In [1]:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, cross_val_sco
import seaborn as sns
from matplotlib import pyplot as plt
%matplotlib inline
import tensorflow as tf
tf.debugging.set_log_device_placement(False)
import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```
tf.random.set_seed(14)
```

In [3]:

```
raw_data = pd.read_csv("gas_turbines.csv")
raw_data.head()
#TEY is the variable we should predict.
```

Out[3]:

	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

In [4]:

```
df = raw_data.copy()
df = df.drop(['AFDP','GTEP','TIT','TAT','CDP','CO','NOX'],axis=1)
df.head()
```

Out[4]:

	AT	AP	AH	TEY
0	6.8594	1007.9	96.799	114.70
1	6.7850	1008.4	97.118	114.72
2	6.8977	1008.8	95.939	114.71
3	7.0569	1009.2	95.249	114.72
4	7.3978	1009.7	95.150	114.72

In [5]:

```
df.info() #No null values
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15039 entries, 0 to 15038
Data columns (total 4 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   TEY     15039 non-null  float64
dtypes: float64(4)
memory usage: 470.1 KB
```

In [6]:

```
df.describe()
```

Out[6]:

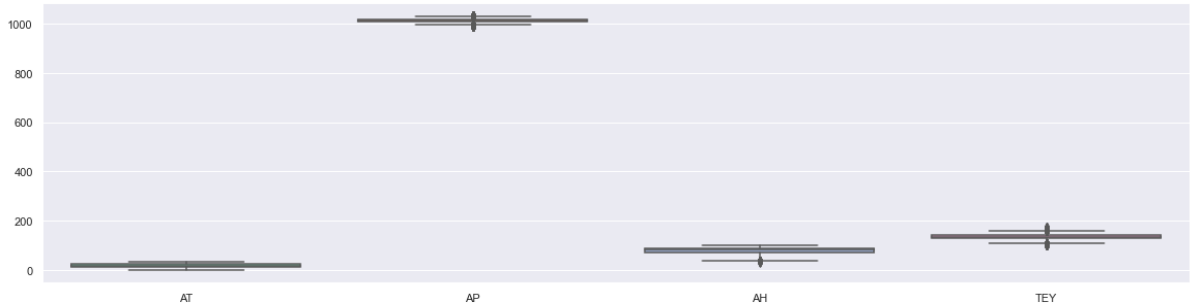
	AT	AP	AH	TEY
count	15039.000000	15039.00000	15039.000000	15039.000000
mean	17.764381	1013.19924	79.124174	134.188464
std	7.574323	6.41076	13.793439	15.829717
min	0.522300	985.85000	30.344000	100.170000
25%	11.408000	1008.90000	69.750000	127.985000
50%	18.186000	1012.80000	82.266000	133.780000
75%	23.862500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

In [7]:

```
sns.set(rc={'figure.figsize':(20,5)})
sns.boxplot(data=df, orient="v", palette="Set2")
```

Out[7]:

```
<AxesSubplot:>
```



Train | Split dataset

In [8]:

```
X =df.iloc[:,:-1]
Y = df.iloc[:,-1]


X_train, X_test, y_train, y_test = train_test_split(X,Y, test_size
```

In [9]:

```
y_train=np.reshape(y_train.to_numpy(), (-1,1)) #https://stackoverfl
y_test=np.reshape(y_test.to_numpy(), (-1,1))
```

In [10]:

```
from sklearn.preprocessing import MinMaxScaler

scaler_x = MinMaxScaler()
scaler_y = MinMaxScaler()

print(scaler_x.fit(X_train))
xtrain_scale=scaler_x.transform(X_train)

print(scaler_x.fit(X_test))
xtest_scale=scaler_x.transform(X_test)

print(scaler_y.fit(y_train))
ytrain_scale=scaler_y.transform(y_train)

print(scaler_y.fit(y_test))
ytest_scale=scaler_y.transform(y_test)
```

```
MinMaxScaler()
MinMaxScaler()
MinMaxScaler()
MinMaxScaler()
```

In [11]:

