In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import mutual_info_regression,SelectKBest
import tensorflow as tf
from tensorflow.keras.models import load_model,Sequential
from tensorflow.keras import layers
from keras.layers import Dense,Dropout

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

In [2]:
```
data = pd.read_csv('gas_turbines.csv')
data.head(50)
```

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	3.15470	82.722
1	6.7850	1008.4	97.118	3.4998	19.728	1059.3	550.00	114.72	10.598	3.23630	82.776
2	6.8977	1008.8	95.939	3.4824	19.779	1059.4	549.87	114.71	10.601	3.20120	82.468
3	7.0569	1009.2	95.249	3.4805	19.792	1059.6	549.99	114.72	10.606	3.19230	82.670
4	7.3978	1009.7	95.150	3.4976	19.765	1059.7	549.98	114.72	10.612	3.24840	82.311
5	7.6998	1010.7	92.708	3.5236	19.683	1059.8	549.97	114.72	10.626	3.44670	82.409
6	7.7901	1011.6	91.983	3.5298	19.659	1060.0	549.87	114.71	10.644	3.48740	82.440
7	7.7139	1012.7	91.348	3.5088	19.673	1059.8	549.92	114.71	10.656	3.60430	83.010
8	7.7975	1013.8	90.196	3.5141	19.634	1060.1	550.09	114.72	10.644	3.39430	82.284
9	8.0820	1015.0	88.597	4.0612	23.406	1083.0	550.21	131.70	11.679	1.90810	82.782
10	8.3047	1016.0	86.343	4.0870	23.747	1085.3	550.20	133.67	11.703	1.71180	81.995
11	8.4684	1016.1	86.491	4.0513	23.734	1085.1	550.14	134.24	11.775	1.46720	80.638
12	8.8856	1016.2	82.974	4.0503	23.869	1085.9	550.17	134.69	11.864	1.71130	80.533
13	9.3714	1016.6	79.980	4.0427	23.916	1085.9	550.09	134.68	11.860	1.66370	80.538
14	9.7962	1017.1	78.061	4.0613	23.962	1085.9	549.97	134.68	11.855	1.40610	80.463
15	9.6996	1017.9	76.382	4.0719	23.945	1085.9	549.84	134.69	11.850	1.40720	80.522
16	9.3111	1018.8	75.559	4.0578	23.890	1085.9	549.72	134.69	11.845	1.51930	80.648
17	8.6702	1019.8	77.476	4.0727	23.824	1085.9	549.60	134.69	11.840	1.42010	80.374
18	8.1375	1020.6	79.741	4.0432	23.832	1085.4	549.99	134.68	11.835	1.61560	80.639
19	7.0396	1021.2	83.102	4.0110	23.655	1084.9	550.03	134.68	11.830	1.69290	81.223
20	6.5345	1021.7	86.253	4.0678	23.801	1085.7	549.87	134.67	11.867	1.49750	84.556
21	6.4893	1022.2	86.190	4.1476	24.098	1087.6	549.66	133.78	12.014	1.34300	86.110
22	6.4935	1022.4	84.820	4.1916	24.053	1088.1	550.03	134.11	11.908	1.30340	86.607
23	6.0790	1022.8	85.053	3.5210	19.259	1057.0	548.09	112.80	10.565	4.64580	105.150
24	5.4134	1023.8	84.432	3.8312	21.433	1070.4	548.85	122.33	11.186	4.11650	105.730
25	5.4755	1024.0	82.830	4.1239	23.893	1087.2	549.96	133.94	11.968	1.46050	87.298
26	5.9958	1024.2	82.376	4.6570	28.128	1099.7	543.28	150.33	12.935	0.78039	82.101
27	6.7455	1024.3	80.521	4.0011	23.628	1085.0	550.06	134.24	11.913	1.43270	84.386
28	7.9304	1023.7	76.271	4.6570	28.433	1096.2	540.40	150.02	13.055	1.12220	75.403
29	9.3818	1022.8	56.158	4.6340	28.221	1100.1	543.62	150.50	12.971	0.91143	83.766
30	10.0050	1022.4	55.943	4.5008	27.325	1100.3	547.24	147.72	12.813	0.99325	87.119
31	10.1970	1022.3	57.235	4.6544	28.375	1099.8	543.21	150.60	12.873	0.93630	80.353
32	10.3620	1022.4	58.863	4.0960	24.085	1085.9	549.89	134.10	11.974	1.46240	82.310

