```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns

np.set_printoptions(precision=3, suppress=True)

import tensorflow as tf

from tensorflow import keras
from tensorflow.keras import layers


print(tf.__version__)

url = '/content/tmpsInfect.csv'
column_names = ['tx', 'nbr', 'txVaccin', 'txQuar', 'txInfect',
                'tmpsInfect', 'tmpsQuar', 'tmpsVoyage']

raw_dataset = pd.read_csv(url, names=column_names,
                          na_values='?', comment='\t',
                          sep=',', skipinitialspace=True)

dataset = raw_dataset.copy()
dataset.tail()

unit = 'tmpsInfect'
```

```
     2.7.0
```

```
train_dataset = dataset.sample(frac=0.8, random_state=0)
test_dataset = dataset.drop(train_dataset.index)

train_dataset.describe().transpose()
```

	count	mean	std	min	25%	50%	75%	max
tx	517.0	67.379110	4.942137e+01	1.00	4.00	93.00	113.00	120.00
nbr	517.0	150.000000	0.000000e+00	150.00	150.00	150.00	150.00	150.00
txVaccin	517.0	0.200000	1.778077e-15	0.20	0.20	0.20	0.20	0.20
txQuar	517.0	0.000000	0.000000e+00	0.00	0.00	0.00	0.00	0.00
txInfect	517.0	0.300000	3.333895e-16	0.30	0.30	0.30	0.30	0.30
tmpsInfect	517.0	4.907408	2.756545e+00	0.05	2.62	4.85	7.19	9.98
tmpsQuar	517.0	1.000000	0.000000e+00	1.00	1.00	1.00	1.00	1.00
tmpsVoyage	517.0	100.000000	0.000000e+00	100.00	100.00	100.00	100.00	100.00

```
sns.pairplot(train_dataset[['tx', unit]], diag_kind='kde')
```

```
<seaborn.axisgrid.PairGrid at 0x7f783f59fc10>
```

```
#coder fct pour supprimer les valeurs trop absurdes
train_features = train_dataset.copy()
test = train_features.pop(unit)
toto = train_features.pop('tx')
l1 = test.values.tolist()
l2 = toto.values.tolist()

print(len(l1))

fin = []

for i in range (0,20):
  a = i/2
  b = a + 0.5

  l = []
  for j in range(len(l1)):
    p = l1[j]
    if (p >= a) and (p <= b):
      l.append(j)


  sumi = 0

  for u in l:
    sumi = sumi + l2[u]

  if len(l) == 0:
    print()
  else:
    sumi = sumi / len(l)
    #print(sumi)
    error = sumi / 10
```

```
    for k in l:
      if (l2[k] < sumi + error) and (l2[k] > sumi - error):
        fin.append(k)

print(fin)
```

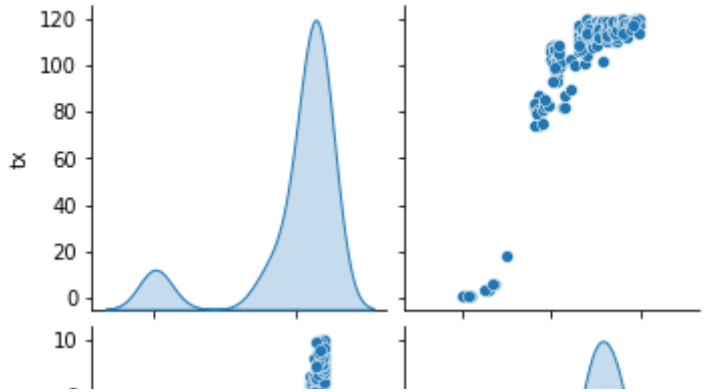
```
     517
     [1, 3, 15, 60, 100, 135, 159, 233, 243, 272, 277, 278, 284, 338, 362, 373, 421, 460, 463, 486, 237, 498, 213, 339, 355, 0, 23,
```

```
data = []
for j in fin:
  data.append([l1[j],l2[j]])

df = pd.DataFrame(data, columns = [unit, 'tx'])


df.describe().transpose()
```

	count	mean	std	min	25%	50%	75%	max
tmpsInfect	245.0	6.669633	2.598107	0.05	5.25	7.24	8.51	9.98
tx	245.0	98.902041	34.333206	1.00	102.00	113.00	117.00	120.00

```
sns.pairplot(df[['tx', unit]], diag_kind='kde')
```

```
<seaborn.axisgrid.PairGrid at 0x7f78331be390>
```




```
dataset = df.copy()
dataset.tail()

train_dataset = dataset.sample(frac=0.8, random_state=0)
test_dataset = dataset.drop(train_dataset.index)

train_dataset.describe().transpose()
```

