]:
```
1 gas_data.describe()
```

Out[4]:

	AT	AP	AH	AFDP	GTEP	TIT	TAT	TEY	CDP	
count	15039.000000	15039.00000	15039.000000	15039.000000	15039.000000	15039.000000	15039.000000	15039.000000	15039.000000	15
mean	17.764381	1013.19924	79.124174	4.200294	25.419061	1083.798770	545.396183	134.188464	12.102353	
std	7.574323	6.41076	13.793439	0.760197	4.173916	16.527806	7.866803	15.829717	1.103196	
min	0.522300	985.85000	30.344000	2.087400	17.878000	1000.800000	512.450000	100.170000	9.904400	
25%	11.408000	1008.90000	69.750000	3.723900	23.294000	1079.600000	542.170000	127.985000	11.622000	
50%	18.186000	1012.80000	82.266000	4.186200	25.082000	1088.700000	549.890000	133.780000	12.025000	
75%	23.862500	1016.90000	90.043500	4.550900	27.184000	1096.000000	550.060000	140.895000	12.578000	
max	34.929000	1034.20000	100.200000	7.610600	37.402000	1100.800000	550.610000	174.610000	15.081000	

In [6]:
```
#Model Building
from sklearn.preprocessing import StandardScaler
import sklearn
import keras
from keras.wrappers.scikit_learn import KerasRegressor
from keras.models import Sequential
from keras.layers import InputLayer,Dense
import tensorflow as tf
```

In [7]:
```
x = gas_data.loc[:,['AT','AP','AH','AFDP','GTEP','TIT','TAT','CDP','CO','NOX']]
y = gas_data.loc[:,['TEY']]
```

In [8]:
```
scaler=StandardScaler()
x=scaler.fit_transform(x)
y=scaler.fit_transform(y)
```

In [9]:
```
def baseline_model():
    model = Sequential()
    model.add(Dense(10, input_dim=10, activation='tanh'))
    model.add(Dense(1))
    model.compile(loss='mean_squared_error', optimizer = 'adam')
    return model
```

In [12]:
```
#Model Validation
from sklearn.model_selection import cross_val_score, KFold, train_test_split
from sklearn.metrics import mean_squared_error

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

In [13]:
```
estimator = KerasRegressor(build_fn=baseline_model, nb_epoch=50, batch_size=100, verbose=False)
kfold = KFold(n_splits=10)
results = cross_val_score(estimator, x, y, cv=kfold)
print("Results: %.2f (%.2f) MSE" % (results.mean(), results.std()))
```

```
Results: -0.19 (0.14) MSE
```

In [15]:
```
estimator.fit(x,y)
prediction = estimator.predict(x)
```

In [16]:
```
prediction
```

Out[16]:
```
array([-0.99548817, -0.97138625, -0.97187394, ..., -0.73313797,
       -0.7611553 , -0.81777555], dtype=float32)
```

In [17]:
```
a = scaler.inverse_transform(prediction)
a
```

Out[17]:
```
array([118.430695, 118.8122  , 118.80449 , ..., 122.58348 , 122.13999 ,
       121.24374 ], dtype=float32)
```

In [18]:
```
b = scaler.inverse_transform(y)
b
```

Out[18]:
```
array([[114.7 ],
       [114.72],
       [114.71],
       ...,
       [110.19],
       [110.74],
       [111.58]])
```

In [19]:
```
mean_squared_error(b,a)
```

Out[19]:
```
54.16840609454398
```

In [20]:
```
X_train,X_test,y_train,y_test = train_test_split(x,y,test_size=0.3)
```

In [22]:
```
estimator.fit(X_train,y_train)
prediction = estimator.predict(X_test)
```

In [23]:
```
1 prediction
```

Out[23]:
```
array([ 1.3786262e+00, -1.1677210e+00,  6.4089811e-01, ...,
       -9.2577701e-04, -7.1702474e-01,  1.7670633e+00], dtype=float32)
```

In [24]:
```
1 c = scaler.inverse_transform(prediction)
```

In [25]:
```
1 d = scaler.inverse_transform(y_test)
```

In [26]:
```
1 mean_squared_error(d,c)
```

Out[26]:
```
40.61129044346486
```

In [ ]:
```
1
```