The dataset contains 36733 instances of 11 sensor measures aggregated over one hour (by means of average or sum) from a gas turbine. The Dataset includes gas turbine parameters (such as Turbine Inlet Temperature and Compressor Discharge pressure) in addition to the ambient variables.

Problem statement: predicting turbine energy yield (TEY) using ambient variables as features.

*Attribute Information:

The explanations of sensor measurements and their brief statistics are given below.

*Variable (Abbr.) Unit Min Max Mean

*Ambient temperature (AT) C â€“6.23 37.10 17.71

*Ambient pressure (AP) mbar 985.85 1036.56 1013.07

*Ambient humidity (AH) (%) 24.08 100.20 77.87

*Air filter difference pressure (AFDP) mbar 2.09 7.61 3.93

*Gas turbine exhaust pressure (GTEP) mbar 17.70 40.72 25.56

*Turbine inlet temperature (TIT) C 1000.85 1100.89 1081.43

*Turbine after temperature (TAT) C 511.04 550.61 546.16

*Compressor discharge pressure (CDP) mbar 9.85 15.16 12.06

*Turbine energy yield (TEY) MWH 100.02 179.50 133.51

*Carbon monoxide (CO) mg/m3 0.00 44.10 2.37

*Nitrogen oxides (NOx) mg/m3 25.90 119.91 65.29

```
import pandas as pd
```

```
data = pd.read_csv('gas_turbines.csv')
data
```

	AT	AP	AH	AFDP	GTEP	TIT	TAT	TEY	CDP	CO	
0	6.8594	1007.9	96.799	3.5000	19.663	1059.2	550.00	114.70	10.605	3.1547	82.
1	6.7850	1008.4	97.118	3.4998	19.728	1059.3	550.00	114.72	10.598	3.2363	82.
2	6.8977	1008.8	95.939	3.4824	19.779	1059.4	549.87	114.71	10.601	3.2012	82.
3	7.0569	1009.2	95.249	3.4805	19.792	1059.6	549.99	114.72	10.606	3.1923	82.

```
data.isna().sum()
```

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

```
data.describe(include='all')
```

	AT	AP	AH	AFDP	GTEP	T
count	15039.000000	15039.00000	15039.000000	15039.000000	15039.000000	15039.00000
mean	17.764381	1013.19924	79.124174	4.200294	25.419061	1083.7987
std	7.574323	6.41076	13.793439	0.760197	4.173916	16.5278
min	0.522300	985.85000	30.344000	2.087400	17.878000	1000.8000
25%	11.408000	1008.90000	69.750000	3.723900	23.294000	1079.6000
50%	18.186000	1012.80000	82.266000	4.186200	25.082000	1088.7000
75%	23.862500	1016.90000	90.043500	4.550900	27.184000	1096.0000
max	34.929000	1034.20000	100.200000	7.610600	37.402000	1100.8000

```
data=data.drop(['TAT','TEY','CDP','CO','NOX','AFDP','GTEP'],axis=1)
data
```

	AT	AP	AH	TIT
0	6.8594	1007.9	96.799	1059.2
1	6.7850	1008.4	97.118	1059.3
2	6.8977	1008.8	95.939	1059.4
3	7.0569	1009.2	95.249	1059.6
4	7.3978	1009.7	95.150	1059.7
...	...	...	...	...

```
# Normalization
def norm_func(i):
  X = (i-i.min())/(i.max()-i.min())
  return(X)
scaled_data=norm_func(data)
scaled_data
```

	AT	AP	AH	TIT
0	0.184182	0.456050	0.951314	0.584
1	0.182020	0.466391	0.955881	0.585
2	0.185295	0.474664	0.939003	0.586
3	0.189922	0.482937	0.929126	0.588
4	0.199830	0.493278	0.927708	0.589
...	...	...	...	...
15034	0.247272	0.408480	0.975092	0.489
15035	0.214075	0.414685	0.984153	0.455
15036	0.195962	0.422958	0.989922	0.369
15037	0.188443	0.433299	0.982936	0.424
15038	0.186173	0.441572	0.961821	0.491

15039 rows × 4 columns

```
x=data.drop('TIT',axis=1)
y=data[['TIT']]
```

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

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=12)
```

```
import tensorflow as tf
from tensorflow import keras
```

```
#Model Building
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Dense(5,input_dim=3, activation='relu'))   # Hidden Layer
model.add(tf.keras.layers.Dense(10,activation='relu'))                 # Hidden Layer
model.add(tf.keras.layers.Dense(1))                                    # Output Layer

model.summary()
```

```
Model: "sequential_8"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_27 (Dense)            (None, 5)                 20

 dense_28 (Dense)            (None, 10)                60

 dense_29 (Dense)            (None, 1)                 11

=================================================================
Total params: 91
Trainable params: 91
Non-trainable params: 0
_________________________________________________________________
```

```
# Model Compilation
model.compile(optimizer='adam',loss='mae',metrics=['mae'])
```

```
#model Traing
model.fit(x_train,y_train,epochs=10)
```

```
Epoch 1/10
376/376 [==============================] - 1s 1ms/step - loss: 1115.9911 - mae: 1115
Epoch 2/10
376/376 [==============================] - 1s 1ms/step - loss: 1082.8569 - mae: 1082
Epoch 3/10
376/376 [==============================] - 1s 1ms/step - loss: 1082.1239 - mae: 1082
Epoch 4/10
376/376 [==============================] - 1s 1ms/step - loss: 1024.6748 - mae: 1024
Epoch 5/10
376/376 [==============================] - 1s 1ms/step - loss: 22.2365 - mae: 22.2365
Epoch 6/10
376/376 [==============================] - 1s 1ms/step - loss: 13.3415 - mae: 13.3415
Epoch 7/10
376/376 [==============================] - 1s 1ms/step - loss: 13.0677 - mae: 13.0677
Epoch 8/10
376/376 [==============================] - 1s 1ms/step - loss: 12.8847 - mae: 12.8847
Epoch 9/10
376/376 [==============================] - 1s 1ms/step - loss: 12.7848 - mae: 12.7848
Epoch 10/10
376/376 [==============================] - 1s 1ms/step - loss: 12.7288 - mae: 12.7288
<keras.callbacks.History at 0x7f176f153b50>
```

```
result = model.evaluate(x_test,y_test)
result
```

```
94/94 [==============================] - 0s 1ms/step - loss: 12.6954 - mae: 12.6954
[12.695395469665527, 12.695395469665527]
```

```
print('accuracy : ',round(result[1],4))
print('Loss     : ',round(result[0],4))
```

```
accuracy :  12.6954
Loss     :  12.6954
```