```
len(xtrain_scale)
```

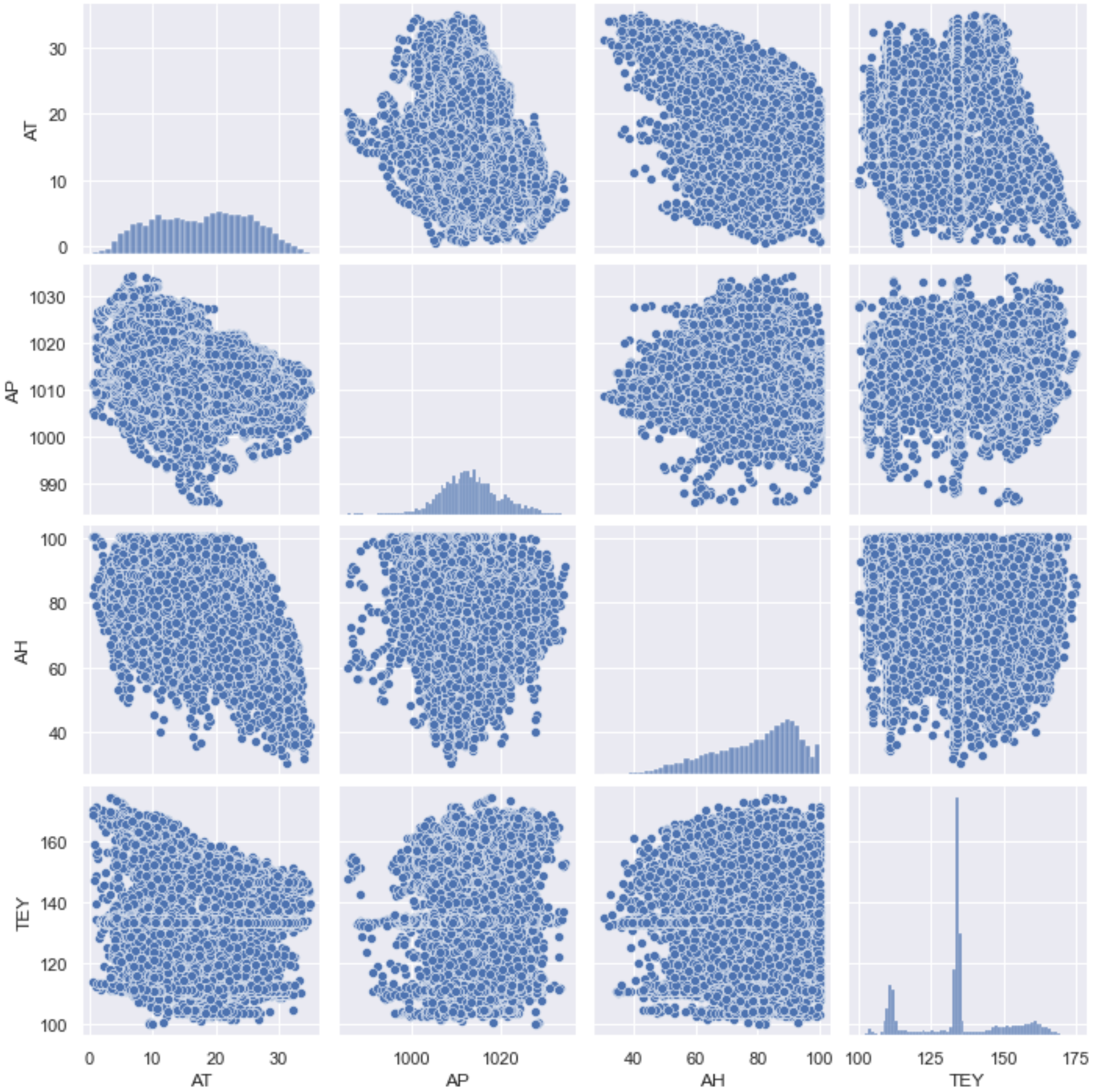
Out[11]:

```
10527
```

Visualizing the data

```
In [12]: sns.pairplot(df,palette='deep')
```

```
Out[12]: <seaborn.axisgrid.PairGrid at 0x7fd16b46fe80>
```



Neural Network Modelling

```
In [13]: import keras
         from keras.models import Sequential
         from keras.layers import Dense
         import keras
         keras. __version__ #init method is not available in this mdethod
```