	AT	AP	AH	AFDP	GTEP	TIT	TAT	TEY	CDP	CO	NOX
33	10.0710	1022.3	62.720	4.2234	25.219	1092.3	550.07	139.70	12.124	1.26940	81.801
34	9.0326	1022.6	66.693	4.2483	25.607	1095.0	550.06	142.04	12.327	1.17870	84.197
35	8.6055	1022.7	69.720	4.3266	26.342	1099.2	550.33	145.76	12.551	0.98319	86.667
36	7.8150	1022.4	73.101	4.4504	27.330	1100.1	546.95	149.63	12.771	0.86157	84.144
37	6.3292	1022.4	79.928	3.9692	23.639	1083.5	549.39	134.62	11.845	1.66840	83.796
38	5.2115	1022.4	85.430	4.0247	23.447	1083.4	550.05	134.37	11.783	1.70050	84.643
39	4.9731	1022.0	85.730	4.0163	23.363	1082.7	550.00	133.75	11.789	2.33590	83.760
40	4.5728	1021.7	87.397	4.0737	23.505	1083.9	549.96	134.70	11.883	1.72790	84.253
41	4.2643	1021.5	88.365	4.0944	23.544	1083.9	549.79	134.32	11.889	1.70170	84.737
42	4.2732	1021.3	88.375	4.2089	24.291	1088.1	550.10	135.89	12.042	1.51610	85.062
43	4.2691	1021.2	87.766	4.1960	24.088	1086.9	549.98	135.05	11.939	1.47710	85.051
44	4.3930	1021.5	86.622	3.8506	21.467	1037.9	528.08	116.72	11.141	31.97100	99.229
45	3.8907	1021.8	88.810	4.1002	23.675	1084.6	550.09	135.07	11.750	1.78590	85.129
46	3.7124	1022.3	90.196	4.2672	24.149	1087.4	550.16	134.61	11.960	1.54460	85.929
47	5.1136	1022.8	91.462	5.2454	31.325	1099.6	532.50	156.91	13.671	0.81911	66.938
48	6.3887	1022.8	92.302	5.1564	30.830	1099.8	534.63	158.69	13.508	0.74091	67.781
49	8.2123	1022.4	94.780	4.8574	28.643	1100.1	542.50	151.81	13.113	0.97347	76.702

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

Out[3]: (15039, 11)

In [4]:
```
data.describe()
```

Out[4]:

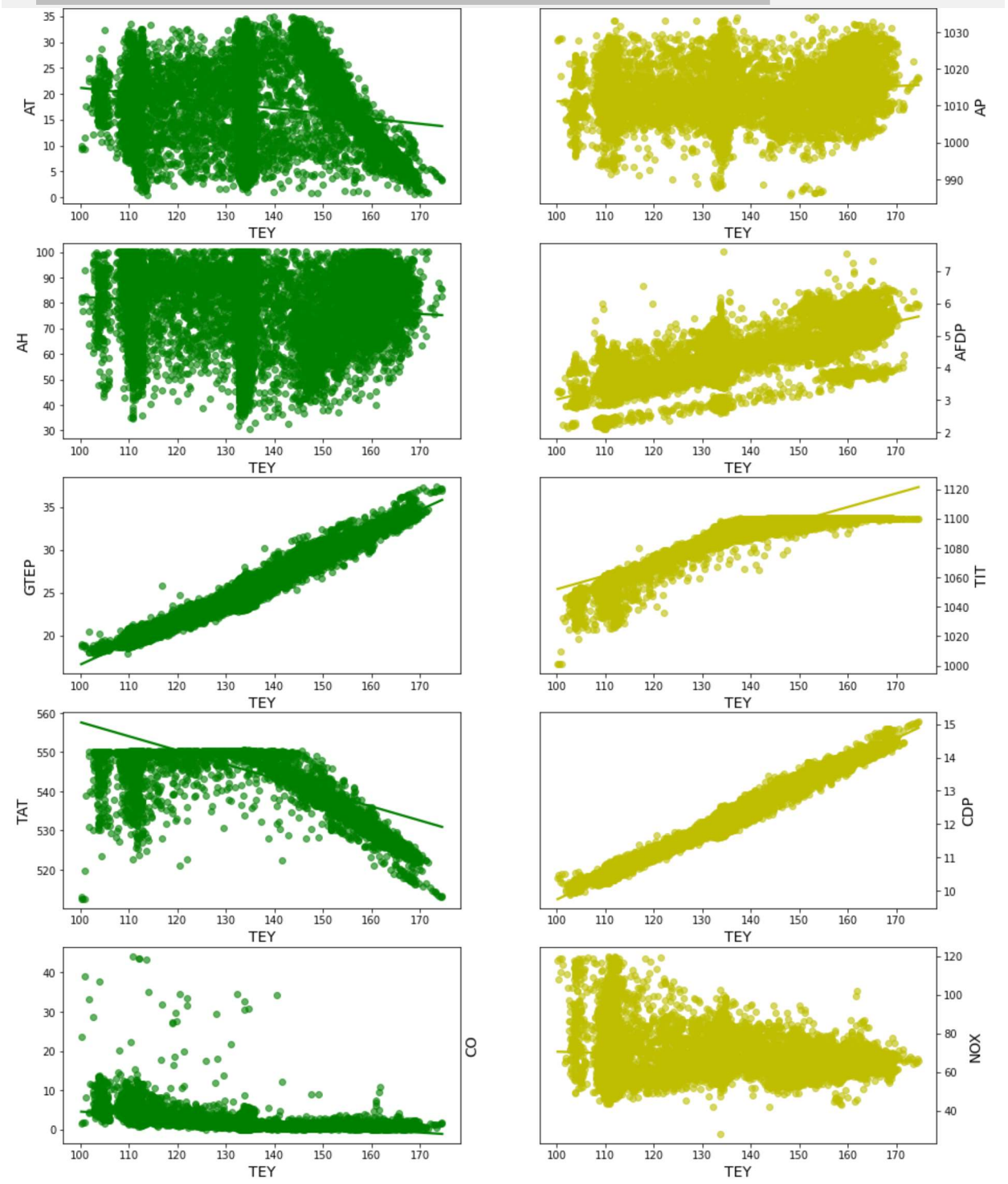
	AT	AP	AH	AFDP	GTEP	TIT	
count	15039.000000	15039.00000	15039.000000	15039.000000	15039.000000	15039.000000	15039.0
mean	17.764381	1013.19924	79.124174	4.200294	25.419061	1083.798770	545.3
std	7.574323	6.41076	13.793439	0.760197	4.173916	16.527806	7.8
min	0.522300	985.85000	30.344000	2.087400	17.878000	1000.800000	512.4
25%	11.408000	1008.90000	69.750000	3.723900	23.294000	1079.600000	542.1
50%	18.186000	1012.80000	82.266000	4.186200	25.082000	1088.700000	549.8
75%	23.862500	1016.90000	90.043500	4.550900	27.184000	1096.000000	550.0
max	34.929000	1034.20000	100.200000	7.610600	37.402000	1100.800000	550.6

In [5]:
```
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 [10]:
```
f, axes = plt.subplots(5,2, figsize=(15,20))
sns.regplot(x = 'TEY', y = 'AT', data = data, ax = axes[0,0], color = 'g', scatte
axes[0,0].set_xlabel('TEY', fontsize = 14)
axes[0,0].set_ylabel('AT', fontsize=14)
axes[0,0].yaxis.tick_left()

sns.regplot(x = 'TEY', y = 'AP', data = data, ax = axes[0,1], color = 'y',scatter
axes[0,1].set_xlabel('TEY', fontsize = 14)
axes[0,1].set_ylabel('AP', fontsize=14)
axes[0,1].yaxis.set_label_position("right")
axes[0,1].yaxis.tick_right()

sns.regplot(x = 'TEY', y = 'AH', data = data, ax = axes[1,0], color = 'g',scatter
axes[1,0].set_xlabel('TEY', fontsize = 14)
axes[1,0].set_ylabel('AH', fontsize=14)
axes[1,0].yaxis.tick_left()

sns.regplot(x = 'TEY', y = 'AFDP', data = data, ax = axes[1,1], color = 'y',scatt
axes[1,1].set_xlabel('TEY', fontsize = 14)
axes[1,1].set_ylabel('AFDP', fontsize=14)
axes[1,1].yaxis.set_label_position("right")
axes[1,1].yaxis.tick_right()

sns.regplot(x = 'TEY', y = 'GTEP', data = data, ax = axes[2,0], color = 'g',scatt
axes[2,0].set_xlabel('TEY', fontsize = 14)
axes[2,0].set_ylabel('GTEP', fontsize=14)
axes[2,0].yaxis.tick_left()

sns.regplot(x = 'TEY', y = 'TIT', data = data, ax = axes[2,1], color = 'y',scatte
axes[2,1].set_xlabel('TEY', fontsize = 14)
axes[2,1].set_ylabel('TIT', fontsize=14)
axes[2,1].yaxis.set_label_position("right")
axes[2,1].yaxis.tick_right()

sns.regplot(x = 'TEY', y = 'TAT', data = data, ax = axes[3,0], color = 'g',scatte
axes[3,0].set_xlabel('TEY', fontsize = 14)
axes[3,0].set_ylabel('TAT', fontsize=14)
axes[3,0].yaxis.tick_left()

sns.regplot(x = 'TEY', y = 'CDP', data = data, ax = axes[3,1], color = 'y',scatte
axes[3,1].set_xlabel('TEY', fontsize = 14)
axes[3,1].set_ylabel('CDP', fontsize=14)
axes[3,1].yaxis.set_label_position("right")
axes[3,1].yaxis.tick_right()

sns.regplot(x = 'TEY', y = 'CO', data = data, ax = axes[4,0], color = 'g',scatter
axes[4,0].set_xlabel('TEY', fontsize = 14)
axes[4,0].set_ylabel('CO', fontsize=14)
axes[4,0].yaxis.set_label_position("right")
axes[4,0].yaxis.tick_left()

sns.regplot(x = 'TEY', y = 'NOX', data = data, ax = axes[4,1], color = 'y',scatte
axes[4,1].set_xlabel('TEY', fontsize = 14)
axes[4,1].set_ylabel('NOX', fontsize=14)
axes[4,1].yaxis.set_label_position("right")
axes[4,1].yaxis.tick_right()
```

```
plt.show()
```



In [11]:
```
X = data.drop('TEY',axis=1)
y = data['TEY']
```

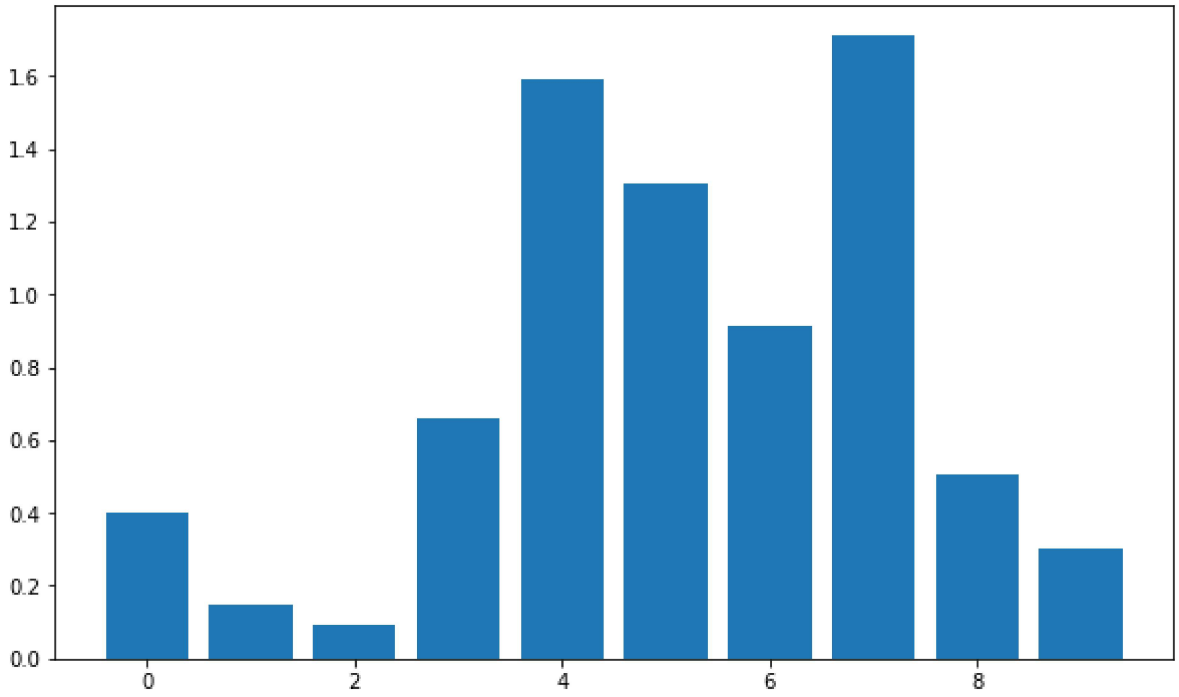
In [12]:
```
def select_features(X_train, y_train, X_test):
 # configure to select all features
    fs = SelectKBest(score_func=mutual_info_regression, k='all')
 # learn relationship from training data
    fs.fit(X_train, y_train)
 # transform train input data
    X_train_fs = fs.transform(X_train)
 # transform test input data
    X_test_fs = fs.transform(X_test)
    return X_train_fs, X_test_fs, fs
```

In [13]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_
X_train_fs, X_test_fs, fs = select_features(X_train, y_train, X_test)
```