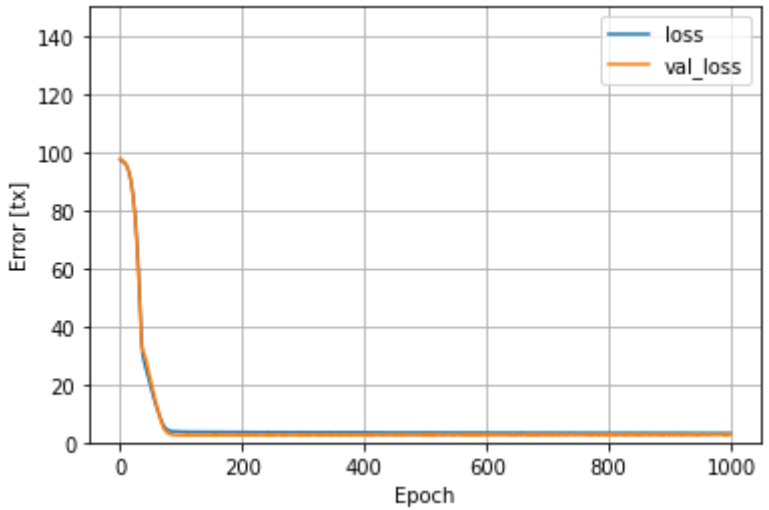
	count	mean	std	min	25%	50%	75%	max
tmpsInfect	196.0	6.537143	2.679027	0.05	5.14	7.195	8.4225	9.98
tx	196.0	97.489796	36.180245	1.00	101.75	113.000	117.0000	120.00

```
train_features = train_dataset.copy()
test_features = test_dataset.copy()

train_labels = train_features.pop('tx')
test_labels = test_features.pop('tx')


normalizer = tf.keras.layers.Normalization(axis=-1)
normalizer.adapt(np.array(train_features))

horsepower = np.array(train_features[unit])
```

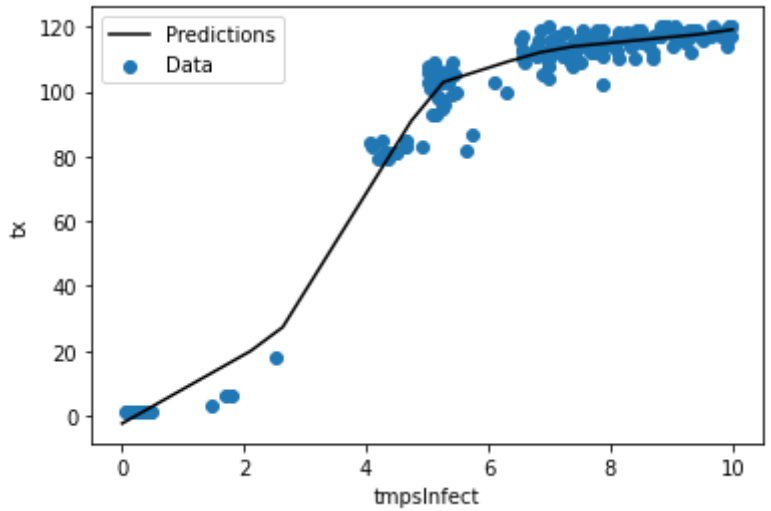
```
horsepower_normalizer = layers.Normalization(input_shape=[1,], axis=None)
horsepower_normalizer.adapt(horsepower)


def build_and_compile_model(norm):
  model = keras.Sequential([
      norm,
      layers.Dense(64, activation='relu'),
      layers.Dense(64, activation='relu'),
      layers.Dense(1)
  ])

  model.compile(loss='mean_absolute_error',
                optimizer=tf.keras.optimizers.Adam(0.001))
  return model


dnn_horsepower_model = build_and_compile_model(horsepower_normalizer)
dnn_horsepower_model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 normalization_1 (Normalizat  (None, 1)                3
 ion)

 dense (Dense)               (None, 64)                128

 dense_1 (Dense)             (None, 64)                4160

 dense_2 (Dense)             (None, 1)                 65

=================================================================
Total params: 4,356
Trainable params: 4,353
Non-trainable params: 3
_________________________________________________________________
```

```
%%time
history = dnn_horsepower_model.fit(
    train_features[unit],
    train_labels,
    validation_split=0.2,
    verbose=0, epochs=1000)

     CPU times: user 30.6 s, sys: 1.57 s, total: 32.1 s
     Wall time: 41.5 s


def plot_loss(history):
  plt.plot(history.history['loss'], label='loss')
  plt.plot(history.history['val_loss'], label='val_loss')
  plt.ylim([0, 150])
  plt.xlabel('Epoch')
  plt.ylabel('Error [tx]')
  plt.legend()
  plt.grid(True)

plot_loss(history)
```



```
x = tf.linspace(0.0, 10, 20)
y = dnn_horsepower_model.predict(x)

def plot_horsepower(x, y):
  plt.scatter(train_features[unit], train_labels, label='Data')
  plt.plot(x, y, color='k', label='Predictions')
  plt.xlabel(unit)
  plt.ylabel('tx')

  plt.legend()

plot_horsepower(x, y)
```




```
dnn_horsepower_model.evaluate(
    test_features[unit], test_labels,
    verbose=0)
```

```
 4.190896034240723
```

```
dnn_horsepower_model.predict([3])
```

```
     array([[38.47]], dtype=float32)


dnn_horsepower_model.save('/content/model.h5')

#lancer avec txInfect entre 0.2 et 0.4
```