```
Out[13]: '2.8.0'
```

In [14]:

```
# create model
model1 = Sequential()
model1.add(Dense(4, input_dim=3, kernel_initializer='normal', activ
model1.add(Dense(2106,kernel_initializer='normal', activation='relu
model1.add(Dense(1, activation='linear'))
# Compile model
model1.compile(loss='mean_squared_error', optimizer='adam', metrics
# Fit the model
hist1 = model1.fit(xtrain_scale, ytrain_scale, validation_split=0.3
#At epoch 50, mse and mae just keeps oscillating back and forth
```

```
- val_mae: 0.1572
Epoch 65/100
48/48 [==============================] - 1s 15ms/step - loss: 0.03
92 - mse: 0.0392 - mae: 0.1581 - val_loss: 0.0388 - val_mse: 0.038
8 - val_mae: 0.1555
Epoch 66/100
48/48 [==============================] - 0s 6ms/step - loss: 0.038
9 - mse: 0.0389 - mae: 0.1571 - val_loss: 0.0397 - val_mse: 0.0397
- val_mae: 0.1599
Epoch 67/100
48/48 [==============================] - 0s 7ms/step - loss: 0.039
4 - mse: 0.0394 - mae: 0.1589 - val_loss: 0.0394 - val_mse: 0.0394
- val_mae: 0.1571
Epoch 68/100
48/48 [==============================] - 1s 12ms/step - loss: 0.03
87 - mse: 0.0387 - mae: 0.1562 - val_loss: 0.0386 - val_mse: 0.038
6 - val_mae: 0.1570
Epoch 69/100
48/48 [==============================] - 0s 5ms/step - loss: 0.038
7 - mse: 0.0387 - mae: 0.1570 - val_loss: 0.0390 - val_mse: 0.0390
- val_mae: 0.1567
```

In [15]:

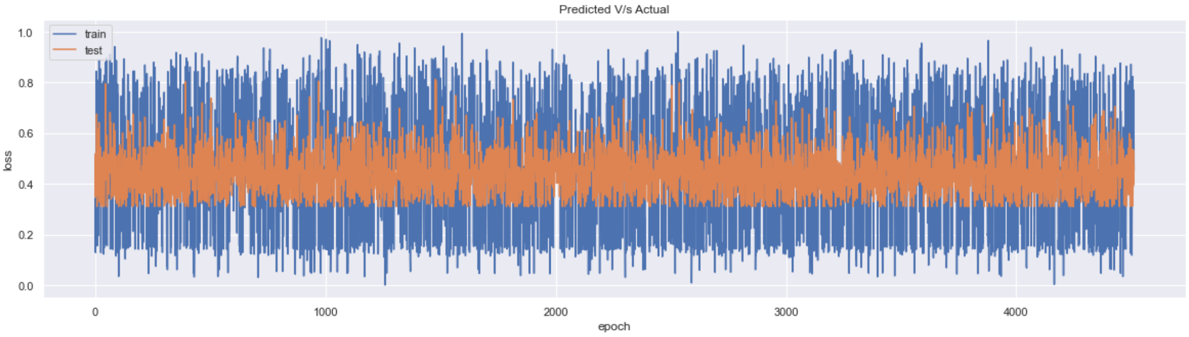
```
model1.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 4)                 16

 dense_1 (Dense)             (None, 2106)              10530

 dense_2 (Dense)             (None, 1)                 2107

=================================================================
Total params: 12,653
Trainable params: 12,653
Non-trainable params: 0
_________________________________________________________________
```

Model Evaluation

In [16]:
```
y_predict = model1.predict(xtest_scale)
```

In [17]:
```
plt.plot(ytest_scale)
plt.plot(y_predict)
plt.title('Predicted V/s Actual')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show() #Neural Networks is not a good model for predicting a re
```



In [18]:
```
print(hist1.history.keys())
```