In [14]:
```
for i in range(len(fs.scores_)):
    print('Feature %d: %f' % (i, fs.scores_[i]))
```

```
Feature 0: 0.403060
Feature 1: 0.146701
Feature 2: 0.091132
Feature 3: 0.656951
Feature 4: 1.591014
Feature 5: 1.301394
Feature 6: 0.911848
Feature 7: 1.710346
Feature 8: 0.506349
Feature 9: 0.302945
```

In [16]:
```
plt.subplots(figsize=(10, 6))
plt.bar([i for i in range(len(fs.scores_))], fs.scores_)
plt.show()
```



In [17]:
```
X = data.drop(['TEY','AT','AP','AH','CO','NOX'], axis = 1)
y = data['TEY']
```

In [18]:
```
std=StandardScaler()
std.fit(X)
```

Out[18]:
```
StandardScaler()
```

In [19]:
```
y = StandardScaler().fit_transform(y.values.reshape(len(y),1))[:,0]
```

In [20]:
```
scaled_features=std.transform(X)
data_head=pd.DataFrame(scaled_features,columns=X.columns)
data_head
```

Out[20]:

	AFDP	GTEP	TIT	TAT	CDP
0	-0.921232	-1.379101	-1.488376	0.585240	-1.357331
1	-0.921495	-1.363528	-1.482325	0.585240	-1.363676
2	-0.944385	-1.351309	-1.476275	0.568715	-1.360957
3	-0.946884	-1.348194	-1.464173	0.583969	-1.356424
4	-0.924389	-1.354663	-1.458123	0.582698	-1.350985
...	...	...	...	...	...
15034	-0.865850	-1.498657	-2.063184	0.103453	-1.543161
15035	-0.913470	-1.438759	-2.268905	-0.276638	-1.513247
15036	-0.951488	-1.410967	-2.789257	-1.026650	-1.467922
15037	-0.988848	-1.447624	-2.456474	-0.528337	-1.422598
15038	-1.016605	-1.464635	-2.051083	0.057689	-1.377273

15039 rows × 5 columns

In [21]:
```
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_s
```

In [22]:
```
model = Sequential()
model.add(Dense(10, input_dim=5, kernel_initializer='he_uniform', activation='tar
model.add(Dense(6, kernel_initializer='he_uniform', activation='tanh'))
model.add(Dense(1, kernel_initializer='he_uniform', activation='linear'))
```

In [23]:
```
model.compile(loss='mean_squared_error', optimizer='adam', metrics=['mse'])
```

In [24]:
```
model.fit(x_train,y_train, epochs=100, batch_size=40)
```

```
264/264 [==============================] - 1s 2ms/step - loss: 0.9957 - mse:
0.9957
Epoch 8/100
264/264 [==============================] - 0s 2ms/step - loss: 0.9965 - mse:
0.9965
Epoch 9/100
264/264 [==============================] - 1s 2ms/step - loss: 0.9961 - mse:
0.9961
Epoch 10/100
264/264 [==============================] - 0s 2ms/step - loss: 0.9958 - mse:
0.9958
Epoch 11/100
264/264 [==============================] - 0s 2ms/step - loss: 0.9953 - mse:
0.9953
Epoch 12/100
264/264 [==============================] - 1s 2ms/step - loss: 0.9947 - mse:
0.9947
Epoch 13/100
264/264 [==============================] - 1s 2ms/step - loss: 0.9948 - mse:
0.9948
```

In [25]:
```
scores = model.evaluate(x_test, y_test)
print((model.metrics_names[1]))
```

```
141/141 [==============================] - 0s 1ms/step - loss: 1.0172 - mse: 1.
0172
mse
```

In [ ]:
```
```