```
dict_keys(['loss', 'mse', 'mae', 'val_loss', 'val_mse', 'val_mae']
)
```

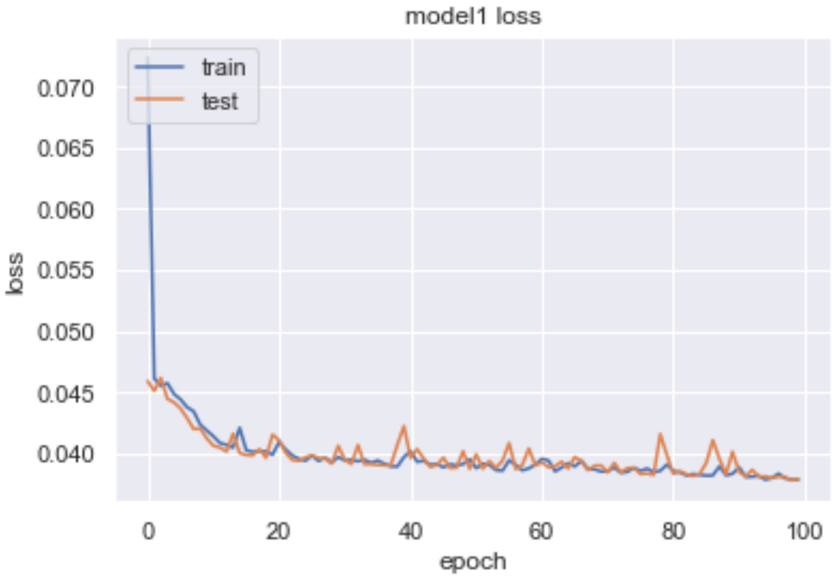
In [19]:
```
hist1_df = pd.DataFrame(hist1.history)
hist1_df["epoch"]=hist1.epoch
hist1_df.tail()
```

Out[19]:

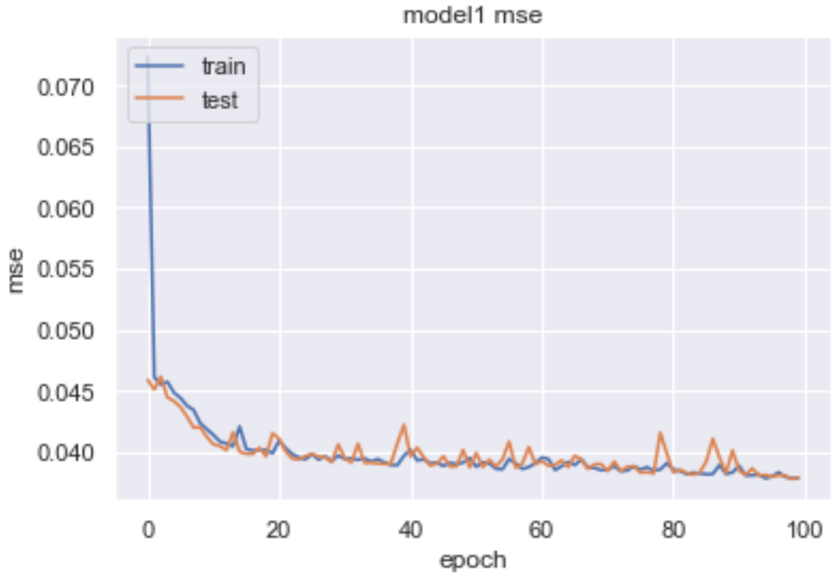
	loss	mse	mae	val_loss	val_mse	val_mae	epoch
95	0.037992	0.037992	0.155483	0.037985	0.037985	0.154435	95
96	0.038346	0.038346	0.155203	0.038096	0.038096	0.154923	96
97	0.037965	0.037965	0.154611	0.038079	0.038079	0.153172	97
98	0.037860	0.037860	0.154089	0.037801	0.037801	0.152929	98
99	0.037860	0.037860	0.153667	0.037914	0.037914	0.153416	99

Visualize Training History

```
In [20]: # summarize history for Loss

         sns.set(rc={'figure.figsize':(6,4)})

         plt.plot(hist1.history['loss'])
         plt.plot(hist1.history['val_loss'])
         plt.title('model1 loss')
         plt.ylabel('loss')
         plt.xlabel('epoch')
         plt.legend(['train', 'test'], loc='upper left')
         plt.show()
```



In [21]:
```
# summarize history for loss
plt.plot(hist1.history['mse'])
plt.plot(hist1.history['val_mse'])
plt.title('model1 mse')
plt.ylabel('mse')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



In [ ]: